TESE DE DOUTORAMENTO

LOW POWER CMOS VISION SENSOR FOR FOREGROUND SEGMENTATION

Daniel García Lesta

ESCOLA DE DOUTRAMENTO INTERNACIONAL DA UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

PROGRAMA DE DOUTORAMENTO EN INVESTIGACIÓN EN TECNOLOXÍAS DA INFORMACIÓN

SANTIAGO DE COMPOSTELA
2021

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

DECLARACIÓN DO AUTOR DA TESE

Low power CMOS vision sensor for foreground segmentation

Don Daniel García Lesta

Presento a miña tese, seguindo o procedemento adecuado ao Regulamento, e declaro que:

1. *A tese abarca os resultados da elaboración do meu traballo.*
2. *De selo caso, na tese faise referencia ás colaboracións que tivo este traballo.*
3. *A tese é a versión definitiva presentada para a súa defensa e coincide coa versión enviada en formato electrónico.*
4. *Confirmo que a tese non incorre en ningún tipo de plaxio doutros autores nin de traballos presentados por min para a obtención doutros títulos.*

En Santiago de Compostela, Abril de 2021

Asdo. Daniel García Lesta

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

AUTORIZACIÓN DO DIRECTOR/TITOR DA TESE

D. **Víctor Manuel Brea Sánchez**
En condición de: **Director/Titor**
Título da tese: **Low power CMOS vision sensor for foreground segmentation**

INFORMA:

Que a presente tese, correspóndese co traballo realizado por D. **Daniel García Lesta**, baixo a miña dirección/titorización, e autorizo a súa presentación, considerando que reúne os requisitos esixidos no Regulamento de Estudos de Doutoramento da USC, e que como director/titor desta non incorre nas causas de abstención establecidas na Lei 40/2015.

En Santiago de Compostela, Abril de 2021

Sinatura electrónica

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

AUTORIZACIÓN DO DIRECTOR/TITOR DA TESE

Dna. **Paula López Martínez**
En condición de: **Directora**
Título da tese: **Low power CMOS vision sensor for foreground segmentation**

INFORMA:

Que a presente tese, correspóndese co traballo realizado por D. **Daniel García Lesta**, baixo a miña dirección/titorización, e autorizo a súa presentación, considerando que reúne os requisitos esixidos no Regulamento de Estudos de Doutoramento da USC, e que como director/titor desta non incorre nas causas de abstención establecidas na Lei 40/2015.

En Santiago de Compostela, Abril de 2021

Sinatura electrónica

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Á miña familia

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

It matters not what someone is born, but what they grow to be.

Albus Dumbledore

An expert is a person who has made all the mistakes that can be made in a very narrow field.

Niels Bohr

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Agradecementos

Esta tese non tería sido posible sin a axuda de moitas persoas. Sirvan estas líneas para expresar o meu profundo agradecemento a todas elas.

En primeiro lugar, ós meus directores de tese, Víctor Brea e Paula López, por confiar en min para levar a cabo este traballo, e por toda a axuda recibida durante a súa realización. Quero agradecer tamén ó Prof. Diego Cabello, por compartir a súa gran experiencia sempre que o necesitei.

To Prof. Piotr Dudek, Stephen Carey, and all the people of D2, for the exceptional treatment I received during my stay in Manchester, and for all the rich and enlightening discussions we had. To Miguel, for being the best flatmate I could have.

Ó meus compañeiros do CiTIUS, especialmente ós do grupo de Visión Artificial, por todas as conversacións relacionadas coa investigación, e máis aínda polas que non.

A Cintia, polo seu apoio incuestionable, e por toda a axuda recibida durante a realización desta tese. Ós meus amigos, por recordarme a importancia de desconectar de cando en vez e sair un pouco da casa.

Finalmente, quixera agradecer á miña familia, especialmente ós meus pais, por facerme ver a importancia dunha boa formación e por dotarme de todo o necesario para poder recibila.

Tamén debo agradecer ós proxectos que financiaron esta tese: Consellería de Cultura, Educación e Ordenación Universitaria (2016-2019, ED431G/08 and ED431C2017/69), Ministerio de Economía, Industria y Competitividad (BES-2016-076501, TEC2015-66878-C3-3-R and RTI2018-097088-B-C32) e European Regional Development Fund (ERDF).

Abril de 2021

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Resumo

Nas últimas décadas o crecente número de cámaras fixo inviable analizar toda a información capturada por elas de maneira manual, polo que, de non empregarse técnicas de procesado de vídeo automatizado, unha gran parte desta información desaproveitaríase. Os métodos clásicos de procesado de imaxe baséanse en capturar vídeo con cámaras comerciais para despois enviar estes datos a outra plataforma para ser procesados, tales como un PC ou un servidor na nube. Esta transmisión pode ser a través dunha conexión cableada ou vía inalámbrica, coas implicacións enerxéticas e de ancho de banda que iso conleva. A información enviada pola cámara pode ser procesada en tempo real, se se deben tomar accións rápidas en función dos datos analizados, ou poden ser procesados a posteriori.

Esta metodoloxía de análise de datos é a mellor posible para moitas aplicacións, onde a opción de utilizar técnicas do estado da arte con imaxes de alta calidade e resolución ofrece os mellores resultados posibles. Con todo, en moitas ocasións isto non é unha opción por diversos motivos. Un dos máis importantes son as ligaduras enerxéticas, por exemplo, cando o dispositivo non pode ser conectado á rede eléctrica por estar situado nunha localización remota. Neses casos, o sistema debe ser alimentado por unha batería, que pode ser recargada por algún sistema de recolección de enerxía, como unha célula solar. Co obxectivo de aumentar tanto como sexa posible o tempo de vida do sistema, ou mesmo facelo perpetuo, o consumo de potencia debe ser tan baixo como sexa posible.

Outra ligadura importante é o ancho de banda dispoñible para o envío da información. Relacionado co punto anterior, algúns dispositivos localizados en lugares remotos non poden ser dotados dunha conexión de alta velocidade, necesaria para enviar as imaxes capturadas sen procesar. Neses casos, o procesamento debe ser realizado in-situ, e o envío de datos soamente se poderá producir cando sexan detectados determinados eventos.

Outro elemento que gañou gran relevancia nos últimos anos, respecto ao envío de imaxes capturadas en lugares públicos, é a privacidade. Con este asunto adquirindo cada vez máis importancia na sociedade, é moi importante ser coidadoso co envío de vídeo a través da rede, aínda máis cando este envío se fai de maneira inalámbrica. Loxicamente, o vídeo capturado pode ser encriptado antes do seu envío para mellorar o aspecto de seguridade. Con todo, esta capa adicional de procesamento engade un nivel extra de complexidade ao sistema.

Para dar solución a estes problemas, nesta tese utilizouse a computación no borde, é dicir, o paradigma de dotar aos sensores con capacidade de computación. Estas capacidades poden ser empregadas para a implementación desde tarefas de visión temperá, como algoritmos de

filtrado da imaxe capturada, ata algoritmos complexos tales como detección de obxectos. Con este procesamento sendo executado preto do sensor, o aforro enerxético é considerable debido a que xa non é necesario enviar os datos capturados crus, se non que só se enviarán os eventos detectados, reducindo tamén en varias ordes de magnitude o ancho de banda requirido.

A computación no borde pódese levar a cabo de varias maneiras. A primeira é utilizando dispositivos dixitais comerciais, tales como unha Nvidia Jetson ou unha Raspberry Pi. Estes dispositivos ofrecen un consumo de potencia baixo, cando son comparados con servidores de computación, ou PCs con GPUs comerciais. Ademais, ofrecen unha facilidade de programación alta, comparada coas outras opcións sinaladas a continuación. Outra posibilidade con menor consumo de potencia é a utilización de circuítos integrados dixitais, tales como FPGAs ou ASICs dixitais deseñados exprofeso para o caso de uso. Estes elementos elevan a complexidade de deseño, xa que son programados con linguaxes de descrición hardware (HDL), que tipicamente require dun maior tempo de deseño que as linguaxes software, pero ofrecen un consumo preto dunha orde de magnitude menor.

Nesta tese búscase o menor consumo de potencia posible. Para iso, as opcións anteriores foron descartadas, optando polo deseño de circuítos integrados en sinal mixto. Esta opción ofrece un menor consumo de potencia, co prezo dunha dificultade e tempo de deseño tipicamente maior que os demais candidatos, ao ser a súa funcionalidade implementada a nivel de transistor. Ademais, de maneira diferente aos circuítos dixitais, que poden ser implementados fisicamente con ferramentas automáticas de ”place and route”, os layouts de deseños analóxicos deben ser realizados manualmente.

Así, o obxectivo principal desta tese é o deseño dun chip de visión, implementado utilizando unha tecnoloxía CMOS estándar, que integre un procesamento por píxel para a subtracción de fondo, ou de maneira equivalente, a detección de fronte. Este tipo de procesamento utilízase para detectar a rexión de interese dunha escena, e aplicar nela algoritmos máis complexos, tales como detectores de características ou redes neuronais de seguimento de obxectos.

Hoxe en día, as clasificacións de subtracción de fondo, como *changedetection*, están dominados por algoritmos baseados en redes neuronais de aprendizaxe profunda. Con todo, ao comezo deste doutoramento, estas clasificacións estaban copados por algoritmos heurísticos baseados en regras. Por tanto, o algoritmo elixido para a súa implementación no plano focal foi o PBAS, debido aos seus bos resultados e as súas características de procesamento por píxel, que o facían un gran candidato para este traballo.

No Capítulo 2 desta tese explícanse as súas principais características, e a adaptación levada a cabo para unha implementación viable no plano focal, i.e., no dominio analóxico, resultando no algoritmo chamado HO-PBAS. Como se dixo previamente, PBAS está deseñado para ter un procesamento por píxel, onde cada un ten un modelo do fondo construído por mostras previas da escena. Estes modelos de fondo baseados en representacións non paramétricas da escena supuxeron un salto cualitativo con respecto a algoritmos previos. Un dos primeiros en introducilo foi o algoritmo ViBe, que ademais incluíu un mecanismo baseado na selección aleatoria da mostra do modelo a substituír para a súa actualización. Este mecanismo substitúe ao esquema tradicional de first-in first-out (FIFO), e implica a necesidade de xerar números aleatorios para a execución do algoritmo. Así, cada vez que se procesa unha imaxe, o algoritmo decide aleatoriamente que píxeles, que non fosen segmentados como fronte, actualizarán o seu modelo de fondo, e aqueles que o fagan, elixirán tamén aleatoriamente que mostra actualizar. Ademais, para incluír obxectos que permaneceron estáticos por un tempo no modelo de fondo, ViBe inclúe un mecanismo chamado polos autores difusión. Despois de levar a cabo o paso de actualización do modelo de fondo, aqueles píxeles que fosen segmentados como fondo decidirán aleatoriamente se executar a difusión ou non, e aqueles que así o decidan inducirán a un dos seus veciños para actualizar o seu modelo. Este veciño é elixido aleatoriamente, e procederá a actualizar unha mostra do seu modelo de fondo repetindo o mecanismo anterior, independentemente do resultado da segmentación. Desta maneira, os obxectos estáticos na escena serán incluídos no fondo aos poucos desde o seu contorno.

O mecanismo de segmentación deste tipo de algoritmos de subtracción de fondo baséase na comparación da imaxe capturada con todas as mostras do modelo. Cada vez que chega un novo valor da imaxe para cada píxel, constrúese unha esfera no espazo de cor de traballo centrada neste valor, cun radio fixo, e cóntase cantas mostras do modelo do fondo se atopan no seu interior. Se o número é maior que un certo limiar, considerarase que o novo valor é similar ao fondo e clasificarase como tal, e en caso contrario será segmentado como fronte. Isto xera un gran rendemento para a segmentación en escenas con fondos dinámicos, xa que o modelo pode conter mostras de diferentes obxectos (por exemplo, no caso dun fondo que conteña unha árbore cuxas ramas se moven polo vento).

PBAS engadiu a ViBe un mecanismo de realimentación para axustar diferentes parámetros do algoritmo en función da escena, facendo ademais estes parámetros axustables para cada píxel. Isto mellorou aínda máis os resultados de ViBe. Como o algoritmo debía ser implementado en hardware no dominio do sinal mixto, PBAS foi modificado para axustar as ecuacións

usadas no seu procesamento, así como para eliminar aquelas accións no camiño de datos con menor impacto no resultado da segmentación. Tras estas modificacións, o algoritmo resultante, chamado HO-PBAS, foi comparado contra a base de datos pública *changedetection*, mostrando mellores resultados en valor medio que o algoritmo orixinal, utilizando menos mostras para o modelo de fondo.

Tras o estudo e deseño do algoritmo a utilizar, no Capítulo 3 os circuítos necesarios para a súa implementación foron deseñados. En primeiro lugar, realizouse un estudo do estado da arte, onde se analizaron diferentes traballos publicados nos últimos anos. A maioría destes traballos utilizan, en maior ou menor medida, un procesamento no dominio analóxico para obter un consumo de potencia baixo. Ademais, moitos deles almacenan información en memorias analóxicas, implementadas en modo voltaxe ou corrente. Con todo, as características de HO-PBAS en termos de acceso a memorias antes de ser actualizadas, ou o tempo que deberán ser almacenadas, xera unha peculiaridade con respecto a estes traballos previos. Para solucionar este aspecto, levouse a cabo un estudo profundo sobre as características principais de cada tipo de memoria analóxica en modo voltaxe, sendo descartadas as memorias de corrente. Tras ser analizadas a través de simulacións eléctricas, os erros de todas elas (lectura, escritura e deterioración temporal debido ás fugas de corrente) foron incluídos no código fonte desenvolto en C++ e OpenCV de HO-PBAS. Esta versión máis próxima ao hardware do algoritmo foi posta a proba coa base de datos *changedetection*, estudando o impacto das non-idealidades na calidade da segmentación. Baseándose na análise dos resultados, decidiuse implementar memorias cunha arquitectura de lazo aberto e un seguidor de tensión para cada memoria individual.

O primeiro chip deseñado para o HO-PBAS, chamado HOPBAS1K, basea o seu funcionamento en operacións analóxicas implementadas con circuítos de condensadores en conmutación, controlados por unha unidade de control dixital que envía as mesmas sinais a todos os píxeles. O circuíto principal que executa as operacións linealizadas deseñadas para HO-PBAS é a Unidade Aritmética, baseada nun amplificador realimentado de alta ganancia con dous condensadores, que ofrece unha ganancia e offset programables.

Para reducir a área requirida por píxel, e aumentar así a resolución espacial do sensor de visión, optouse por unha arquitectura onde algunha circuitería é compartida por varios píxeles. Para iso, primeiro identificáronse aqueles bloques que pola súa función debían estar implementados individualmente en cada píxel. Algúns dos máis importantes son, entre outros, a lóxica local para a toma de decisións da actualización do modelo de fondo, o bloque

de memorias analóxicas para o almacenamento do modelo, ou o bloque funcional que implementa o filtrado pasa baixa da imaxe capturada. Este último bloque, cun gran impacto na calidade final da segmentación, foi desenvolto cunha rede de condensadores conmutados entre os píxeles que, por compartición de carga, implementan un filtrado Gaussiano, cuxa dispersión pode ser controlada co número de ciclos de reloxo que se repite esta operación.

Pola outra banda, todos aqueles bloques de procesamento que poden ser compartidos integráronse nunha Unidade de Procesamento (PU), que xunto aos píxeles para os que é compartida forma o Elemento de Procesamento (PE). Tras un estudo sobre as implicacións de compartir unha PU con máis ou menos píxeles, unha arquitectura con relación de catro píxeles por cada PU foi elixida. O layout deste PE foi desenvolto de maneira que os píxeles están homoxéneamente distribuídos, evitando con iso posibles artefactos na captura da imaxe. HOPBAS1K ofrece como saídas o resultado da segmentación codificado en 1 bit por cada píxel, así como a imaxe capturada en 8 bits, tras ser convertida ao dominio dixital por un conversor analóxico a dixital de rampla, implementado nunha arquitectura dun conversor por columna. Debido ao número limitado de pads no chip, o control foi implementado dentro do chip, utilizando ferramentas de síntese dixital, co inconveniente de non poder facer modificacións ao mesmo unha vez o chip fose fabricado. HOPBAS1K foi fabricado nunha tecnoloxía CMOS estándar, ocupando unha área de 1.6×3.2 mm^2, e cunha resolución espacial de 24×56 píxeles.

No Capítulo 4 explícase a experimentación co chip fabricado. Para iso, deseñouse un circuíto impreso que integra un conector para HOPBAS1K, a circuitería necesaria para a alimentación, unha matriz de potenciómetros para xerar as voltaxes de referencia e polarización necesarios, e os elementos necesarios para conectar a FPGA CMOD A7 35T (utilizada para a xeración dos sinais de reloxo e para a lectura da imaxe capturada e procesada), e o microcontrolador Particle Photon (utilizado para a experimentación cos bloques individuais de testeo de HOPBAS1K, así como para xerar os sinais de configuración do chip). En primeiro lugar, os bloques individuais de testeo foron analizados, sendo estes aqueles necesarios para a execución do algoritmo HO-PBAS, así como o conversor analóxico a dixital ou o xerador de números aleatorios, necesario para a actualización do modelo de fondo, e deseñado ad-hoc para esta tese.

Para a xeración de números aleatorios empregouse un sistema caótico, implementado cun circuito non lineal, cunha función de transferencia en forma de tenda de campaña. Este tipo de circuitos funcionan empregando como entrada a saída da súa iteración anterior, xerando series de datos pseudo-aleatorios. Este circuito non lineal creouse empregando unha unidade

aritmética con dúas configuracións posibles, correspondentes ás dúas rectas que forman a tenda de campaña. Combinando varias unidades deste tipo, con diferentes formas da tenda de campaña, e un postprocesado das sinais obtidas con portas lóxigas do tipo XOR, conseguiuse un xerador de números aleatorios analóxicos e dixitais cunha baixa correlación temporal. A análise dos datos obtidos de este circuito, a partir de probas experimentais, mostraron unha saída analóxica distribuida en todo o rango de valores posibles, e unha saída dixital de baixo sesgo. Ademais, para mellorar a calidade da saida deste xerador de números aleatorios, deseñouse un sistema no cal o circuito opera a baixa frecuencia durante o tempo de exposición da imaxe, e os valores necesarios se suministran rapidamente cando son necesarios. Os resultados para estas probas foron os esperados.

Tras a análise dos bloques de testeo individuais, procedeuse á comprobación a nivel de sistema. Nestas probas detectáronse problemas coa captura da imaxe, aparecendo píxeles que ofrecían un nivel de offset independente da escena, aleatoriamente distribuídos na matriz de píxeles, así como unha esquina que sempre está saturada próxima ao bloque dixital de xeración de sinais de control. Estes problemas non puideron ser solucionados para HOPBAS1K, pero si para a segunda versión do chip, o HOPBAS10K. Con respecto ao resultado da segmentación de fronte, detectouse un problema grave no almacenamento do resultado na memoria dixital implementada en cada píxel, sendo posible soamente ler o resultado do último píxel procesado en cada PU, co resultado de todos os demais igual a 1, correspondente ao valor de fronte. Tras unha análise exhaustiva, a orixe deste problema foi atopado no solape de dous sinais de control, responsables da escritura do valor da segmentación na memoria e do reinicio do bloque de procesamento que calcula ese valor.

Finalmente, no Capítulo 5 detállase o deseño dun segundo chip que implementa HOPBAS, chamado HOPBAS10K. Esta segunda iteración do traballo foi realizada para solucionar os problemas detectados durante as probas realizadas con HOPBAS1K. A área de silicio dispoñible para este chip foi de 5×5 mm^2, o que permitiu incrementar a resolución espacial a 98×98 píxeles. Ademais, a opción de utilizar sinais de control xeradas no exterior foi engadida, o que permitiu solucionar algúns problemas detectados nas probas realizadas con este chip.

En primeiro lugar, solucionáronse os problemas coa imaxe capturada. O primeiro, o correspondente aos píxeles aleatoriamente distribuídos que ofrecían o valor capturado cun offset constante, foi solucionado reducindo a voltaxe de reinicio do fotodiodo. Seguidamente, chegouse á conclusión de que os efectos de borde da matriz de píxeles, como o problema da

esquina saturada próxima ao bloque de control dixital, estaban orixinados en axustes con ruído eléctrico da periferia. A solución a isto foi cambiar o esquema de operación, parando todos os sinais dixitais mentres a imaxe estaba a ser capturada. Ademais da imaxe capturada, HOPBAS10K ofrece a opción de visualizar a imaxe despois do filtrado Gaussiano, implementado por compartición de carga entre condensadores que unen os píxeles. As probas realizadas para o filtrado Gaussiano mostraron resultados favorables, sendo posible controlar a dispersión da distribución Gaussiana co número de ciclos que se repetía esta operación.

A continuación, comprobouse que, efectivamente, o problema de non almacenar correctamente o resultado da segmentación estaba orixinado polo solape de sinais previamente descrito, ofrecendo correctamente HOPBAS10K o resultado da segmentación de todos os píxeles. Isto permitiu realizar múltiples probas con diferentes parámetros de configuración do chip, tales como a dispersión do filtrado Gaussiano, os valores máximos e mínimos dos parámetros do algoritmo, ou a velocidade de procesamento. Ademais, o proceso de difusión, polo cal os obxectos estáticos na imaxe son incorporados no modelo de fondo, puido ser testeado, verificando que a súa implementación foi a correcta.

Ao poder acceder correctamente ao resultado da subtracción de fondo, unha análise cualitativa do mesmo puido ser levado a cabo. Para o devandito análise, dous vídeos de 600 imaxes cada un, elaborado en condicións de laboratorio e cunha frecuencia de operación de 50 imaxes por segundo, foron capturados e procesados por HOPBAS10K, sendo almacenados os resultados de captura e segmentación nun PC. Tras iso, os resultados de HOPBAS10K foron comparados cos obtidos tras procesar as mesmas imaxes cunha versión do algoritmo implementado en C++ e a librería de procesamento de imaxes OpenCV. O resultado observado foi unha perda de calidade da subtracción do 20%, compatible co esperado tras unha implementación hardware no dominio analóxico dun algoritmo tan complexo.

Unha das consecuencias da complexidade do algoritmo foi o consumo de potencia de HOPBAS10K, bastante superior a outros traballos do estado da arte. Para entender este alto consumo, é necesario lembrar que o longo camiño de datos do algoritmo implicou o uso de circuitería activa con baixo erro, xa que o uso de circuítos pasivos sen seguidores de tensión entre os diferentes bloques orixinaría un gran erro acumulado, provocando con iso unha calidade da subtracción de fondo moi baixa. Ademais, as estratexias utilizadas para obter un consumo de potencia non foron as máis arriscadas, xa que tratar de utilizar solucións demasiado complexas, ou arquitecturas máis avanzadas para os circuítos implementados, faría perigar o funcionamento dos chips proba-de-concepto desenvolvidos nesta tese.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Contents

CHAPTER 1

INTRODUCTION

1.1 Introduction

In the last decades the growing number of cameras made unfeasible to analyze all the data coming from them by human operators. Thus, if computer-based video analysis is not applied, a huge amount of relevant information is being thrown away without taking fully advantage of it. Classical approaches of video processing are based on commercial cameras capturing some type of scene and sending the raw data to a computer unit. This transmission can be both through a cable connection or wireless. In the computer unit, data are processed online or offline and decisions are taken if required.

This approach is the best one for many different applications, where the option of using the highest possible resolution multi-channel images with the top most powerful computer systems will offer the best possible results. Nevertheless, in those scenarios where this is not an option, other type of systems must be used. Some of these reasons might be, among others:

- Energy constraints: in some cases a power line is not available, for example if the device is placed in a remote location. In these cases, the system must be powered from a battery, which can be somehow recharged with an energy harvester such as a solar cell, but in order to optimize the lifetime or even to make the system perpetual, the power consumption should be reduced as much as possible [1].
- Bandwidth requirements: similarly to the previous point, devices in remote locations or places with hard access are difficult to be provided with a cable connection. Also, on many occasions a wireless connection is not an option or it is not fast enough for the

raw data transmission. Furthermore, fast data connections are power hungry, even more in the wireless case [2, 3].

- Privacy issues: raw data transmission naturally comes with the problem of sensitive data exposure. With this matter being of increasing relevance to the society in the recent years, it is of a great importance to be careful when sending video streaming through a network, even more, when the transmission is wireless. Of course, data transmission can be codified to avoid non-desired receivers to get all the information that was sent. However, to add a data encryption layer robust enough adds an additional level of challenge.

In order to overcome these issues, in this thesis we will focus on edge-computing, i.e., the paradigm of providing the sensors with computing capabilities to perform different types of processing as close to the sensor as possible [4, 5], without off-device assistance like cloud computing. This processing can be used to perform from early vision tasks, like image filtering up to complex algorithms such as object detection. With the processing included on the system, the required amount of data to be transmitted is much smaller, reducing the power consumption and the bandwidth by a wide margin. Also, as only the processed data is sent, many privacy issues are solved with no additional effort.

Edge-computing solutions can be implemented on a wide variety of devices, which can be mainly divided into three different categories, namely: *i)* commercial embedded digital platforms, such as the NVidia Jetson family or the Raspberri Pi [6, 7, 8], *ii)* custom digital platforms, such as FPGAs or digital ASICs [9, 10, 11], or *iii)* custom made mixed-signal ASICs [12, 13]. Whereas the first ones show the highest versatility and ease of use, they are the most power hungry, as they commonly need to power complex processor architectures such as CPUs and/or GPUs. In a medium point are the custom digital approaches, with an intermediate power consumption compared with the other two options. The same happens with the system design, normally developed with the help of Hardware Description Languages (HDLs) and digital synthesis tools, which stands between the pure-software platforms and the custom made mixed-signal ASICs, where the analog part and its interface with the digital part must be manually designed. Finally, despite the system design complexity, mixed-signal systems typically feature the lowest possible power consumption. In this work, the possible lowest power consumption is targeted. Hence, we will focus on the custom-made mixed-signal ASICs.

The main goal of this thesis is to build a system able to perform real-time video processing with a low power consumption. This video processing will be background subtraction, or equivalently, foreground detection, which consists in the detection of moving objects on a scene [14]. This information is then used to separate the foreground from the background of the image in order to focus the attention of higher level algorithms such as object recognition, classification or activity analysis, making the later steps more efficient [15].

Some examples of applications that rely on background subtraction are:

- Traffic monitoring: the ability to identify the different vehicles on a road is extremely useful for subsequent tasks such as accident detection, vehicle counting or vehicle identification [16, 17].
- Video surveillance: many applications might be included into this category, starting with homeland security (for example by monitoring people behaviour in transport networks, town centers or public facilities), to an efficient management and control of transport key points, such as road or railroad crossings or traffic lights [18].
- Industrial applications: industry commonly requires fast and reliable detections. A typical application is to detect the objects in a production line in order to perform some kind of analysis [19] (e.g. to analyze the color distribution of a vegetable to discard the ones in bad condition or to look for strange objects in the final product).

When it comes to the background subtraction algorithms used in real applications compared with the ones developed in fundamental research, the existing gap between them is remarkable [20], not to mention the ones implemented in silicon, that are often based on frame difference with some refinement or feedback mechanisms [21, 13]. This gap usually comes from the hardware requirements of complex algorithms, which lead to a reduced frame rate in software-based solutions and large pixel pitches in hardware implementations. In this thesis, one of the main objectives is to balance algorithm complexity, which is usually paired with accuracy, and pixel pitch, which calls for simplicity, while keeping the ability to solve real-life applications. This idea is implemented on a CMOS vision sensor chip in standard technologies, achieving a high computing power per energy unit.

1.2 Objectives

The main objectives of this thesis are:

O1 To select the most appropriate background subtraction algorithm for a hardware implementation in the analog domain.

O2 To perform a study of the possible modifications that can be applied to the selected algorithm in order to reduce the implementation complexity while maintaining the segmentation performance.

O3 To design the circuitry that implements the algorithm designed in objective O2.

O4 To design the system and architecture for the circuitry depicted in objective O3, studying which parts can be shared between pixels to reduce the pixel fill-factor.

O5 To implement the designed system into integrated circuits and to fabricate the experimental setup required for the testing of the chips.

O6 To test the fabricated chips, assessing their proper behavior and analyzing their output qualities.

1.3 Contributions

The main contributions of this dissertation are:

C1 Design of a Hardware-Oriented version of the PBAS (HO-PBAS) algorithm and study of the system non-idealities impact on the algorithm performance.

This contribution can be found in the following publications:

1 **García-Lesta, D**., López, P., Brea, V. M., & Cabello, D. (2018). In-pixel analog memories for a pixel-based background subtraction algorithm on CMOS vision sensors. International Journal of Circuit Theory and Applications, 46(9), 1631-1647.

2 **García-Lesta, D**., Brea, V. M., López, P., & Cabello, D. (2017, June). Effect of temporal and spatial noise on the performance of hardware oriented background extraction algorithms. In 2017 15th IEEE International New Circuits and Systems Conference (NEWCAS) (pp. 45-48).

3 **García-Lesta, D**., Brea, V. M., López, P., & Cabello, D. (2018, May). Impact of analog memories non-idealities on the performance of foreground detection algorithms. In 2018 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-5).

4 **García-Lesta, D**., Brea, V. M., López, P., & Cabello, D. (2018, May). Shannon Entropy as Background Dynamics Estimator In Foreground Detector Algorithms. In 2018 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-4).

5 Blanco-Filgueira, B., **García-Lesta, D**., Fernández-Sanjurjo, M., Brea, V. M., & López, P. (2019). Deep learning-based multiple object visual tracking on embedded system for iot and mobile edge computing applications. IEEE Internet of Things Journal, 6(3), 5423-5431.

C2 Implementation of the HO-PBAS on a custom-made mixed-signal CMOS vision sensor using standard CMOS technologies.

The publication where this contribution can be found is listed below:

1 **García-Lesta, D**., López, P., Brea, V. M., & Cabello, D. (2020, October). A CMOS Vision Sensor for Background Subtraction. In 2020 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-5).

1.3.1 Outline

This thesis manuscript is divided in six chapters. This first chapter serves as introduction. Chapter 2 covers the introduction to background subtraction algorithms and the design of the so-called HO-PBAS. Then, in Chapter 3 the design of a proof-of-concept chip that implements the HO-PBAS, HOPBAS1K, is shown. Experimental setup and results from the first tape-out or iteration of the chip can be found in Chapter 4. In Chapter 5 a second iteration of the proof-of-concept chip, HOPBAS10K, with larger image resolution, and where the main issues of the first iteration are solved, is depicted. Finally, on Chapter 6 the future work and main conclusions are shown.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 2

BACKGROUND SUBTRACTION

Introduction

In order to segment the foreground, different options can be selected. This chapter covers the algorithmic part of the thesis, where the background subtractor method is chosen (Section 2.3). Once an algorithm is selected, the modifications proposed to make it suitable for a hardware implementation will be depicted, showing how the algorithm performance is affected by them (Section 2.4).

2.1 Foreground Detection

One of the first steps in computer-based video analysis is to detect the region of interest of the scene and then use this information for many different tasks, i.e., visual surveillance of human activities, visual observation of animals and insects behaviors, visual observation of natural environments, content-based video coding... [20, 14]. Regions of interest commonly correspond with moving objects that can be detected by comparing a reference frame with the current scene. This can be taken as the basic principle of a background subtractor, or equivalently, a foreground detector, an algorithm which analyzes the scene by comparing it with a background model to determine when each pixel is background (if the incoming value is similar to the background model) or foreground.

In the literature, many background subtraction techniques, with their different background model and segmentation mechanisms, have been proposed and several surveys can be found

about this topic; see for example [22, 23, 24]. According to the literature, a set of important features that these algorithms must include are:

- Adjustment to global illumination changes: this feature is of great relevance in outdoor scenarios, where the sunny illumination is in constant change, and the algorithm must be continuously adjusting the background model to include this phenomenon and avoid false positives.
- Inclusion of static foreground objects into the background model: these static objects can be either an object that stops in the scene for a long time (a car that has been just parked) or the silhouette that an object leaves when it moves from a position where it was already included into the background model, commonly known as their ghost. Through some mechanism, the algorithm should include both cases into the model.
- A mechanism to deal with dynamic backgrounds: in some situations there are moving objects that should not be considered as foreground, such as sea waves or tree branches moving because of the wind. These cases are challenging due to the difficulty of distinguishing these objects from foreground.

In the last two decades many solutions for foreground detection have been proposed. Although those based on deep learning strategies are clearly the ones with the best performance, as shown in public rankings such as *changedetection* [25], its complexity for an on-chip implementation in the analog domain discards them for this work. On the other hand, ruled-based algorithms are good candidates for such purpose, as shown in the literature [26]. Furthermore, pixel-wise solutions can be naturally deployed on smart imagers, forming massively parallel image processors. Thus, in this thesis we will focus on the latest.

In this line, we have chosen algorithms with local connectivity among pixels, which favor their implementation on CMOS vision sensors. ViBe [27] and PBAS [28] were top-ranked solutions at the starting time of this PhD.

2.2 ViBe

The problem addressed by background subtraction involves the comparison of the observed image with a model of the scene with no objects of interest, also referred as the background model. Thus, after this foreground segmentation, the output will be a binary image with the regions of interested marked as in Figure 2.1.

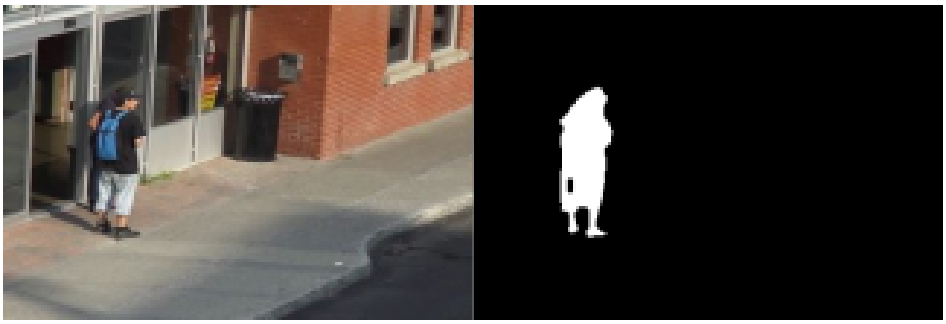

Figure 2.1: Background subtraction algorithm example. The left side shows a scene with two people, which are the interesting objects of the image. The right side corresponds to the binary mask that highlights the area where those people are placed. Source: *changedetection*.

To model the background different approaches have been proposed in the past years. Parametric models of the background gained a great popularity due to their good results, with the Gaussian Mixture Model (GMM) being the most popular [29]. However, despite their good performance compared with other algorithms, they were outperformed by algorithms based on non-parametric models of the background, such as ViBe [27]. In ViBe, the per-pixel background model is formed by N previous samples of the scene:

$$\mathbf{B}(x_i) = \{B_1(x_i), ..., B_k(x_i), ..., B_N(x_i)\} \tag{2.1}$$

This background model is used to be compared against the incoming pixel value $I(x_i)$. This comparison is performed by counting how many samples are inside the sphere with radius R and centered in $I(x_i)$, as shown in Figure 2.2, and checking if this number is below a fixed value $\#_{min}$. This procedure could be summarized as follows:

$$S_n(x_i) = \begin{cases} 1, & \#\{dist(I(x_i), B_k(x_i)) < R\} < \#_{min} \\ 0, & \text{else} \end{cases} \tag{2.2}$$

where $S_n(x_i)$ is the segmentation result for the frame n at pixel x_i. This result will be 1, i.e., foreground, if the number of samples inside the sphere does not exceed the minimum value $\#_{min}$, or 0, i.e. background, if otherwise.

Regarding the background model update mechanism, this is performed in two different ways. The first one is meant to adjust the model to slow changes, such as the global illumi-

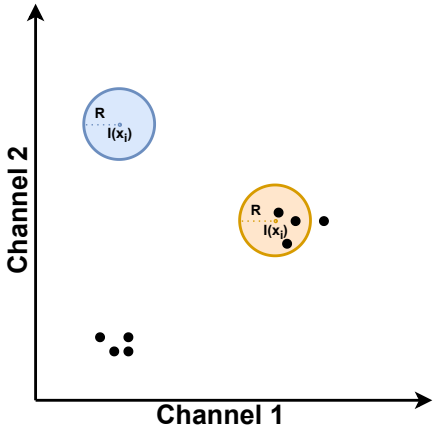

Figure 2.2: Segmentation mechanism example for a two channels color space in ViBe (27). In the case of the orange new pixel value three background model samples, represented as black dots, are inside the segmenter sphere. Thus, it will be considered background. On the other hand, the blue sphere has not any sample inside, which corresponds to a foreground pixel.

nation evolution during the day or the shadow projection variation of static objects caused by the movement of the sun, and it works by replacing a sample of the background model $B_k(x_i)$ by the new pixel value $I_n(x_i)$. This process is carried out with a probability $1/T$, where T is a fixed parameter called "learning parameter", which affects only over the pixels that were segmented as background. Also, those pixels that are going to update their background model select randomly the sample of their background model to be replaced, differently from the conventional first-in first-out approach [30].

This mechanism alone is not able to deal with situations where a foreground object stops in the scene and needs to be included into the background model after some time, as all of its pixels are segmented as foreground. This is achieved with the so-called diffusion mechanism, which acts as follows: after the first step of the background model update, each pixel segmented as background decides, again with a probability of $1/T$, if said diffusion will be performed. If the decision is affirmative, the algorithm randomly selects a neighbor of its vicinity to update its background model, even if it was previously segmented as foreground, in the same way as in the first step of the background model update mechanism. Therefore, after a certain number of frames, static foreground objects will be "eaten-up" from the outside until they are completely included into the model.

2.3 PBAS

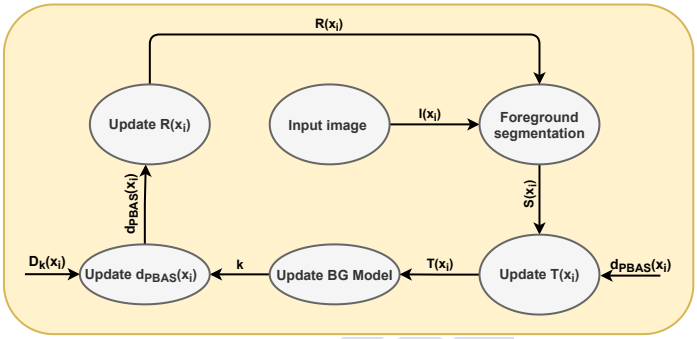

Figure 2.3: PBAS per-pixel feedback mechanism for algorithm parameters update.

Based on ViBe, the Pixel-Based Adaptive Segmenter (PBAS) incorporated a per-pixel feedback mechanism to adjust some algorithm parameters as a function of the scene, among other minor changes, which leads to better results than with ViBe [28]. This feedback scheme is summarized in Figure 2.3, where it can be seen that the main sources of the feedback are the segmentation result and the background dynamics estimator $d_{PBAS}(x_i)$. This background dynamics estimator tries to measure how chaotic the background model is. For that purpose, besides the background model, an array of minimal decision distances for every pixel $\mathbf{D}(x_i) = \{D_1(x_i), ..., D_k(x_i), ..., D_N(x_i)\}$ is also stored. Thus, every time the background model is updated substituting the k-sample of $\mathbf{B}(x_i)$, the minimal decision distance between the incoming pixel value and all the background model samples is incorporated into the k-position of $\mathbf{D}(x_i)$. The background dynamics estimator $d(x_i)$ is calculated as the average value of the minimal distances array:

$$d_{PBAS}(x_i) = \frac{1}{N}\sum_{k=0}^{N} D_k(x_i) \tag{2.3}$$

The background dynamics estimator is used to adjust two of the most important parameters of the algorithm, namely the pixel-dependant learning parameter $T(x_i)$, and the segmentation sphere radius $R(x_i)$, which also depends on the pixel in this algorithm. In PBAS, the sphere radius is adjusted to be higher in the zones with larger dynamics, thus reducing the

false positives. This operation is carried out as follows:

$$R(x_i) = \begin{cases} R(x_i) \cdot (1 - R_{inc\_dec}), & R(x_i) > d_{PBAS}(x_i) \cdot R_{scale} \\ R(x_i) \cdot (1 + R_{inc\_dec}), & \text{else} \end{cases} \tag{2.4}$$

where R_{inc_dec} and R_{scale} are fixed parameters. For a constant $d_{PBAS}(x_i)$ the radius of the sphere tends to $d_{PBAS}(x_i) \cdot R_{scale}$, and once there, a fast change of the background dynamics leads to a (slow) change towards the new value. Also, if the minimum value for $R(x_i)$ is not set, those zones with static backgrounds would highly suffer from false positives, as tiny variations caused by image noise or other sources would produce false positives. Hence, the result (2.4) is limited to a minimum value of R_{min}.

On the other hand, the learning parameter $T(x_i)$ is not only adjusted as a function of the background dynamics but it also depends on the segmentation result through:

$$T(x_i) = \begin{cases} T(x_i) + \frac{T_{inc}}{d_{PBAS}(x_i)}, & \text{if } S_n(x_i) = 1 \\ T(x_i) - \frac{T_{dec}}{d_{PBAS}(x_i)}, & \text{else} \end{cases} \tag{2.5}$$

where T_{inc} and T_{dec} are fixed parameters. Also, after (2.5) is applied, the result is constrained into the fixed range (T_{min},T_{max}). Therefore, the learning parameter is increased in those areas where the foreground objects appear more often, reducing the update probability $p(x_i) = 1/T(x_i)$, and consequently the chances of including wrong samples into the background model. One illustrative example of this situation is a road with moving vehicles on a lane and parking areas at its sides. When a car stops on the lane due to a traffic light indicator, because many cars have passed before, the learning parameter in this area will be high and, as a consequence, it is unlikely that the diffusion mechanism begins to actuate over it. Nevertheless, the situation changes when this same car stops at the side of the road, where the learning parameter will be some value near to the minimum, leading the cars that once were foreground to background.

In addition to the segmentation result, equation (2.5) also weights the variation of $T(x_i)$ with the background dynamics estimator $d_{PBAS}(x_i)$, allowing faster adjustments to those pixels belonging to areas with more static backgrounds as they will suffer less from missclassifications.

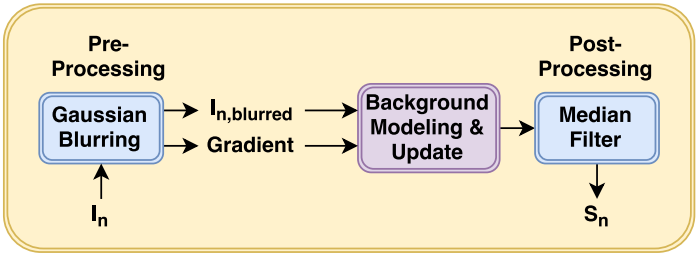

Figure 2.4: PBAS pre- and post-processing pipeline.

2.4 HO-PBAS

In order to avoid large pixels and low fill-factors, the PBAS algorithm needs to be simplified with the lowest possible degradation of quality metrics [31]. All the proposed modifications are to ensure a feasible hardware implementation, as it will be shown in Chapter 3, trying to simplify as much as possible the number of operations while maintaining the overall performance or slightly affecting it.

The first modification concerns the median filter. HO-PBAS removes it from its pipeline (see Figure 2.4), as rank filters might lead to very complex pixels and the performance degradation from the lack of the median filter is less relevant to the final accuracy of the algorithm [28]. However, the Gaussian blurring is kept as this is a critical step in background subtraction algorithms, and feasible in a close to the sensor implementation, i.e., in the analog domain [32, 33].

The second change made with respect to the original PBAS algorithm was to use only the Euclidean distance for pixel value comparison, without the image gradient contribution. Also, the neighborhood interaction was reduced from 8- to 4-connectivity, and the minimum number of background samples needed was also assessed in order to limit the number of in-pixel memories.

In terms of the algorithm itself, the PBAS equations were also simplified towards a more lineal model. In particular, the background model update mechanism was modified to avoid divisions, replacing (2.5) by (2.6), where $p(x_i)$ is the inverse of $T(x_i)$ and p_{dec} and p_{inc} are

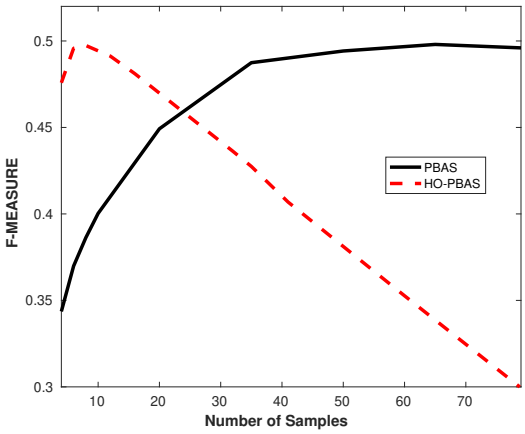

Figure 2.5: F-Measure of PBAS and HO-PBAS as a function of the number of samples N in the background model.

fixed parameters. In our approach, $d(x_i)$ is in the [0,1] range.

$$p(x_i) = \begin{cases} p_{min}, & \text{if } S(x_i) = 1 \\ p(x_i) + [1 - d(x_i)] \cdot p_{inc}, & \text{else} \end{cases} \tag{2.6}$$

Regarding the background dynamics estimator, an in-depth analysis was performed, studying different possibilities such as the standard deviation of the background model values, the Shannon Entropy of the model distribution [34] or the range of that distribution, among others. This analysis showed that the Shannon Entropy was the best estimator. However, its complexity makes unrealistic a feasible hardware in-pixel implementation. Hence, as a tradeoff between algorithm performance and design feasibility was selected. Therefore, the background dynamics estimator, $d(x_i)$, in (2.3) is replaced by (2.7), where β is a fixed parameter. Thus, the maximum difference between background model samples is measured, which should be larger in highly dynamic background zones of the scene than in static ones.

$$d(x_i) = \beta \cdot [max_k(B_k(x_i)) - min_k(B_k(x_i))] \tag{2.7}$$

The segmentation sphere's radius is obtained through an iterative process where the value tends to $d(x_i) \cdot R_{scale}$, as shown in (2.4). Through different simulations it was observed that with a set of optimized algorithm parameters $R(x_i)$ reaches that value within only 3 to 5 frames

Table 2.1: F-Measure for PBAS and HO-PBAS with number of samples of the background model N=8 and N=35 (best results in bold).

	N=8		N=35	
	Original	Modified	Original	Modified
shadow	0.6864	**0.7367**	**0.8058**	0.6249
badWeather	0.5492	**0.6987**	**0.7226**	0.5870
PTZ	0.0447	**0.1220**	0.0718	**0.1056**
dynamicBackground	0.1555	**0.4994**	0.3839	**0.5342**
cameraJitter	0.2387	**0.5585**	**0.4285**	0.2820
thermal	**0.6791**	0.3694	**0.6703**	0.2472
intermittentObjectMotion	**0.4107**	0.2793	0.4187	**0.6903**
turbulence	0.0625	**0.6718**	0.2112	**0.6281**
baseline	**0.7512**	0.7052	**0.7674**	0.6281
lowFramerate	0.3725	**0.5078**	**0.4941**	0.4071
nightVideos	0.2482	**0.4009**	**0.3417**	0.2331
Overall	0.3863	**0.4981**	**0.4874**	0.4275

and remains there unless $d(x_i)$ changes. Hence, in order to reduce the hardware implementation complexity (2.4) was substituted by (2.8).

$$R(x_i) = d(x_i) \cdot R_{scale} \tag{2.8}$$

This set of changes in the PBAS pipeline and the background model equations results in what was called by the authors the hardware-oriented PBAS (HO-PBAS) [31].

To assess the impact of HO-PBAS on the algorithm performance the public dataset *changedetection* [25] was used. This dataset contains 53 videos with 11 challenging categories for background subtraction, namely: dynamic background, camera jitter, intermittent object motion, shadows, thermal signatures, challenging weather, low frame-rate, acquisition at night, PTZ capture and air turbulence. Also, each video comes with a hand annotated groundtruth image for each frame, allowing to compute both true and false positives (TP and FP), and false negatives (FN).

HO-PBAS was implemented in C++ with the OpenCV library, and a quantitative performance evaluation was carried out through the F-Measure as figure of merit, calculated according to (2.9), with Precision = $TP/(TP+FP)$ and Recall = $TP/(TP+FN)$.

$$\text{F-Measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.9}$$

The results of this study are shown in Figure 2.5, where the average value of the F-Measure of all categories in the whole dataset is plotted as a function of N. It can be seen that, as expected, the original PBAS performance increases with the number of background samples N, reaching saturation around $N = 35$ [28]. However, its performance with fewer samples is clearly worse than that of HO-PBAS, which reaches the maximum F-Measure at N=8 and decreases with larger N. The disaggregate metrics are shown in Table 2.1 for N=8 and N=35. In 8 out of 11 categories and the overall average, HO-PBAS clearly outperforms the original PBAS for a reduced number of samples. Also, HO-PBAS with $N = 8$ outperforms PBAS with $N = 35$ for some categories, reinforcing the conclusion drawn from Figure 2.5 of the suitability of HO-PBAS for focal-plane processing.

2.5 Conclusions

As a conclusion, at the beginning of this thesis, a careful analysis of the state of the art in background subtraction was made, choosing PBAS as one of the top-ranked solutions of the *changedetection* dataset more suitable for a CMOS vision sensor with focal plane processing. Subsequently, a set of changes to make PBAS more hardware oriented led to HO-PBAS. It was shown that these changes keep or even feature higher accuracy than the conventional PBAS for some categories in *changedetection* [25]. Next chapter addresses the implementation of a CMOS vision sensor for HO-PBAS.

CHAPTER 3

HOPBAS1K DESIGN

Introduction

The objective of this chapter is to describe the design of the proof-of-concept CMOS vision sensor chip that implements the proposed background subtraction algorithm HO-PBAS, HOPBAS1K, described in Chapter 2. This chip is designed in standard 0.18 μm CMOS technology and will incorporate a 24×56 pixel array with on-chip control signals generation and a column parallel single-slope ADC. Inside the array, the pixels will be divided in groups of four and each group will have a shared processing unit to reduce the pixel pitch.

3.1 Related work

Many examples of focal-plane processors can be found in the literature[26]. In this section the state-of-the-art will be covered, showing the main features of recent and classic work. Also, due to the special memory requirements of HO-PBAS, i.e., storage time of the order of 10 seconds and ultra-low reading degradation, as explained in Chapter 2, the analog memories implemented at each work will be detailed.

A good example of a general purpose CMOS vision sensor is reported in [35]. In this work a massively parallel image processor is developed, where arithmetic and digital operations are executed in-pixel. Analog operations are performed using current-mode circuits taking advantage of their current memories. Also, each pixel features digital memories able to perform digital operations. In addition, 4-neighbor pixel connectivity is implemented, offering the possibility of executing different early vision algorithms based on n×n-kernel convolutional

operations. Although this platform is suitable for different real-time algorithms [12, 36], its in-pixel analog memories can not hold stored values for long times, which makes it inappropriate for applications that require long-term storage.

Reference [37] shows another work where a Single-Instruction Multiple-Data (SIMD) vision sensor with in-pixel processing capabilities is developed. Pixel architecture includes, in addition to the photodiode, two analog memories and amplifier and multiplexer structures which can be used to dissociate the acquisition of the current frame in the first memory and the processing of the previous frame in the second memory. In addition, each pixel features an analog arithmetic unit based on four analog multipliers which performs the linear combination of the four adjacent pixels using a 2×2 convolution kernel. Memory capacitors are about 40 fF and they are able to store a value for 20 ms with an error lower than 4%.

The solution reported in [21] is a low-power focal-plane processor in standard CMOS technology with a per-pixel approach that provides foreground segmentation. The algorithm implemented is based on frame difference with adaptive thresholds according to the pixel activity. This chip includes per-pixel analog memories in an open-loop topology to store the segmentation thresholds. The same capacitor of 220 fF is used for the memory and the low pass filter required in the algorithm. These capacitors are isolated with PMOS switches at the input node and they are continuously connected to a comparator at the output. Authors report a long-term storage degradation with the voltages drifting toward the supply voltage because of the junction leakage current at the PMOS switches.

Based on the double-threshold dynamic background subtraction algorithm developed in [21], the work reported in [38] implements a VGA sensor array with motion detection working in the digital domain. The system is able to perform the motion detection to a QQVGA subsampled image with a power consumption of 344 μW with per column parallel processing. When anomalous motion is detected in the scene the vision chip delivers gray-scale 8-bit VGA images with associated local binary pattern coding at 8 fps and 1.35 mW.

A vision chip that can work in an always-on motion detection is reported in [39]. This chip implements an in-pixel frame differencing algorithm with on/off event detection using analog capacitive memories. It also can provide the full captured image as well as saliency detection. Using a low-voltage ping-pong Pulse Width Modulation (PWM) pixel and multi-mode operation, it achieves high-speed low-power full-resolution motion detection, consecutive event frame reporting, and image capture functions. It consumes 71.2 μW at 360 fps with a figure-of-merit of 48.3 pJ/pixel·frame.

Another example of a smart CMOS vision sensor chip, also called retina chip, with per-pixel analog memories is [40], where an 80 $\times$ 80 pixel array is implemented for various operational modes: event generation, motion tracking and video output mode in full-resolution or compressed by the region of interest. The chip operation relies on a motion detection unit, shared by 4 active pixel sensors, which detects the pixels where an event is happening based on the previous one and current frame difference. The memory capacitors, designed with a size that makes the impact of parasitic capacitors negligible, are used for both storage and subtraction tasks. Their isolation from the rest of the circuit is carried out by an NMOS switch at the input, and the output is connected to the subtraction circuit.

Reference [41] is a recent work where a 96 $\times$ 96 array of 30 μm $\times$ 30 μm pixels was developed. The milestone of this circuit is to provide a wide bias voltage swing (both negative and positive) for each individual pixel with region of interest enhancement capabilities and a solid base for infrared multispectral acquisition. For that purpose it integrates two different types of memories. The first one is implemented through a capacitor of 55 fF with one plate connected to ground to hold the pixel reset voltage, which is different for each pixel, before being applied through a differential amplifier in voltage follower configuration to the photodiode. The second one is an in-pixel sample and hold circuit in open-loop configuration between the photodiode in accumulation mode and the output buffer implemented with a capacitor of 57 fF buffered by a source follower.

The work [42] is also a retina used for stereo vision based on the focal plane paradigm. This hybrid chip integrates on the same die a 320 $\times$ 240 pixel array with 3 $\times$ 320 analog memories in open-loop configuration with 50 fF capacitors and a stereo matching digital processor. As in the previous examples, the values just need to be stored the time required to compute the stereo matching algorithm, which is under a millisecond, and in this process they are not corrupted as the readings are performed through individual source followers for each capacitor.

Combining digital and analog resources can be also a good approach for object detection. By using the main advantages of each domain, in the work [13] an ultra low-power mixed signal IC for robust object detection is achieved. In this work a per-pixel analog memory is implemented with an MIM capacitor for short-term storage of a previous frame to be used in the frame differencing mechanism. In addition to the in-pixel memory, an off-chip SRAM is used to store a one-frame background model. The video processing is implemented in a column-parallel reconfigurable analog processor. These processors perform a row-wise frame

differencing using the current pixel value and the previous value stored in the per-pixel memory and, as depending on the result, it might or might not perform the background subtraction using the background model stored off-chip. This off-chip digital information is converted to the analog domain using a per-column DAC in order to be used within the analog circuitry. Finally, the information gathered by these processes is used to highly optimize the analog-to-digital conversion of the captured image in terms of power consumption.

Despite the good results of the previous examples, none of the computer vision algorithms implemented has the HO-PBAS singularities in terms of memory accesses, 160 on average before they are updated in a typical implementation, or the long time elapsed before a memory is rewritten, commonly between 5 to 10 seconds.

3.2 Core Circuitry

All the functions required by the HO-PBAS algorithm are implemented near to the sensor in the analog domain, with the circuits controlled by digital signals generated in the periphery of the array of pixels. The main part of the core that will be scaled up to form the array is the Processing Element (PE), with four pixels that share a Processing Unit (PU).

3.2.1 Processing Element Schematic

Circuitry that is not shared among pixels will be implemented in-pixel. In this thesis, in-pixel means a photodiode and its associated circuits which only operate on values from its corresponding photodiode. These circuits are: 3 transistors active pixel sensor (3T-APS) for image capturing, Correlated Double Sampling (CDS) for reset noise removal, the frame buffer to hold the captured image until is read, the Gaussian diffusion block for image pre-processing, local logic circuits and the Analog Random Access Memories (ARAM) for the background model storage.

Figure 3.1 shows the simplified schematic of the pixel with the interconnections between its blocks. Each block is controlled by common control signals coming from the periphery as well as by internal blocks. These signals, which will be explained in detail in Section 3.5, are responsible for operating all switched-capacitor circuits that implement different functions inside the PE, connecting each pixel with its corresponding PU or enabling/disabling active blocks for power consumption reduction, among others.

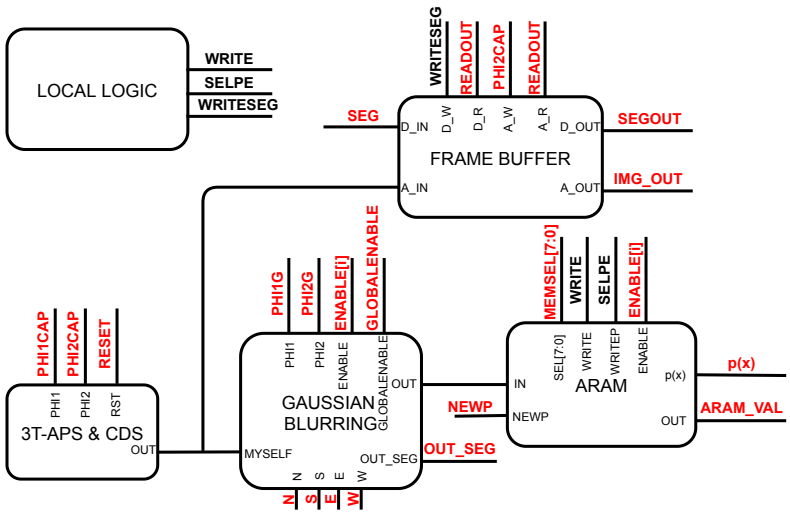

Figure 3.1: Pixel simplified schematic of the HO-PBAS algorithm on the CMOS vision sensors implemented in this PhD.

All the circuits that can be shared by different pixels multiplexing them in time were incorporated into the PU. Thus, when the image processing is being carried out one pixel connects at a time to this unit. This shared PU, shown in Figure 3.2, is formed by the blocks that calculate $d(x)$, $R(x)$ and the updated $p(x)$, as well as the block that takes the segmentation decision according to (2.2), i.e., the segmenter.

3.2.2 Analog primitives

Throughout the design many processing blocks replicate the same functions and circuits designs such as:

- voltage buffers
- voltage comparators
- high gain inverter amplifiers
- arithmetic units

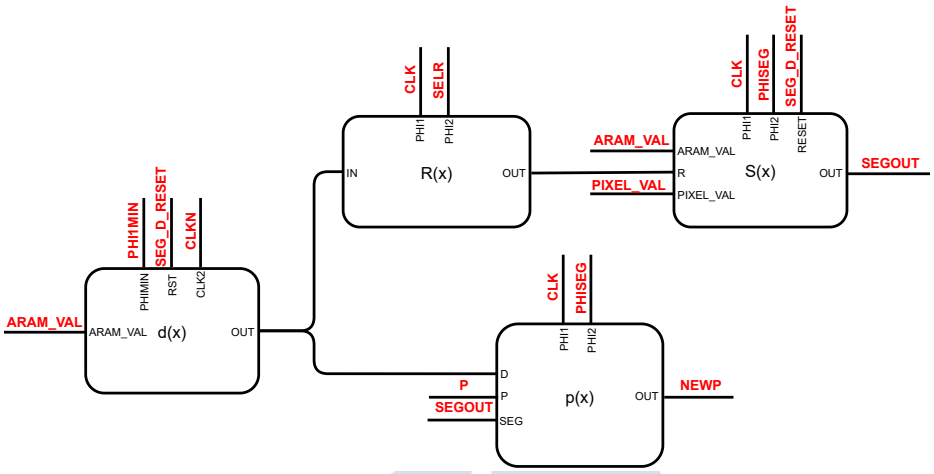

Figure 3.2: PU simplified schematic shared by four pixels.

	M1	M2	M3	M4	M5	M6	M7	M8
Voltage Buffer	0.5/6	0.5/6	1.5/3	1.5/3	0.5/2	2/0.5	-	-
Comparator	0.5/6	0.5/6	1/2	1/2	2/2	2/5	1.5/0.3	0.25/2
Inverter Amp	0.6/0.8	0.6/0.3	0.6/0.18	1/0.18	1/0.28	0.6/0.3	-	-

Table 3.1: Transistor dimensions of the analog primitives used in the HO-PBAS CMOS vision sensor designed in this thesis. All dimensions are expressed as W/L ratio in microns.

making up a set of analog primitives in our design. These circuits were designed with low area and power consumption specifications as they are meant to be in-pixel. Their transistors dimensions are shown in Table 3.2.2.

Voltage Buffer

The selected voltage buffer is an analog primitive designed as an operational amplifier with negative feedback loop, implemented by a differential pair with a current mirror load and an NMOS transistor for biasing, as shown in Figure 3.3 (a). This buffer, which works in the operation range of 0.3 V - 1.35 V, imposed by NMOS and PMOS transistors threshold voltages, features an open-loop gain of 50 dB, providing a maximum output error of 3.5 mV when comparing the input with the output, while working at frequencies up to 100 kHz with a capacitance load of 100 fF. These capabilities make it fast enough to drive the signals used

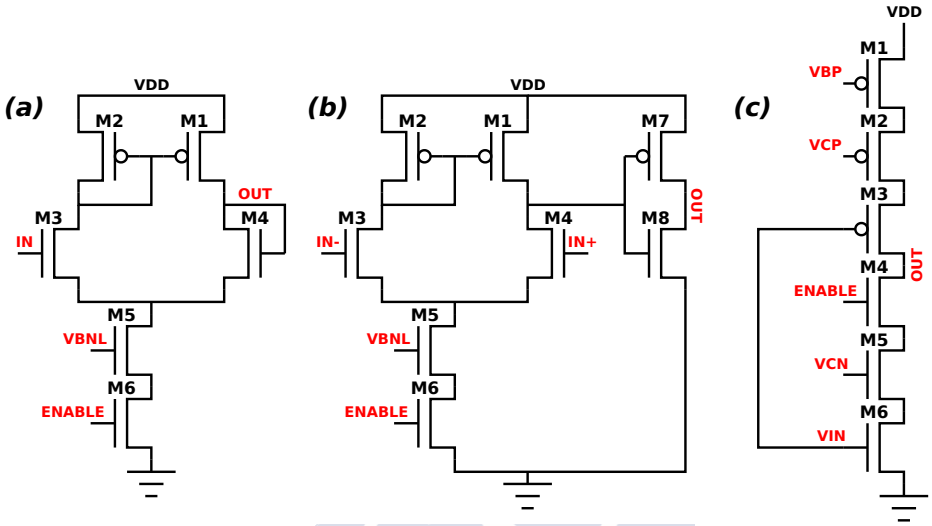

Figure 3.3: Voltage buffer (a), comparator (b) and high gain cascode amplifier (c) schematics. Red labels indicate input/output ports. Transistors M3-6 in (a) and (b) schematics are implemented with medium threshold voltage type.

in this design to the in-pixel circuitry.

Comparator

The comparator used is shown in Figure 3.3 (b). It is designed with two stages: the first one is implemented with the same architecture as the voltage buffer to provide high differential gain. The output stage is an inverter sized to feature a voltage threshold in a range that makes the comparator robust under process variation. This comparator features an average offset of 1 mV and 85 dB of gain for the 0.3 V to 1.4 V range. In terms of dynamic characteristics it offers an average speed of 300 V/μs for the same voltage range with a capacitive load of 100 fF.

High Gain Amplifier

Another analog primitive widely used in this thesis is the cascode inverter amplifier shown in Figure 3.3 (c). This single-input single-output amplifier features a high gain of 81 dB, which

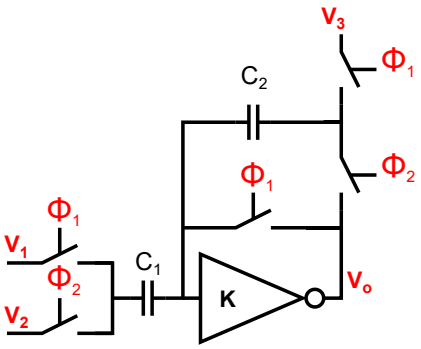

Figure 3.4: Arithmetic unit schematic used in the CDS as well as in different arithmetic operations required for the HO-PBAS algorithm operation.

produces small enough gain errors in the arithmetic unit explained below. Its main drawback is the need of three different bias voltages, which increases the layout complexity.

Arithmetic Unit

As shown in Chapter 2, HO-PBAS introduces several modifications to the PBAS algorithm to make it feasible for an analog hardware implementation whereas maintaining the algorithm performance. One of the most important simplifications was to linearize the equations in the datapath, so that they can be easily executed using switched-capacitor (SC) circuits. The circuit shown in Figure 3.4, which we call here arithmetic unit, will be replicated in all the processing blocks described below. This circuit uses the cascode inverter and capacitors C_1 and C_2 to implement a differential amplifier with adjustable offset and gain, controlled by the two non-overlapped clock signals ϕ_1 and ϕ_2. Assuming that the gain K is infinite, the analysis of the arithmetic unit is as follows. In the first phase, ϕ_1 is in the high state and the charge stored in the capacitors C_1 and C_2 is respectively:

$$\begin{aligned} Q_{1_{\phi_1}} &= C_1(V_1 - V_q) \\ Q_{2_{\phi_1}} &= C_2(V_3 - V_q) \end{aligned} \tag{3.1}$$

where V_q is the quiescent voltage of the inverter amplifier with feedback loop. In the second phase of the process, ϕ_1 is now low and ϕ_2 is high, connecting V_2 to C_1 and introducing C_2

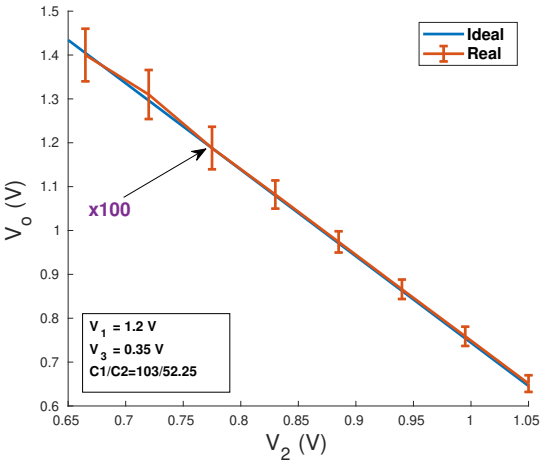

Figure 3.5: Arithmetic unit simulation. Vertical bars indicate variation under Monte Carlo simulations, scaled up 100 times.

into the feedback loop. Then, the charges of the capacitors must be:

$$\begin{aligned} Q_{1_{\phi_2}} &= C_1(V_2 - V_q) \\ Q_{2_{\phi_2}} &= C_2(V_o - V_q) \end{aligned} \tag{3.2}$$

Because of the charge conservation principle, the total charge of the circuit should be the same in both states, meaning that $Q_{1_{\phi_1}} + Q_{2_{\phi_1}} = Q_{1_{\phi_2}} + Q_{2_{\phi_2}}$. Thus, combining (3.1) with (3.2), and assuming an infinite gain of the amplifier, which makes V_q equal at both phases, the output voltage V_o can be obtained as:

$$V_o = V_3 + \frac{C_1}{C_2}(V_1 - V_2) \tag{3.3}$$

This simple circuit is a differential amplifier with a gain controlled by the relationship between capacitors C_1 and C_2 and an offset voltage V_3. Figure 3.5 shows electrical simulations run on the arithmetic unit, where the great robustness under process variation and mismatch can be appreciated. This is due to the use of a closed loop and the fact that the gain control relies on a ratio of two capacitors.

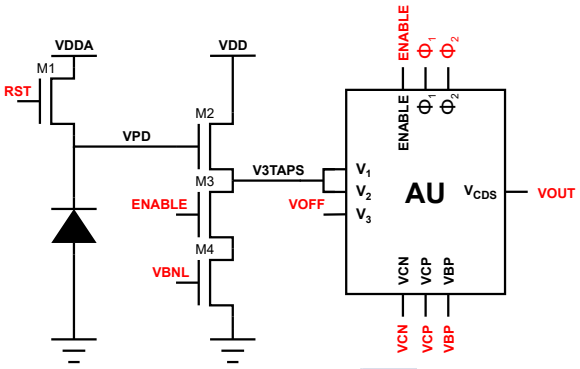

Figure 3.6: 3T-APS circuit with CDS for image acquisition.

3.2.3 Image Acquisition and Preprocessing

This subsection addresses the image acquisition and preprocessing blocks of our design. As mentioned above, part of these functions are implemented with the analog primitives described in Section 3.2.2.

3T-APS and CDS

For the image acquisition stage an NWELL photodiode in integration mode was used with a 3T-APS as the conditioning circuit. A global shutter strategy is carried out by a sample and hold circuit placed after the CDS. The photodiode is implemented with an 8×8 μm^2 NWELL over p-substrate. This decision was taken based on different aspects such as previous experience of the authors with photodiodes in standard CMOS technologies, similar work [35, 43], and studies about the performance of different photodiodes [44].

Figure 3.6 shows the schematic of our image acquisition block implemented with a 3T-APS and CDS. Transistor M1 is sized with minimal dimensions as it is used for resetting the photodiode voltage to VDDA, which is set to 1.4 V. This voltage ensures M1 to work in the saturation region and reduces the reset value variation due to threshold voltage mismatch. Transistors M2, M3 and M4 form a source follower with enable switch to buffer the pixel voltage of the photodiode. M3 has minimal dimensions as it is only used as a power switch. On the other hand, M2 and M4 are sized with W/L relationship of 1.2/0.7 and 1.2/0.3 in μm with the objective of making the gain as close as possible to unity for the voltage range 0.3

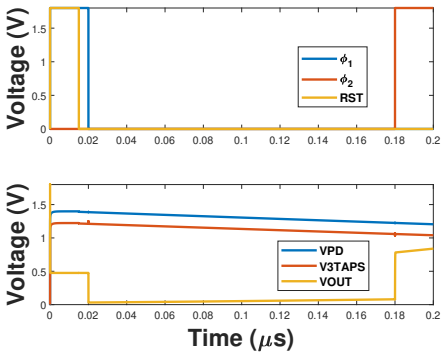

Figure 3.7: 3T-APS acquisition simulation with control signals (top) and nodes voltages (bottom).

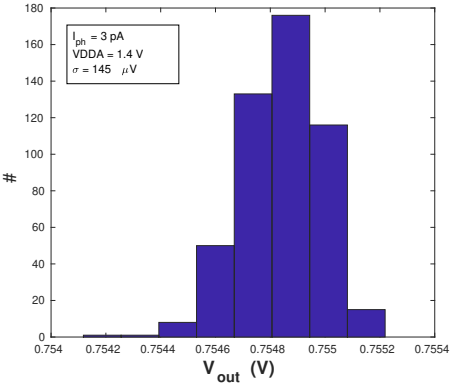

Figure 3.8: Monte Carlo simulation for the output of the CDS.

- 1.4 V. Also, to increase the input range, M2 is implemented with a low voltage NMOS transistor.

In order to improve the image quality an in-pixel CDS is included. This circuit, using the arithmetic unit previously described (see Figure 3.4 and their companion explanation), samples VPD when is reset and outputs the difference with the value at a certain time, as shown in Figure 3.7. In addition, it offers the possibility of adding a given gain and offset to the output as pointed out in (3.3), which is useful to set the output range according to the input range of subsequent circuits. Experimental tests with a chip that implements a photodiode equal to the one used in this chip show a voltage discharge on the photodiode of around 150 mV while sensing a grey area with standard indoors illumination. Thus, in this system we set an offset voltage of 300 mV and a voltage gain of 3, using a relationship between the capacitors C_1/C_2 of 150 fF/50 fF, as with these parameters, voltages between 0.3 V and 1.4 V are expected for usual scenes, being this voltage range adequate for the circuitry implemented in this work. With this setup, a Monte Carlo simulation of 500 points was executed (Figure 3.8) showing, an average variability of just 145 μV for V_{out} with a photocurrent I_{ph}=3 pA discharging the diode modeled by the spectre model provided by the foundry, which corresponds to standard indoors illumination. This variability barely changes across different lighting conditions.

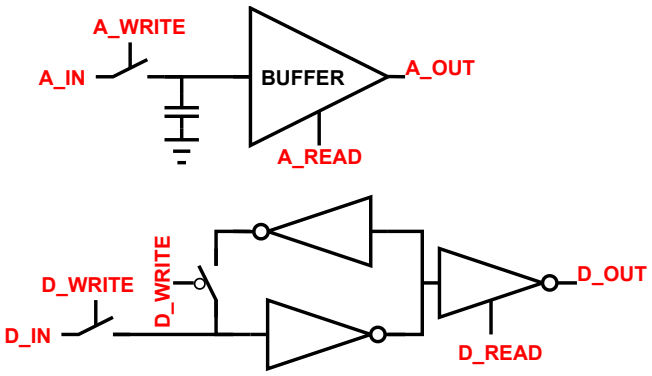

Figure 3.9: Frame buffer schematic to hold HO-PBAS pixels information before being read out.

Frame Buffer

The readout of the information provided by the CMOS vision sensor chips designed in this PhD is performed through an analog to digital conversion step, so that every time a frame is captured it must be hold until the analog to digital conversion is executed. All the pixels perform the integration at the same time, implementing a global shutter approach, and after that, the output value of the CDS is written into a sample and hold circuit with an MIM capacitor of 50 fF. The output buffer enable signal will be controlled by the row selector signal, connecting the value to the column bus when required.

Also, as shown in the schematic of Figure 3.9, a digital register with a tri-state output buffer is included to hold the segmentation result until it should be connected to the column bus for the readout process.

Gaussian Blurring

After the CDS, a low pass filter is applied to the captured image to reduce noise, improving the algorithm performance [45]. Different filters might be applied, however, Gaussian filters are very appropriate to remove the high frequency components of the image, that might be caused by fixed pattern noise, temporal noise of the imager or from digital artifacts after image compression [27, 28]. The kernel coefficients for the Gaussian filters are obtained from

the Gaussian distribution:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.4}$$

where σ is the width of the Gaussian distribution and x and y the spatial coordinates of the image. From (3.4) the filtered image $I_{filt}(x,y)$ can be obtained by the spatial convolution of the captured image $I(x,y)$ with the Gaussian kernel:

$$I_{filt}(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{3.5}$$

This convolution can be implemented in hardware through an RC network joining pixels in a 4-neighborhood connectivity scheme. Each pixel will hold the captured value in a capacitor C connected to each neighbor through a resistor R. The voltage stored at each capacitor will evolve as a function of the charge propagation in the RC network, and the time evolution of this magnitude can be modeled by:

$$C\frac{dV_{i,j}}{dt} = \frac{V_{i+1,j} + V_{i,j+1} + V_{i-1,j} + V_{i,j-1} - 4V_{i,j}}{R} \tag{3.6}$$

which is actually the continuous-time heat differential equation with diffusion coefficient $D = 1/RC$, where t is the time and $V_{i,j}$ the corresponding voltage to illumination intensity $I(x,y)$ [46]. In the case of a Gaussian blurring, D determines the degree of blurring through the expression $\sigma = \sqrt{2Dt}$, which leads to:

$$\sigma_{RC} = \sqrt{\frac{2t}{RC}} \tag{3.7}$$

One possible solution using this technique was explored in [47], where the Gaussian diffusion was performed connecting the voltage held in the stage capacitors through MOS transistors operating in the ohmic region. The non-linearity of active resistive links and the time uncertainty of the sampling mechanism degrade the accuracy of the diffusion process in these type of networks. Thus, a different approach was implemented in this work based on the solution of reference [32], where the resistive connection between capacitors was made using a switched-capacitor circuit as shown in Figure 3.10, where ϕ_1 and ϕ_2 are two non-overlapped clock control signals. With this circuit (3.7) becomes:

$$\sigma_{SC} = \sqrt{\frac{2n_{gauss}C_E}{C}} \tag{3.8}$$

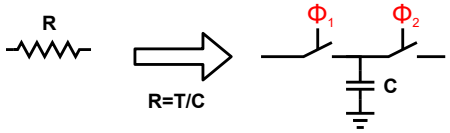

Figure 3.10: Switched-capacitor circuit example. Equivalent resistance can be obtained as a function of the frequency of clock signals ϕ_1 and ϕ_2 as $R = \frac{1}{fC}$.

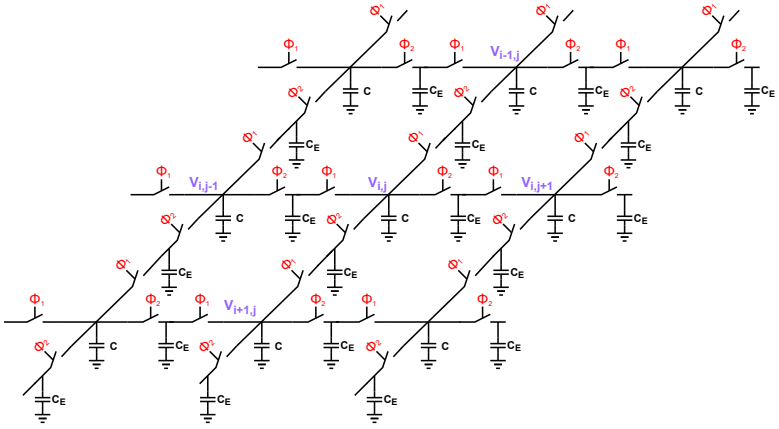

Figure 3.11: Single-Euler configuration of the SC network that performs the low-pass filter of the captured image.

where n_{gauss} is the number of complete cycles of the clock signals ϕ_1 and ϕ_2 that the operation is repeated. This approach is implemented in two dimensions with the circuit of Figure 3.11 to run it on 2D images. Both the stage capacitor C and the exchange capacitor C_E are MIM devices to ensure the linearity of the circuit, and with sizes 133 fF, and 35 fF, respectively. These values for the capacitors allow to select σ ranging from 0 to 6 (validated through image simulations to be enough by a great margin) with a number of clock cycles n below 63. Each pixel will write the captured voltage V_{ij} into the central capacitor C and the Gaussian diffusion will be performed by charge sharing through the exchange capacitors C_E.

To implement such a network, each pixel incorporates the cell described in Figure 3.12. This schematic shows all the elements present in the cell, with their input/output connections. The input port is controlled by signal *write*, which will be connected to ϕ_2 signal of the CDS

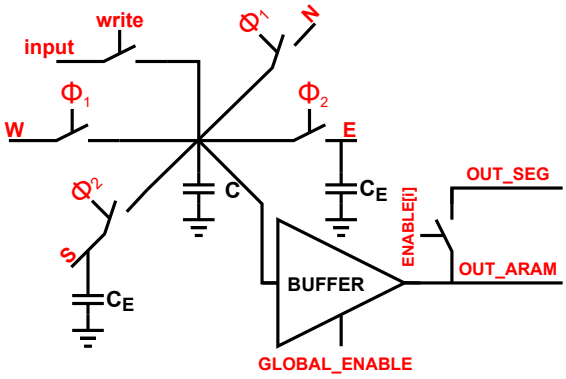

Figure 3.12: Configuration of the SC network that performs the low-pass filter of the captured image.

circuit. Thus, every time a new pixel value is captured, it will be written to the Gaussian blurring cell. Each Gaussian diffusion block incorporates two exchange capacitors at South and East nodes, thus when the cells are connected as in Figure 3.11 the required exchange capacitors at North and West nodes will be at the neighbors' cells.

The output voltage of the Gaussian blurring is buffered by the differential amplifier with negative feedback loop described in Section 3.2.2, which is controlled by the power gating signal *GLOBAL_ENABLE*. The cell provides the output through two different ports: one of them permanently connected to the ARAM input, and the other one through a switch, controlled by the pixel enable signals (explained in Section 3.5), that will be connected to the processing circuitry when required.

To test this architecture a simulation of an array of 9×9 Gaussian diffusion cells was executed with all the pixels' capacitors voltages except the pixel under study initialized at 350 mV, and the capacitor voltage under study at 900 mV. It can be seen in Figure 3.13 how the stored value evolves through the clock cycles and stabilises when all the capacitors are approximately at the same value.

3.2.4 Analog Random Access Memory

This subsection addresses the memories implemented for storing the background model of HO-PBAS. The features of HO-PBAS running on the focal plane are key to determine both

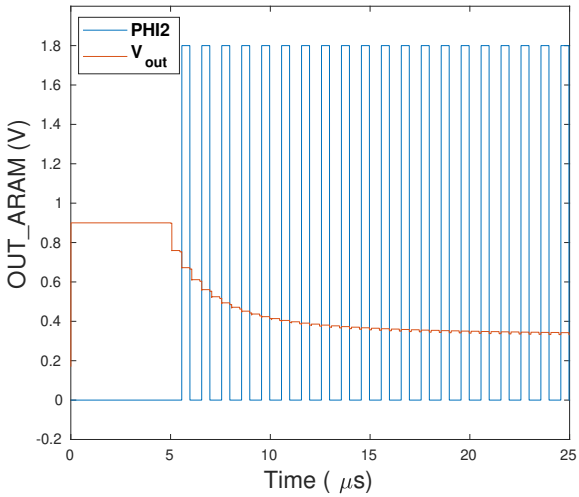

Figure 3.13: Electrical simulations of the SC network for the Gaussian diffusion of the HO-PBAS CMOS vision sensor chip.

the memory topology and the main design parameters to optimize. As an example, in the case of a pixel with N=8 analog memories for running HO-PBAS with a probability of updating a pixel value of $p(x) = 0.05$, every analog memory would be read on average $N/p(x) = 160$ times before being updated with a new pixel value. Regarding the time that the samples must be stored before an update, this strongly depends on the processing framerate and the $p(x)$ of each specific pixel. To have an idea of the order of this magnitude, if a framerate of 25 fps and a $p(x) = 0.1$ are considered, this time would be 4 seconds on average. This poses stringent design demands in terms of low read and long-term storage degradation errors.

Although there are CMOS vision sensors with current mode processing and/or current memories [48, 49, 35], our CMOS vision sensor chips for HO-PBAS have switched-capacitor voltage memories.

As seen in Figure 3.14, there are three main analog voltage memory topologies based on switched-capacitor circuits, namely open-, closed-loop and integrator architectures [50]. All of them share a scheme of an input buffer followed by a storage capacitor and an output buffer. In our design, the input voltage to the memories that store the background model of HO-PBAS is in the range of 0.3-1.4 V, which is within the output voltage limits provided by

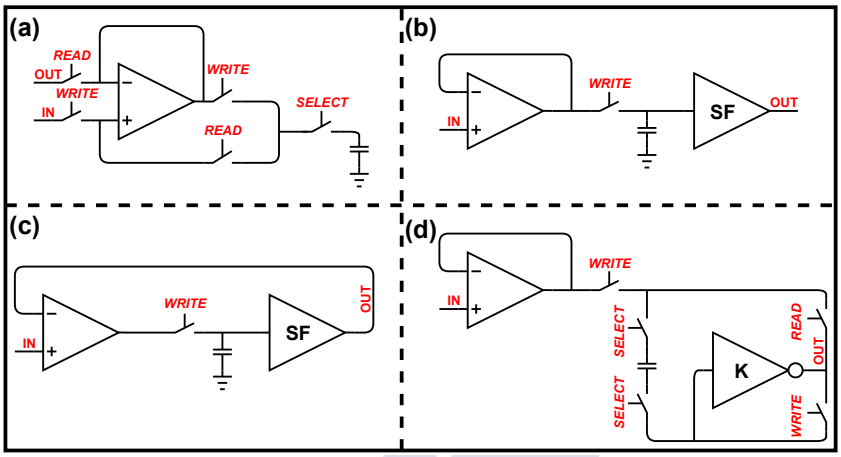

Figure 3.14: Voltage-mode analog memories architectures: (a) Open-loop (b) Open-loop with individual output buffer (c) Closed-loop (d) Integrator.

the CDS output of every pixel.

An in-depth study of the effects of current memory implementations on HO-PBAS performance was developed in [45]. In that work, the four type of memories shown in Figure 3.14 were compared, studying their electrical performance: write and read errors, long-term degradation due to the off resistances of the switches and value corruption after many readings.

The open-loop architecture shares the output buffer for all the memories. Thus, every time a capacitor is connected through its switch it will share its charge with the input capacitance of the buffer, degrading the stored value, as shown in Figure 3.15. One possibility to overcome this problem is to provide each capacitor with its own voltage buffer, with the main drawback of increasing area and power consumption. This solution is the one shown in Figure 3.14 (b), where the output buffers are implemented with source followers in order to reduce area. Besides, this approach solves the charge sharing issue but adds new errors stemmed from the actual implementation of the source followers.

The next circuit that was studied is the closed-loop analog memory, shown in Figure 3.14 (c). This circuit adds the output buffer, also implemented with source followers, in the feedback loop of the input buffer. With that, the voltage stored in the capacitor is the one that produces an output of the buffer equal to the input value. However, this solution comes

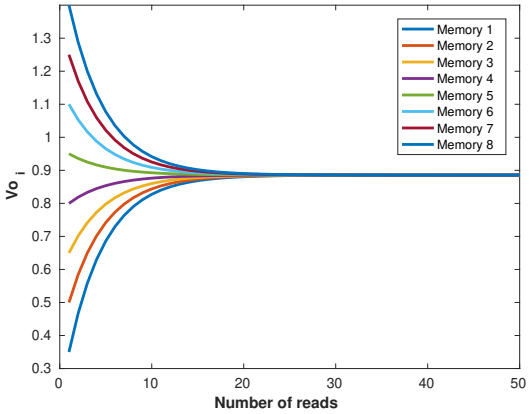

Figure 3.15: Signal integrity electrical simulation for the memory with open-loop architecture as number of reads, showing the effect of charge sharing with eight capacitances connected to only one amplifier.

with two drawbacks. The first one is that it needs a dedicated amplifier for the input buffer or to increase the design complexity if it needs to be shared (the input buffer of the other approaches will be the output buffer of the previous stage). The other issue is that for values larger than 1.3 V the voltage that needs to be held in the capacitor is out of the operating range of the switches, which might lead to considerable leakage currents or undesired states with the switches in consideration here.

Finally, the integrator architecture shown in Figure 3.14 (d) works by placing the memory capacitor in the feedback loop of the amplifier instead of having one plate connected to ground. Thus, the value is stored as the difference between the input value and V_q of the amplifier. If the capacitors are isolated with two switches instead of only one the accuracy of the system increases as pointed out in [51]. The disadvantages of this solution are mainly that it suffers from charge sharing each time that a certain memory is selected (see Figure 3.16), and that it needs two different hold voltages, increasing layout routing complexity.

In addition, the degradation of single write/read errors was studied. These errors might come from different sources. The most important ones for the write operation are the charge injection or clock feedthrough. Also, the finite gain of the amplifier that implements the input/output voltage buffers affects both the writing and reading operation. The results of this study are shown in Figure 3.17. As seen, the closed-loop architecture has an acceptable error

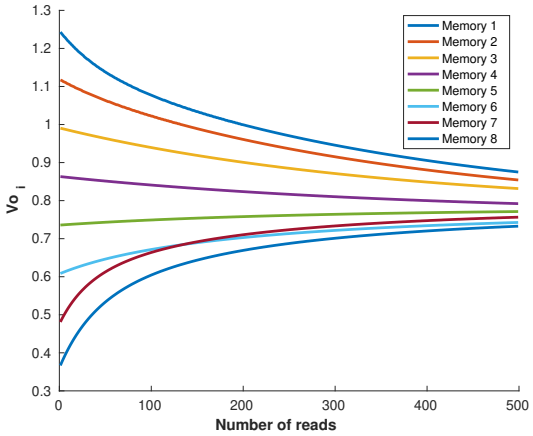

Figure 3.16: Electrical simulation for the memory topology with the integrator architecture that shows how the signal stored at each capacitor worsens with the number of consecutive readings.

for all the input range. Similarly, the integrator topology features a small write-read error but, as it will be shown, it is seriously affected in the case of multiple memory reads. As expected, the open-loop approach has a poor performance due to the charge sharing between the storing capacitors and the parasitic input capacitance of the output buffer, which discourages the use of this topology. Finally, the open-loop with source followers as output buffers shows a large error. However, as the results presented below show, it is possible to cancel out this effect by inserting a source follower between the 3T-APS pixel and the circuit that would perform HO-PBAS segmentation. This new source follower shifts the voltage of the incoming pixel to a voltage level similar to that of a past value of the pixel under study stored in the analog memory.

Taking into account this study it is clear that all of the possibilities feature some benefits and suffer from any or other drawback. Nevertheless, as they are intended for the specific application of running HO-PBAS, the decision of which one should be used on the chip relies on which one will provide the best algorithm performance. To check this point the results extracted from the electrical simulations were included into the algorithm C++ source code of HO-PBAS and assessed against the dataset *changedetection* [25]. In this simulation, two different important issues were included in the case of the open-loop topology with individual

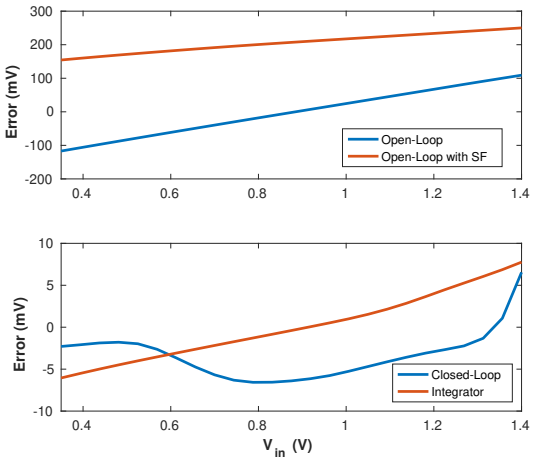

Figure 3.17: Global error at write-read operation of all the architectures of memories outlined in Figure 3.14.

source follower for each capacitor memory. As these circuits will suffer from mismatch and process variation, statistical information obtained from Monte Carlo electrical simulations was included into the software simulations implemented in C++ too. Also, to mitigate the effect of the transfer function of these voltage buffers, an additional source follower will be added in the path from the Gaussian blurring circuit to the stage responsible for the segmentation decision (see Figure 3.18). By doing that, both signals are affected by a similar transfer function.

Results for this test are shown in Table 3.2. It can be seen that the open-loop with individual source follower for each memory is the one with the best performance, and hence, it will be the one to be included on the CMOS vision sensor chip for HO-PBAS designed in this thesis. The output buffers will have an enable transistor with a twofold purpose: to cut down the power consumption when they are not used, and to connect or disconnect from the output bus, as shown in the schematic of Figure 3.19. Also, logic circuitry is added to differentiate when a memory is selected to be read or to be written. All the memory capacitors are implemented with 150 fF MIM devices. The rationale behind MIM capacitors instead of MOS capacitors is that MOS capacitors are expected to suffer from parasitic light sensitivity, which comes as leakage currents from the incident light as the stored charge in the transistor channel

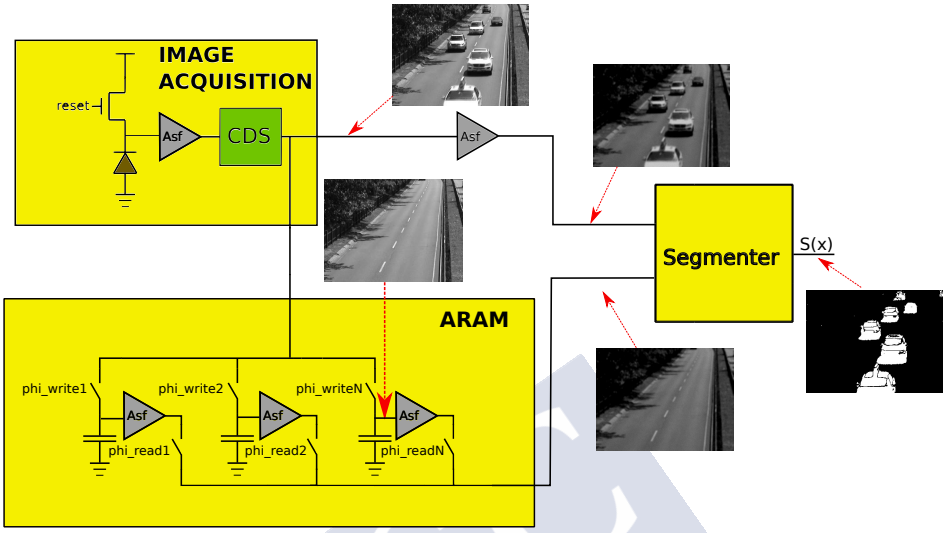

Figure 3.18: Solution for the open-loop memory topology with source follower architecture. Accounting for the change in brightness and contrast made by the memories output buffers, an additional SF was added in the path from the CDS output to the segmenter input.

Table 3.2: Ratio of F-Measure with hardware non-idealities to ideal F-Measure (higher is better).

	Open-loop	**Open-loop with SF**	**Closed-loop**	**Integrator**
shadow	26.18	79.61	62.03	22.76
badWeather	72.50	83.81	71.77	76.88
PTZ	9.26	77.16	82.98	29.70
dynamicBackground	7.92	79.87	51.35	9.92
cameraJitter	4.52	88.35	16.35	6.58
thermal	14.82	82.43	66.88	20.08
interObMotion	13.44	88.87	75.03	16.91
turbulence	13.44	80.80	72.59	20.03
baseline	36.77	72.82	76.78	44.52
lowFramerate	7.66	86.33	32.46	11.23
nightVideos	13.44	74.79	105.7	20.27
Overall	19.61	81.35	64.90	25.35

will generate a photocurrent due to the pn junction of the parasitic diode created. Thus, on this occasion, it is preferable to exchange area for accuracy, searching for the best quality of the foreground/background segmentation of the scene. In addition, the parasitic diodes of the

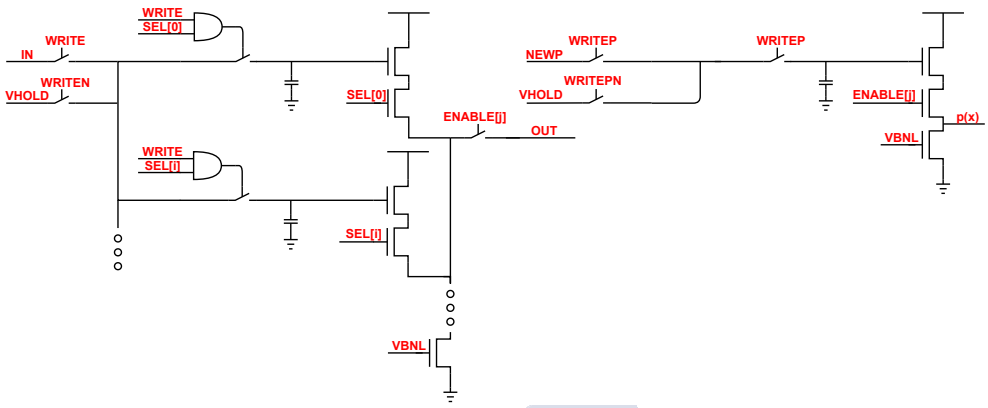

Figure 3.19: Schematic of the ARAM of the HO-PBAS CMOS vision sensor chips of this PhD. The left part holds the background model samples, whereas the right part is the responsible of storing $p(x)$.

MOS switches will produce a large impact on the stored value degradation, as the pn junctions at the drain/source will also produce parasitic photocurrents. As PMOS transistors will need an NWELL, they are expected to produce more photocurrent than NMOS transistors [44]. An ideal option would be to use transmission gates with matched sizes to eliminate the photocurrent contribution. However, due to the complexity of the photocurrent generation estimation and the lack of information about doping profiles of standard CMOS technologies, minimum width NMOS transistors were selected for the switches. The size of the parasitic diodes does not increase with channel length, which permits to change channel lengths to decrease current leakages in switches caused by its off-state resistance.

3.2.5 Algorithm Parameters Update

Background dynamics estimator

HO-PBAS incorporates a feedback scheme to update some key parameters to adjust them according to the specific scene that is being captured. These parameters are the radius of the segmentation sphere $R(x)$ and the update probability $p(x)$, both pixel-dependant. As explained in Chapter 2, they both depend on the background dynamic estimator $d(x)$ which needs to be calculated for each pixel using (2.7). This value is extracted from the background model using the circuit in Figure 3.20, which works as follows.

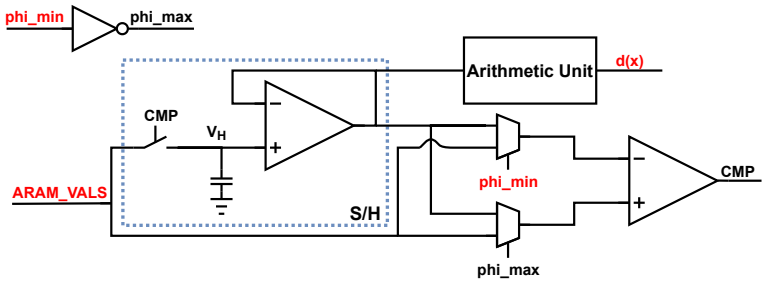

Figure 3.20: Background dynamic estimator. The background model values are introduced into the ARAM_VALS node one by one and depending on which one of signals *phi_max* or *phi_min* is set to high, the maximum or the minimum is stored in the S/H circuit at the end of the cycle.

First, signal *phi_min* is set to low connecting the comparator inputs in a configuration where the maximum value is searched. All the values stored in the ARAM are provided one by one, multiplexed in time, to the ARAM_VALS node and they are compared with the current value in the sample and hold (S/H) circuit. For each value above V_H the comparator output *CMP* triggers the S/H input switch and updates the stored value. Once all the ARAM values have been compared against each other the S/H output will be the $max_k(B_k(x))$, which is sampled by the arithmetic unit. Then, *phi_min* is set to high and the process is repeated. As the comparator inputs were swapped, the S/H will output the minimum value at the end of the cycle. This value is the input of the arithmetic unit, configured with a capacitor relationship of C_1/C_2 = 150/75 fF, resulting in an output value equal to $d(x)$.

To check the validity of the design, this circuit was simulated for different input values. Figure 3.21 shows one of them, with seven samples of the background model comprised between 700 mV and 750 mV and the remaining sample, V_{var}, used as the independent variable to set the x-axis. It can be seen how the circuit detects the minimum and the maximum values for each scenario and how it calculates $d(x)$ according to 2.7.

Radius of the segmentation sphere

The background dynamics estimator can now be used in a circuit that obtains the segmentation sphere radius $R(x)$ through (2.8). For that purpose a sample and hold circuit is added to maintain the value of $d(x)$ while it is being used. The output of that circuit is connected to the block shown in Figure 3.22, which will obtain the value of $R(x)$ that will be used in

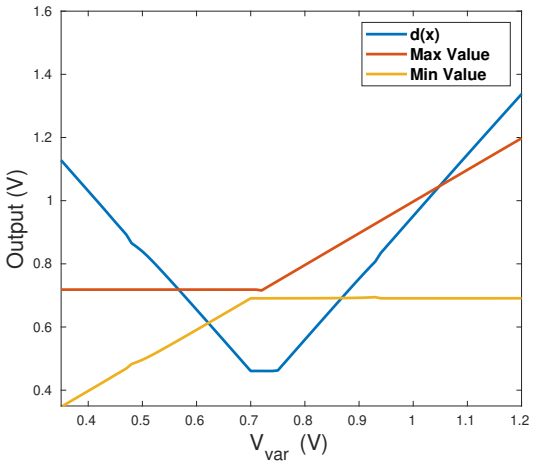

Figure 3.21: Electrical simulation for the circuit that calculates $d(x)$, which grows with the difference of the maximum minus the minimum value.

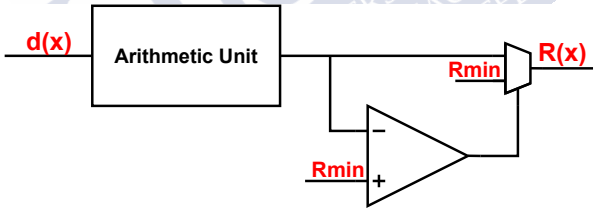

Figure 3.22: Segmentation sphere's radius is obtained from $d(x)$ and compared with the minimum value R_{min}.

the iteration under study. The arithmetic unit is implemented with capacitor values C_1/C_2 in femtofarads of 50/200. This block needs an additional functionality to check if the new value is below a certain minimum. In that case the comparator connected to an analog multiplexer will set the output to R_{min}. The circuit functioning can be seen in Figure 3.23, where the output value is represented as a function of the input $d(x)$.

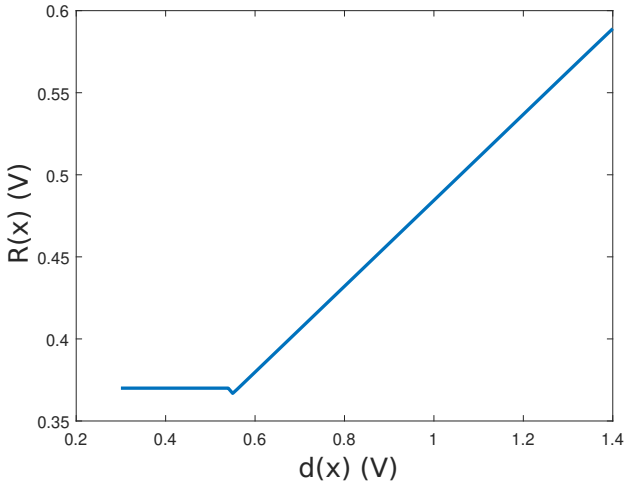

Figure 3.23: Parametric simulation for the circuit that calculates $R(x)$ of the HO-PBAS CMOS vision sensors of this PhD.

Update probability

The last circuit that updates an algorithm parameter is the one responsible for the update probability $p(x)$. This parameter is adjusted as a function not only of $d(x)$ but also of the segmentation result. Therefore, its operation is activated once the segmenter circuit has finished. Differently from $R(x)$, the update probability also depends on its previous value and it must be stored in a dedicated memory of the ARAM. Thus, the circuit shown in Figure 3.24 implements (2.6) using an arithmetic unit, and it connects the execution result into an analog multiplexer that, depending on the segmentation result, it will select which part of (2.6) is used. Then, the output of the first multiplexer is compared with p_{max} to assure that the final result is not bigger. Finally, this new value is written to the memory to be used when required.

Figure 3.25 shows a simulation for the block that calculates new update probability based on its previous value and on the segmentation result. It can be seen how the new value increases according to $d(x)$ when the segmentation result is background ($S(x) = 0$) and how the result is always p_{min} when $S(x) = 1$.

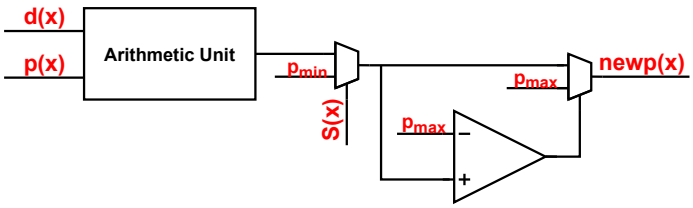

Figure 3.24: Schematic of the circuit responsible for updating $p(x)$ of the HO-PBAS CMOS vision sensors of this PhD.

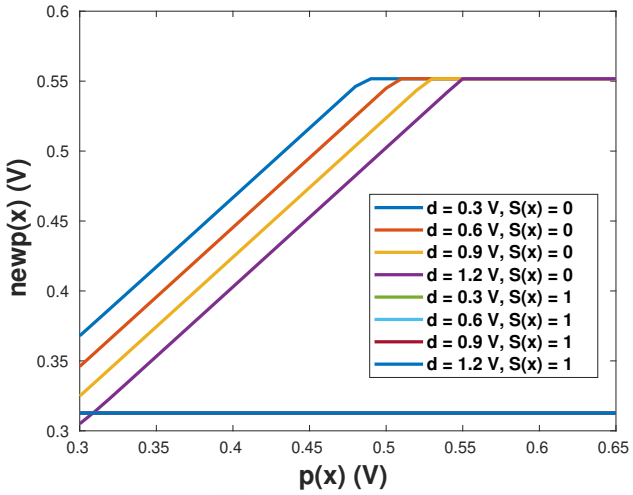

Figure 3.25: Electrical simulation for the circuit that calculates the update probability $p(x)$ of the background model for the new frame based on its previous value.

3.2.6 Foreground Detection

The last circuit required for the image processing is the one responsible for the segmentation decision. Based on the previous image samples stored in the ARAM and the incoming pixel value, this circuit must calculate the one-to-one absolute difference between the incoming pixel value and the ARAM values, and count how many are inside the segmentation sphere. Figure 3.26 shows the schematic of the segmenter, where the different stages can be seen. The first one is the absolute difference circuit. It is implemented with the arithmetic unit described in Section 3.2.2 configured with unity gain and with a multiplexer for each input controlled by

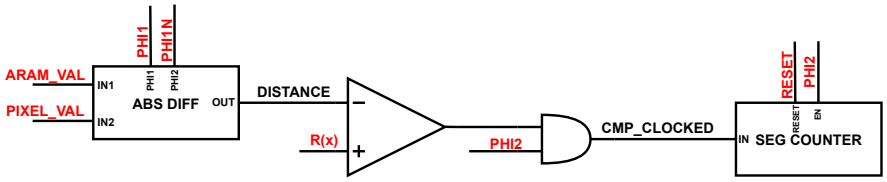

Figure 3.26: Segmenter schematic.

a comparator in such a way that the highest input is always connected to V_1; see (3.3). With this setup, the result of the difference will always be a positive value plus the offset voltage.

Once the absolute difference of the background model sample, connected to the *ARAM_VAL* node of Figure 3.26, and the input image is obtained, the result is compared with the radius of the sphere $R(x)$. Thus, if the difference is smaller than the radius $R(x)$ it will mean that the sample is inside the sphere. To avoid the effect of transient voltages an AND gate is added to the output of the comparator. Thereby, the counter will only receive pulses when *PHI2* is high, preventing the effect of glitches to cause false positives.

Different options exist for the counter implementation. In this circuit only the samples inside the sphere are being counted. Besides, the algorithm only needs two of them to consider the pixel background, as explained in Chapter 2. Thus, a circuit that triggers its output after two input pulses was designed instead of using a standard CMOS counter based on D-type flip flops, as they would substantially increase the circuit area. Figure 3.27 shows its schematic. After the reset, nodes V_1 and V_{12} are connected to ground and V_2 is set to VDD, and every time that *PHI2* is high V_{12} goes to high if the circuit is receiving a pulse. The same process happens with V_2 if V_{12} was previously set to high, activating the circuit output if at least two pulses were received after the reset.

Figure 3.28 shows a segmenter simulation for two different pixel values. For each one of them the background model is the same: four samples of 0.5 V and four samples of 0.7 V. As it can be seen in the graph, the simulation begins by resetting all the circuits during two clock cycles. Then, the system connects each sample with the input of the arithmetic unit that calculates the difference between the pixel value and the background model. The result is then compared with the radius of the sphere, $R(x)$, and for every distance smaller than the radius the AND gate shown in Figure 3.26 produces a positive pulse when *PHI2* is high. That is the case in the first part of the simulation, from 0 to 15 μs, when an input value of 550 mV

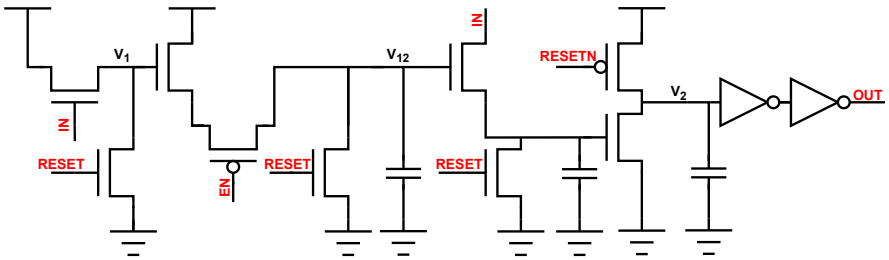

Figure 3.27: 2 bit pseudo-counter schematic used in the segmenter block. All transistors are of minimal sizes and capacitors are implemented with PMOS transistors of 1.5μm$\times 1.5\mu$m dimensions.

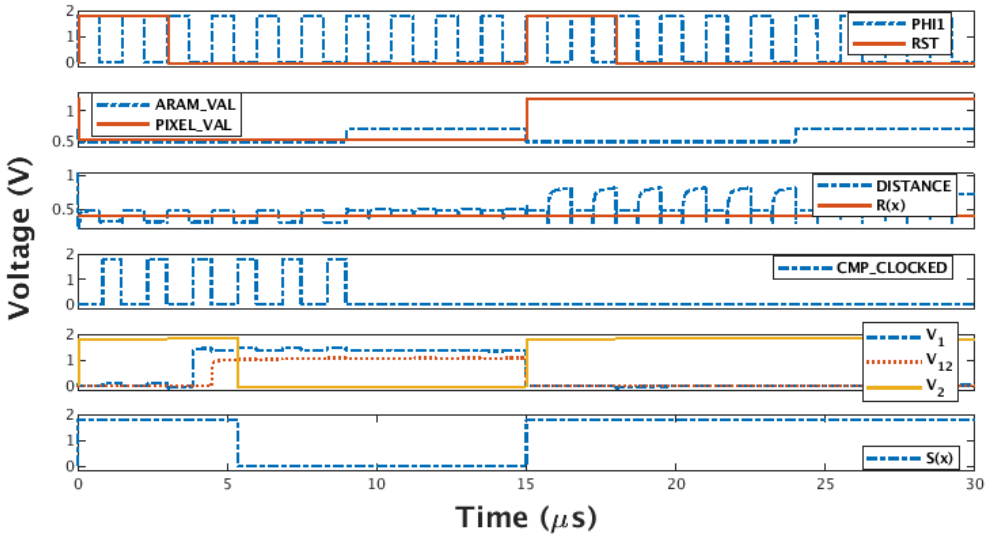

Figure 3.28: Segmenter simulation with two different input values (550 mV and 1.2 V) against the same background model ($B_k(x) = \{0.5, 0.5, .5, 0.5, 0.7, 0.7, 0.7, 0.7\}$ V). $S(x)$ must be read at the end of each cycle.

is compared against the background sample values of 500 mV. When this same input value is compared with the remainder samples of the background model no pulses are generated, as their correspondent distances are bigger than $R(x)$. The pulses generated after the system reset produce the desired impact into the counter. As shown in the fourth subplot, the first pulse results in a rise in V_1, which forces V_{12} also to go high. This allows V_2 to go low at the next pulse, making the output of the circuit equal to 0 V, the corresponding voltage of background.

The following pulses in this first part of the simulation do not produce any change in the circuit outcome, as expected. At 15 μs the system is reset, the input value changed to 1.2 V and the segmentation process repeated. As the input value is different from all of the samples of the background model, none of them are inside the segmentation sphere. Thus, no pulses are generated and at the end of the cycle the segmentation value remains at VDD, meaning that this pixel will be segmented as foreground.

3.2.7 Local Logic

In such a complex design it is convenient to handle different control signals inside the pixel array for different purposes. The main reason is to reduce the number of signals that come from the periphery, decreasing the layout complexity. Also, because some control signals are generated as a function of different pixel-dependant parameters, such as the update probability $p(x)$, which are stored or calculated in-pixel. Therefore, this block comprises different circuits to carry out this task.

The first part of the local logic block is the one that generates control signals based on global control signals to reduce the number of connections from the periphery and hence, the layout complexity. The schematic of this last part can be seen in Figure 3.29 (a), and its main purpose is to generate two local signals. The first one is *SELP_LOCAL*, that connects the update probability $p(x)$ of the selected pixel through the *ENABLE* signal to the PU that is going to update it. The other one is $WRITESEG$ which writes the segmentation result into a local register to be read when required.

As explained before, one important feature of HO-PBAS is that different algorithm decisions are taken in-pixel. These decisions, corresponding to perform a background model update and to execute the diffusion mechanism to a neighbor, are taken if two conditions are met: the analog random value *RNDIN* is below $p(x)$ and the pixel is segmented as background. The circuit that computes this logic is shown in Figure 3.29 (b). It can be seen that a 3 input NOR gate was used with its inputs inverted (obtained by swapping the comparator inputs, with the segmentation result as it is and by using CLK instead of CLKN) to implement an AND gate with the result available at the second part of the clock cycle. The other circuit in this part of the local logic is used for the $WRITE$ generation, connected to the ARAM write input, and that is set to high if either $GLOBAL\_WRITE$ or $LOCAL\_WRITE$ are high.

The *DECISION* signal is used in the background model update unit of Figure 3.29 (c) and whose schematic is shown in Figure 3.30. This block is responsible of three tasks. The

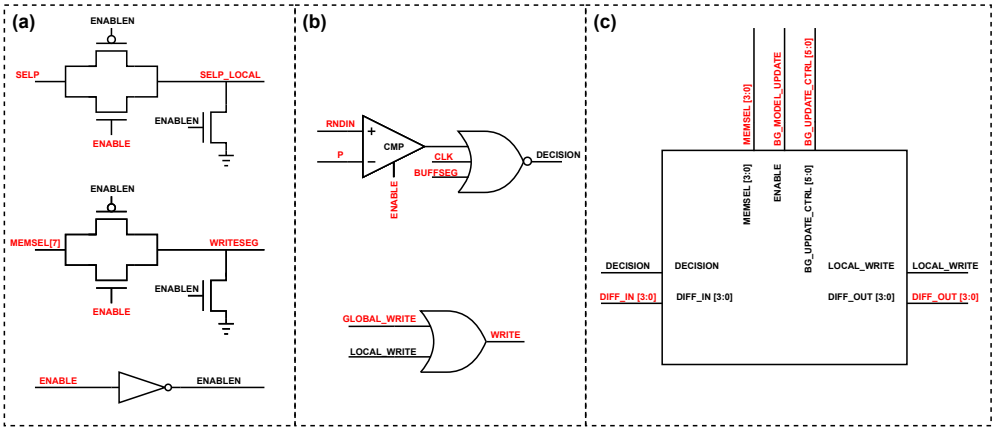

Figure 3.29: Local logic schematic: (a) global signals gate (b) local signals generation (c) background model update unit.

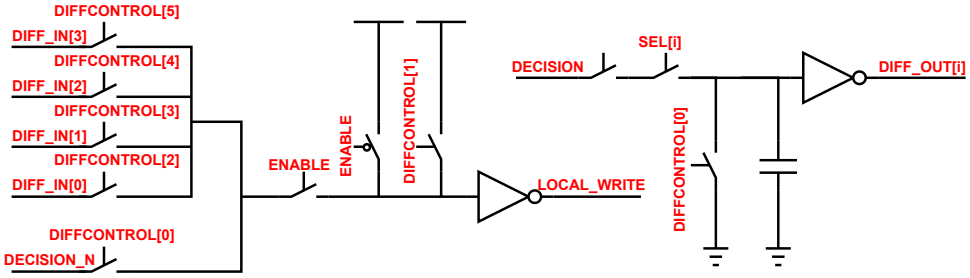

Figure 3.30: Schematic of the background model update unit that controls the background model update mechanism, both the self-update and the diffusion. The right part is replicated for i=0,1,2,3, corresponding to the output for the North, East, South and West neighbors.

first one is to send the $WRITE$ signal to the analog memory bank that stores the background model if the pixel has decided to update its background model. The second one is to send the appropriate signal to a random neighbor if the pixel has decided to perform the diffusion. The last one is to check the neighbors connections to detect a diffusion command from any of them and to generate the write pulse to the ARAM accordingly. This procedure is split in six steps controlled by the global control signals $DIFFCONTROL[5:0]$ plus an enable signal that minimizes glitches. At each step one of these signals is set to high and all of the others to

low, managing the system in the following way:

Step 0) The input circuit sets $LOCAL\_WRITE$ to the value of $DECISION$, taken from the circuit previously explained. This signal is connected to the WRITE input of the ARAM through the logic shown in Figure 3.29 (b). Also, the output circuits are reset, setting $DIFFOUT[3:0]$ to VDD. These signals are the ones that are connected to each one of the neighbors.

Step 1) In this step the block decides whether to perform diffusion or not. This is done by writing the decision result in one of the four output circuits. The $SEL[3:0]$ signal, which comes from a periphery circuit, sets towards which neighbor this is going to be performed by randomly selecting which one of the four signals is high. The selected circuit, which was previously reset, changes its value to the decision result, that might be high or low. This value is hold for the remainder steps of the diffusion process.

Steps 2-5) Now, in these cycles, the input values from the neighbors are multiplexed into $DECISION$. If any of the neighbors decided to perform the diffusion, the input value will be low provoking $DECISION$ to be high and setting the write signal of the ARAM.

During all this process, at each step, the memory selector of the ARAM is randomly chosen using the input from a random number generator that will be described in following sections.

3.3 Periphery Circuits

Although most of the image processing is performed inside the core array, some functionality needs to be implemented at the array periphery. This circuitry is responsible of tasks such as control signals generation, image and segmentation readout or random numbers generation.

3.3.1 Readout Circuitry

After the image is captured, its corresponding value is held in a frame buffer in order to be read when required. Thus, it is necessary both row and column decoders to select which specific pixel is going to be connected to the output, converting the binary value generated by the global control unit to a one-hot format (i.e., all the bits set to zero except the selected one). These circuits can be implemented with different approaches, as the full-custom analog

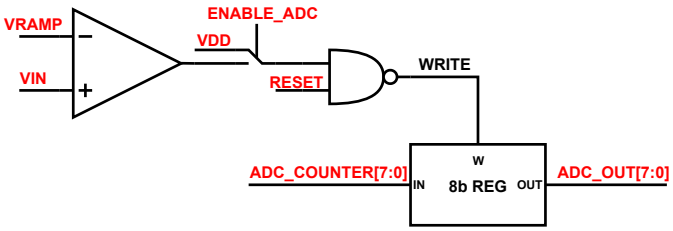

Figure 3.31: Schematic of the per column 8 bit SS-ADC of the HO-PBAS CMOS vision sensors of this PhD.

circuit used in [32]. However, in this work, a pure digital synthesized circuit will be used. It decodes the binary value to a one hot line if the input is valid, or to all zero otherwise. Our first HO-PBAS chip, HOPBAS1K, features an array of 24 $\times$ 56 pixels, so that a 5-bit row decoder and 6-bit column decoders are needed.

The CMOS vision sensor chips for HO-PBAS designed in this thesis can provide the raw analog value if required. In addition, a column-wise ADC is incorporated at the bottom part of each column. Different options can be found for the ADC architecture in the literature [52], each of them with its benefits and drawbacks. SAR and cyclic ADCs commonly offer the best figure of merit [53]. However, in this thesis a single slope ADC (SS-ADC) was selected because of its control simplicity and great linearity. Although for large arrays they might not be the best option due to the low-speed and high power consumption caused by the fast and accurate voltage ramp generation, in this proof-of-concept small design it is a good candidate. Also, it will be implemented in a per-column approach, making the conversion time only proportional to the number of rows and not to the whole array size. The voltage ramp and ADC counter can be shared between all the ADC cells, as it will be explained below.

The working principle of the SS-ADC is based on the comparison of the voltage to be converted with a known voltage externally supplied. It can be seen how the circuit is implemented in Figure 3.31, where V_{RAMP} is generated in an off-chip DAC, and $ADC\_COUNTER$ is an 8 bit counter signal created in the global control block at the array periphery inside the HO-PBAS chip. The circuit operation begins by resetting the 8 bit register with the reset signal connected to the NAND gate, as shown in Figure 3.32. This sets the $WRITE$ signal to high and writes a 255 value to the 8-bit register, supplied from the $ADC\_COUNTER$ bus. Then, if the ADC is enabled, the NAND gate input is connected to the comparator output, and the reset value is turned low. At this moment, the voltage ramp, V_{RAMP}, begins to decrease synchro-

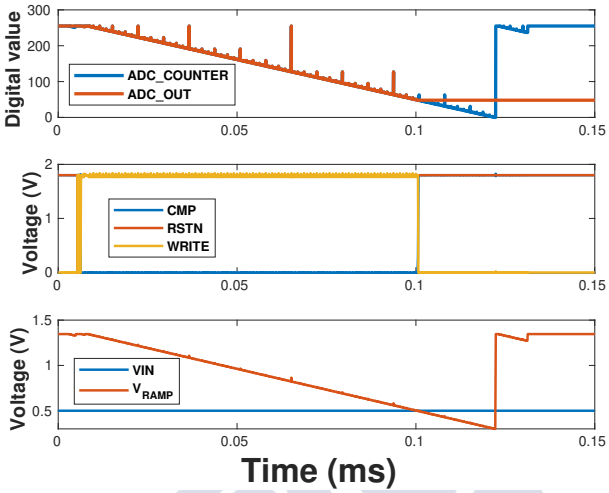

Figure 3.32: 8-bit single slope ADC simulation output used as per-column ADC on the HO-PBAS CMOS vision sensors of this PhD.

nized with the $ADC\_COUNTER$. While V_{RAMP} goes below VIN, the 8 bit register is being written by the counter signal until the comparator output goes low and the register stops being updated, holding the last 8-bit count until the next reset. Finally, V_{RAMP} reaches the minimum value and the systems holds in this state for the required time to transfer the conversion result to the next block in the data readout path. One advantage of the SS-ADC, among its control simplicity, is that the voltage ramp can be shared by all of the column ADCs. Control signals can also be shared, and no other local logic is required besides the NAND gate driven by the local comparator (Figure 3.31).

The conversion result can be used following different strategies. One possibility is to hold it in the register until it is read through multiplexers. This option highly reduces the needed area as any additional circuitry is not required to buffer the result. However, the ADC should remain idle while the readout process is being carried out not to destroy the converted value, increasing the total conversion time of the whole frame. A different option is to add a digital frame buffer and to store the converted image, available to be read when required. This approach would offer the possibility of asynchronous readings, which would be beneficial for some applications. Nevertheless, the required area for such a buffer would highly reduce the available space for other circuitry as larger arrays of pixels. Thus, our HO-PBAS CMOS

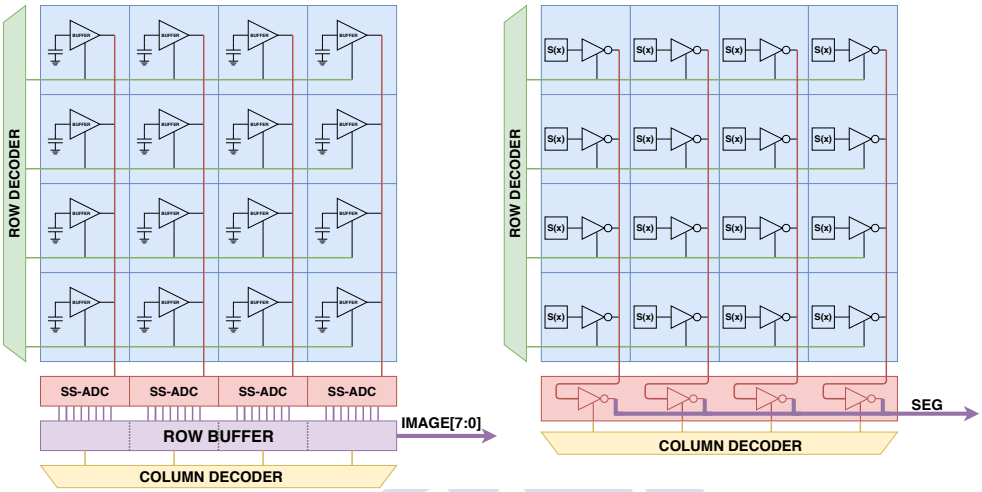

Figure 3.33: Image and segmentation result readout scheme of our HO-PBAS CMOS vision sensor chips. The row and column decoders are shared for both outputs.

vision sensors will use a strategy in between both options, shown in Figure 3.33, by using a row buffer that holds a conversion result from a row while the next one is being converted. Hence, while the row *i+1* is being converted, the results from the row *i* are being outputted through a digital multiplexer based on tri-state inverters to the 8-bit output bus.

The HO-PBAS pixel holds both image in the analog domain and the foreground segmented image in 1-bit format. As seen in Figure 3.33, the row decoder output lines are connected to the analog buffer that holds the captured image and also to the register with the segmentation result, both of them inside the pixel.

Different options exist for the readout process. One possibility would be to wait until the segmentation process had finished, and then read in parallel the 8-bit captured image and the 1-bit segmentation result. However, in our HO-PBAS vision chip the 8-bit image is read out while the segmentation process is carried out and, when this is finished, the read out process is repeated again for the segmented image.

3.3.2 Random Numbers Generation

The HO-PBAS needs a random source for decision making at the background model update step. This random source must provide both analog and digital values: analog values are

required for the comparison with $p(x)$ and digital values are used for the decision of which sample is updated or towards which neighbor the diffusion mechanism is applied. Due to the importance of true random number generators (TRNGs) in information security applications, such as key generation and digital signatures, many papers about them can be found in the literature [54, 55]. Typically, TRNGs work by sampling and post processing a physical entropy source. Depending on the type of this entropy source they can be divided roughly in two different categories: analog or digital TRNGs. The major part of the developed TRNGs are designed for an only-digital output. Those with an analog entropy source might be modified to also provide a random analog stream.

Analog TRNGs might extract randomness from analog physical sources such as thermal noise, the photoelectric effect [56] or chaotic maps. Whereas the former ones assure the true randomness because of the natures of its source, they would impose hard constraints on the overall design as they would be hard to implement in standard CMOS technologies. Also, their modeling and the design of the appropriate circuit conditioning would be very complex. Conversely, switched-capacitor circuits allow us to implement TRNGs based on chaotic maps with a great level of robustness under mismatch and process variation. These circuits work by using a non-linear transfer function, which can be of different shapes, to produce output streams using as the input the output of the previous iteration. With the non-linear transfer function chaotic streams may be obtained.

As the chaotic maps are built ad-hoc for our design they will feature the appropriate specifications in terms of output voltage range or bit rate, among others. In this thesis, a variation of the solution reported in [57] will be used. This work performs an analysis of different chaotic maps and it shows simulations for an implementation of the skew-tent map, proving that this circuit is able to produce random streams that pass the NIST test [58].

A skew-tent map with a unity output range can be defined as:

$$V_{n+1} = \begin{cases} \mu(V_n - Et) + 1, & \text{if } V_n < Et \\ -\frac{\mu}{\mu - 1}(V_n - Et) + 1, & \text{else} \end{cases} \tag{3.9}$$

where μ is the parameter that controls the position of the skew peak and $Et = 1/\mu$. This non-linear circuit works by using the output of iteration n, V_n, as the input for the iteration $n+1$. Thus, defining the initial value V_0 the whole stream will be defined. As stated in [57], for a value of $\mu > 1$ these circuits may produce output streams chaotic enough to pass the NIST, which means that it can be used with a certain level of confidence in practical applications.

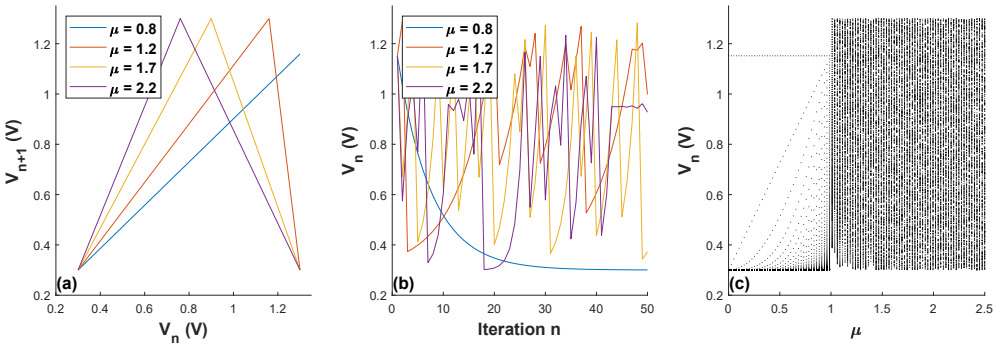

Figure 3.34: MATLAB simulations: (a) Different skew tent maps as a function of μ. (b) Output stream for the skew tent maps of (a). (c) Bifurcation map of the designed skew tent maps.

In this work, different chaotic maps will be implemented with arithmetic units. These circuits need to operate with an offset voltage to be compatible with the voltage range of subsequent circuits. Thus, (3.9) is manipulated to incorporate this offset and rearranged to be in a more convenient form to be implemented, resulting in:

$$V_{n+1} = \begin{cases} \mu(V_n - V_{ref}) + V_{ref}, & \text{if } V_n < Et + V_{ref} \\ \frac{\mu}{\mu-1}(V_{ref2} - V_n) + V_{ref}, & \text{else} \end{cases} \tag{3.10}$$

V_{ref} is the offset voltage and $V_{ref2} = 1 + V_{ref}$. Figure 3.34 (a) shows different ideal MATLAB skew-tent maps of (3.10), as a function of μ. Their corresponding output streams are also plotted in Figure 3.34 (b) for an initial value of V_0=1.15 V. The result of iterating over different maps is also shown in Figure 3.34 (c), where for each value of μ in the x axis, y axis represents multiple outputs in what is called the bifurcation map. This representation deletes the temporal information but allows to study how chaotic and how spread is the output as a function of μ. It is clear from the graph that for $\mu < 1$ the output becomes non-chaotic, due to the fact that in this region the skew tent map transforms into a straight line (see μ = 0.8 at Figure 3.34 (a)).

The arithmetic unit previously discussed in Section 3.2.2 can be used once again to implement such a skew-tent map, as it is composed of two different linear functions. Both parts of (3.10) can be carried out by setting the circuit inputs according to Table 3.3. Such a configuration is obtained with the circuit shown in Figure 3.35, where the inputs are connected as a function of the comparison result between V_n and $E_t + V_{ref}$ with analog switches imple-

Table 3.3: Arithmetic unit inputs for the RNG of our HO-PBAS CMOS vision sensor chips.

Condition	V_1	V_2	V_3	C_1/C_2
$V_n < Et$	V_n	V_{ref}	V_{ref}	μ
$V_n > Et$	V_{ref2}	V_n	V_{ref}	$\frac{\mu}{\mu-1}$

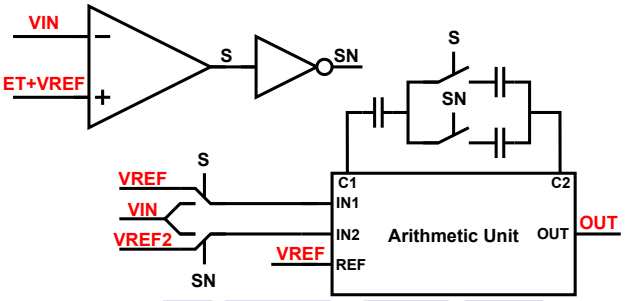

Figure 3.35: Skew-tent map schematics for random number generation of our HO-PBAS CMOS vision sensor chips.

mented with CMOS transmission gates. Monte Carlo simulations are shown in Figure 3.36 for the transfer function of the chaotic map. It is shown that the main non-ideality appears for $V_n \approx E_t + V_{ref}$ caused by the offset error of the comparator. Also, as the slopes are based on the ratio of two MIM capacitors, the circuit shows a great robustness under process variation. In addition, the effect of the reset comparator in the case of a value smaller than the minimum appears as the vertical line at the left part of the graph.

In order to extract time series of random analog values the circuit shown in Figure 3.37 is used, with two sample and hold circuits to introduce the output of one cycle as the input to the next one. Additionally, a comparator is included to check if the input value is below the minimum value due to circuit non-idealities, as in that case the output will tend to zero forever. If this happens, the output of the comparator changes the map input to the initial value and the process is reset. Another failure source might be input values greater than V_{ref2}. However, in this case the corresponding output value will be smaller than the minimum value and the problem will be solved by the strategy previously explained.

Even when the analog output is chaotic by construction, a strong temporal correlation exists. With the intention of reducing this correlation and improving the output quality of the bitstream the combination of the four different skew-tent maps shown in Figure 3.38 was

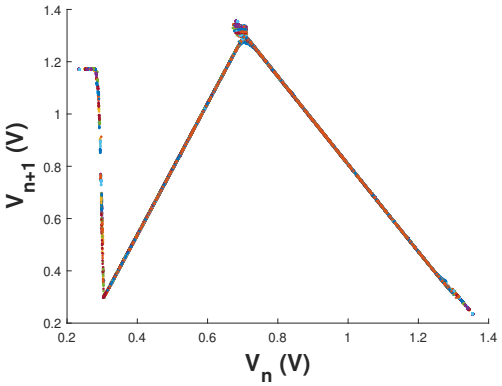

Figure 3.36: Monte Carlo simulations of the circuit shown in Figure 3.35 for the skew ten map.

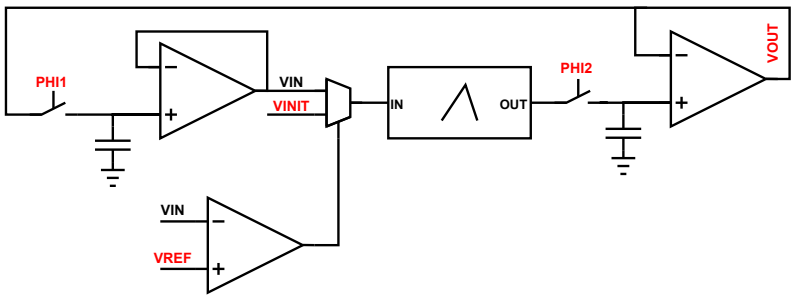

Figure 3.37: Sample and hold circuits combined with the skew-tent map to provide random numbers. Additional circuitry is added to increase the reliability of the system.

developed. The idea of this system is to obtain the output of two different RNG cells and randomly select one or the other for the analog outcome. To select which one is used a digital random value generated by other two different RNG cells is included. First, each of the analog values obtained from the cells corresponding to μ_3 and μ_4 is compared against their characteristic voltages, $E_{t3}+V_{ref}$ and $E_{t4}+V_{ref}$, respectively. To reduce the bias of these two bitstreams a simple but effective post processing is applied to them. This post processing is just an XOR gate that, despite its simplicity, highly reduces the output bias [57]. Once the analog output is selected, it is compared with its characteristic voltage and the digital result connected to an additional XOR gate to be combined with the digital value used before. Thus,

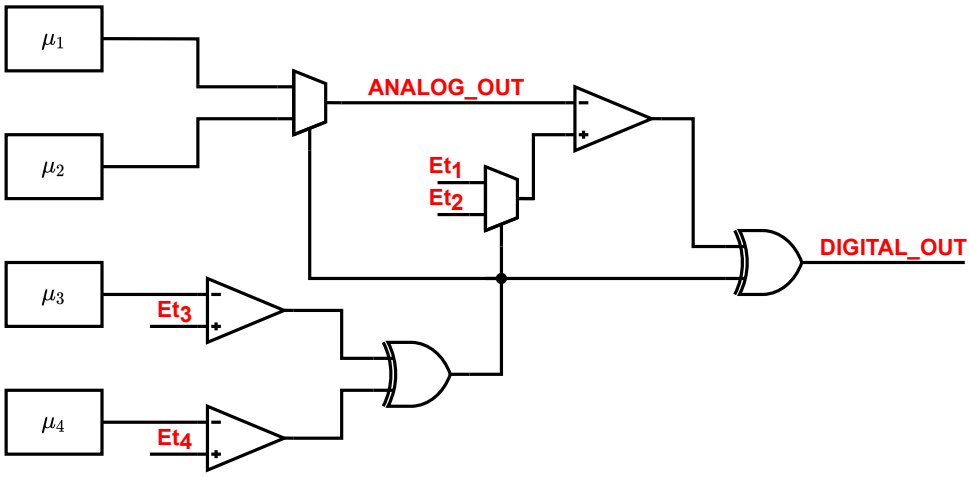

Figure 3.38: Combination of RNG cells to reduce the output temporal correlation and improve the bitstream quality.

the digital output of the system will be the result of this final XOR operation. Figure 3.39 shows a simulation of the final RNG system with the analog and digital outputs, showing the validity of the design.

Ideally, each pixel will need its own TRNG. However, its circuit complexity and required area make this option unfeasible. To check the impact of sharing the same random source by all the pixels in the segmentation performance, image tests implemented in C++ with OpenCV were repeated with this feature. The results of such simulation show that the impact is marginal, with only a reduction in the performance of 1 % for the *changedetection* benchmark [25].

Also, circuit simulations showed that skew-tent quality decreases with output frequency. Thus, it is better to obtain random values slowly at the image exposure stage and store them until they are needed by the processing. As these memories will drive the whole array the fanout will be considerable. Thus, instead of using one buffer per cell, eight of them in parallel are used for each one. Analog values are stored in MIM capacitors of 200 fF and digital bits in the digital block generated by synthesis that also controls the TRNG.

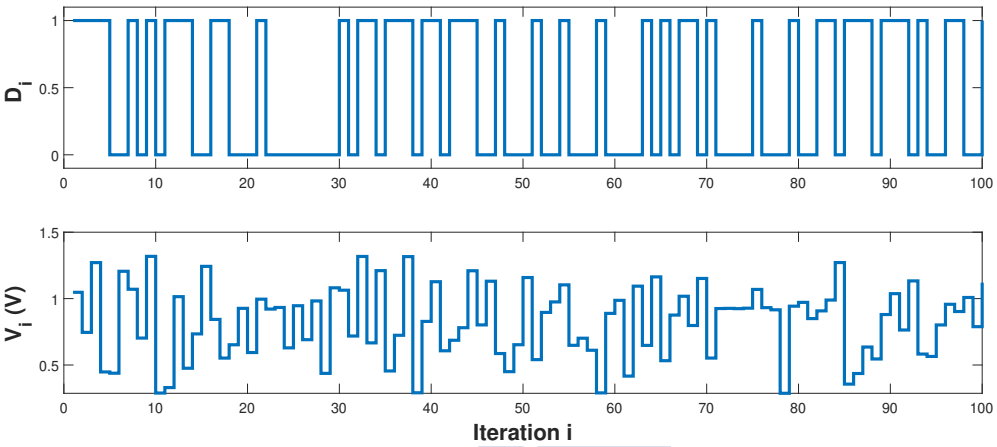

Figure 3.39: Electrical simulation for the full scheme of the random number generator for both digital (top) and analog (bottom) outputs on our HO-PBAS CMOS vision sensor chips.

3.4 System Architecture

In Sections 3.2 and 3.3 an in-detail description of the blocks used on the proof-of-concept chip was shown. However, the architecture design is also a key point in such a complex system as decisions taken in this stage may have a big impact on the system performance metrics (speed, power consumption or output quality).

3.4.1 Processing Element

It is worth remembering here that in order to have a reasonable area per pixel, several pixels share circuitry. Every pixel is a photodiode with their local circuitry. Every group of pixels with a shared Processing Unit (PU) makes a Processing Element (PE). Once the blocks that form part of the PE that will be replicated in the array are known, an important decision is to choose the appropriate ratio of the number of pixels per PU. For this, it should be taken into account considerations such as the maximum processing speed, fill factor optimization or design complexity. Also, the PU sharing strategy must be chosen (one for the whole array, per row/column of group of rows/columns or per group of pixels).

In this academic proof-of-concept work a pixel array that fits into a small area was fabricated, and, in order to make the conclusions extracted from it valid for future work, the scalability of the design is a main concern. Hence, to select the sharing scheme the process-

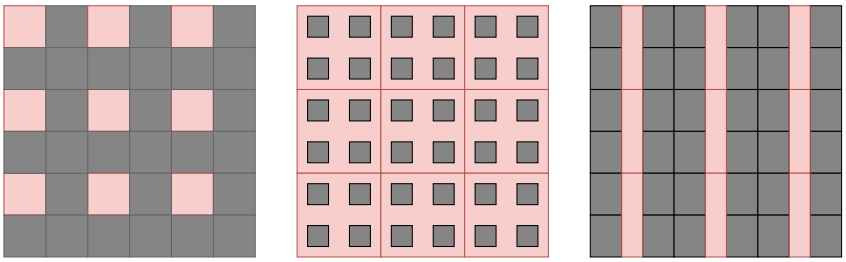

Figure 3.40: Different strategies for the shared processing circuitry placement inside the array: substituting the space of a pixel (left), spread in between all the pixels (center) or in between the group of pixels (right). Pink areas represent shared circuitry and grey areas are the individual pixels.

ing time of the whole array was used as the main criteria. Table 3.4 shows the total array processing time, T_{proc} as a function of array size and sharing strategy. N and M are the number of rows and columns, respectively, and t_{proc} the required time of one pixel to be processed. The processing time of the first two options of the Table 3.4 is proportional to the number of rows and columns, or to the number of columns, respectively. The option of a processing block per group of pixels guarantees the scalability of the design, as the processing time only depends on the pixels per PU. Nevertheless, the fact that the processing circuitry is placed inside the array increases the pixel fill factor, decreasing the image spatial resolution.

Table 3.4: Total processing time as a function of selected strategy for PU sharing among pixels and pixel array size.

Type	$\mathbf{T_{proc}}$
Single block	$N \times M \times t_{proc}$
One per n columns	$n \times N \times t_{proc}$
One per n pixels	$n \times t_{proc}$

For the one shared processing unit per group of n pixels approach different options exists in the literature. One simple option would be to substitute a pixel with the PU, losing the visual information that would have been gathered by the pixel in that position in the array (Figure 3.40 (a)). A more sophisticated approach was introduced in [59], where different processing resources were distributed at different pixels, forming macro-pixel structures. Nevertheless, with this approach depending on the implemented algorithm the processed image needs to be downsampled, as for each processed pixel resources from different array locations are

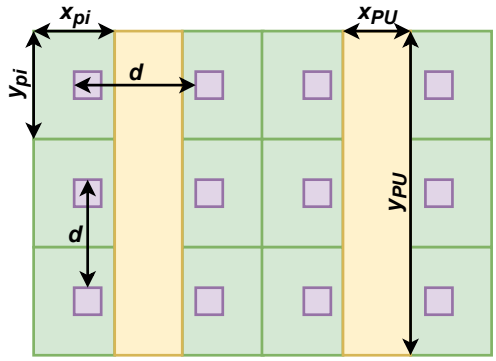

Figure 3.41: System architecture scheme of our HO-PBAS CMOS vision sensor chips. Pixels are represented in green, photodiodes in purple and the PU in yellow. In this example the PU is shared by 6 pixels.

required. A different strategy to avoid the loss of information could be the one used in [60], where the processing resources are located between the pixels in the group, maintaining the pixel distribution homogeneity (Figure 3.40 (b)). However, by using this approach for our analog design the layout complexity would increase substantially, differently from the digital design reported in [60], where the placement and routing of the standard cells used is executed by a CAD tool. Furthermore, the processing circuitry used in this work is designed with full-custom methodology at transistor level, and the design must be laid out manually. Thus, rectangular regions for circuits placement are desired.

In this thesis, a novel approach for the processing circuitry distribution is proposed. Figure 3.40 (c) shows the idea of the architecture used in this work: the PU (in pink in Figure 3.40 (c)) is placed between n pixels. To preserve a homogeneous distribution of the photodiodes, the pixels are placed mirrored in the same group, with the photodiode moved from the center in a way that they are at the same distance from all of the neighbors' photodiodes, as shown in Figure 3.41. This architecture imposes hard constraints on the pixel (x_{pi},y_{pi}) and PU (x_{PU},y_{PU}) dimensions. As all pixels have the same dimensions, the vertical distance between photodiodes d must be equal to the pixel length y_{pi}. Also, as photodiodes are equally spaced in both axis, the width of the PE formed by the pixels and the PU must have twice the length of a pixel. Thus:

$$2x_{pi} + x_{PU} = 2y_{pi} \tag{3.11}$$

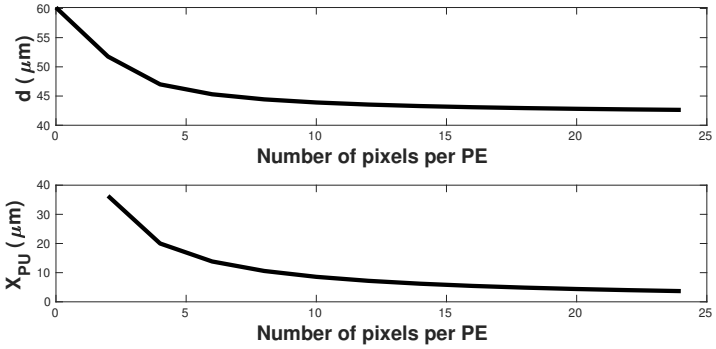

Figure 3.42: Pixel pitch (top) and PU width (bottom) as a function of the number of pixels per PE.

By examining Figure 3.41, it is also clear that the length of the PU is $n/2$ times the length of the pixel, i.e., $y_{PU} = \frac{ny_{pi}}{2}$, where n is the number of pixels per PE. In addition, if we define pixel area as $A_{pi} = x_{pi}y_{pi}$ and shared processing unit area as $A_{PU} = x_{PU}y_{PU}$, a system of 5 equations and 5 variables is obtained, allowing us to express the distance between photodiodes as a function of the required areas of each part and the number of pixels per PU:

$$d = \sqrt{A_{pi} + \frac{A_{PU}}{n}} \tag{3.12}$$

Equation (3.12) is of great relevance, as it shows the photodiodes distance d, what is equivalent to the pixel pitch in a design without a shared processing unit, as a function of the sharing degree with the number of pixels in a PE, n, as a parameter. Manipulating the previous equations, all the dimensions can be calculated. From all of them, the one that has more impact on the layout complexity is the PU width, as it will be the shortest one and it can be expressed as:

$$x_{PU} = \frac{2A_{PU}}{nd} \tag{3.13}$$

Hence, the procedure followed in this work for the array floorplan was the following: first, the analog primitives were laid out and an estimation of all the blocks area was performed, resulting in an approximate similar area of $A_{pi} \approx A_{PU} \approx 1750\ \mu\text{m}^2$. These data can be used to plot in Figure 3.42 the mentioned magnitudes and select a reasonable value of n that provides a PU wide enough for a feasible layout. From the plot we can see that n = 4, which means

four pixels per PU, offers a great improvement compared with $n = 2$. Sharing PU by six pixels would be also a good solution, as the reduction in pixel pitch with respect to $n = 4$ is substantial. However, its correspondent $x_{PU} = 13.83$ μm is too narrow to embed all the functions of our PU, which makes a better candidate the solution of four pixels per PU with a correspondent pixel pitch of 47 μm. It is worthy to mention that with the not sharing strategy the approximate pixel pitch, that could be calculated by (3.14) assuming a square pixel and that the saved area from the local logic responsible of sharing control is negligible, would have a value of 60 μm. Thus, the group of pixels sharing strategy leads to more than 27% of area saving. In other terms, with the available area for the CMOS vision chip of this thesis, this pixel pitch would lead to a spatial resolution of 18×43 pixels. If this resolution is compared with the achieved 24×56 pixels on the same silicon area, it can be concluded that the increase in the design complexity of this approach compensates for the effort.

$$d_{nosharing} = \sqrt{A_{PU} + A_{pi}} \tag{3.14}$$

Even when this analysis was made with estimated layout areas and the the final dimensions might no be exact, it is of great help as once selected the sharing degree n of the architecture, if any of the areas increases considerably with respect of the first estimation the calculation might have to be redone to know the new layout parameters and adjust it in consequence. Without this formalism, the layout task would turn into a trial and error process that would considerable increase the design time. In the case of this work shapes obtained from analytical analysis resulted in the final ones, as the little extra space after the final layout was used for the blocks metal connections. Figure 3.43 shows the final layouts of the pixel, PU and PE with its parts labeled. As some extra space was unoccupied in the pixel an output selector for testing different signals of the pixel was added at the top right corner. Regarding the PU it was implemented in a rectangle of 94 × 20 μm^2 adjusting the shapes of all the processing blocks to fit in it.

To build up the PE that will form part of the array, a few considerations regarding neighborhood connectivity should be taken into account. The first one is that the pixels at one side of the processing unit are slightly different from the others. This occurs because as they are mirrored, diffusion block input/output connections will not match its corresponding output/input pin and ports positions should be swapped. Similarly, for the Gaussian diffusion block if the pixels were identical a processed image distortion would happen as East ports would be connected to the East ports of their neighbors and the same would happen to the

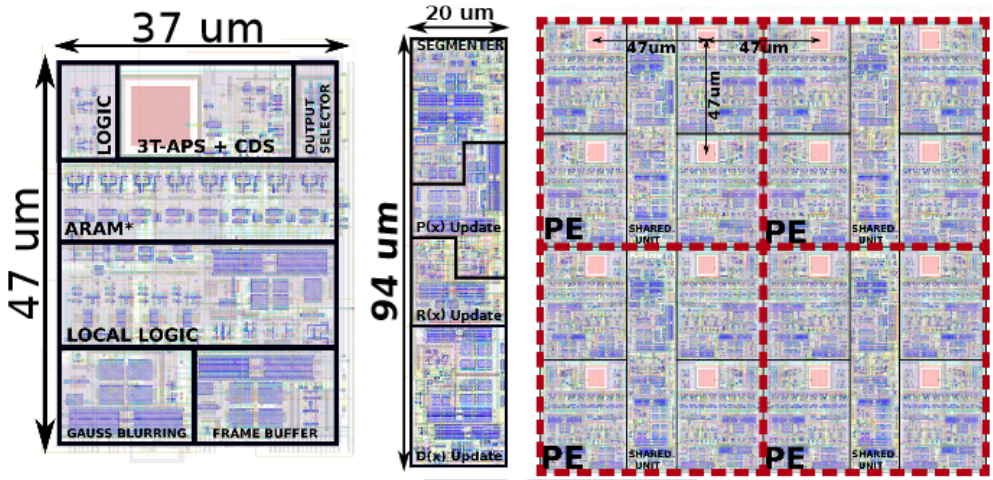

Figure 3.43: Chip architecture. From left to right: pixel, PU and four PEs (formed by four pixels and a PU each one) that make part of the chip core of the HO-PBAS CMOS vision sensor chips designed in this thesis.

West direction. Thus, two layout versions of the pixel were created depending if they are placed at the left or right side of the processing unit. Figure 3.43 shows four PEs of four pixels per PU, where pixels homogeneously spaced can be seen.

3.4.2 PE Array

The PE is designed for an array deployment based on metal connections overlap. Thus when the PE cells are put together and correctly aligned and spaced, they will share bias connections in the horizontal axis and digital control signals in the vertical direction, with the intention of reducing the cross-coupling between them. Power supply distribution was performed by a power ring surrounding the whole array and adding a power stripe for each column of PEs. Then, in the vertical direction, each column is provided with a two wire bus to send the captured image (or the selected output for the pixels) and the segmentation result when its row is selected by the row decoder. This bus is connected to the circuitry placed at the bottom part of the column which includes the single slope ADC, a tri-state register for the segmentation result multiplex and an 8-bit digital buffer to hold the ADC conversion, as explained in Section 3.3.1.

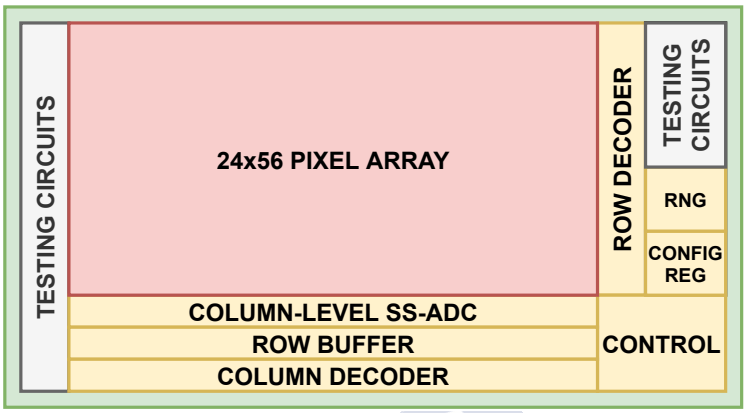

Figure 3.44: HOPBAS1K floorplan.

3.4.3 Chip Floorplan

As a proof-of-concept work, this design was fabricated in the TSCM 180 nm standard CMOS technology using a two reticle mini ASIC resulting in a chip area of 1.6×3.2 mm^2. After considering the area required by the pads and the periphery circuitry, a 24×56 pixel array was implemented. Figure 3.44 details the chip floorplan with the circuit blocks distribution, composed of:

- Pixel array: 24×56 pixel array grouped into a 12×28 PE array for the image capture and segmentation processes.
- ADC & Readout circuitry: formed by the row decoder that connects a row to the column-wise ADC, the ADC itself and the row buffer which holds the conversion result while it is being accessed by the column decoder.
- TRNG: random number generator which provides the pixels with analog and digital values required by the HO-PBAS foreground detection algorithm.
- Configuration register and control signal generator: these blocks, explained in detail in Section 3.5, read user configuration input and generate the digital control signals accordingly.
- Testing circuits: in order to fully characterize processing blocks designed in this thesis some of them were implemented individually on the periphery area, with their control,

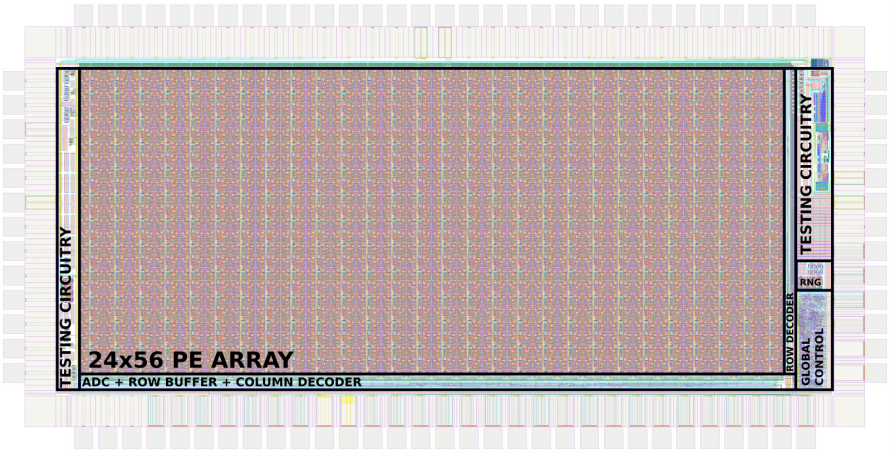

Figure 3.45: Final layout of the 24×56 pixel array HO-PBAS proof-of-concept foreground detection CMOS chip, HOPBAS1K.

input and output signals connected to dedicated pads. These blocks are the segmenter, $d(x)$, $R(x)$ and $p(x)$ update blocks and the TRNG.

An in-depth description of the chip input and output connections is detailed in Appendix A. However, an overview of the I/O is also shown in Figure 3.44. The chip main outputs are the captured image offered in an 8-bit parallel bus or through an analog pad and the segmentation result. Also, for debugging purposes, testing probes were added to the PE in the top right corner of the array, each of them with its own output analog pad to allow the visualization of all of them at the same time. The chip inputs include clock and configuration signals (both explained in the next Section). In addition, individual testing circuits are provided with their own input/output pads as explained before. The final layout of the chip is shown in Figure 3.45.

3.5 Control Unit

An analog on top mixed-signal system is designed in this thesis. By using this strategy most of the image processing is performed in the analog domain. However, many of the analog units are based on switched-capacitor circuits that need clock signals. Furthermore, all the circuitry needs to be synchronized to operate at the appropriate time to offer its result for the

subsequent blocks at the right moment. This task is carried out by a shared control unit that generates the control signals for each block and it might be on-chip or at the outside. The advantages of generating such control signals at the outside are, among others, the possibility of reconfiguration. For an academic work such as this one it would be a great point, as this feature would offer the possibility of fixing mistakes made at control design, or to try different configurations if desired. Nevertheless, this strategy could not be even an option due to the limited number of available pads or it would obligate to highly reduce the number of testing points in the design. Therefore, this option had to be sacrificed and the digital control unit was implemented on-chip, using digital synthesis techniques, and taking as input only clock signals and a three-wire bus for system configuration.

This control unit is the responsible for generating all the control signals required by all the circuits on HOPBAS1K that, due to the different tasks required in the system operation, will need to manage four different frequency domains corresponding to image capture and processing, random number generation, analog to digital conversion and image readout. The TRNG will operate in the slowest domain to provide a random output of good quality. The analog processor placed inside the array needs a faster clock signal to execute the algorithm in the required time slot for the desired frame rate. Finally, the ADC and readout circuitry are handled by two clock signals considerably faster than the processing clock as the single-slope ADC and decoding system need to perform a huge number of operations compared with the algorithm execution.

A timing scheme of the global system operation is shown in Figure 3.46. After a global reset, the first image is captured until the configured exposure time is reached. Then, a new frame begins to being captured, independently from other blocks of the system in continuous mode. Every time a new frame is obtained, a signal is sent to the other blocks. When this occurs, at the first frame, the background model is initialized, setting all the samples to the captured value after going through the Gaussian blurring block. After that, the captured image is processed with the unique purpose of generating signals required by the readout block for proper functioning, as the segmentation result is straightforward due to a comparison of the image against itself. For the following frames, the image is processed right after it is captured, while the new frame is being integrated, and the background model updated as the final step of the image processing. Also, after the background model update mechanism has used the random values stored in the TRNG, a new set of values generation begins.

Regarding the analog-to-digital conversion of the image and its readout process, their

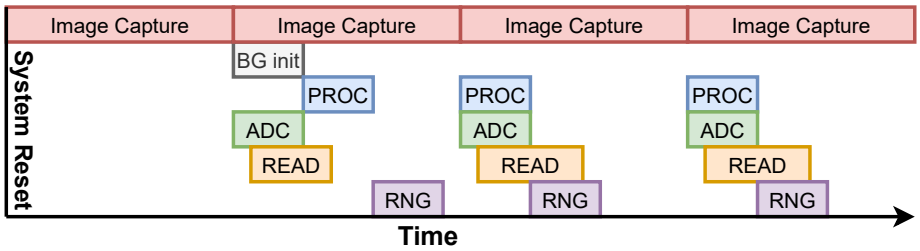

Figure 3.46: Global control time scheme for the different operations of the HO-PBAS CMOS vision sensor chip.

corresponding blocks are connected in a master/slave configuration. When the readout block receives the new frame signal it begins its operation selecting the first row and activating the ADC block. The ADC receives its control commands and performs the conversion. When the process is finished, the readout block selects the next row and resets the ADC block, repeating the procedure until the whole array is converted. At that moment the readout block waits until the image segmentation is executed, and then it repeats the array decoding process connecting the segmentation result with the outside.

To implement this operation scheme different control sub blocks are designed with the Verilog source code in Appendix A, and implemented with digital synthesis tools. The remainder of this section will explain them in detail.

3.5.1 Control Input

This block handles user configuration inputs through a three-wire bus connected to a 33 bits register. The reading data process is performed using the little-endian system, i.e., beginning from the least significant bit. Table 3.5 shows the register structure:

Table 3.5: Configuration register structure for the exposure time, Gaussian diffusion σ, control output multiplexer, pixel output multiplexer and analog output value.

Magnitude	n_{exp}	n_{gauss}	$data\_mux$	$pixel\_mux$	ana
Bit	[0:19]	[20:23]	[24:29]	[30:31]	[32]

Each part of the configuration registers sets different options of the chip:

- n_{exp}: number of clock cycles that the image is integrated controlled by the f_{proc} frequency. Exposure time would be $t_{exp} = n_{exp}/f_{proc}$.

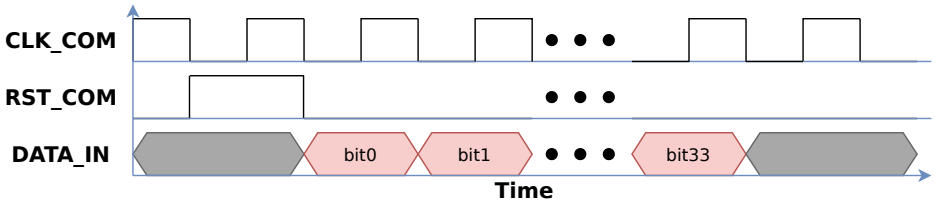

Figure 3.47: Timing diagram for the configuration register write operation of our CMOS vision sensor chips.

- n_{gauss}: number of times that the Gaussian blurring is repeated to control the σ parameter defined in (3.8).
- *data_mux*: control signal of six bits for multiplexing all the control signals generated on-chip to an output pad for debugging purposes.
- *pixel_mux*: pixel output selector. Possible outputs are (0) captured image, (1) low-pass filtered image, (2) contents of memory number 3 and (4) stored $p(x)$ value.
- *ana*: bit that selects if the analog value connected to the per-column ADC is attached to the output pad.

Write operation, displayed on Figure 3.47, begins with a positive pulse of the *RST_COM* signal. After that, a new data bit is read from the *DATA_IN* line every positive edge of the *CLK_COM*, which should be of any frequency slower than 20 MHz.

3.5.2 Core

Core control handles various tasks executed in parallel in the pixel array. These tasks are image capturing and pre-processing, algorithm parameters update, image segmentation and background model maintenance. All the signals are generated from a clock signal of frequency 400 kHz for the signals *CLK* and *CLKN*. Image capturing is carried out as a function of the n_{exp} parameter, which controls the number of clock cycles that photodiodes are integrating the light. The process begins with a photodiode reset and sampling the reset value, as detailed in Figure 3.48. *PHI1CAP* remains high for one additional clock cycle to sample the reset value after the charge injection caused by the reset transistor, which would produce an undesired image offset voltage otherwise. Then, after n_{exp} clock cycles, *PHI2CAP* goes high

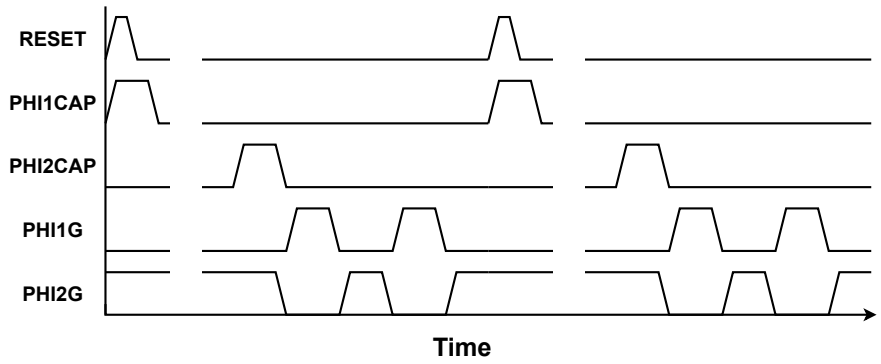

Figure 3.48: Image capture and pre-processing signals of the HO-PBAS CMOS vision sensor chip.

executing the second phase of the CDS and storing the captured value into the frame buffer and the Gaussian blurring block. At this moment, the Gaussian diffusion is performed n_{gauss} times, where n_{gauss} ranges from 0 to 15 and the result is held in the same central capacitor used for the diffusion connected to a voltage buffer to be non-destructively read when desired.

While a new image is being captured, the system is also processing the previous frame in parallel. As mentioned before, first each of the pixels will be processed one by one by the PU and then, when the segmentation result is known, the background model of each one will be updated based on their individual parameters ($p(x)$ and segmentation result). This final step is performed in parallel by all of the pixels as the circuits required by this task are not shared.

Individual pixel processing begins by selecting the first pixel of the PE and connecting it to the PU. Figure 3.49 details all the control signals involved in this process. First, the *ENABLE* bus sets the line connected to the first pixel of the PE to high, connecting all the relevant input/output ports to the PU. Then, the segmentation process begins by extracting $d(x)$ from the background model. Hence, the execution always begins by resetting the block that calculates $d(x)$ with a positive pulse of the *SEG_D_RESET* signal (which is also shared with the segmenter) and with the signal *PHI_MIN* tied low, as required by the block (see Section 3.2.5). With the block correctly initiated, all the memories of the ARAM are connected one by one with the *MEM_SEL* signal bus and when the last one is reached, the process is repeated with the *PHI_MIN* changed to high.

When the execution reaches the last memory of the ARAM $d(x)$ will be calculated and held by an S/H circuit controlled by the *PHI_MIN* signal. Based on this result, $R(x)$ is gen-

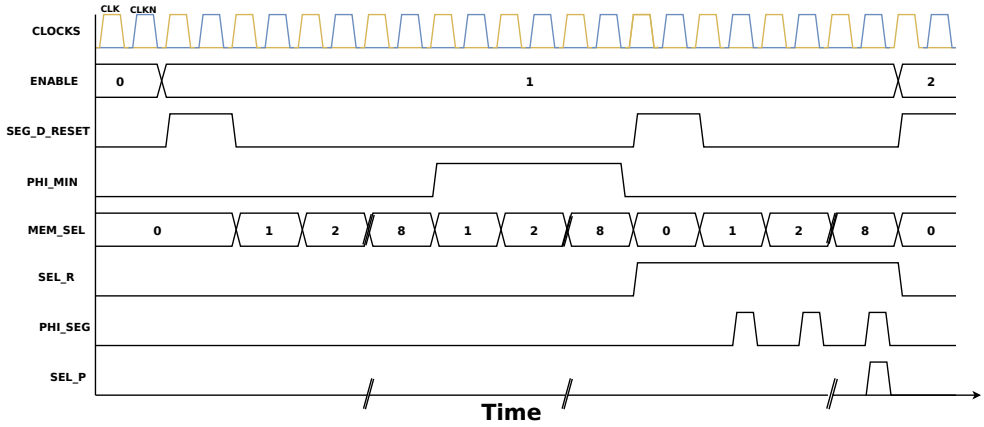

Figure 3.49: Control signals generated for the array core of our 24×56 HO-PBAS CMOS vision sensor chip (see timing breaks in the x axis). Number in buses indicates the bit in high state, all of the others are in low state and zero means that none of them is high.

erated according to (2.8). This block gets its *PHI*1 signal from *CLK* and *PHI*2 from *SEL_R*. Thus, at the next clock cycle, when the segmenter is being reset, $R(x)$ is obtained and maintained for the whole segmentation process. With the segmenter reset and the value of $R(x)$ connected to the segmenter input, the segmentation process only needs to process all of the ARAM values one by one with its *PHI*1 connected to *CLK* and *PHI*2 to the *PHI_SEG* signal, which generates pulses according *CLKN* only when the segmentation is being executed. Finally, when the last memory sample is analyzed, the output from the block that calculates $p(x)$ will be valid as the segmentation result is definitive at this point and it is written into its memory with the *SEL_P* signal.

After the first pixel of the PE is processed and its result stored in the frame buffer, the *ENABLE* signal selects a new pixel and the whole process is repeated until the last pixel of the PE is reached. Then, the background model update is executed in parallel by all the pixels using the signals detailed in Figure 3.50. The background model update begins enabling the background model update unit with signal *BG_MODEL_UPDATE* and setting the first bit of *DIFF_CONTROL* high. At this point, the part of the block that handles the diffusion between pixels is being reset and the part that takes care of the self update is active. Thus, by using the analog random value coming from the periphery, the decision is taken and the randomly selected sample updated or not in consequence. It should also be noted that *DIFF_CONTROL*

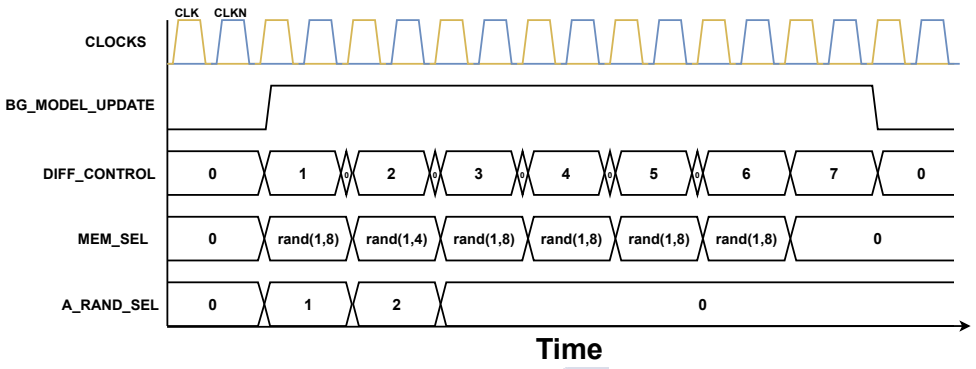

Figure 3.50: Background model update control sequence in the HO-PBAS CMOS vision sensor chip. Number in buses indicates the bit in high state, all of the others are in low state and zero means that none of them is high.

control signals are set low at the falling edge of CLKN, as overlaps between them may produce undesired glitches into the *WRITE* port of the ARAM. The next step is to decide whether to induce a diffusion into a neighbor as a function of a different analog random value and segmentation result or not. If the decision is positive, the neighbor is selected using the *MEM_SEL* bus, which will have one of the first four bits in high state. Finally, each pixel will listen one by one all of the four neighbors to check if any of them is inducing diffusion and, in that case, a randomly selected sample will be updated. At the end of the cycle, the internal signal *DIFF_CONTROL*[6] sends a positive pulse to the TRNG's control logic to indicate that it should generate a new set of random values.

3.5.3 TRNG

As explained in previous sections the quality of the TRNG's output increases as the frequency decreases. Thus, a slow clock, *CLK_RNG* of frequency 50 kHz was included to produce the random values used for the background model update, both analog and digital, with the lowest possible speed while fast enough to provide a different set of values for each frame. Thus, the control block for the TRNG uses the independent clock signal *CLK_RNG* to produce *PHI*1*RNG* and *PHI*2*RNG* as detailed in Figure 3.51. Also, the required signal *ENABLE_RNG* is provided to enable all the circuitry each time the TRNG is generating a new value, cutting off the power when the block is in idle state. Every positive pulse of *PHI*2*RNG* a new set of

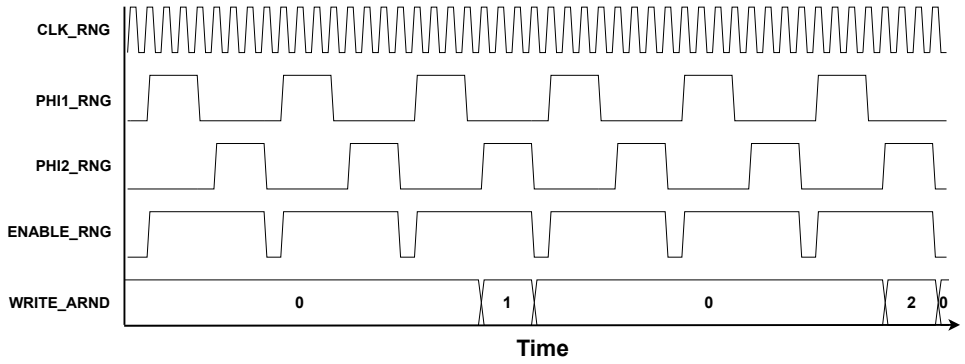

Figure 3.51: Random number generation control signals of the HO-PBAS CMOS vision sensor chips. Number in buses indicates the bit in high state, all of the others are in low state and zero means that none of them is high.

digital and analog values is created: the digital value is always stored in the digital circuitry and, as the algorithm just needs two analog values per processed frame, only two analog values are stored in S/H circuits, at the third and sixth cycle. This process is repeated 18 times as the background model update mechanism needs six sets of three bits to generate digital values from 1 to 8.

3.5.4 ADC & Readout

HOPBAS1K offers the option of performing the analog to digital conversion off-chip by setting a bit high in the configuration register. However, as explained before, it also features a column-wise single-slope ADC that shares control signals and needs a voltage ramp generated outside the chip. This ramp will be synchronized with the control signals of Figure 3.52 using *PHI1_ADC*. All the signals involved in the conversion are generated using the *CLK_ADC* positive edges, configured with an independent frequency of 3 MHz.

The conversion process begins by selecting the first row of the array. This is done with the positive edge of the *PHI1_ADC* signal that tells the readout block that it should increase the *ROW* value. Also, the positive edge of the same signal resets the counter, as detailed in the control signals diagram. At the next clock cycle, the ADC register is reset, setting all the bits to high. Thus, if the input voltage is bigger than the initial value of the voltage ramp, the full-scale byte will be already written into the RAM. Then, the next step will be

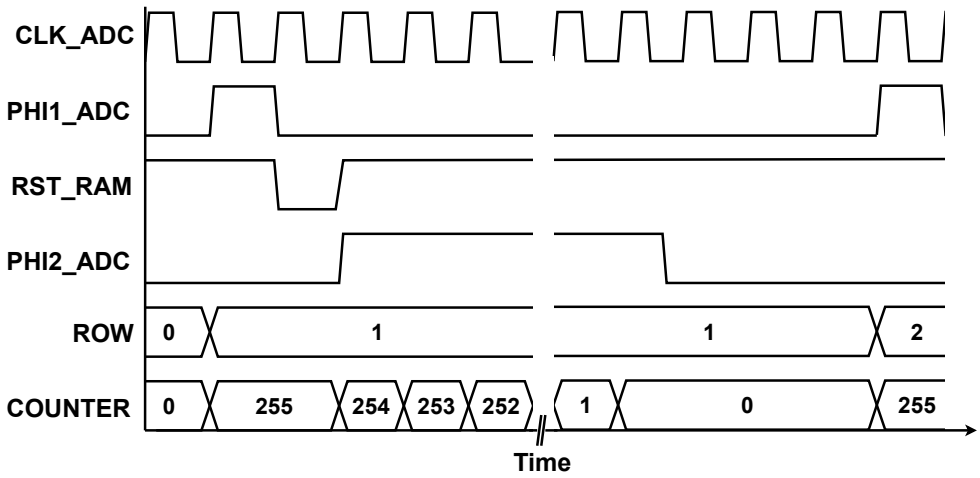

Figure 3.52: Analog to digital conversion control signals of the HO-PBAS CMOS vision sensor chips (be aware of the time axis break).

to enable the conversion comparator with the signal *PHI2_ADC*, connecting the comparator output to the NAND gate that controls the write operation into the register. If the ramp voltage is bigger than the input, the counter value will be written. Otherwise, the register will hold the previously written value. This is repeated until the counter reaches zero at the same time that the minimum vale of the ramp is being used. At this point, *PHI2_ADC* goes low and the system remains idle until the readout logic finishes to read the data stored in the row buffer. Finally, when all the columns were read, the conversion begins again by generating a positive pulse of *PHI1_ADC* that increments the *ROW* val and resets the counter. Furthermore, this control signal writes the value stored in the ADC RAM into the row buffer before it is destroyed by the *RST_RAM* negative pulse.

While a row is being converted, each of the row buffer's 8-bit registers is connected to the output pads for two clock cycles of *CLK_READOUT*, which runs at 2 MHz. As the first row is available to read when the second row is being converted, the data coming from the chip when the first row is connected to the ADC is thrown away and the last row is converted twice. Finally, when the entire array has been read, the readout control block checks if the image was already processed and, if this true, it repeats the process once again connecting the segmentation result one by one to the output pad for one clock cycle.

3.6 Conclusions

This chapter has covered the design of HOPBAS1K. This full custom IC implements the background subtraction algorithm designed in Chapter 2 using analog blocks based on switched capacitors circuits. Main circuit non-idealities, such as the effect of current leakage in the analog memories or design considerations as the sharing of some functions by several pixels, have been included into image simulations performed with C++ and OpenCV, to assess the impact on the algorithm performance.

A PE was designed containing four pixels and a PU. This structure was scaled up forming a 24×56 pixel array with column-level SS-ADC for the captured image conversion. All the control required by the CMOS vision chip was designed using digital synthesis tools and included on-chip. Finally, this chip was fabricated in a 0.18 μm standard CMOS technology, occupying 1.6×3.2 mm^2. The experimental results obtained from the chip tests will be shown in Chapter 4.

CHAPTER 4

HOPBAS1K EXPERIMENTAL RESULTS

Introduction

This chapter covers the design of the experimental setup required to test the first HO-PBAS CMOS vision sensor chip designed in this thesis, HOPBAS1K, along with its experimental results.

4.1 Experimental Setup

In order to test the fabricated chip different parts are required. One of said parts is the socket that provides mechanical support to the die and eases the connectivity with the pads through wire bonding between them and the physical pins of the socket. The decision of which carrier to use was done taking into account our previous experience, the number of pins, and the ease of the chip replacement with a female socket soldered into the system Printed Circuit Board (PCB). With these considerations taken into account, the chip carrier selected was the Ceramic Pin Grid Array CPGA100. A microphotograph of the chip bonded to the chip carrier can be seen in Figure 4.1, and the full pin correspondence can be found in Appendix A.

The packaged chip can be soldered to a PCB to be tested or plugged into a female socket. In a mass production prototype the former option would be used, as it would reduce fabrication costs. However, in this academic work it is preferred to use a female socket to test different chips with the same PCB. The selected female socket was a Zero-Insertion-Force (ZIF) PGA compatible with the CPGA100 that allow to plug and unplug the chips without damaging them.

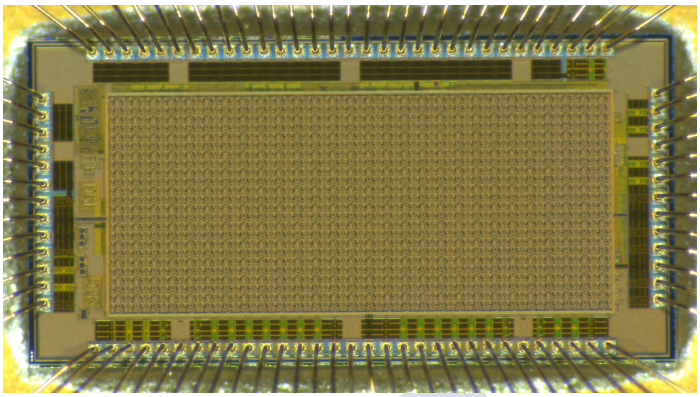

Figure 4.1: Microphotograph of the first CMOS vision sensor chip designed in this thesis bonded to the CPGA100 socket.

The testing PCB, shown in Figure 4.2, includes different circuitry required for the chip test. On the left hand side part of the picture it is the FPGA board CMOD A7 35T [61], used for clock signals generation and chip readout. On the right hand side part the microcontroller Particle Photon can be seen [62]. This microcontroller operates at a clock frequency of 120 MHz and features an 8 channel 12-bit ADC and two 12-bit DACs, as well as 18 digital ports. It is mainly used for the communication with the chip, implementing the custom made protocol explained in Section 3.5. Also, it was employed to generate the required signals for the experimental tests depicted in Section 4.2 and to measure their outputs. The system could be designed with only the FPGA, which would integrate also the custom made communication protocol and test signals generation. However, the use of the microcontroller highly reduces the design time as to implement the explained functionality with a software language such as C++ is commonly faster and easier than to do it with HDLs.

The rest of the PCB circuitry comprises an array of potentiometers used for bias and voltage reference generation, placed at the left bottom corner, the TLC7524 DAC [63] for the voltage ramp generation required by the SS-ADC, controlled by the FPGA, the power supply circuitry placed on top of the microcontroller, and the line of male headers placed on the top part, that offers a physical connection with the CPGA100 socket pins.

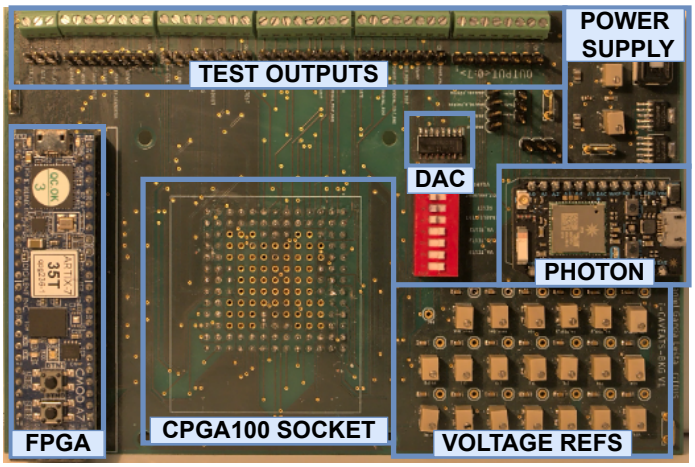

Figure 4.2: Custom made experimental platform formed by the female chip socket, the FPGA, the microcontroller and auxiliary circuitry for HOPBAS1K.

4.2 Individual Test Blocks

As explained in Chapter 3, HOPBAS1K incorporates different testing circuitry on the periphery. This circuitry is meant to test individual block of the design. In this section, such tests performed are shown.

4.2.1 Processing Circuitry

The first circuits that were tested are the ones responsible for the algorithm processing, namely: the blocks that calculate the background dynamics estimator $d(x)$, the segmentation sphere's radius $R(x)$ and the new update probability $p(x)$, and the segmenter that performs the classification process.

Background Dynamics Estimator

The background dynamics estimation is carried out through (2.7), which requires to calculate the maximum and minimum values of the background model, as explained in Section 3.2.5. In this test, the digital control signals were connected to the Photon device, and the background models samples were generated with its DAC. This background model was set in two groups of four equal values v_1 and v_2, i.e., $B(x) = \{v_1, v_1, v_1, v_1, v_2, v_2, v_2, v_2\}$. In the test a full sweep

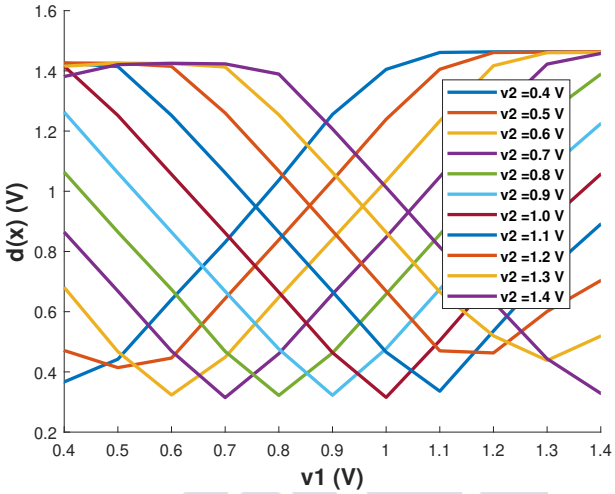

Figure 4.3: Background dynamics estimator $d(x)$ block test. Each curve shows the correspondent $d(x)$ for a background model formed by 4 samples of $v1$ and four samples of $v2$. These experimental data were extracted with an analog input of the Photon device.

over the entire range of the background models was executed and the output $d(x)$ for each scenario was measured through a Photon's analog input. The results are shown in Figure 4.3. It can be seen how the $d(x)$ reaches the minimum value for $v_1 = v_2$ and how it increases linearly with the background model range until it reaches the maximum output value of the arithmetic unit, around 1.4 V, as expected.

Update Probability

The update probability parameter $p(x)$ is calculated at every frame according to (2.6), and it takes as inputs the segmentation result $S(x)$, the background dynamics estimator $d(x)$ and its previous value, which is stored in the ARAM. The individual test block for this function needs p_{min} and p_{max}, which are configured with two potentiometers on the PCB, two analog values for $d(x)$ and $p(x)$, which are provided by the Photon's DACs, and a digital input for $S(x)$, also coming from the Photon device.

The digital control signals, *phi1* and *phi2*, are generated with the Photon unit. First, the inputs are set and, after a clock cycle of each of the control signals, the output is measured and stored. This process was repeated for the entire possible range of $p(x)$ and for different values

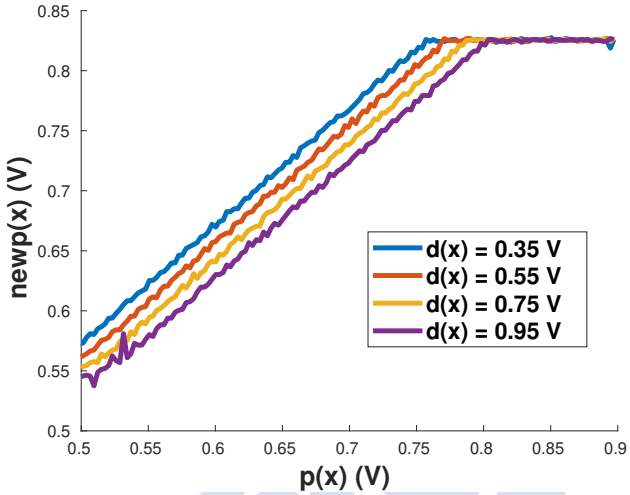

Figure 4.4: Test results for the block that updates the $p(x)$ parameter. In this test the segmentation result, that is used as an input, is set to background, and the parameters p_{min} and p_{max} to 0.54 V and 0.84 V respectively. In the case of foreground the result is always p_{min}. Experimental data obtained from an analog input of the Photon microcontroller.

of $d(x)$ with p_{min} and p_{max} set to 0.54 V and 0.84 V, respectively. The satisfactory results of this test are shown in Figure 4.4. It can be seen how the circuit features a good linearity and how the output result is cropped by the maximum allowed value p_{max}.

Sphere's Radius

The circuit responsible for calculating the segmentation threshold, the radius of the sphere $R(x)$, only requires the background dynamics estimator $d(x)$ as input, provided by the Photon DAC, and the control signals *phi1* and *phi2*. Figure 4.5 shows the output for the entire range of $d(x)$. It can be seen that the output is cropped by the minimum value R_{min}, set by a potentiometer on the PCB. When this value is reached, the comparator connects the output to the arithmetic unit (see Figure 3.22). Due to the comparator offset and its finite gain, at that point the circuit shows a non-ideality. Also, the circuit shows a non-linearity as a function of the input value, probably caused by the non-ideal behaviour of the arithmetic unit. Nevertheless, the actual value of the sphere's radius obtained experimentally is a monotone increasing function with the background dynamics of the scene, which might not be an insurmountable

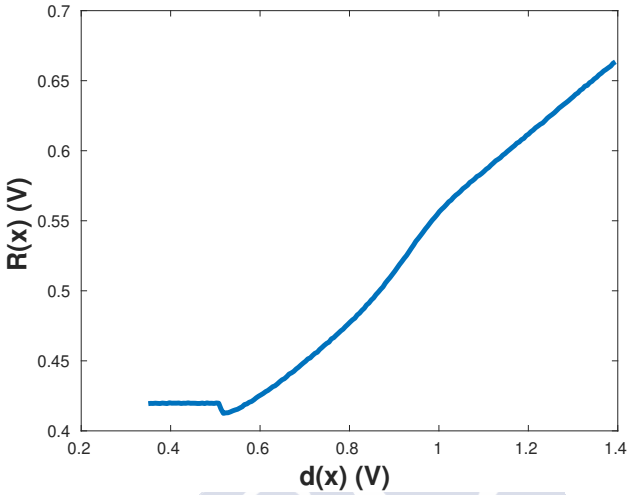

Figure 4.5: $R(x)$ individual block test results for different values of $d(x)$ for the HOPBAS1K.

barrier for the practical use of HO-PBAS on our chip.

Segmenter

The last block belonging to the PU tested is the segmenter. This circuit is the responsible for deciding if a pixel value is similar to the background model or not. It needs two analog inputs corresponding to the ARAM and pixel values, which, for this test, were connected to the Photon's DACs. The controls signals *reset*, *phi1* and *phi2* are controlled also with the Photon unit. Finally, as this is a transient test, the output is captured with a the mixed-signal oscilloscope Tektronix MDO4034C.

The first test performed was to simulate a background model of 8 samples, with four of them being 0.6 V, and the remaining 1 V. Then, three different input pixel values were processed without modifying the background model. The results are shown in Figure 4.6. After each reset cycle, represented in yellow at the top plot, a set of 8 cycles of the control signals *phi1* and *phi2* is repeated. For each cycle of the control signals *phi1* and *phi2*, a new ARAM value is loaded, and the absolute difference from the input pixel and the ARAM values is compared with the radius of the sphere, set to 420 mV in this test. Thus, for the first pixel value, set to 650 mV, right after the second cycle the segmentation results goes to zero, as

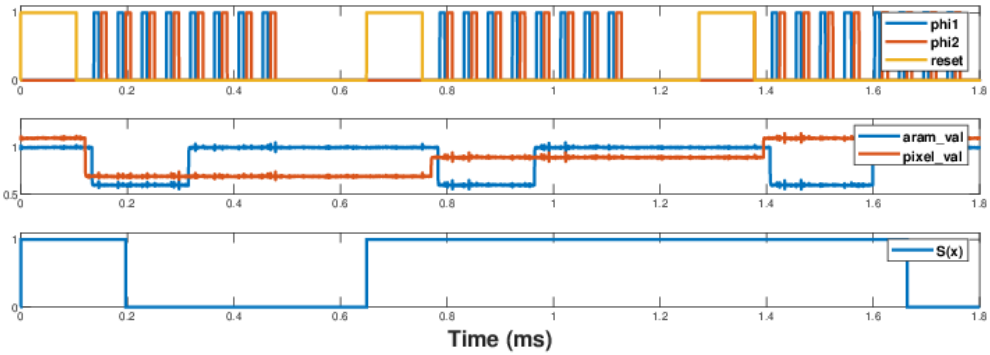

Figure 4.6: Transient test for the segmenter block of HOPBAS1K with the background model $B(x) = \{0.6, 0.6, 0.6, 0.6, 1, 1, 1, 1\}$ V, captured with the mixed-signal oscilloscope Tektronix MDO4034C. Digital signals are represented with digital values 1 or 0.

it was compared with the two first background samples with value 600 mV. The second test with pixel value set to 850 mV begins with the reset slightly after 0.6 ms. At that moment, the segmentation result resets to digital 1, and it remains there for the entire segmentation process. That means that all the background samples were outside the segmentation sphere, centered in 850 mV, leading to the segmentation output of foreground. Finally, the last iteration uses 1.05 V for the input pixel value. In this case, it is not until the sixth cycle that the segmentation output goes to zero, as the 1 V background model samples begin to be compared with the input value at the fifth cycle.

This test was repeated for the entire input range of the pixel value. The results are shown in Figure 4.7, where the segmentation result after the eight clock cycles is plot as a function of the input pixel value. Ideally, as the background model consists of two sets of four samples with value 600 mV and 1 V, it should show a background output on the regions centered in these values. However, as it can be seen in the figure, these regions are shifted to the right. The reason of that is the additional source follower placed in the path from the input of the segmenter and the circuit that calculates the absolute difference to compensate the effect of the source follower implemented at each memory cell of the ARAM, as explained in Section 3.2.4.

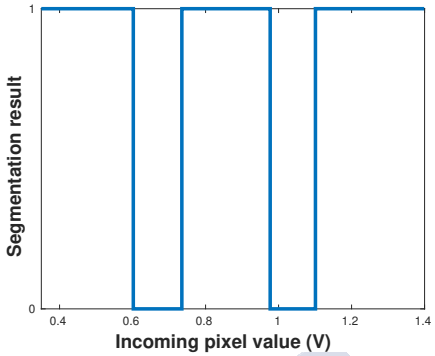

Figure 4.7: Segmentation result as a function of the incoming pixel value for the background model $B(x) = \{0.6, 0.6, 0.6, 0.6, 1, 1, 1, 1\}$ V captured with an analog input of the Photon unit.

4.2.2 Random Number Generation

Another important circuit that was placed on the chip periphery to be individually tested is the random number generator (RNG). This test circuit requires two non-overlapped clock signals, *phi1* and *phi2*, and at each clock cycle it outputs one analog random voltage in the range of 0.35 V to 1.35 V and a random digital bit. To test the circuit a random stream of 500 000 values was extracted with a clock frequency of 50 kHz, which is the maximum speed required for the chip operation. A temporal stream for both output types can be seen in Figure 4.8.

As this algorithm strongly relies on the random number generation for the background model update, it is of great importance to get an unbiased, or little biased, output stream. This can be assessed for the analog output by checking the output distribution shown in Figure 4.9. This histogram shows that the output values are distributed through the entire output range. For the digital output the bias of the 500000 samples is 51.6 %, which seems very reasonable for such a simple design, and for providing high enough quality of foreground segmentation on a HO-PBAS CMOS vision sensor chip.

4.2.3 Analog to Digital Converter

The vision chip incorporates a column-level 8-bit single slope Analog to Digital Converter (ADC). Column ADCs share the voltage ramp, which is fed from the outside, and the 8-bit counter, implemented on-chip in the synthesized control block. One of the main difficulties of

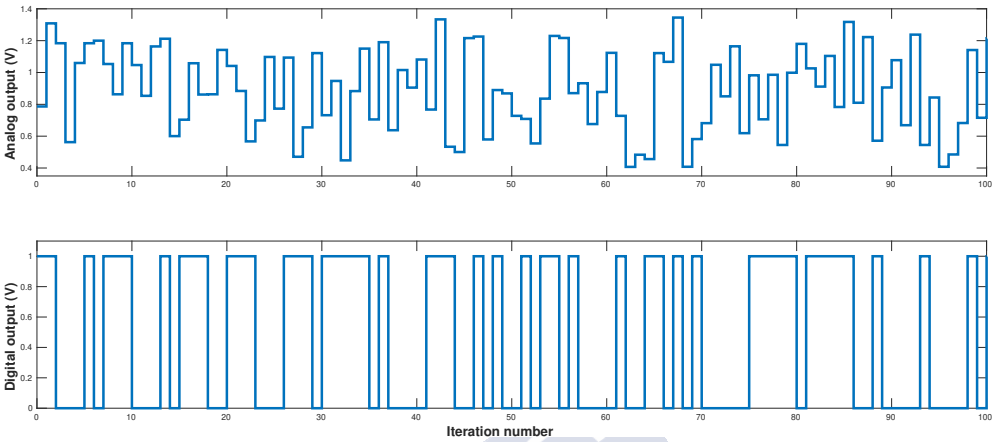

Figure 4.8: Experimental RNG temporal output stream of the digital an analog.

an SS-ADC is to synchronize the counter with the voltage ramp, as deviations from that will lead to offset conversion errors. In the case of this work, some parts of the internal control are replicated on the FPGA, which is also the responsible for generating the clock signals. Thereby, as the external DAC is controlled by the FPGA, the generated voltage ramp will be synchronized with the counter.

In order to test the SS-ADC without having to replicate all the circuitry (control, counter and the ADC itself), one of the ADCs used for the image conversion was provided with a multiplexer at its input, allowing to select between the pixel values of that column or an external voltage. This external input was connected to one Photon's DAC and a voltage sweep from 0.35 V to 1.35 was configured, setting the voltage ramp from 0.4 V to 1.35 V. This test was repeated 25 times and the result of the average response is shown in Figure 4.10, where the great linearity of the system can be appreciated. However, the conversion result suffers from some non-idealities at the last 100 mV of the input range, which can be caused by the resetting time of the voltage ramp and because the ADC's comparator is working at the end of its operation range, where non-linearity errors are more likely.

4.3 System Test

In order to test the system the camera prototype of Figure 4.11 was built. It adds to the experimental setup a custom made 3D printed case to hold the 35 mm lens, which features

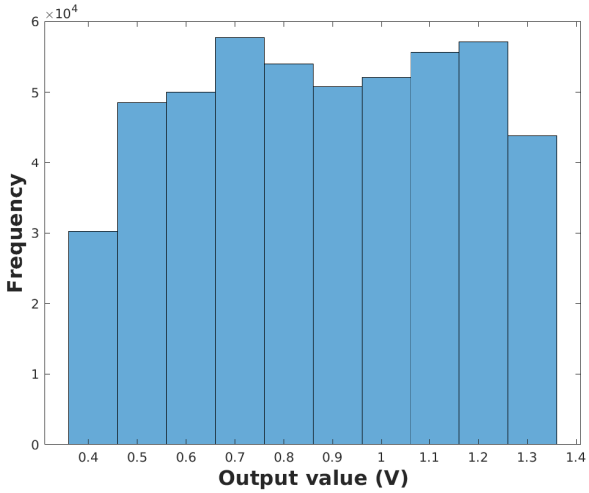

Figure 4.9: Histogram of 500 000 samples obtained from the analog output of the RNG implemented on the HOPBAS1K, captured with an analog input of the Photon unit.

an adjustable diaphragm and focus distance. As mentioned before, for theses tests the FPGA is used to feed the clock signals onto the chip and to read out the captured image, as well as the segmentation result. Then, these data are sent to a PC through a serial port implemented on the FPGA. Configuration registers of the chip are written from the Photon microcontroller, which also communicates with the PC through a different serial port. Thus, the PC software reads the data coming from the chip through one serial port and writes the chip configuration to the Photon unit through a different serial port.

4.3.1 Image Capture

An example of captured images by the CMOS vision sensor chip is shown in Figure 4.12. These images are converted using the 8-bit column-parallel SS-ADC and read out by the FPGA. The left image shows an oscilloscope front button panel. Even with this low resolution, 24×56, image it is possible to differentiate four different buttons with their shadows. However, it is not possible to identify the label over them.

This image shows a number of brighter pixels, which is a static effect that affects randomly to different pixels, and varies from one chip to other. This effect causes that some pixels show

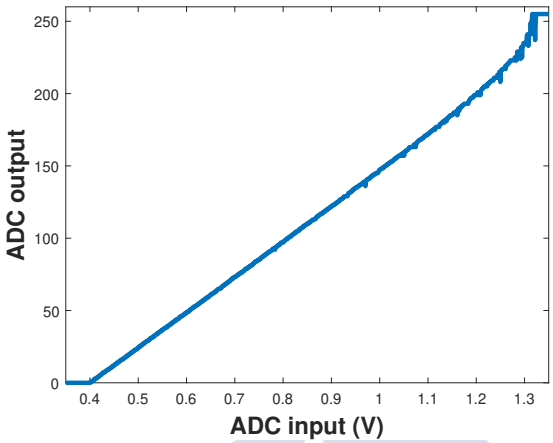

Figure 4.10: Full range ADC output with the voltage ramp configured from 1.35 V to 0.4 V.

the captured value plus a certain offset, which depends on the array position. In the right image of Figure 4.12 a captured frame with the lens of the camera covered is shown. There, it can be seen how the affected pixels are the same as in the left image of the same figure. The reason of this problem was not detected during the tests of HOPBAS1K. However, this issue was further analyzed in the tests of HOPBAS10K, the second chip designed in this thesis, and the conclusions extracted from this analysis will be explained in Section 5.2.2.

Another important problem that arose in this test can be seen in the top right corner, where a white semicircle covers this part of the image. This effect occurs in all the chips with the same magnitude. Also, tests were performed repeating the image capturing at different temperatures, ranging from 20 °C to 80 °C, with the size of the semicircle or its intensity not varying. After these tests, a deep review of the chip layout was carried out, looking for some error in the area of the semicircle not detected by the Layout-Versus-Schematic (LVS) tool. After these checks, a short circuit in the padring was detected, caused by a misplaced metal line that slightly covered the abstract cell of a pad. As in this technology the pads are provided as abstract cells, the LVS did not detect the connection of VDD and GND. As this short circuit was placed near the affected area, it was a possible reason of the problem. To assess if this was the reason of the non-working corner of Figure 4.12 some naked dies were sent to be treated by a dual beam scanning electron microscope. The dual beam electronic microscope offered electronic images with a large zoom and also a 30 kV focused ion beam

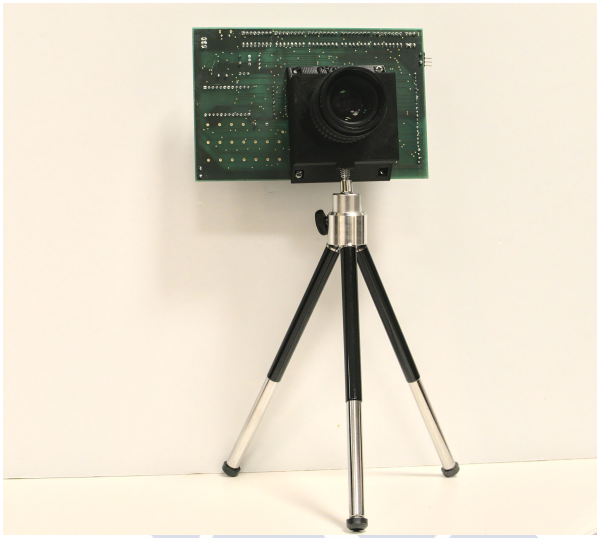

Figure 4.11: Camera prototype with a 35 mm lens. HOPBAS1K is below the custom-made 3D-printed holder of the lens.

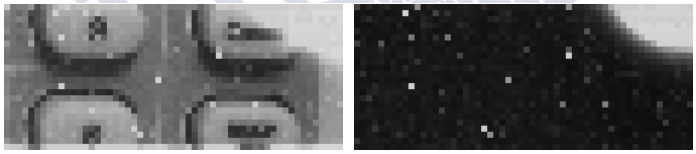

Figure 4.12: Images taken with the HOPBAS1K with exposure time of 3 ms. The left image shows a button panel of an oscilloscope, and the right image is taken with the lens covered.

of Ga. This ion beam was applied into the area where the short existed, as shown in Figure 4.13, removing the short circuit without affecting other parts of the chip. This could be tested by checking that the circuitry that worked before continued to work after the post-processing, and companion circuitry from other project on the same die that did not work properly before they did operate properly afterwards.

Unfortunately, after this post-processing, the non-working corner of the vision sensor remained. That led to a second revision of the layout, looking this time for differences of that corner with other parts of the array. After this revision, it was concluded that the only possible part that somehow could be affecting the non-working corner was the control signal generation block. Some of the generated control signals probably coupled with some internal node

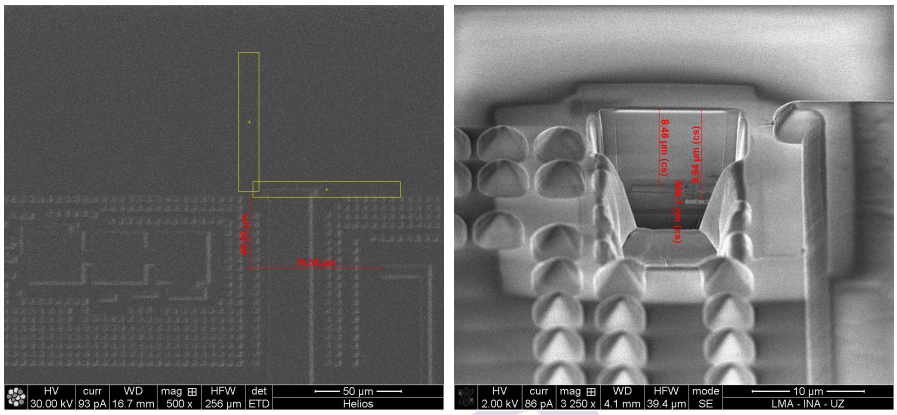

Figure 4.13: Images of the chip taken with an STEM microscope while removing a short circuit in the padring that once was thought to be the reason behind the non-working part in the top right hand side corner of HOPBAS1K (see Figure 4.12).

of the closest pixels, provoking that effect. As it will be seen in the second version of the chip, shown in Chapter 5, this was verified to be the reason of the problem.

4.3.2 Segmentation Result

Even if the imager suffers from different non-idealities is clear enough to analyze the scene. Furthermore, as addressed in [31], the HO-PBAS shows a great robustness under inter-pixel variation. As the per-pixel background model samples are captured and processed by the same circuitry, they are affected by the same non-linearities and second-order effects, which highly reduces their impact.

To check the segmentation result the same setup as in the previous section was used. In this case, after reading the output image, the chip sends the segmentation output through a digital pad. This is done once the image is processed in the array, and the result stored in the frame buffer. Then, the row and column decoders connect the digital result stored in each pixel with the output pad. This result is read by the FPGA and sent to the PC through the serial port.

Two frames of a recorded video performing this test are shown in Figure 4.14. Figure 4.14 (a) shows the input frame without any foreground object, being the background the same button panel as in the previous Section. The correspondent segmentation output, which should be all black as it is a pure background image, is displayed in Figure 4.14 (b). A frame with a

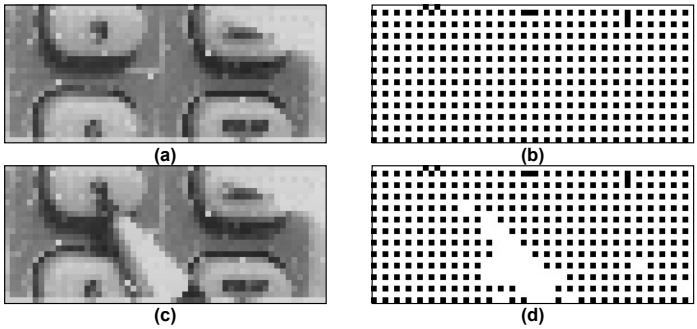

Figure 4.14: HOPBAS1K full output for: (a) scene with no foreground objects; (b) corresponding output for (a); (c) scene with a foreground object; (d) segmentation result of (c).

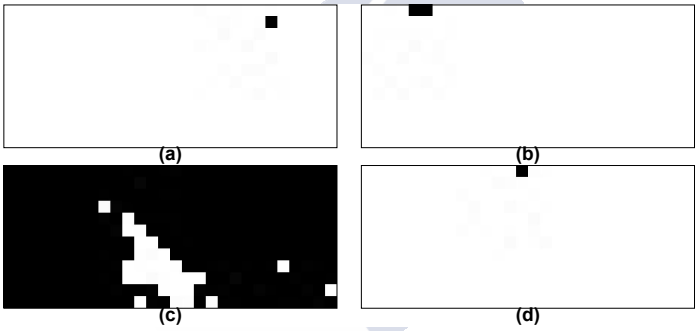

Figure 4.15: Segmentation output of Figure 4.14 (d) split in four parts. Each part corresponds to the output of each pixel of the PEs.

pencil tip as a foreground object and its correspondent segmentation output are displayed in Figure 4.14 (c) and (d), respectively. As seen in the figure, there is a clear problem with the output, showing a $\left(\begin{smallmatrix}1 & 1\\ 1 & 0\end{smallmatrix}\right)$ pattern for the background areas, and only changing the '0' of this pattern for the foreground areas, as it can be seen in Figure 4.14 (b) and (d). If the output is split in four parts, where each part represents the output for each individual pixel of the four pixels group that form the PEs, Figure 4.15 is obtained. There it can be seen how only one of the pixels of each PE is working properly.

This chip was provided with as many test structures and signals as possible, with the main constraint being the number of available output pads. Some of these test signals were the internal signals that connect the pixels with the PU in the PE placed at the bottom right corner of the array. From these testing signals it was possible to check that the output of

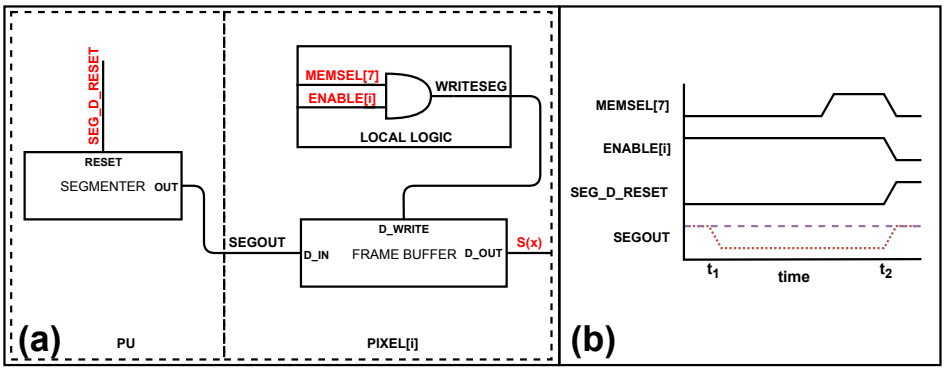

Figure 4.16: Explanation of the control signals overlap that cause that only one of the four pixels group stores correctly the segmentation result of HOPBAS1K.

the segmenter was the correct one for all of the four pixels in the PE, and how it changed accordingly to the input pixel value and the background model samples. Thus, the source of the problem had to be related with the frame buffer that stores the result before it was read out, and it could be at the writing process, due to some signal glitch while the value was stored or during the readout process. Unfortunately, no additional test signals were available to check these hypothesis. Also, as the control signals generation was tied to the on-chip synthesized digital control block, it was not possible to change it. The only possible change in that sense was to change the speed of the readout process by modifying the frequency of the clock signals generated by the FPGA, with these tests not showing any useful information. Therefore, the only possibility to detect the problem source was through deduction and simulation.

After a series of post-layout simulations, and simulations where parasitic devices were manually included in nodes where the parasitic extraction tool could not properly model these effects, were performed, the cause of the problem was detected. The reason of the chip not correctly providing the correct result for three of the four pixels groups that form the PE was an overlap between the control signals that reset the segmenter block and the one that writes the digital input into the frame buffer. The circuits where this signal overlap occurs can be seen in Figure 4.16 (a), which shows how the $WRITESEG$ signal is generated with the *AND* operation of $MEMSEL[7]$ and $ENABLE[i]$. As the time diagram of Figure 4.16 (b) shows, at the same time instant that these signals go to low the SEG_D_RESET signal used to reset the segmenter in order to be used by the next pixel in the PE goes to high, provoking that regardless of the resulting value of $SEGOUT$ for the selected pixel the bit that is finally

written will be always high. This is not the case for the last processed pixel in the PE, as the segmenter is not reset again, and this explains why only the result of the last selected pixel is properly written into the frame buffer.

4.4 Conclusions

This chapter covered the testing of the first prototype of the proof-of-concept 24×56 CMOS vision sensor for foreground detection in standard 180 nm CMOS technology, HOPBAS1K. In these experiments the individual test blocks were tested with favorable results. Although some blocks suffer from non-idealities or second-order effects, these are typically expected in hardware analog designs and they not compromise the chip correct behavior.

Regarding the system test, results were shown for both the imager itself and also for the segmentation result. Imager tests show that the vision part of the chip and the designed architecture for the ADC worked as expected. However, two vision artifacts arose during the tests: some random pixels offer a captured value with an offset with respect to the others and the top right corner pixels of the array saturate independently of the exposure time and external conditions. The first issue might be related to the use of a standard CMOS technology not specialized in vision sensors and, as explained in Section 4.3.1, does not compromise the algorithm operation. The non-working corner problem source was identified to be an electrical coupling with the control signal generation block, and it will be studied more deeply in Chapter 5.

Finally, the tests for the segmentation results discovered a severe problem in the output binary map. Regardless of the input image, three of the four pixels of each PE showed a segmentation result of foreground. The remaining pixel was able to classify background or foreground input values with respect to its background model, and it maintained this background model properly. The architecture and the system idea seem to be alright, as internal testing signals show that the segmenter processes correctly all the pixels and the problem is related to the writing process of these results. However, without all the pixels working properly the diffusion mechanism can not be tested and it can not be assured that all the desired features of the HO-PBAS were accurately implemented. Therefore, a second version of the chip was designed to solve these issues. In the next chapter the design process and the experimental results of this new iteration will be shown.

CHAPTER 5

HOPBAS10K DESIGN AND TEST

Introduction

After HOPBAS1K was tested, some design problems and other minor issues were detected. In this section possible solutions are proposed. Then, they are implemented on the second version of HOPBAS1K, named HOPBAS10K, with a spatial resolution of 98×98 pixels. This chip was fabricated in standard 180 nm CMOS technology, and its experimental results are shown in this chapter.

5.1 HOPBAS10K Design

In Chapter 4 the experimental results of HOPBAS1K for the individual test blocks and for the HO-PBAS CMOS vision sensor array were explained. These tests showed that the architecture and the main functionalities of HOPBAS1K behave as expected. However, some parts of the design did not work properly, being the most important ones:

- Only one of the four pixels of each PE stores correctly its segmentation result into the frame buffer. This effect, caused by a signal overlap, was explained in Section 4.3.2. The proposed solution was to delay the reset signal of the segmenter unit one clock cycle.
- The top right corner of HOPBAS1K is in saturation regardless of the exposure time or the illumination conditions. Although this problem's source was not perfectly determined, we hypothesized that this is caused by interferences from nearby digital signals

and HOPBAS10K must offer some flexibility for the control signals generation to explore and solve it.

- Some pixels of the array provide an output with a constant offset, independently of the system configuration. Similarly to the previous problem, this issue will be further explored in this new iteration.

Also, during the tests of HOPBAS1K, some other minor issues appeared. Those are:

- Supply voltages were not properly connected to their pads, with less vias than those recommended from the top to the bottom layer metals that are connected to the power ring. This added a series resistance on the power lines and, even if it did not cause any visible problems, due to the low current consumption of HOPBAS1K and because the power supply pads were redundantly placed on different parts of the padring, it should be corrected and taken into account. Furthermore, the second iteration of the CMOS vision sensor chip designed in this thesis, HOPBAS10K, will feature ten times more pixels, with the corresponding increase in power consumption, and this issue might become an important problem.

- The per-pixel output multiplexer, included to offer the option of selecting which magnitude is connected to the per-column ADC, did not work when an output different from the captured image was selected. After analyzing the pixel schematics, the source of this problem was discovered to be that the row enable signal, coming from the row decoder, was only connected to the enable input of the frame buffer. When any of the other outputs was selected, all of the pixels connected their outputs at the same time to the column line, making the resulting signal totally useless. The solution for this was to add an additional enable transistor to the multiplexer controlled by the row enable signal.

- HOPBAS1K provides a poor segmentation quality while working at the design processing clock frequency. If this frequency is reduced from 400 kHz to 100 kHz, the chip behaves as expected. Even when a loss of performance can be expected when a CMOS chip is fabricated, due to parasitic capacitances non-properly modeled by post-layouts simulations tools, or due to off-chip parasitic effects, in the case of HOPBAS1K this effect is of a great importance in situations where the comparator designed in Section 3.2.2 had to operate with voltages below 0.5 V. Post-layout simulations repeated for this

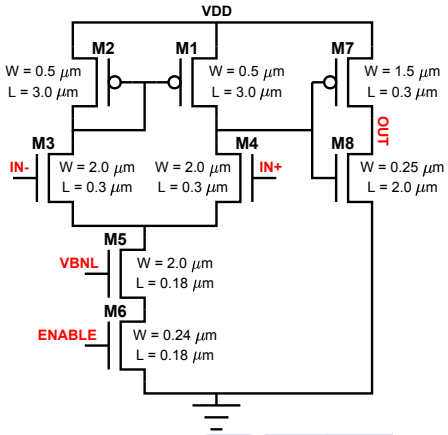

Figure 5.1: Schematic of the comparator used in HOPBAS10K.

comparator showed a big decrease in the comparison speed for the input range 0.35 V to 0.5 V. This problem was solved by redesigning the comparator, focusing on achieving a comparison speed fast enough in the entire voltage operation range to solve this issue. The resulting comparator schematic is shown in Figure 5.1, and it features similar characteristics of the one designed for HOPBAS1K, but for the entire operation range and in a slightly smaller area.

HOPBAS10K was fabricated in a silicon area of 5×5 mm^2, resulting in an array of 98×98 pixels. Thus, apart from the problems detected in the tests of HOPBAS1K, some modifications needed to be performed into the system to adjust to the new pixel array requirements. One of the first modifications was to provide each column with column drivers between the control signal generator and the pixels, as for the bigger resolution it is required to buffer those signals into the pixel array. Also, similarly to the control signals coming from the digital control block, the output to the RNG was provided with one analog voltage buffer for every four columns to reduce the impact of the increased output load. This voltage buffer was implemented with eight of the voltage buffers designed in Section 3.2.2 connected in parallel.

Another important adjustment into the system designed for the second version of the HOPBAS CMOS vision sensor designed in this thesis was in the control blocks. In this case, both the row and column decoders were manually designed, featuring a 7-bit input and 98 one-hot outputs with driving capabilities adjusted to the number of columns, and the readout control

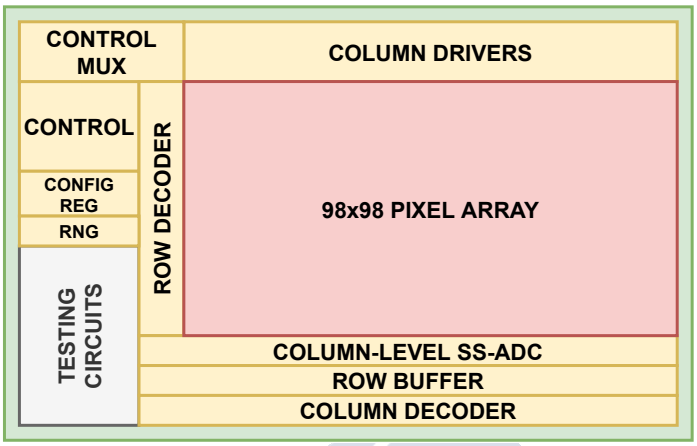

Figure 5.2: Floorplan of the second version of the HO-PBAS CMOS vision sensor chip designed in this thesis, HOPBAS10K.

block changed accordingly to the new size.

Regarding to the analog-to-digital conversion, the new array size imposes new temporal constraints if the same readout speed as in HOPBAS1k must be achieved. Simulations with faster conversion speed to reach the same image output rate were executed, proving that this can be accomplished without any additional modification. That was also experimentally tested by increasing the ADC clock frequency for HOPBAS1K with the result of the converted images experimenting no noticeable effect.

The last modification performed into the system was to add the option of using external control signals. The bigger silicon area makes the number of available pads no longer a constraint, allowing us to feed all the required control signals from the outside. Thus, a digital multiplexer was included before the column drivers, providing the option to choose between the internal control signals or the ones coming from the outside. This additional degree of freedom in the system operation permits to test different changes into the algorithm workflow, as well as to correct possible mistakes made during the digital control design.

These proposed system modifications were implemented in HOPBAS10K, a 98×98 pixels array CMOS vision sensor, reusing the same architecture as in HOPBAS1K. The chip floorplan is shown in Figure 5.2. It can be seen that the only difference with respect to HOPBAS1K, apart from the blocks position on the chip, is the addition of the control multiplexer, which based on an external signal selects between the internal or the external generated con-

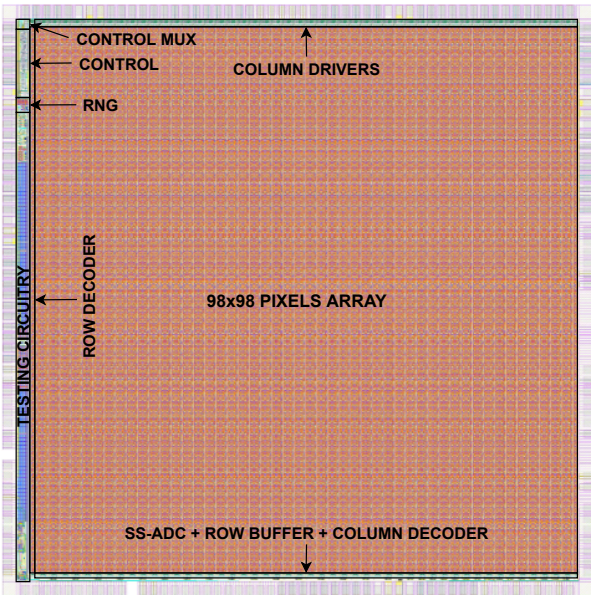

Figure 5.3: HOPBAS10K CMOS vision sensor layout with its main blocks labeled.

trol signals. Also, the labeled chip layout is shown in Figure 5.3.

5.2 Experimental Results

5.2.1 Experimental Setup

The experimental setup used for the test of HOPBAS10K is very similar to the one used in Chapter 4 for HOPBAS1K. In this case the chip carrier needs to be changed, as the larger number of pads requires more pins to be wire-bonded to. Thus, the package used is the CPGA208, which features 208 pins whose correspondence with the chip pads can be found in Appendix B. Figure 5.4 shows the 5×5 mm^2 die bonded to the package.

Using the same strategy as in the testing experimental setup for HOPBAS1K, the designed PCB features a ZIF female socket compatible with the CPGA208. This socket is placed in the center of the PCB and covered by a custom-made 3D plastic case that supports the lens, as shown in Figure 5.5. The rest of the PCB components are the same as in the test of HOPBAS1K, with the exception of the top female headers, which are connected to the external

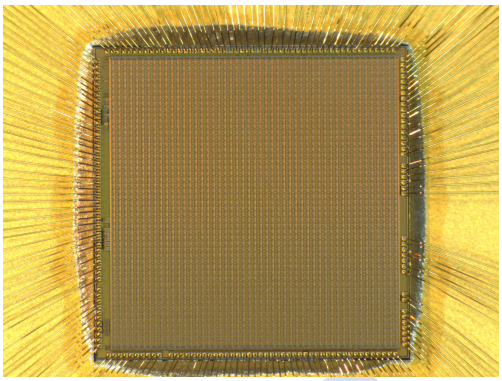

Figure 5.4: HOPBAS10K CMOS vision sensor microphotograph.

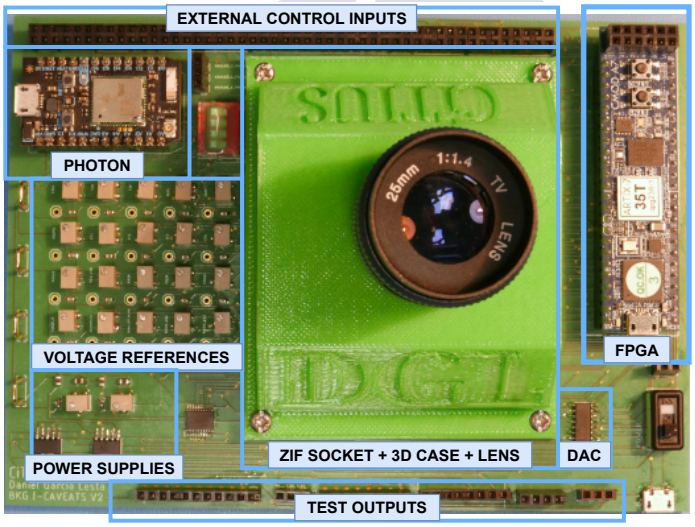

Figure 5.5: HOPBAS10K CMOS vision sensor test PCB with the main parts labeled. HOPBAS10K is in the socket below the custom-made 3D-printed holder of the lens.

control signals inputs. These inputs could have been placed as a female socket compatible with a FPGA board, similar to the FPGA used for the clock signals generation of the internal control, or with a standard connector to a different platform. However, as the internal control was expected to work and only have to use the external control for specific tests, it was preferred to make the PCB smaller and put these connections with a small footprint.

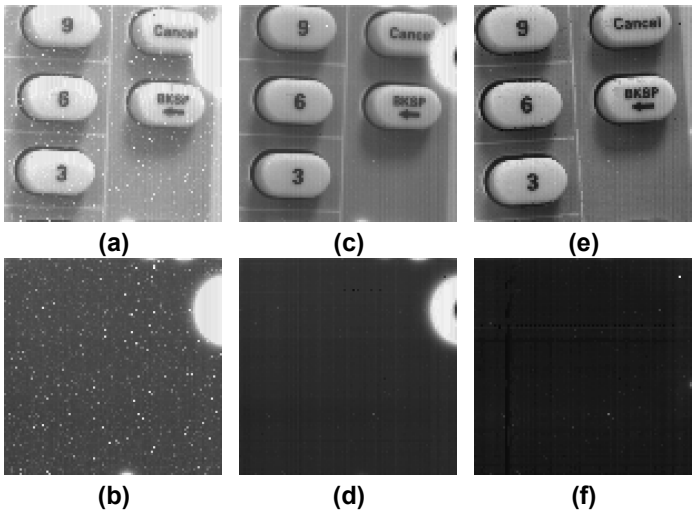

Figure 5.6: Captured images from HOPBAS10K: (a) image suffering from pixel mismatch and a non-working zone near to the digital control block; (b) same case as (a), with the lens covered; (c) pixel mismatch effect reduced by decreasing the photodiode reset voltage to 500 mV; (d) same as (c) with the lens covered; (e) clear image taken, with the effect of the issues previously explained reduced, by decreasing the photodiode reset voltage and turning off the processing and readout blocks while the image is being captured; (f) same as (d), with the lens covered.

5.2.2 Captured Images

The first test performed with HOPBAS10K was to analyze the quality of the captured images. Images from HOPBAS1K suffer from an always saturated corner and from some pixels outputting an offset value independently of the captured image. In this first test the images are captured with the internal control selected and with all the system blocks working (algorithm processing, ADC, readout and RNG). Figure 5.6 (a) shows an image of a button control panel of an oscilloscope, acquired with an exposure time of 2 milliseconds. It can be seen that, before applying any changes to the system configuration, the same image problems of HOPBAS1K appear in this new version of the chip. These issues can be better seen in Figure 3.48 (b), where the image is taken in the same conditions but with the lens covered. As with HOPBAS1K, some pixels give their outputs with an offset value, whose spatial distribution varies with the selected chip. This effect points toward a mismatch problem in the pixel array that not appeared in Monte Carlo simulations.

Exploring different causes and solutions for the problem of the pixels mismatch, it was discovered that by decreasing the reset voltage of the photodiode, the impact of the pixels mismatch on the pixels offset was highly reduced, as it can be appreciated in the images of Figure 5.6 (c) and (d), where the photodiode reset voltage was set to 500 mV. This dependency with the photodiode reset voltage allows us to narrow down the source of the problem. If the origin of the problem were located in the 3T-APS circuit, the offset contribution of the mismatch would be eliminated by the CDS circuit. On the other hand, if the mismatch were placed after the CDS unit, the change on the photodiode voltage reset would have no impact on the output value, as the CDS circuit output is proportional to the difference of the reset value and the voltage level of the photodiode after the integration time, and does not depend on the their dc level. Thus, the conclusion is that the source of the problem must be at some part of the CDS circuit and probably related with some charge injection or clock feedthrough effect. Unfortunately, there are not any test signals at this part of the pixel to verify that through experimentation.

Regarding the non working zone that appears at the top right hand side of the captured image on Figure 5.6, it has to be noted that this problem appears again near the control unit that generates the digital control signals, as in the HOPBAS1K. As HOPBAS10K solved the problem with the pixel output multiplexer, it was possible to access and visualize the sample number three of the background model and, during the tests of this feature, it was noticed that right after the reset, when the background model is formed with the first frame captured, the image formed by this sample was totally fine, with the aforementioned area working as expected. This result shows that the source of this problem is not a fabrication defect, and that it has to be caused by some electrical coupling. The only difference from the first frame captured and the following ones is that the only control block operating is the one responsible for the image capture, and all of the others remain idle (see Figure 3.46).

To verify that the problem is caused by an electrical coupling with these digital signals, a test was performed feeding the control signals from the outside. These signals were generated using two synchronized CMOD A7 35T FPGAs that replicate the internal control. Results show that with the external control the non working area near the control block disappears, but this problem appears at different places of the array. After studying the HOPBAS10K layout it was concluded that these new non working areas arise in the vicinity of the entry points to the chip of some external control signals. Thereby, the only possibility to eliminate this problem would be to electrically isolate the pixels array from the sources of electrical noise. This is

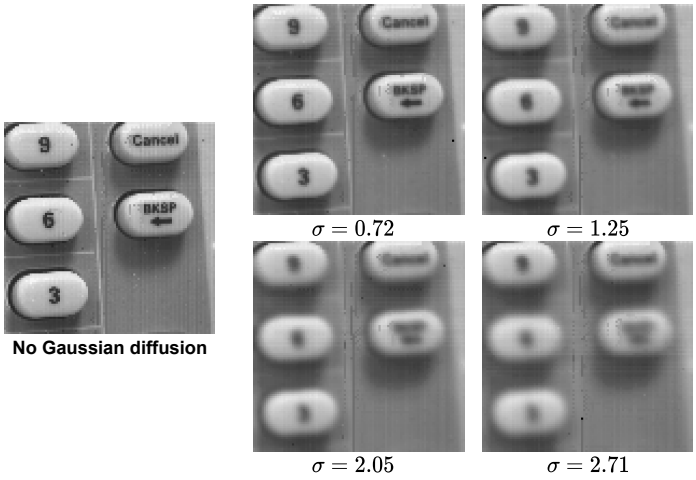

Figure 5.7: Low-pass filtered image captured and processed with HOPBAS10K for different values for the σ parameter of the Gaussian distribution (see Equation (3.8)).

not an option for HOPBAS10K and a different approach was adopted. As the captured image is damaged during the exposure time, a test was performed modifying the operation scheme by delaying the image capture until all the processing blocks have finished their tasks. This clearly affects the chip performance, as some of the parallelization is destroyed, but allows to use all the information gathered by the vision sensor. Results of these tests are shown in Figure 5.6 (e) and (f), where the modified external control signals were fed into HOPBAS10K.

Another important feature that could not be tested in HOPBAS1K, due to the design error in the pixel output multiplexer, is the performance of the Gaussian blurring. This block implements a low-pass filter using a Gaussian diffusion network, with in-pixel circuitry, through charge sharing between exchange capacitors with each neighbor and a capacitor that holds each pixels' value. By modifying the number of cycles that the charge sharing is performed, the σ parameter of the Gaussian distribution can be controlled (see (3.8)). Results for these tests with different values of the σ parameter are shown in Figure 5.7. These tests were performed using the external control to avoid the non working corner, and the images are taken using the on-chip SS-ADC, which probably is the responsible for some image artifacts, such as Fixed Pattern Noise (FPN) that can be appreciated in Figures 5.6 and 5.7.

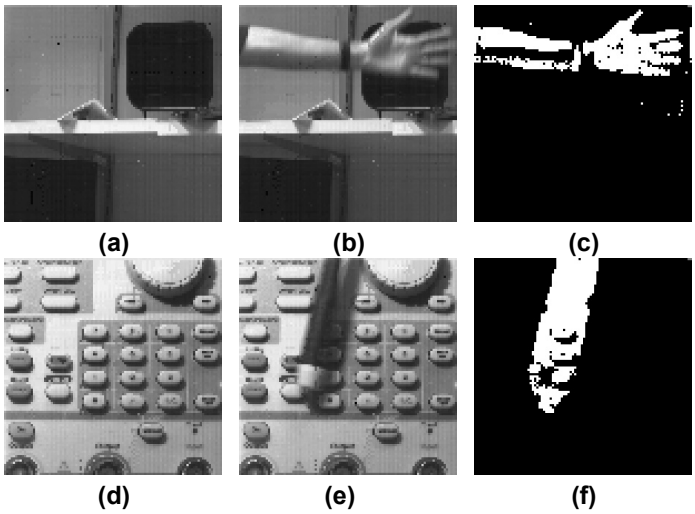

Figure 5.8: Segmentation results of the HOPBAS10K for two different input frames.

5.2.3 Segmentation Result

One of the main problems of HOPBAS1K is that only one of the four pixels of each PU is able to store its segmentation result. As explained in Section 4.3.2, our hypothesis is that an overlap between the signal that resets the segmenter unit, and the signal that writes the segmentation result into the frame buffer, is the responsible. HOPBAS1K does not incorporate the option of using external control signals, and it could not be tested if that was really the source of the problem. Thus, the control block of HOPBAS10K was changed to avoid this overlap.

To test the segmentation result the external control was used, avoiding the problem of the closest pixels to the control unit saturating regardless of the exposure time. As these pixels can not detect any input variation due to their saturated output, when the control signal is selected, this part of the array always classifies the input as background. The results for the segmentation are shown in Figure 5.8. It can be seen how the problem of HOPBAS1K was indeed the signals overlap previously explained, and how the HOPBAS10K is able to detect the input objects on the scene.

Figures 5.8 (a) and (d) show the background scene of two examples, whereas in (b) and (d) the foreground objects appear. For the first case, the segmentation result can be seen in Figure 5.8 (c). In that image it can be seen how a region on the bottom part of the arm is wrongly

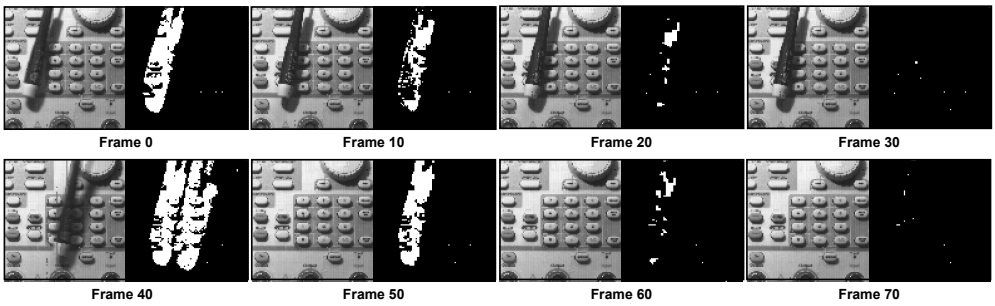

Figure 5.9: Diffusion mechanism test of HOPBAS10K. A screwdriver is segmented as foreground. After 30 frames, it is included into the background model. Then it is moved again and both the screwdriver and the silhouette left behind are detected as foreground objects. Again, after 30 frames, its ghost is included into the background mode.

classified as background, caused by the camouflage of the shadowed part of the arm with the background. Another important problem inherent to background subtraction algorithms can be appreciated in Figure 5.8 (f), where both the shape and the shadow of the screwdriver are detected.

Another major feature that, due to the segmentation result storage could not be tested from HOPBAS1K, is the diffusion mechanism responsible of incorporating static foreground objects into the background model after a certain time, which can be adjusted as a function of the scene. Figure 5.9 shows how a foreground object that is left static on the scene is gradually eaten up and incorporated into the background model. Then, the object is moved again and its ghost, i.e., the silhouette left behind, is also slowly included into the background model again.

Finally, a quantitative analysis of the segmentation performance was carried out to assess the decrease in the segmentation quality compared with the ideal algorithm response. This study was performed as follows: both the segmentation result and the captured image are read and stored by a PC. Then a software implemented in C++ and OpenCV reads the captured images and processes them with the software version of the HO-PBAS. Finally, the same performance metrics as the ones used in Chapter 2 are extracted, using the ideal algorithm response as the groundtruth.

Two different videos of 600 frames each one, recorded in laboratory conditions, were analyzed. In those videos different objects, such as an screwdriver or a human arm, appear in the scene. At some frames they stop for a long time, to assess the impact of the diffusion mechanism, and then they move again. Figure 5.10 shows an example of one of those videos,

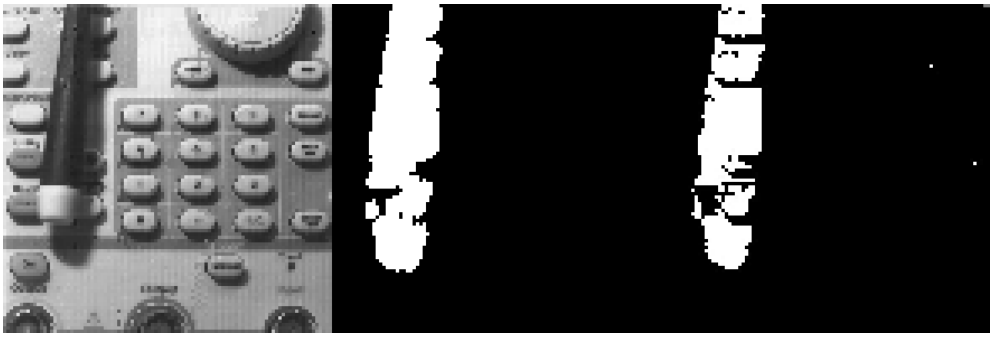

Figure 5.10: Comparison between the ideal segmentation result from the software version of HO-PBAS and the output of HOPBAS10K.

with the software and the HOPBAS10K segmentation results. It can be seen how the HOPBAS10K result misses more pixels than the ideal result. The overall analysis of the videos provides a result of Recall = 0.71, Precision = 0.91, and F-Measure = 0.80. The F-Measure result is dominated by the Recall figure of merit, which means that HOPBAS10K misses more true positives than the software version of the algorithm. On the other hand, it features a great precision. Thereby, it can be said that the output performance, in terms of segmentation quality, is in good agreement with the expected response. Furthermore, HOPBAS10K provides a great level of programmability for the algorithm parameters, and these results could be further improved as a function of the input scene.

5.2.4 Power Consumption Analysis

The last important HOPBAS10K's feature analyzed is its power consumption. This system was intended to be low-power, and for that the strategy of all-analog with power gating processing strategy was adopted. However, for this academic proof-of-concept system, a control design as simple as possible was implemented, trying to maximize the success of the chip operation. Thus, the power gating strategy was implemented simply by turning on all the processing circuitry during the background subtraction algorithm execution, and off when the pixels where just capturing the image.

Also, in order to get an accurate measure of the PEs power consumption, specific power supply lines for these blocks should have been included. However, in HOPBAS10K the core

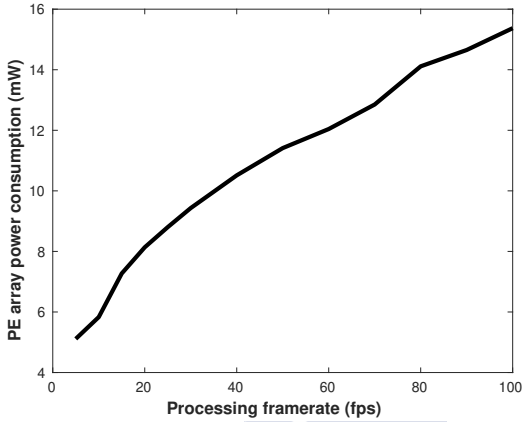

Figure 5.11: PE array of HOPBAS10K average power consumption as a function of the processing framerate. Each point of these experimental data is measured by averaging the input current for 30 seconds with the multimeter Keysight U1282A.

power supply is shared between all the analog circuitry, which includes circuits like as the ADCs and the RNG block. Thus, to get a power consumption estimation the following approach was applied: first, by using the external control, the current flowing to the analog voltage supply pads was measured with all the processing blocks turned off. Then, the same measurement was performed with the processing blocks turned on at different framerates. Finally, the power consumption was estimated making the difference of these two magnitudes. Results from this test are a fixed power consumption of 19.2 mW for the non-processing part of the chip, and the power consumption as a function of the framerate is shown in Figure 5.11. As expected, the power consumption of the array is an increasing monotonic function with the framerate, from the power consumption of the processing circuits, with an offset value that comes from the idle power consumption of the circuits turned off.

5.3 Comparison with The State-of-the-Art

To make a comparison of HOPBAS10K with similar work is not straightforward due to its peculiarities. HOPBAS10K is a specific-purpose IC that implements a top-ranked complex algorithm, differently from state-of-the-art solutions which apply more simple strategies for motion detection, such as Frame Differencing (FD), Double-Threshold Dynamic Background

Table 5.1: Comparison of HOPBAS10K with the state-of-the-art.

	[39]	[21]	[38]	[13]	HOPBAS10K
Process	180 nm 1P6M	350 nm 2P3M	110 nm 1P4M	180 nm 1P6M	180 nm 1P6M
Supply A/D (V)	0.8	3.3	1.2/3.3	1.2/0.8	1.8/3.3
Pixel Size (μm^2)	10×10	26×26	4×4	7.9×7.9	47×47
Fill factor (%)	20	12	49	33.7	2.9
Pixel array	132×104	64×64	640×480 (160×120)	256×216 (128×108)	98×98
Motion detection	FD	DT-DBS	DT-DBS	FD + BS	BS
Domain	Mixed	Mixed	Digital	Mixed	Mixed
Processing architecture	Pixel	Pixel	Column	Column	Group of pixels
Processing framerate (fps)	510	13	8	15	100
Processing power (μW)	74.4	33	344	2.36 (14FD+1BS)	15 372
FoM (pJ/pixel·frame)	35.6	620	2240	11.4	16 005

Subtraction (DT-DBS), or a combination of FD and BS. Due to the complexity of HO-PBAS, the long datapath imposes the use of low-error active circuitry to reduce the impact on the algorithm performance caused by the accumulated error after many subsequent operations.

Table 5.1 shows a comparison with recent work that perform foreground detection. All of them implement much simpler algorithms, and in some cases, with a down-sampled version of the captured image. Also, looking for smaller pixel pitch, some works apply column-wise processing. The impact of these considerations is clearly seen in the fill-factor, where HOPBAS10K has the lowest one with a great margin.

Another performance metric where HOPBAS10K loses is the power consumption, both in the total processing power consumption and in the Figure-Of-Merit, defined as the energy required for each pixel and frame. The large values of HOPBAS10K are explained as follows:

- The use of active circuitry for the operations required by the algorithm, instead of passive circuits.
- The basic power gating strategy, that did not optimize the activation or deactivation of specific blocks at each step of the algorithm execution.
- In this PhD, a reliable IC implementation was targeted. Thus, functional blocks, such as voltage buffers or comparators, were implemented with basic structures, differently from more advanced architectures, which could have offered a reduced power consumption.

5.4 Conclusions

HOPBAS10K was designed to solve the issues detected during the tests of HOPBAS1K and to increase its spatial resolution. As seen in this chapter, the problem of only one pixel of the PE properly storing the segmentation result was successfully solved. Also, other minor issues such as the one related to the pixel output multiplexer was also corrected, allowing the testing of the Gaussian diffusion network. The pixel mismatch effect on the captured image was highly reduced by decreasing the photodiode reset voltage. Additionally, the non-working zone affecting the pixels near to the digital control unit was identified to be caused by a signal coupling during the exposure time. With the use of the external control, this was fixed by changing the operation scheme.

Segmentation output tests, performed by comparing the chip output with the ideal HOPBAS, showed good results but with a reasonable performance decrease. Regarding the power consumption of the pixel array, working with a processing clock frequency of 400 kHz and processing at 100 fps, a pixel array power consumption of 15.3 mW is achieved, resulting in a figure of merit of 16 nJ/pixel·frame. Considerations such as the algorithm complexity and the basic power gating strategy must be taken into account to understand this large power consumption when compared with the state-of-the-art.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Conclusions

The main contributions of this thesis are: 1) the design of HO-PBAS, an adapted to hardware version of PBAS, one of the top-ranked background subtraction algorithms at the beginning of this PhD, and 2) the design of two proof-of-concept chips that implement this algorithm on the focal-plane with analog processing circuitry: HOPBAS1K and HOPBAS10K.

The algorithmic part of the thesis was about the design of the foreground detector that was used in HOPBAS1K and HOPBAS10K. Nowadays, deep-learning based foreground detectors outperform rule-based algorithms [25]. However, their implementation complexity makes them unfeasible for a focal-plane design. Furthermore, at the beginning of this PhD, heuristic algorithms dominated ranking contests, such as *changedetection*. PBAS was a top-ranked background subtractor with per-pixel operation, which made it a great candidate for this thesis. After a deep analysis of its main features and the relevance of each one on the algorithm performance, a hardware-oriented version was developed for a feasible implementation in the analog domain. As explained in Chapter 2, HO-PBAS introduced modifications such as reducing the background model size, linearizing the equations used in the algorithm datapath or removing median filters for the segmentation result post-processing. These modifications were included into the source code developed in C++ with OpenCV, showing that HO-PBAS outperformed PBAS for some video categories, and slightly improved PBAS results on average. To assess the impact of hardware non-idealities, software simulations with these effects were performed, proving the feasibility of the design.

A 1.6×3.2 mm^2 24×56 pixel array 180 nm standard CMOS vision sensor that implements

HO-PBAS was designed. This chip, named HOPBAS1K, features full-array background subtraction, implemented with an architecture of 4 pixels sharing a Processing Unit (PU), forming the Processing Element (PE). HOPBAS1K includes per-column 8-bit Single-Slope Analog-to-Digital Converters (SS-ADC), a custom-made mixed-signal Random Number Generator (RNG) and an on-chip digital synthesized control block. The outputs from HOPBAS1K are the 1-bit segmentation result and the 8-bit captured image, which is read through a parallel bus. Experimental results from HOPBAS1K tests showed the expected behaviour from the processing blocks, as well as from the ADC and RNG. Due to an error design the Gaussian diffusion block could not be tested in this chip. Finally, systems tests showed a pixel mismatch effect on the captured image, that was solved for HOPBAS10K, and that the pixels on the vicinity of the digital control unit provided a saturated output regardless of the input scene. An important problem detected during these tests was that the segmentation result was only well stored for one of the four pixels of each PE. After a deep analysis, it was concluded that the reason of this issue was the overlap between two signals in the PE, the one responsible of resetting the segmenter unit and the one that activated the write input of the segmentation result into the frame buffer.

After testing and analyzing HOPBAS1K, a new CMOS vision sensor was designed in an IC of 5×5 mm^2, featuring a 98×98 pixel array and, similarly to HOPBAS1K, a column-wise 8-bit SS-ADC. This new chip, named HOPBAS10K, has the option of using external control signals, which was very useful for testing it. The source of the imaging problems detected in HOPBAS1K was narrowed down and solved. Also, the problem of only one pixel of each PE storing properly the segmentation result was fully solved in this new iteration. This allowed to test the diffusion mechanism, an important feature of the algorithm, with good results. A quantitative analysis of the segmentation results showed a performance decrease with respect to the original PBAS close to the 20%, which is a reasonable value for a hardware mixed-signal implementation. Finally, HOPBAS10K implements a complex top-ranked foreground detection algorithm, and as seen during the power consumption analysis, that came with the cost of a big power consumption.

To implement at pixel level a state-of-the-art computer vision algorithm, and therefore, a top-ranked algorithm on benchmarks, datasets or current contests, is a huge challenge with two strong requirements: *i)* to redesign the algorithm to make it lightweight, while reducing as much as possible its performance: in this PhD done with the evolution from PBAS to HO-PBAS, and *ii)* a design at circuit level with techniques to optimize both the area and the power

consumption, that in this thesis for instance it has been the strategy of the PUs and PEs, and the power gating. Although the HO-PBAS algorithm is not currently at the top of the list in the *changedetection* contest, these ideas can be extrapolated to more recent algorithms, such as those based on deep learning.

Future Work

After analysing the main conclusions of this thesis, different options exist as future work. At system level, a new prototype could be designed optimising the PCB area. The experimental setup used in this thesis was designed with the purpose of testing all the chip outputs, and with reasonable size to be easily manipulated. A second iteration of the camera prototype should reduce the system size as much as possible. Also, all the chip interface should be implemented on the FPGA, without any microcontroller unit. This FPGA could be integrated on the main PCB, to further reduce the prototype size. Furthermore, different functionalities could be developed on the FPGA, such as segmentation images post-processing to increase the output quality (median filters, erosion-dilation operations...), or other high-level algorithms that use the HOPBAS10K's output. A few possible applications that can be developed are: a tracking by detection algorithm, an algorithm that calculates speed vector based on the tracking, or an always-on system that wakes up a commercial camera to take a high-resolution picture when an object bigger than a certain threshold is detected on the captured image.

Regarding the chip itself, many improvements could be developed. For instance, some issues detected during the tests of HOPBAS10K could be solved, such as the high power consumption, by redesigning the processing circuity with low power blocks or refining the power gating strategy, or the Fixed Pattern Noise (FPN), caused by the per-column SS-ADCs. Also, as the main functionalities of HOPBAS10K have been proved to work properly, some parts of the system could be redesigned to occupy less silicon area, improving with that the image spatial resolution, or the system could be redesigned in an advanced CIS technology, to further improve the image quality. An additional step would be to use non-standard technologies, such as back-side illuminated (BSI) CMOS image sensors, or 3D stacked CMOS ICs, to further improve the image quality by increasing the spatial resolution and the pixel fill-factor.

Additional functionalities could be integrated on-chip making use of the foreground detection result. An interesting option would be for example to build an analog-to-information converter, such as the one developed in [13], where the information obtained from the back-

ground subtraction is applied during the image readout to reduce the ADC power consumption and/or the data bandwidth. With this information, the background part of the image could be converted to the digital domain less frequently and with less bits than the foreground part, achieving with that a highly reduced power consumption and data bandwidth requirement.

APPENDIX A

HOPBAS1K

Source code

All the source code developed for HOPBAS1K can be found in the following repository:

`https://gitlab.citius.usc.es/thesis-dgl/hopbas1k`

HOPBAS10K pinout

#	Name	I/O	Functionality	Value
1	D	O	Testing point connected to the output of the block that calculates d(x) in a PU inside the PE array.	
2	P	O	Testing point connected to the input of the block that calculates p(x) in a PU inside the PE array.	
3	NEWP	O	Testing point connected to the output of the block that calculates p(x) in a PU inside the PE array.	
4-6	NC			
7	O_TO_SEG	O	Testing point connected to the input of the segmenter in a PU inside the PE array.	
8	VDD_PIX	I	Photodiode reset voltage.	500 mV
9	GND	I	Ground.	0 V
10	p_{max}	I	Maximum value of p(x).	800 mV
11	p_{min}	I	Minimum value of p(x).	600 mV

#	Name	I/O	Functionality	Value
12	VDDA	I	Analog voltage supply.	1.8 V
13	R_{min}	I	Minimum value of R(x).	420 mV
14	VOFF	I	Offset voltage of the processing blocks.	350 mV
15	VHOLD	I	Voltage connected to the ARAM input when it is not being written.	900 mV
16	V_{RAMP}	I	Voltage ramp used for the SS-ADC.	
17	VBNL	I	Low bias voltage for the NMOS transistors.	400 mV
18	VBP	I	High bias voltage for the PMOS transistors.	1.2 V
19	VCN	I	High bias voltage for the NMOS transistors.	650 mV
20-22	NC			
23	VCP	I	Low bias voltage for the PMOS transistors.	950 mV
24	MEM_TESTS	O	Testing analog memories placed on the array periphery output.	
25	TEST_AOUT	O	Analog output of the captured image.	
26	D_T_RNG	O	Digital output of the individual RNG implemented for testing.	
27	SEG_I_TEST	O	Output of the individual test block for the segmentation.	
28-32	D_TEST[5:0]	I	Digital inputs for the individual test blocks.	
33	VDD	I	Digital voltage supply.	1.8 V
35	E_TEST	I	Enable signal for the test blocks.	
36	D_O_RNG	O	Testing point connected to the digital output of the RNG	
37	SEG_TEST	O	Testing point connected to the output of the segmenter in a PU inside the PE array.	
38	MUXOUT	O	Output of the digital multiplexer implemented in the internal control signals generation.	
39	PHI2ADC	O	Control signal of the SS-ADC, connected to the outside for the synchronization with the DAC.	
40	SEGOUT	O	Segmentation result.	
41-48	OUTPUT[0:7]	O	Captured image converted by the per-column SS-ADC.	
49	CLK_READ	I	Readout clock.	2 MHz
50	CLK_RNG	I	RNG clock.	50 kHz
51	CLK_ADC	I	ADC clock.	3 MHz
52	CLKN	I	Inverted processing clock.	100 kHz
53	CLK	I	Processing clock.	100 kHz
54-57	NC			

#	Name	I/O	Functionality	Value
58	VDD3.3	I	I/0 Voltage supply	3.3 V
59	CLK_COM	I	Control input clock	500 kHz
60	DATA_IN	I	Data signal for HOPBAS1K configuration.	
61	RST_COM	I	Reset signal for the HOPBAS1K configuration.	
62	RESET	I	HOPBAS1K global reset.	
63	NC			
64	1.8V	I	Analog voltage supply.	1.8 V
65	VSS	I	Ground.	0 V
66-83	NC			
84	A_O_RNG	I	Testing point connected to the analog output of the RNG.	
85	ET1	I	Reference voltage used by the RNG.	750 mV
86	ET2	I	Reference voltage used by the RNG.	770 mV
87	ET3	I	Reference voltage used by the RNG.	790 mV
88	VSS	I	Ground.	0 V
89	VDDA	I	Analog voltage supply.	1.8 V
90	ET4	I	Reference voltage used by the RNG.	810 mV
91	VREFII	I	Reference voltage used by the RNG.	1.35 V
92-94	A_TEST[1:3]	I	Analog voltages used for the inidividual test blocks.	
95	OUT_DCALC	O	Output of the individual test block that calculates d(x).	
96	O_RUPDATE	O	Output of the individual test block that calculates R(x).	
97	O_PUPDATE	O	Output of the individual test block that calculates p(x).	
98	A_T_RNG	O	Analog output of the individual RNG implemented for testing.	
99	ARAM_VAL	O	Testing point connected to the output of the ARAM in a PU inside the PE array.	
100	R	O	Testing point connected to the output of the block that calculates R(x) in a PU inside the PE array.	

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

APPENDIX B

HOPBAS10K

Source code

All the source code developed for HOPBAS10K can be found in the following repository:

`https://gitlab.citius.usc.es/thesis-dgl/hopbas10k`

HOPBAS10K pinout

#	Name	I/O	Functionality	Value
1-6	ROW[6:1]	I	External control input: Row selector.	
7	GND	I	Ground.	0 V
8	VDDA	I	Analog power supply.	1.8 V
9	ROW[0]	I	External control input: Row selector.	
10	S	I	Internal/external control selector.	
11	CLK_RNG	I	RNG clock.	50 kHz
12	CLK_READ	I	Readout clock.	2 MHz
13	CLK_ADC	I	ADC clock.	3 MHz
14	CLK	I	Processing clock.	400 kHz
15	CLKN	I	Inverted processing clock.	400 kHz
16	DIG_RNG	O	Digital output of the RNG.	
17	MUXOUT	O	Internal control signal generator multiplexer output.	
18	RESET	I	HOPBAS10K global reset.	
19	NC			
20	VDD	I	Digital power supply.	1.8 V

#	Name	I/O	Functionality	Value
21	GND	I	Ground.	0 V
22	DATA_IN	I	Data signal for HOPBAS10K configuration.	
23	RST_COM	I	Reset signal for the HOPBAS10K configuration.	
24	CLK_COM	I	Control input clock.	500 kHz
25	E_RNG_TEST	I	Enable the output tests from the RNG.	
26-32	COL[0:6]	I	External control input	
33	VDD	I	Digital power supply.	1.8 V
34	VSS	I	Ground.	0 V
35	ET4	I	Reference voltage used by the RNG.	810 mV
36	ET3	I	Reference voltage used by the RNG.	790 mV
37	ET2	I	Reference voltage used by the RNG.	770 mV
38	ET1	I	Reference voltage used by the RNG.	750 mV
39	VREFII	I	Reference voltage used by the RNG.	1.35 V
40	TEST_AOUT	O	Analog output of the captured image.	
41-63	NC			
64	VSS	I	Ground.	0 V
65	VDD	I	Digital power supply.	1.8 V
66	IN_MEMTEST	I	Input of the memories array implemented for testing purposes.	
67	OUT_MIM	O	Output of the memories array implemented for testing purposes.	
68	W_MEMTEST	I	Write signal of the memories array implemented for testing purposes.	
69	E_MEMTEST	I	Enable signal of the memories array implemented for testing purposes.	
70-71	ROW_M[1:0]	I	Select signal of the memories array implemented for testing purposes.	
72-73	COL_M[1:0]	I	Select signal of the memories array implemented for testing purposes.	
74	SEL_G_MTEST	I	Select signal of the memories array implemented for testing purposes.	
75	OUT_NCAP	O	Output of the memories array implemented for testing purposes.	
76	OUT_PCAP	O	Output of the memories array implemented for testing purposes.	
77	SEGOUT	O	Sementation result.	
78-85	OUTPUT[0:7]	O	Captured image converted by the per-column SS-ADC	
86	NC			

#	Name	I/O	Functionality	Value
87	VDD	I	Digital voltage supply.	1.8 V
88	GND	I	Ground.	0 V
89	VDD33	I	I/O voltage supply.	3.3 V
90	CMP	O	Output of the block that calculates d(x) implemented for testing purposes.	
91	CLK2	I	Input of the block that calculates d(x) implemented for testing purposes.	
92	PHIMAX	I	Input of the block that calculates d(x) implemented for testing purposes.	
93	E_DCALC	I	Enable of the block that calculates d(x) implemented for testing purposes.	
94	RESET_TEST	I	Reset of the block that calculates d(x) implemented for testing purposes.	
95	SEG_TEST	O	Testing point connected to the output of the segmenter in a PU inside the PE array.	
96	VDDA	I	Analog voltage supply.	1.8 V
97	A_RNG	O	Analog output of the RNG.	
98	A_DCALC	O	Output of the block that calculates d(x) implemented for testing purposes.	
99	GND	I	Ground.	0 V
100	MIN_MAX	O	Output of the block that calculates d(x) implemented for testing purposes.	
101	VDDA	I	Analog voltage supply.	1.8 V
102	TEST_AOUT	O	Analog output of the captured image.	
103	GND	I	Ground.	0 V
104	DCALC_T_I	I	Input of the block that calculates d(x) implemented for testing purposes.	
105	V_{RAMP}	I	Voltage ramp used for the SS-ADC	
106	ARAM_VAL	O	Testing point connected to the output of the ARAM in a PU inside the PE array.	
107	O_TO_SEG	O	Testing point connected to the input of the segmenter in a PU inside the PE array.	
108	NEWP	O	Testing point connected to the output of the block that calculates p(x) in a PU inside the PE array.	
109	D	O	Testing point connected to the output of the block that calculates d(x) in a PU inside the PE array.	

#	Name	I/O	Functionality	Value
110	P	O	Testing point connected to the input of the block that calculates p(x) in a PU inside the PE array.	
111	R	O	Testing point connected to the output of the block that calculates R(x) in a PU inside the PE array.	
112	NC			
113	GND	I	Ground.	0 V
114	VDDA	I	Analog voltage supply.	1.8 V
115-117	NC			
118	VOFF	I	Offset voltage of the processing blocks.	350 mV
119	VDD_PIX	I	Photodiode reset voltage.	500 mV
120	VCP	I	Low bias voltage for the PMOS transistors.	950 mV
121	VBP	I	High bias voltage for the PMOS transistors.	1.2 V
122	VCN	I	High bias voltage for the NMOS transistors.	650 mV
123	VBNL	I	Low bias voltage for the NMOS transistors.	400 mV
124	VHOLD	I	Voltage connected to the ARAM input when it is not being written.	900 mV
125-126	NC			
127	GND	I	Ground.	0 V
128-129	NC			
130	GND	I	Ground.	0 V
131	NC			
132	GND	I	Ground.	0 V
133	VDDA		Analog voltage supply.	1.8 V
134	GND	I	Ground.	0 V
135	p_{max}	I	Maximum value of p(x).	800 mV
136	p_{min}	I	Minimum value of p(x).	600 mV
137	R_{min}	I	Minimum value of R(x).	420 mV
138	NC	I		
139	GND	I	Ground.	0 V
140	PHI2RNG		External control input: second clock phase of the RNG.	
141	PHI1RNG	I	External control input: first clock phase of the RNG.	
142	ENABLE_RNG	I	External control input: enable of the RNG.	

#	Name	I/O	Functionality	Value
143	RESET_RAM	I	External control input: SS-ADC memory RAM reset.	
145	ACT_OUT<0>	I	External control input: connect first analog random sample stored in the RNG with the array.	
146	PHI2ADC	I	External control input: second clock phase of the ADC.	
147	NC			
148	VDD	I	Digital power supply.	1.8 V
149	GND		Ground.	0 V
150	VDD33	I	I/O voltage supply.	3.3 V
151	PHI1ADC	I	External control input: first clock phase of the ADC.	
152-155	S_TEST[3:0]	I	External control input: pixel multiplexer output selector.	
156	TEST_A_CTRL	I	External control input: activate the analog output for the captured image.	
157-159	DIFFCTRL[5:3]	I	External control input: background model upadte unit control signals.	
160	WRITE<1>	I	External control input: write second analog random sample in the RNG.	
161-163	DIFFCTRL[2:0]	I	External control input: background model upadte unit control signals.	
164-167	ENABLE[0:3]	I	External control input: PE pixel selector to be connected to the PU.	
168	E_PROCESSING	I	External control input: enable array processing.	
169	GLOBALWRITE	I	External control input: initialize the background model writing the captured image to all the background model samples.	
170-171	MEMSEL[7:6]	I	External control input: ARAM memory selector.	
172	VDD	I	Digital power supply.	1.8 V
173	GND	I	Ground.	0 V
174	VDD33	I	I/O voltage supply.	3.3 V
175-180	MEMSEL[5:0]	I	External control input: ARAM memory selector.	
181	WRITE<0>	I	External control input: write first analog random sample in the RNG.	
182	VDDA	I	Analog voltage supply.	1.8 V

#	Name	I/O	Functionality	Value
183	GND	I	Ground.	0 V
184	VDDA	I	Analog voltage supply.	1.8 V
185	GND	I	Ground.	0 V
186	VDDA	I	Analog voltage supply.	1.8 V
187	PHI1CAP	I	External control input: CDS first clock phase.	
188	PHI1G	I	External control input: Gaussian diffusion first clock phase.	
189	PHI2CAP	I	External control input: CDS second clock phase.	
190	PHI2G	I	External control input: Gaussian diffusion first clock phase.	
191	PHIMIN	I	External control input: d(x) calculation second clock phase.	
192	PHISEG	I	External control input: segmenter second clock phase.	
193	RESET_PIX	I	External control input: photodiode reset.	
194	SEG_D_RST	I	External control input: segmenter and block that calculates d(x) reset.	
195	SELP	I	External control input: second phase of the block that calculates p(x).	
196	SELR	I	External control input: second phase of the block that calculates R(x).	
197	BGMODELUP	I	External control input: enable for the background model update unit.	
198-200	ADC_CNT[7:5]	I	External control input: ADC counter.	
201	VDD	I	Digital power supply.	1.8 V
202	GND	I	Ground.	0 V
203	VDD33	I	I/O voltage supply.	3.3 V
204-208	ADC_CNT[4:0]	I	External control input: ADC counter.	

Bibliography

[1] D. García-Lesta, D. Cabello, E. Ferro, P. López, and V. M. Brea, "Wireless sensor network with perpetual motes for terrestrial snail activity monitoring," *IEEE Sensors Journal*, vol. 17, no. 15, pp. 5008–5015, 2017.

[2] W. Vereecken, W. Van Heddeghem, M. Deruyck, B. Puype, B. Lannoo, W. Joseph, D. Colle, L. Martens, and P. Demeester, "Power consumption in telecommunication networks: overview and reduction strategies," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 62–69, 2011.

[3] J. Baliga, R. Ayre, K. Hinton, and R. S. Tucker, "Energy consumption in wired and wireless access networks," *IEEE Communications Magazine*, vol. 49, no. 6, pp. 70–77, 2011.

[4] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE ACCESS*, vol. 6, pp. 6900–6919, 2017.

[5] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[6] B. Blanco-Filgueira, D. García-Lesta, M. Fernández-Sanjurjo, V. M. Brea, and P. López, "Deep learning-based multiple object visual tracking on embedded system for iot and mobile edge computing applications," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5423–5431, 2019.

[7] P. Popat, P. Sheth, and S. Jain, "Animal/object identification using deep learning on raspberry pi," in *Information and Communication Technology for Intelligent Systems*, pp. 319–327, Springer, 2019.

[8] T. N. Ngo, K.-C. Wu, E.-C. Yang, and T.-T. Lin, "A real-time imaging system for multiple honey bee tracking and activity monitoring," *Computers and Electronics in Agriculture*, vol. 163, p. 104841, 2019.

[9] A. Linares-Barranco, F. Perez-Peña, D. P. Moeys, F. Gomez-Rodriguez, G. Jimenez-Moreno, S.-C. Liu, and T. Delbruck, "Low latency event-based filtering and feature extraction for dynamic vision sensors in real-time fpga applications," *IEEE Access*, vol. 7, pp. 134926–134942, 2019.

[10] J. Weberruss, L. Kleeman, D. Boland, and T. Drummond, "Fpga acceleration of multilevel orb feature extraction for computer vision," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–8, IEEE, 2017.

[11] Q. Liu, O. Richter, C. Nielsen, S. Sheik, G. Indiveri, and N. Qiao, "Live demonstration: Face recognition on an ultra-low power event-driven convolutional neural network ASIC," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[12] J. N. P. Martel, L. K. Müller, S. J. Carey, J. Müller, Y. Sandamirskaya, and P. Dudek, "Real-time depth from focus on a programmable focal plane processor," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, pp. 925–934, March 2018.

[13] X. Zhong, M. Law, C. Tsui, and A. Bermak, "A fully dynamic multi-mode cmos vision sensor with mixed-signal cooperative motion sensing and object segmentation for adaptive edge computing," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1684–1697, 2020.

[14] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: An overview," *Computer Science Review*, vol. 11, pp. 31–66, 2014.

[15] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Transactions on image processing*, vol. 17, no. 7, pp. 1168–1177, 2008.

[16] B.-J. Huang, J.-W. Hsieh, and C.-M. Tsai, "Vehicle detection in hsuehshan tunnel using background subtraction and deep belief network," in *Asian Conference on Intelligent Information and Database Systems*, pp. 217–226, Springer, 2017.

[17] R. R. Fratama, N. D. A. Partiningsih, E. H. Rachmawanto, C. A. Sari, P. N. Andono, *et al.*, "Real-time multiple vehicle counter using background subtraction for traffic monitoring system," in *2019 International Seminar on Application for Technology of Information and Communication (iSemantic)*, pp. 1–5, IEEE, 2019.

[18] K. Goyal and J. Singhai, "Review of background subtraction methods using gaussian mixture model for video surveillance systems," *Artificial Intelligence Review*, vol. 50, no. 2, pp. 241–259, 2018.

[19] P. Tumas and A. Serackis, "Effective background subtraction algorithm for food inspection using a low-cost near infrared camera," in *2017 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pp. 1–4, 2017.

[20] B. Garcia-Garcia, T. Bouwmans, and A. J. R. Silva, "Background subtraction in real applications: Challenges, current models and future directions," *Computer Science Review*, vol. 35, p. 100204, 2020.

[21] N. Cottini, M. Gottardi, N. Massari, R. Passerone, and Z. Smilansky, "A 33 uw 64 x 64 pixel vision sensor embedding robust dynamic background subtraction for event detection and scene interpretation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 850–863, 2013.

[22] M. Piccardi, "Background subtraction techniques: a review," in *Systems, man and cybernetics, 2004 IEEE international conference on*, vol. 4, pp. 3099–3104, IEEE, 2004.

[23] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, IEEE, 2008.

[24] S. Y. Elhabian, K. M. El-Sayed, and S. H. Ahmed, "Moving object detection in spatial domain using background removal techniques-state-of-art," *Recent patents on computer science*, vol. 1, no. 1, pp. 32–54, 2008.

[25] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.net: A new change detection benchmark dataset," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference*, pp. 1–8, IEEE, 2012.

[26] A. Zarándy, *Focal-Plane Sensor-Processor Chips*. Springer-Verlag New York, 1 ed., 2011.

[27] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *IEEE Transactions on Image processing*, vol. 20, no. 6, pp. 1709–1724, 2011.

[28] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The pixel-based adaptive segmenter," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference*, pp. 38–43, IEEE, 2012.

[29] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 747–757, 2000.

[30] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *European conference on computer vision*, pp. 751–767, Springer, 2000.

[31] D. García-Lesta, V. M. Brea, P. López, and D. Cabello, "Effect of temporal and spatial noise on the performance of hardware oriented background extraction algorithms," in *New Circuits and Systems Conference (NEWCAS), 2017 15th IEEE International*, pp. 45–48, IEEE, 2017.

[32] M. Suárez, V. M. Brea, J. Fernández-Berni, R. Carmona-Galán, D. Cabello, and Á. Rodríguez-Vázquez, "Low-power CMOS vision sensor for gaussian pyramid extraction," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 2, pp. 483–495, 2017.

[33] J. Fernández-Berni and R. Carmona-Galán, "All-MOS implementation of RC networks for time-controlled Gaussian spatial filtering," *International Journal of Circuit Theory and Applications*, vol. 40, no. 8, pp. 859–876, 2012.

[34] D. García-Lesta, V. M. Brea, P. López, and D. Cabello, "Shannon entropy as background dynamics estimator in foreground detector algorithms," in *2018 IEEE International Symposium on Circuits and Systems*, 2018.

[35] S. J. Carey, D. R. Barr, B. Wang, A. Lopich, and P. Dudek, "Mixed signal simd processor array vision chip for real-time image processing," *Analog Integrated Circuits and Signal Processing*, vol. 77, no. 3, pp. 385–399, 2013.

[36] C. Greatwood, L. Bose, T. Richardson, W. Mayol-Cuevas, J. Chen, S. J. Carey, and P. Dudek, "Tracking control of a uav with a parallel visual processor," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4248–4254, Sep. 2017.

[37] D. Ginhac, J. Dubois, M. Paindavoine, and B. Heyrman, "An simd programmable vision chip with high-speed focal plane image processing," *EURASIP Journal on Embedded Systems*, vol. 2008, no. 1, p. 961315, 2009.

[38] Y. Zou, M. Gottardi, M. Lecca, and M. Perenzoni, "A low-power vga vision sensor with embedded event detection for outdoor edge applications," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 11, pp. 3112–3121, 2020.

[39] T.-H. Hsu, Y.-K. Chen, J.-S. Wu, W.-C. Ting, C.-T. Wang, C.-F. Yeh, S.-H. Sie, Y.-R. Chen, R.-S. Liu, C.-C. Lo, *et al.*, "5.9 a 0.8 v multimode vision sensor for motion and saliency detection with ping-pong pwm pixel," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 110–112, IEEE, 2020.

[40] X. Liu, M. Zhang, and J. Van der Spiegel, "A low-power multifunctional CMOS sensor node for an electronic façade," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 9, pp. 2550–2559, 2014.

[41] G. R. C. Fiorante, J. Ghasemi, P. Zarkesh-Ha, and S. Krishna, "Spatio-temporal bias-tunable readout circuit for on-chip intelligent image processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 11, pp. 1825–1832, 2016.

[42] C. Kim, K. Bong, S. Choi, K. L. Lee, and H.-J. Yoo, "A CMOS Image Sensor-Based Stereo Matching Accelerator With Focal-Plane Sparse Rectification and Analog Census Transform," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2180–2188, 2016.

[43] D. Ginhac, J. Dubois, B. Heyrman, and M. Paindavoine, "A high speed programmable focal-plane simd vision chip," *Analog Integrated Circuits and Signal Processing*, vol. 65, no. 3, pp. 389–398, 2010.

[44] G. Köklü, R. Etienne-Cummings, Y. Leblebici, G. De Micheli, and S. Carrara, "Characterization of standard cmos compatible photodiodes and pixels for lab-on-chip devices,"

in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1075–1078, May 2013.

[45] D. García-Lesta, P. López, V. M. Brea, and D. Cabello, "In-pixel analog memories for a pixel-based background subtraction algorithm on CMOS vision sensors," *International Journal of Circuit Theory and Applications*, vol. 46, no. 9, pp. 1631–1647, 2018.

[46] B. Jähne, H. Haussecker, and P. Geissler, *Handbook of computer vision and applications*, vol. 2. Academic press, 1999.

[47] J. Fernandez-Berni, R. Carmona-Galan, and L. Carranza-Gonzalez, "Flip-q: A qcif resolution focal-plane array for low-power image processing," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 669–680, March 2011.

[48] A. Rodríguez-Vázquez, G. Liñán-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galán, F. Jiménez-Garrido, R. Domínguez-Castro, and S. E. Meana, "ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 5, pp. 851–863, 2004.

[49] M. Laiho, J. Poikonen, and A. Paasio, "Mipa4k: Mixed-mode cellular processor array," in *Focal-Plane Sensor-Processor Chips*, pp. 45–71, Springer, 2011.

[50] R. J. Baker, *CMOS: circuit design, layout, and simulation*, vol. 1. John Wiley & Sons, 2008.

[51] A. Nshare, J.-O. Klein, and A. Dupret, "Improved ARAM for PARIS, an original programmable vision chip," in *Cellular Neural Networks and Their Applications, 2002.(CNNA 2002). Proceedings of the 2002 7th IEEE International Workshop on*, pp. 347–354, IEEE, 2002.

[52] S. Kawahito, "Column-parallel adcs for cmos image sensors and their fom-based evaluations," *IEICE Transactions on Electronics*, vol. 101, no. 7, pp. 444–456, 2018.

[53] J. A. Leñero-Bardallo, J. Fernández-Berni, and Á. Rodríguez-Vázquez, "Review of adcs for imaging," in *Image Sensors and Imaging Systems 2014*, vol. 9022, p. 90220I, International Society for Optics and Photonics, 2014.

[54] B. Lin, B. Gao, Y. Pang, P. Yao, D. Wu, H. He, J. Tang, H. Qian, and H. Wu, "A high-speed and high-reliability trng based on analog rram for iot security application," in *2019 IEEE International Electron Devices Meeting (IEDM)*, pp. 14–8, IEEE, 2019.

[55] R. S. Prasad, A. Siripagada, S. Selvaraj, and N. Mohankumar, "Random seeding lfsr-based trng for hardware security applications," in *Integrated Intelligent Computing, Communication and Security*, pp. 427–434, Springer, 2019.

[56] Y. Liu, R. C. C. Cheung, and H. Wong, "A bias-bounded digital true random number generator architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 1, pp. 133–144, 2017.

[57] J. L. Valtierra, E. Tlelo-Cuautle, and Á. Rodríguez-Vázquez, "A switched-capacitor skew-tent map implementation for random number generation," *International Journal of Circuit Theory and Applications*, vol. 45, no. 2, pp. 305–315, 2017.

[58] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks, *et al.*, *Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications*. National Institute of Standards & Technology, 2010.

[59] J. Le Hir, A. Kolar, and F. V. Dos Santos, "Distributed mixed-signal architecture for programmable smart image sensors," *Analog Integrated Circuits and Signal Processing*, vol. 97, no. 3, pp. 493–501, 2018.

[60] J. A. Schmitz, M. K. Gharzai, S. Balkır, M. W. Hoffman, D. J. White, and N. Schemm, "A 1000 frames/s vision chip using scalable pixel-neighborhood-level parallel processing," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 2, pp. 556–568, 2016.

[61] "CMOD A7." `https://reference.digilentinc.com/reference/programmable-logic/cmod-a7/start`. Accessed: February 2020.

[62] "Paticle Photon." `https://docs.particle.io/photon/`. Accessed: February 2020.

[63] "Texas Instruments TLC7524." `https://www.ti.com/lit/ds/slas061d/slas061d.pdf`. Accessed: February 2020.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Figures

List of Tables