DISSENY I IMPLEMENTACIÓ HARDWARE D'ALGORISMES DE RECONEIXEMENT DE CARÀCTERS PER COMPTADORS D'AIGUA

Memòria del Projecte Fi de Carrera
d'Enginyeria Electrònica
realitzat per
DANIEL CASTILLO HAND
i dirigit per
LLUÍS TERÉS TERÉS
JOSE LUIS MERINO PANADÉS
Bellaterra, 29 de gener de 2007

Resum.

Aquest projecte consisteix en l'estudi, comparació i implementació en Hardware d'algoritmes de reconeixement de caràcters per integrar en un sistema intel·ligent de captura d'imatges. Aquest sistema, integrat per una càmera amb format i característiques específiques i que anirà acoblat a un comptador d'aigua tradicional, en captarà imatges i les enviarà per RF al punt de recepció de la companyia. L'objectiu principal consisteix en aconseguir un disseny que redueixi al màxim la quantitat d'informació per transmetre, tenint en compte les limitacions de l'entorn.

Resumen.

Este proyecto consiste en el estudio, comparación e implementación en Hardware de algoritmos de reconocimiento de caracteres para integrar en un sistema inteligente de captura de imágenes. Este sistema, integrado por una cámara con formato y características específicas y que irá acoplado a un contador de agua tradicional, captará imágenes y las enviará por RF al punto de recepción de la compañía. El objetivo principal consiste en conseguir un diseño que reduzca al máximo la cantidad de información para transmitir, teniendo en cuenta las limitaciones del entorno.

Summary.

This project is about the study, comparison and Hardware implementation of character recognition algorithms to integrate in an image capture intelligent system. This system, integrated by a camera with specific characteristics and format that will be connected to a traditional water meter, will catch images and will send them by RF to the reception point of the water supplier company. The main goal consists in obtaining a design that reduces as much as possible the amount of information that has to be transmitted, considering the limitations imposed by the environment.

Universitat Autònoma de Barcelona

etse

Escola Tècnica Superior d'Enginyeria

El sotasignat, **Lluís Terés Terés**

Professor de l'Escola Tècnica Superior d'Enginyeria de la UAB (departament de Microelectrònica i Sistemes Electrònics),

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en **Daniel Castillo Hand**

I per tal que consti firma la present.

Signat: .....

Bellaterra, 29 de gener de 2007.

Universitat Autònoma de Barcelona

etse

Escola Tècnica Superior d'Enginyeria

El sotasignat, **José Luis Merino Panadés**

Investigador de l'Institut de Microelectrònica de Barcelona, integrant del Centre Nacional de Microelectrònica (CNM),

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat al centre sota la seva supervisió mitjançant conveni

firmat amb la Universitat Autònoma de Barcelona.

Així mateix el centre en té coneixement i dona el vist-i-plau al contingut que es detalla en aquesta memòria.

Signat: .....

Bellaterra, 29 de gener de 2007.

Índex de continguts

1 INTRODUCCIÓ 1

2 OBJECTIUS 3

2.1 Descripció dels objectius. 3
2.2 Entorn de desenvolupament 5
2.2.1 Sistema intel·ligent de captura d'imatges. 5
2.2.2 Format de les imatges. 7
2.3 Antecedents. 9
2.4 Condicionants. 9
2.4.1 Reducció de la quantitat d'informació transmesa 9
2.4.2 Limitació d'àrea, consum i temps 9
2.5 Abast. 10

3 METODOLOGIA DE TREBALL 11

3.1 Etapes i tasques 11
3.2 Planificació temporal. 14

4 RECONEIXEMENT ÒPTIC DE CARÀCTERS (OCR) 15

4.1 Introducció al reconeixement òptic de caràcters. 15
4.2 Procediment per al reconeixement òptic de caràcters. 17
4.2.1 Captura de la imatge. 17
4.2.2 Segmentació. 17
4.2.3 Pre-processament de la imatge. 18
4.2.4 Obtenció de característiques del caràcter. 23
4.2.5 Comparació de característiques amb patrons. 24
4.3 Mètodes d'obtenció de característiques de caràcters. 24
4.3.1 Mètode de Crossings. 24
4.3.2 Mètode de Zoning 25
4.3.3 Altres mètodes. 26

5 DISSENY I IMPLEMENTACIÓ 27

5.1 Disseny software i test d'algoritmes de reconeixement de caràcters. 27
5.1.1 Algoritmes basats en el Mètode de Crossings. 27
5.1.2 Algoritmes basats en el Mètode de Zoning. 34
5.1.3 Comparació dels algoritmes. 37
5.1.4 Descomposició dels algoritmes en funcions 42

5.2 Implementació en hardware de blocs funcionals d'algoritmes OCR. 44
5.2.1 Bloc que realitza el procés de binarització. 44
5.2.2 Bloc que calcula el llindar per al procés de binarització. 44
5.2.3 Blocs que implementen un Filtre de Mitjana. 45
5.2.4 Bloc del Compressor de dades..... 45
5.2.5 Blocs per obtenir els encreuaments ("Crossings"). 46
5.2.6 Blocs que realitzen l'aprimament de la imatge. 47
5.2.7 Bloc que realitza la zonificació de la imatge ("Zoning"). 49
5.2.8 Bloc que calcula els llindars per al procés de zonificació. 49
5.2.9 Blocs combinacionals de control d'accés. 50
5.2.10 Mòduls no sintetitzables: Càmera. 52
5.2.11 Mòduls no sintetitzables: Memòries. 53
5.2.12 Mòduls no sintetitzables: *Imatge_bmp* i *Imatge_gris_bmp*..... 54
5.3 Síntesis i test dels blocs funcionals..... 56
5.4 Disseny hardware de mòduls de processament d'imatges. 59
5.4.1 Arquitectura 1. 59
5.4.2 Arquitectura 2. 61
5.4.3 Arquitectura 3. 63
5.4.4 Arquitectura 4. 65
5.4.5 Arquitectura 5. 67
5.5 Síntesis i test dels mòduls de processament d'imatges..... 69
5.5.1 Test dels mòduls sintetitzats. 69
5.5.2 Test sobre FPGA. 71
5.6 Comparació dels diferents dissenys i determinació del més òptim. 74
5.6.1 Comparació de l'àrea i consum de potència. 74
5.6.2 Comparació de l'àrea del mòdul de memòria sintetitzat. 76
5.6.3 Comparació dels ràtios de compressió de dades. 78
5.6.4 Comparació de resultats gràfics amb imatges disponibles. 79
5.6.5 Determinació del disseny més òptim. 80

6 RESULTATS 83

6.1 Descripció del disseny escollit. 83
6.2 Limitacions del disseny escollit. 87

7 CONCLUSIONS 89

7.1 Valoració dels resultats obtinguts. 89
7.2 Propostes d'ampliació. 90
7.3 Valoració personal. 90

8 REFERÈNCIES..... 93

9 GLOSSARI 95

APÈNDIX A. CONTINGUT DEL CD ADJUNT. A-1

APÈNDIX B. EXEMPLE DE CODIS FONT. B-1

Apèndix B.1 . Exemple d'arxiu font VHDL : arxiu *binaritzador.vhd*. B-1
Apèndix B.2 . Exemple d'arxiu font VHDL sintetitzat. B-4

APÈNDIX C. DADES TECNOLÒGIQUES..... C-1

Apèndix C.1 . Exemple de datasheet d'un component C35B4. ..... C-1
Apèndix C.2 . Dades de memòries RAM sintetitzables de C35B4. ..... C-2

Índex de figures

Figura 2.1. Imatge i esquema bàsic del sistema de captura d'imatges 3
Figura 2.2. Detall dels pixels 5
Figura 2.3. Primera versió de l'ASIC global 5
Figura 2.4. Estructura de la matriu de pixels de la càmera 6
Figura 2.5. Esquema intern del Sistema intel·ligent de captura d'imatges 7
Figura 2.6. Zona de la matriu de pixels sobre la que es treballarà 8
Figura 3.1. Diagrama de flux de les tasques 13
Figura 3.2. Planificació de la distribució temporal de les tasques 14
Figura 4.1. Alteracions en les imatges que dificulten la tasca de l'OCR 16
Figura 4.2. Càlcul del pixel llindar segons mètode 19
Figura 4.3. Pixels considerats en el procés del Filtre de Mitjana 20
Figura 4.4. Exemple d'aplicació del Filtre de Mitjana 20
Figura 4.5. Veïnatge digital segons adjacència considerada 21
Figura 4.6. Resultat de l'aprimament complert d'una imatge 21
Figura 4.7. Procés d'aprimament nord d'una imatge 22
Figura 4.8. Exemple d'aplicació del mètode de "Crossings" 25
Figura 4.9. Exemple d'aplicació del mètode de "Zoning" 26
Figura 5.1. Diagrama de flux de l'algoritme OCR_crossings 28
Figura 5.2. Diagrama de flux de l'algoritme OCR_crossings_filtre 29
Figura 5.3. Diagrama de flux de l'algoritme OCR_crossings_filtre_bin 31
Figura 5.4. Diagrama de flux de l'algoritme OCR_crossings_histograma 32
Figura 5.5. Diagrama de flux de l'algoritme OCR_crossings_aprimament 34
Figura 5.6. Diagrama de flux de l'algoritme OCR_zoning 35
Figura 5.7. Diagrama de flux de l'algoritme OCR_zoning_llindar 37
Figura 5.8. Imatges resultats dels algoritmes OCR 41
Figura 5.9. Exemple de compressió de dades d'una imatge 46
Figura 5.10. Diagrama de flux del bloc aprimament_imatge 48
Figura 5.11. Pixels considerats en el procés de Zoning 50
Figura 5.12. Esquema d'un controlador d'accés 51
Figura 5.13. Procés de simulació de la càmera 52
Figura 5.14. Procés per obtenir la imatge resultant del processament 55
Figura 5.15. Concepte de síntesis 56
Figura 5.16. Resultats de la simulació de blocs funcionals amb *Modelsim* 58
Figura 5.17. Diagrama de flux de l'algoritme de l'arquitectura 1 60
Figura 5.18. Diagrama de flux de l'algoritme de l'arquitectura 2 62
Figura 5.19. Diagrama de flux de l'algoritme de l'arquitectura 3 64

Figura 5.20. Diagrama de flux de l'algoritme de l’arquitectura 4 66
Figura 5.21. Diagrama de flux de l'algoritme de l’arquitectura 5 68
Figura 5.22. Exemple de simulació sense considerar retards de portes 70
Figura 5.23. Exemple de simulació considerant retards de portes 70
Figura 5.24. Interacció amb la FPGA..... 72
Figura 5.25. Entorn de test FPGA i càmera..... 73
Figura 5.26. Imatges resultants del processat, segons arquitectura..... 79
Figura 6.1. Imatges resultants del processament del mòdul-OCR 85
Figura 6.2. Mòdul-OCR integrat a l’ASIC..... 86

Índex de taules

Taula 5.1. Prestacions dels algoritmes OCR implementats 39
Taula 5.2. Recursos de memòria necessaris per implementar els algoritmes OCR 40
Taula 5.3. Ràtios de compressió de dades dels diferents algoritmes OCR 41
Taula 5.4. Relació de blocs combinacionals de control d'accés 51
Taula 5.5. Mòduls de memòria implementats en el disseny 53
Taula 5.6. Arxius resultants del procés de síntesis dels blocs funcionals 57
Taula 5.7. Recursos hardware de l'arquitectura 1 59
Taula 5.8. Recursos hardware de l'arquitectura 2 61
Taula 5.9. Recursos hardware de l'arquitectura 3 63
Taula 5.10. Recursos hardware de l'arquitectura 4 65
Taula 5.11. Recursos hardware de l'arquitectura 5 67
Taula 5.12. Resultats de la síntesis del mòdul-OCR segons arquitectura 74
Taula 5.13. Blocs funcionals integrants del mòdul-OCR segons arquitectura 75
Taula 5.14. Recursos de memòria requerits pel mòdul-OCR segons arquitectura 76
Taula 5.15. Àrees de mòduls de memòria SRAM sintetitzables; font: [AMS02]. 77
Taula 5.16. Capacitat de memòria sintetitzable requerida segons arquitectura 77
Taula 5.17. Àrees de memòria sintetitzable requerides per les arquitectures 78
Taula 5.18. Ràtios de compressió de dades de les diferents arquitectures 78
Taula 5.19. Resum de característiques de les arquitectures 80
Taula 5.20. Comparativa de les diferents arquitectures 81
Taula 6.1. Paràmetres del mòdul-OCR 83
Taula 6.2. Mapa de ports I/O del mòdul-OCR 84

1 Introducció

A mesura que la tecnologia va avançant, cada vegada hi ha més processos, tradicionalment realitzats de forma manual, que es fan de forma automatitzada, on l'acció humana es circumscriu a tasques de guiat i control.

Un exemple és el procés de lectura d'un comptador de consum d'aigua per part de la companyia subministradora. Tradicionalment, aquesta tasca ha sigut realitzada per un operari de la pròpia companyia, que després de desplaçar-se als punts de consum ha subministrat els valors indicats pel comptador a la central. Moltes de les noves instal·lacions urbanes ja disposen de comptadors electrònics als punts de consum de forma que el procés de lectura es realitza automàticament i posteriorment la informació és transmesa a la central per mitjà de connexions per cable o a través de ràdio-freqüència (RF).

Malgrat això, la majoria d'instal·lacions antigues no disposen d'aquests comptadors ni d'aquesta infrastructura. Per això moltes companyies subministradores d'aigua corrent, estan desenvolupant sistemes per poder realitzar la tasca de lectura de comptadors de consum de forma automatitzada, sense les despeses que comportarien les inversions en infrastructura en instal·lacions antigues.

Un sistema capaç de realitzar aquesta tasca consisteix en un sistema intel·ligent de captura d'imatges acoblat al comptador tradicional. La seva funció serà la de capturar la imatge de la zona on hi figuren els dígits del valor de consum, realitzar el processament de la imatge determinant el valor de consum indicat pel comptador, i enviar aquesta informació a través de ràdio-freqüència (RF) a un equip de captura de dades de la companyia corresponent.

El present projecte es desenvolupa en el marc d'un projecte d'investigació orientat a la realització d'aquest sistema intel·ligent de captura d'imatges.

L'objectiu d'aquest projecte és l'estudi, comparació i implementació en Hardware de diferents algoritmes de reconeixement de caràcters per tal de reduir al màxim la quantitat d'informació transmesa per RF, tot conservant la informació necessària per tal de poder identificar el dígit al punt de recepció, tenint en compte les limitacions d'àrea disponible i consum màxim permès per l'esmentat sistema.

La present memòria s'estructura en capítols, el contingut dels quals se sintetitza a continuació.

El capítol 1, el present, és l'introductori; en ell s'introdueixen els continguts del projecte i s'explica l'estructura de la Memòria.

El capítol 2 enuncia els objectius del projecte, descriu els elements de l'entorn en el qual es desenvoluparà el projecte, així com els condicionants a tenir en compte en la realització del mateix. Finalment, indica el nivell d'abstracció fins al qual es desenvoluparà el projecte.

El capítol 3 descriu la metodologia de treball, és a dir, els passos que se seguiran entre el plantejament dels objectius i l'obtenció del resultat final.

Al capítol 4 es fa una explicació teòrica del procés de reconeixement òptic de caràcters (OCR), necessària pel desenvolupament del projecte.

Al capítol 5 és on es desenvolupa el procés de disseny i implementació en Software i en Hardware de diferents algoritmes de reconeixement de caràcters per tal d'aconseguir els objectius fixats. També es realitza una comparativa dels diferents algoritmes i es determina el més adient per l'aplicació.

El capítol 6 descriu els resultats obtinguts, amb el detall del disseny implementat i les limitacions del mateix.

El capítol 7 expressa les conclusions, el 8 cita les referències externes emprades en aquest projecte, i el 9 conté el glossari dels principals acrònims emprats en la redacció d'aquesta memòria.

Els apèndixs de la memòria consten de la relació d'arxius font de diversos codis emprats en el desenvolupament i la seva ubicació en el CD adjunt, d'exemples de codis font desenvolupats, així com de dades tècniques de recursos emprats en el projecte.

2 Objectius

2.1 Descripció dels objectius.

El sistema intel·ligent de captura d'imatges sobre el que treballarem s'acobla a un comptador de consum d'aigua tradicional, tal i com podem veure a la Figura 2.1. La funció principal del sistema és la de capturar i processar la imatge del comptador, on hi figuren els dígits del valor numèric de consum, abans de procedir al seu enviament per RF al punt de recepció de la companyia.

Com podem veure a la Figura 2.1 el sistema consta dels següents elements:

- Càmera CMOS de format específic.
- Mòdul de comunicacions per RF.
- Sistema d'il·luminació per LEDs.
- Alimentació per bateries.
- Sistema de control, encarregat de la gestió de tot el mòdul.

El principal objectiu del present projecte consisteix en minimitzar el volum de les dades a enviar per RF per tal de reduir tan l'ample de banda necessari com el temps en que el mòdul de RF està activat, amb el que reduïm el seu consum.

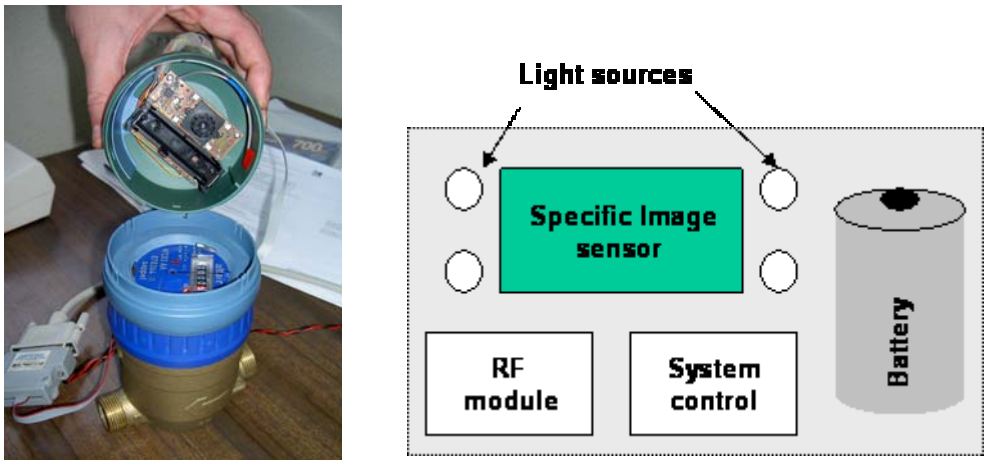


Figura 2.1. Imatge i esquema bàsic del sistema de captura d'imatges

La mínima informació d'un dígit decimal a transmetre seria el valor del dígit (0-9), és a dir 4 bits. La màxima informació d'un dígit seria la seva imatge representada com a mapa de bits. A la primera opció, el sistema de lectura realitza tot el procés d'identificació de la imatge, mentre que a la segona aquesta tasca és realitzada íntegrament per l'equip del punt de recepció. En aquest projecte es pretén implementar un disseny en Hardware que redueixi al màxim la informació transmesa d'un dígit. Interessaria aconseguir, si fos possible, la seva total identificació o, si no fos possible, la transmissió per RF d'una quantitat d'informació notablement inferior a la necessària per transmetre la imatge com a mapa de bits.

Per tal d'assolir aquest objectiu principal, és necessari assolir una sèrie d'objectius parcials.

El primer objectiu parcial del present projecte consisteix en l'estudi de diferents algoritmes genèrics de reconeixement òptic de caràcters (OCR), i desenvolupar-ne d'específics pels nostres 10 dígits ('0'-'9') amb el corresponent format i font.

El següent objectiu parcial consisteix en la implementació en Software dels algoritmes considerats aptes d'acord amb els criteris enunciats anteriorment i validar-ne el seu funcionament.

Finalment, l'altre objectiu parcial és la implementació dels algoritmes OCR en Hardware i la verificació del seu correcte funcionament. Per fer-ho primer s'implementaran els diferents blocs Hardware necessaris per realitzar els algoritmes. Posteriorment, s'implementaran i verificaran diferents arquitectures, on a cadascuna es materialitzi de forma més o menys complerta l'algoritme OCR. D'aquesta manera a cada arquitectura hi haurà una quantitat d'informació major o menor a transmetre i una quantitat de hardware major o menor necessària per implementar l'algoritme. Finalment una comparativa de les diferents arquitectures permetrà determinar la més adient per l'aplicació, tot tenint en compte les limitacions del sistema.

Aquest Hardware dissenyat s'integraria al mòdul de control del sistema en un IC monolític que inclouria la camera, el sistema de control i el bloc analògic pel control del sistema de il·luminació.

2.2 Entorn de desenvolupament.

2.2.1 Sistema intel·ligent de captura d'imatges.

El sistema intel·ligent de captura d'imatges, que serà integrat en un Circuit integrat per aplicacions especifiques (ASIC) [Des93],[PRE96], fet específicament per la corresponent aplicació i al qual es farà referència a partir d'ara com a ASIC, està format per diversos elements, tant analògics com digitals.

El principal element és una càmera, que consisteix en un array de sensors APS (Active Pixel Sensor) integrats en tecnologia CMOS, que capturen la imatge; un conversor analògic-digital de 8 bits, també integrat al mateix CMOS, que converteix els valors analògics proporcionats per l'array a format digital i un circuit de control que regula la captura de la imatge i el procés de conversió AD.

Un cop l'array captura la imatge, els valors capturats a cada pixel es mantenen durant uns 2 segons, de forma que durant aquest temps es comporta com una memòria ROM. Durant aquest període de temps podem obtenir els valors de cadascun dels pixels de la imatge, indicant els valors de la fila i columna corresponent al controlador. A la Figura 2.4 s'il·lustra el diagrama de blocs intern de l'array de pixels de la càmera.

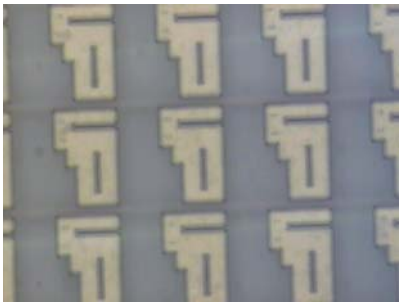


Figura 2.2. Detall dels pixels

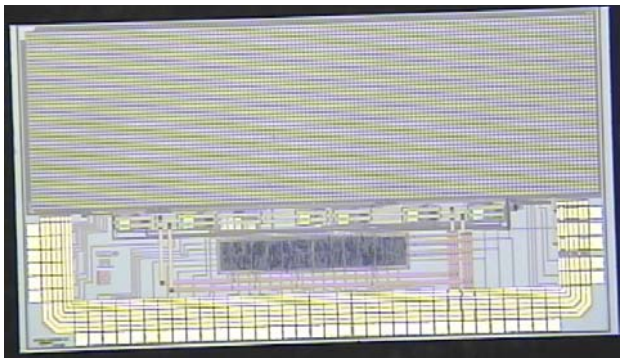


Figura 2.3. Primera versió de l'ASIC global

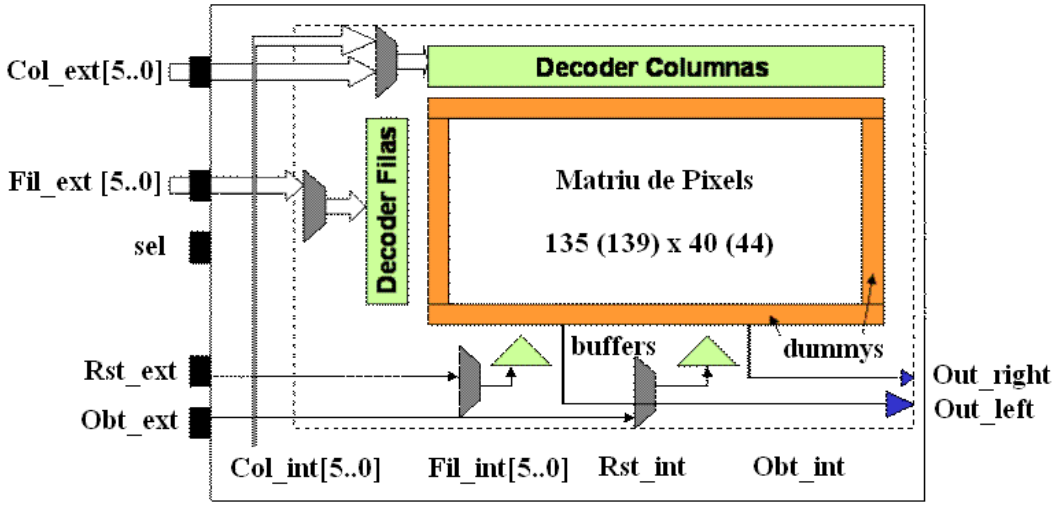


Figura 2.4. Estructura de la matriu de pixels de la càmera

Dos altres mòduls importants son per una banda el sistema de control general, que consistirà en un microcontrolador, o altre dispositiu anàleg, i per altra banda el circuit de comunicació per RF, que farà la funció d'interfície entre l'ASIC i el mitjà físic de transmissió (en el nostre cas RF) per on es transmetran les dades. Aquests dos mòduls es troben encara en fase de disseny.

L'ASIC també disposarà de mòduls de memòria RAM per realitzar l'emmagatzemat de dades durant el processament de la imatge, així com informació rellevant del protocol de comunicació per RF i dades de control.

Finalment, el circuit final també inclourà el sistema Hardware de processament de la imatge desenvolupat en el present projecte, al qual es farà referència a partir d'ara com a mòdul-OCR.

A la Figura 2.5 es mostra un esquema bàsic dels elements que constituiran l'ASIC. En ella es poden veure els mòduls analògics (blocs verds), digitals encara per implementar (blocs vermells), de memòria (bloc blau) i el mòdul-OCR (bloc vermell amb fons groc).

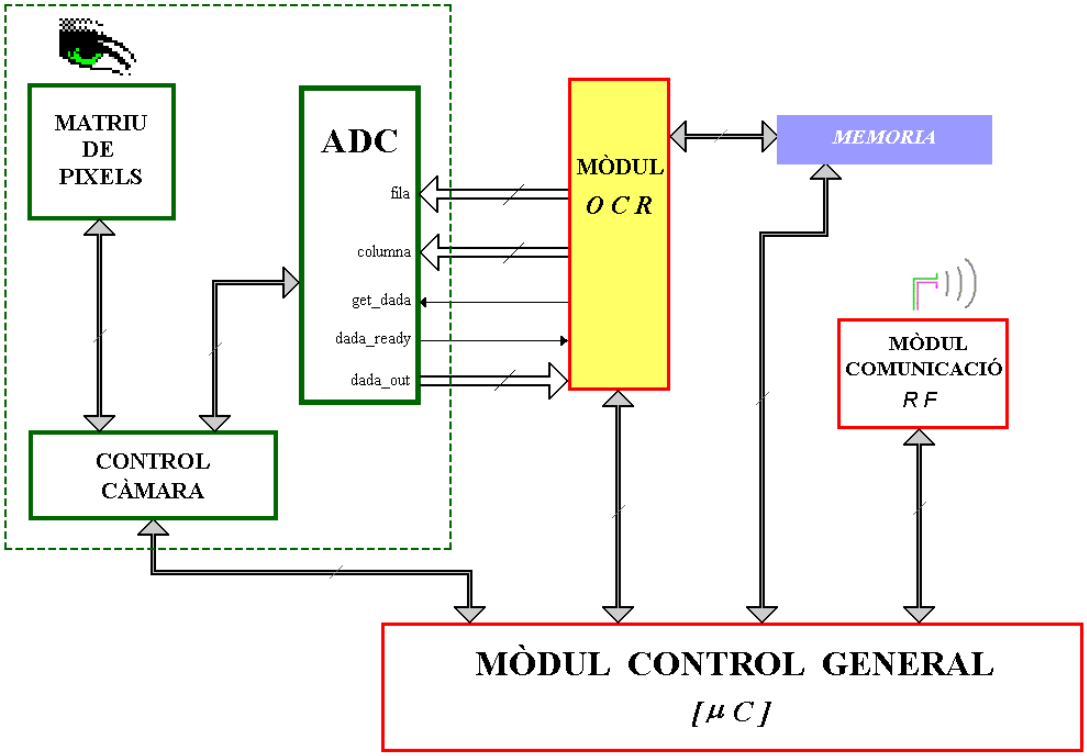


Figura 2.5. Esquema intern del Sistema intel·ligent de captura d'imatges

2.2.2 Format de les imatges.

La matriu de pixels té unes dimensions totals de 44 files i 139 columnes. D'aquestes les dues files i columnes més exteriors no son útils ("dummys"), de forma que les dimensions útils son de 40 files i 135 columnes. L'espai de la matriu correspon al que ocupen 5 dígits decimals del comptador d'aigua.

El processament de la informació de la imatge en el present projecte es farà sobre una zona de la matriu, que correspon a l'espai que ocupa un dígit decimal del comptador, el qual serà l'objecte del procés d'identificació. Degut a la no disponibilitat d'imatges capturades amb la càmera definitiva, es treballarà amb una zona d'imatge de 39 files i 26 columnes, corresponent a les dimensions de la càmera del prototip anterior, idèntica en arquitectura i prestacions, i únicament diferent en dimensions de zona d'imatge. Les proves es realitzaran amb imatges capturades per aquesta càmera. La Figura 2.6 il·lustra les dimensions útils de la matriu de pixels (negre) i la zona sobre la que es treballarà (vermell).

Cada pixel te una àrea de 30 x 30 μm i el seu element sensor es un fotodiode que ocupa el 60% de l'àrea del pixel. Per capturar una imatge el pixel es carrega a la tensió

d'alimentació i, en exposar-ho a la llum, la seva tensió va caient progressivament, depenent de la intensitat de la llum entrant. Una vegada finalitzat el temps d'exposició, s'emmagatzema en una capacitat el valor actual, que es manté durant uns 2 segons.

Si la tensió emmagatzemada és alta, vol dir que no hi ha entrat gaire llum, per tant el valor obtingut serà negre. En canvi, si la tensió ha caigut molt serà perquè ha entrat molt llum, per tant associarem aquesta lectura a un blanc.

La configuració de la matriu de pixels corresponent a la zona sobre la que es treballarà és la següent:

- L'índex de files comença a comptar al costat esquerra.
- L'índex de columnes comença a comptar al la banda de baix.
- La informació de cada pixel es codifica com un byte, amb el color del mateix en escala de 256 nivells de gris. El 00h (0) correspon al color negre i el FFh (255) al blanc.
- Els dígits del comptador d'aigua son de color blanc sobre fons negre.

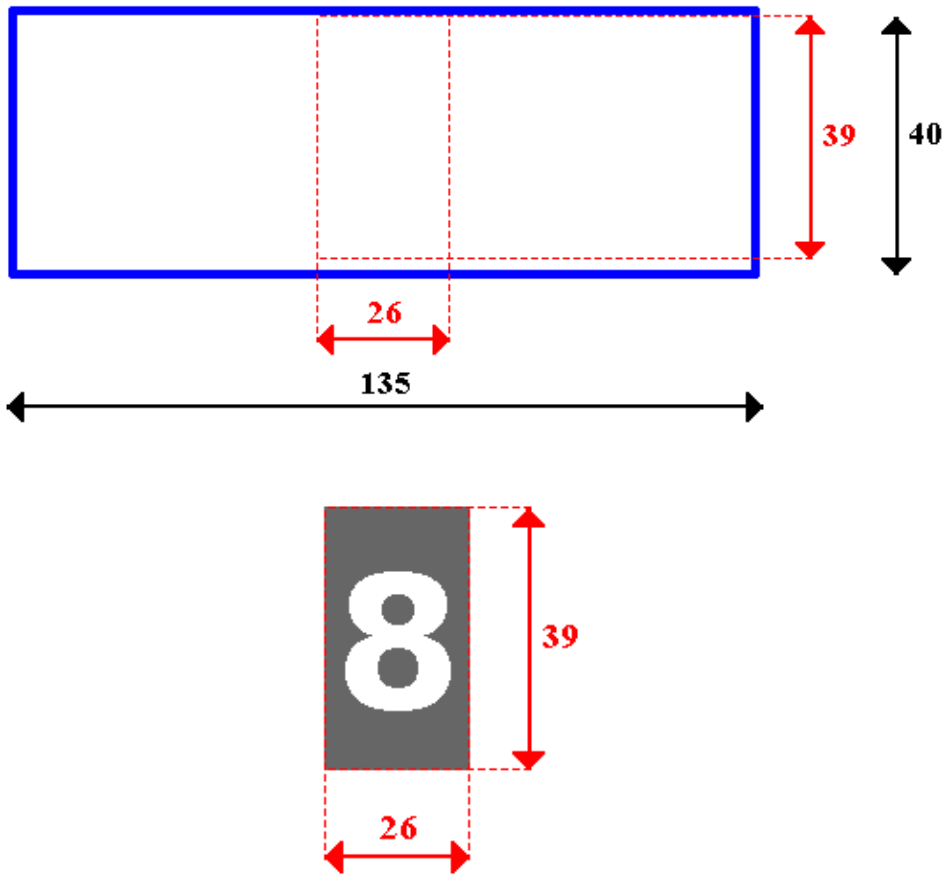


Figura 2.6. Zona de la matriu de pixels sobre la que es treballarà

2.3 Antecedents.

Sobre aquest mateix entorn de desenvolupament s'han realitzat altres projectes, on s'han dissenyat diferents sistemes d'adquisició de la imatge o s'han estudiat diferents protocols de comunicació, com per exemple [Pon03].

En el present cas es parteix d'una arquitectura concreta de l'ASIC, d'un sistema de captura de la imatge dissenyat i provat el seu funcionament i del que es tracta és de centrar-se en l'estudi de diferents arquitectures d'un mòdul concret. Aquest, el mòdul-OCR, tindrà la finalitat d'aconseguir que la quantitat d'informació que es transmet, que posteriorment serà emprada en el procés de reconeixement del dígit, sigui menor que la que es transmetria sense l'esmentat mòdul.

2.4 Condicionants.

Els principals condicionants per la realització del present projecte es detallen a continuació.

2.4.1 Reducció de la quantitat d'informació transmesa.

El principal objectiu per la realització del present projecte és reduir la quantitat d'informació per enviar entre l'ASIC i el punt de recepció de la companyia d'aigua.

La imatge capturada per la càmera sobre la que treballarem consisteix en una matriu de pixels de 39 files i 26 columnes, és a dir 1014 pixels, cadascun dels quals amb una resolució de 8 bits (256 nivells de gris). De manera que transmetre una imatge sense processament implica transmetre 1014 bytes (8112 bits). El requisit principal en la realització del projecte és que un cop realitzat el processat, la quantitat d'informació per transmetre sigui sensiblement menor a l'esmentada quantitat. Aquesta informació ha de contenir suficients característiques de la imatge original de tal manera que permeti obtenir a partir d'ella i amb el corresponent processat de reconeixement, el valor del dígit capturat per la càmera. S'ha de tenir en compte que els possibles dígits per identificar estan acotats entre els caràcters '0' a '9' i amb una font concreta i invariable.

2.4.2 Limitació d'àrea, consum i temps.

Un altre condicionant important per tenir en compte son els límits d'àrea disponible per la implementació del sistema, el consum màxim permès del mateix i el temps màxim en el qual aquest haurà de realitzar el processat de la imatge.

El sistema que es dissenyarà formarà part del circuit integrat ASIC, de forma que el sistema que s'ha de dissenyar té un límit d'àrea relativament elevat. El principal element de l' ASIC és la càmera, la qual ocupa una àrea molt més gran que la que pot ocupar el nostre mòdul i la seva funció no permet modificar la seva estructura geomètrica, ja que aquesta ha de ser la mateixa que la del lector de dades del comptador. Així doncs, les limitacions d'àrea del nostre mòdul no seran geomètriques sinó que vindran donades exclusivament pel cost del silici, que s'afegirà al cost de l'ASIC**.**

La funció que realitzarà el sistema es basa en un sistema d'alimentació aïllat de la xarxa de subministrament (bateries), de forma que la potència que consumeixi el nostre mòdul haurà de ser la menor possible. De fet aquest és el motiu principal pel qual volem reduir el volum de dades a enviar, ja que el mòdul de comunicacions per RF té un consum molt elevat, especialment en el mode de transmissió.

Tal i com s'ha descrit anteriorment, la càmera conserva la imatge un cop capturada durant uns 2 segons, de forma que un cop transcorregut aquest temps, el sistema dissenyat ja no podrà requerir el valor dels pixels. No obstant, el temps màxim en el qual s'haurà d'haver realitzat completament el processat de la imatge és d'un valor relativament elevat, donat que el nostre sistema només realitzarà aquesta tasca amb molt baixa freqüència, de l'ordre d'un cop al dia com a màxim.

2.5 Abast.

El present projecte abastarà el disseny dels algoritmes pel processament de la imatge, la seva descomposició en funcions parcials, la implementació d'aquestes en hardware en llenguatge de descripció de Hardware (VHDL), la síntesis del mateix en tecnologia Austriamicrosystems 0.35 μm C-MOS, amb 4 capes de metalls i 2 de poli-silicis (C35B4), compatible amb TSMC i la simulació per tal de verificar el seu correcte funcionament. També es realitzarà la simulació considerant els retards de les portes lògiques del circuit sintetitzat i finalment es compilaran els circuits implementats sobre una FPGA i es verificarà el correcte funcionament.

3 Metodologia de treball

3.1 Etapes i tasques.

Un pas previ a la consecució dels objectius que es pretenen assolir és la definició de les diferents tasques que es realitzaran. A continuació es relacionen de forma cronològica i es descriuen aquestes tasques.

1) **Estudi previ OCR.** El primer pas és la documentació i familiarització del procediment de reconeixement òptic de caràcters (OCR). Aquesta tasca consisteix en l'estudi dels algoritmes OCR que poden ser emprats amb el format d'imatge del que es disposa i l'anàlisi de les diferents fases d'aquests de cara a la seva implementació parcial en Hardware.

2) **Implementació i test d'algoritmes OCR en llenguatge C.** Posteriorment, es procedirà a la implementació dels algoritmes OCR estudiats en llenguatge C i es generaran arxius executables. Per tal de realitzar el test, s'utilitzaran imatges preses amb la càmera i emmagatzemades com a arxius **.bmp* (mapa de bits). Els programes en C llegiran aquests arxius, n'extrauran la informació de la imatge, realitzaran el corresponent processament i generaran un altre arxiu **.bmp* amb la imatge processada.

3) **Descomposició algoritmes OCR en funcions.** Quan els tests dels algoritmes implementats tinguin èxit, es procedirà a la descomposició d'aquests en les diferents fases dels procediments OCR. Cadascuna d'aquestes fases s'implementarà posteriorment en Hardware per així poder realitzar un disseny on s'implementi de forma més o menys complerta l'algoritme d'OCR.

4) **Implementació en VHDL dels diferents blocs funcionals dels algoritmes OCR.** Aquesta tasca consisteix en implementar en llenguatge de descripció de hardware (VHDL) els diferents blocs que realitzen les funcions corresponents a les distintes fases dels algoritmes OCR. Posteriorment es comprovarà el seu correcte funcionament simulant els blocs implementats a través del programa *Modelsim versió 5.7*. La simulació de la càmera es realitzarà convertint els arxius **.bmp* emprats en la simulació en C a arxius **.dua*. Aquest format és utilitzat en VHDL per simular memòries ROM, i donat que la càmera es comporta en un espai de temps com a tal, aquest arxiu representarà els pixels de la matriu.

5) **Implementació en VHDL d'algoritmes OCR a partir dels blocs funcionals implementats.** Aquesta tasca consisteix en implementar en VHDL diferents algoritmes per realitzar el processament de la imatge emprant diferents combinacions dels blocs implementats. Posteriorment, es comprovarà el seu correcte funcionament.

6) **Síntesis amb tecnologia C35B4 dels algoritmes.** En aquesta fase es realitzarà la síntesis dels algoritmes implementats en VHDL amb la tecnologia Austriamicrosystems 0.35 μm C-MOS (C35B4). Això es farà a través del software *Synopsys Design Analyzer*. Es realitzaran tests per comprovar el seu correcte funcionament, i finalment, es realitzaran tests considerant els retards de les portes lògiques del circuit implementat.

7) **Síntesis dels algoritmes dissenyats sobre FPGA.** Aquesta tasca es realitza amb l'objectiu de comprovar, sobre Hardware físic, el correcte funcionament dels sistemes dissenyats. Consisteix en sintetitzar sobre la FPGA *Apex 20K* de la firma *Altera* aquells algoritmes implementats, la síntesi dels quals no requereixi més recursos que els disponibles per part del dispositiu, i comprovar que el seu comportament és el mateix que el mostrat en els processos de test realitzats anteriorment. A més a més, això permetrà que qualsevol prova amb algun dels algoritmes dissenyats, es pugui realitzar sense que aquest estigui integrat a l'ASIC.

8) **Comparació dels algoritmes dissenyats i determinació del més òptim.** Finalment, a partir dels condicionants imposats per la consecució del projecte, es realitzarà una comparació dels diferents algoritmes implementats i es determinarà el més òptim per les necessitats concretes del nostre sistema.

La Figura 3.1 mostra el diagrama de flux del procés de desenvolupament del projecte.

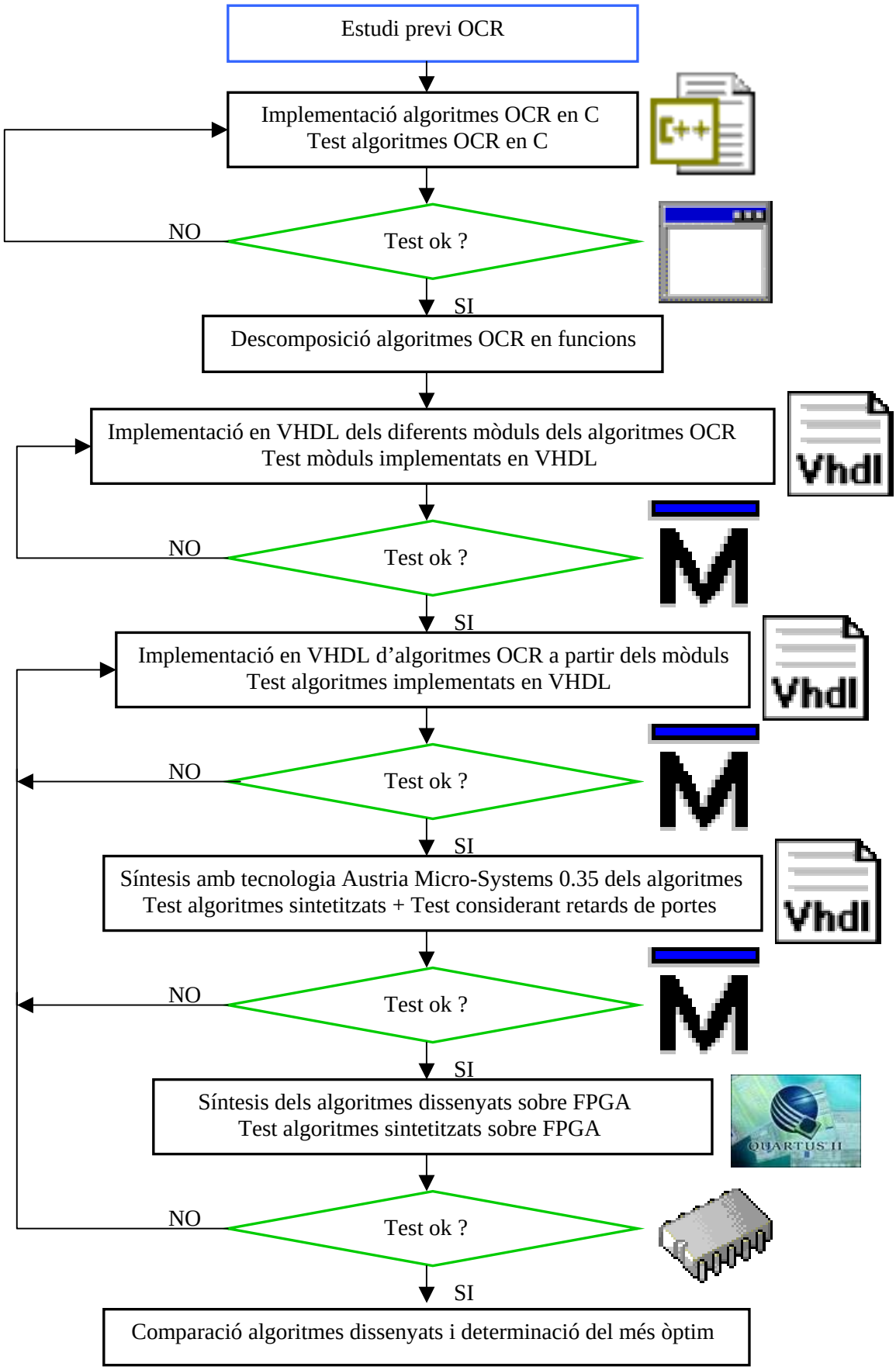


Figura 3.1. Diagrama de flux de les tasques

3.2 Planificació temporal.

La planificació a priori de la distribució temporal de les tasques, és la que es mostra al Diagrama de Gantt de la Figura 3.2.

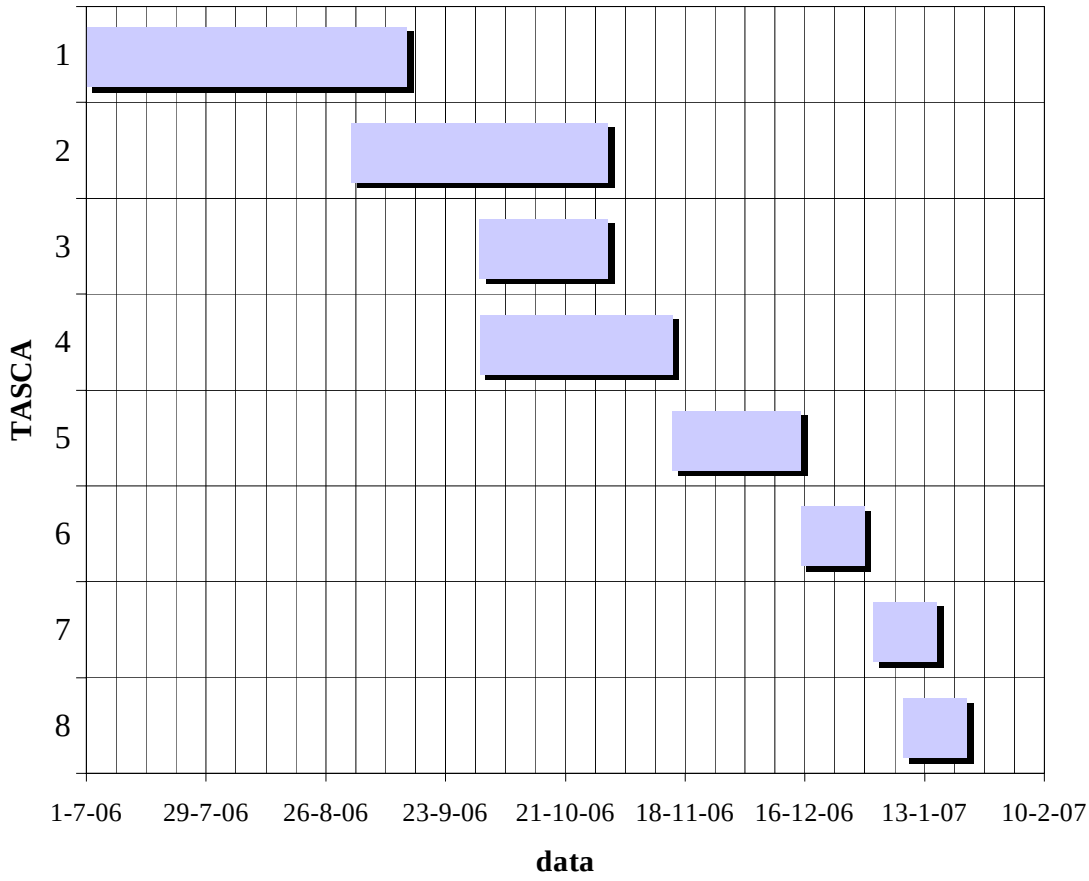


Figura 3.2. Planificació de la distribució temporal de les tasques

4 Reconeixement òptic de caràcters (OCR)

En el present capítol s'introdueixen els conceptes bàsics sobre el procés de reconeixement òptic de caràcters (OCR), els quals s'utilitzaran durant el desenvolupament del projecte.

4.1 Introducció al reconeixement òptic de caràcters.

Tot seguit es descriu què és un Sistema de Reconeixement Òptic de Caràcters (OCR) genèric, a través de la descripció dels algoritmes que s'utilitzen en el desenvolupament d'un sistema d' OCR.

El Reconeixement Òptic de Caràcters és un mètode per reconèixer la part textual d'una imatge digitalitzada. L'OCR rep com entrada la imatge digitalitzada i el resultat és un arxiu de text que pot ser editat i usat com a tal per qualsevol programa o aplicació que ho necessiti.

El Reconeixement Òptic de Caràcters d'una imatge binària, és a dir, d'una imatge on tots els pixels son només blancs o negres, consistirà a reconèixer els caràcters que apareixen a la imatge que han de crear el fitxer de text. El reconeixement d'aquests caràcters es realitzarà bàsicament amb la comparació de cada caràcter de la imatge amb uns patrons o plantilles que contindran tots els possibles caràcters.

Però les imatges reals no són perfectes i quan aquestes imatges són escanejades, pas previ a la majoria d'aplicacions dels algoritmes de OCR, s'introdueixen algunes alteracions, motiu pel qual l' OCR es troba amb diverses dificultats com les següents:

A) El dispositiu, a través del qual s'obté la imatge, pot introduir nivells de grisos en el fons de la imatge donant lloc a nivells de grisos que no pertanyen a la imatge real.

B) La resolució finita d'aquests dispositius pot afectar als pixels que han de ser avaluats.

C) La posició de la imatge a la pàgina escanejada és determinant, ja que un mateix símbol no ocupa el mateix espai horitzontalment que verticalment.

D) La connexió de dos o més caràcters per pixels comuns. Per exemple, quan dos caràcters estiguin units provoquen que existeixin pixels que pertanyin tant a un com a l'altre caràcter.

E) La separació dels caràcters. La no existència d'un espai fix entre caràcters pot provocar errors a l'hora del reconeixement.

F) Soroll a la imatge. El dispositiu que obté la imatge pot inserir nivells de grisos dintre dels pixels que haurien de ser negres i petites regions negres dintre dels pixels que haurien de ser blancs.

La Figura 4.1 mostra els efectes d'aquestes alteracions.

Degut a tots aquests problemes a l'hora de reconèixer els diferents caràcters, un sistema OCR ha de tenir en compte la informació de l'entorn.

Tots els algoritmes d'OCR persegueixen la finalitat de poder reconèixer un text per a poder-lo tractar com a tal posteriorment. Els algoritmes d'OCR es basen en cinc passos fonamentals:

- Captura de la imatge.
- Segmentació de la imatge.
- Pre-processament de la imatge.
- Obtenció de característiques del caràcter.
- Comparació de les característiques amb patrons dels possibles caràcters.

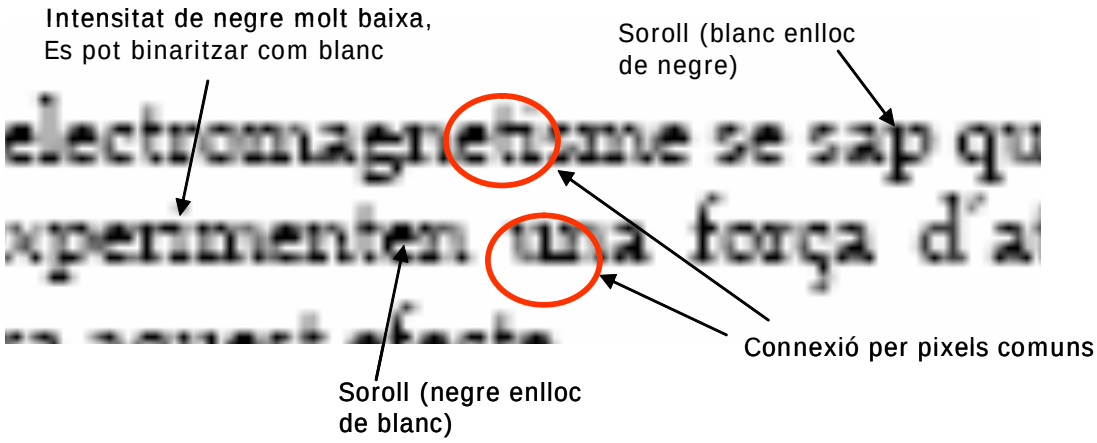


Figura 4.1. Alteracions en les imatges que dificulten la tasca de l'OCR

4.2 Procediment per al reconeixement òptic de caràcters.

A continuació es detallen les diferents fases que realitza un algoritme genèric OCR. En el nostre disseny s'implementaran algunes d'aquestes fases, mentre que d'altres es desestimaran per no ser necessàries o possibles.

4.2.1 Captura de la imatge.

La funció de capturar la imatge consisteix en convertir la llum en senyals electrònics sobre els quals es treballarà. En el nostre cas aquesta fase és realitzada per la càmera descrita a l'apartat 2.2.1.

4.2.2 Segmentació.

Una vegada obtinguda la imatge, aquesta s'ha de fragmentar en les diferents zones que la componen, on a cadascuna d'elles hi hagi un caràcter. Aquest procés constitueix un dels més complexes en cas que el text que es pretén identificar estigui escrit a mà o la seva font no sigui invariable.

La segmentació és l'operació que permet la descomposició d'un text en diferents entitats lògiques. Amb la segmentació s'ha de localitzar les zones d'interès i separar-les. Les zones d'interès estan caracteritzades per uns atributs comuns. No existeix cap mètode genèric per a realitzar la segmentació de la imatge que sigui prou eficaç per a l'anàlisi d'un text. Els mètodes majoritàriament utilitzats són variacions del mètode de Projeccions Verticals. Un altre possibilitat és utilitzar un mètode que segmenti la imatge en entitats que no tinguin cap punt en comú ("Tsujimoto"). En imatges perfectes donaria un resultat favorable. No obstant això, en la realitat i tenint en compte el soroll, això no es produeix. Tots aquests mètodes no poden ser aplicats al cent per cent, i es fa necessari l'estudi de l'entorn per poder realitzar una segmentació a mida i correcta.

En el nostre cas els caràcters per reconèixer estan acotats entre el '0' i '9', amb una font concreta i invariable, i situats a la mateixa posició horitzontal de la imatge cadascun d'ells. De manera que el nostre sistema no realitzarà aquest pas, ja que la càmera està dissenyada amb un format específic que conté cada dígit d'una manera aïllada. L'únic problema que podem trobar és que el dígit no estigui sempre en la mateixa posició vertical degut a que la rotació del comptador pot ser imperfecta i l'aturada per indicar un valor pot produir-se en alçades diferents segons comptador o dígit.

4.2.3 Pre-processament de la imatge.

Prèviament a l'extracció de característiques del caràcter, que posteriorment seran emprades en el procés d'identificació, hi ha una sèrie de procediments que es realitzen per tal d'obtenir una imatge més nítida.

Binarització de la imatge.

La major part dels algorismes OCR estan implementats a partir d'imatges binàries, per la qual cosa és necessari el pas d'una imatge en nivells de grisos (o colors) a una binària, només amb pixels blancs o negres. A més a més, això permet reduir el volum de les dades a tractar.

La binarització d'una imatge digital consisteix en convertir la imatge digital en una imatge en blanc i negre, de tal manera que es preservin les propietats essencials de la mateixa.

Abans de realitzar la binarització pròpiament dita, s'ha de determinar el valor adequat dintre dels nivells de grisos que marcarà la frontera (llindar). Tots els nivells de grisos més foscos que el nivell llindar calculat es convertiran en negre i tots els més clars en blanc.

Hi ha diversos mètodes per obtenir el llindar.

1) Un dels mètodes és mitjançant l'histograma d'aquesta imatge. A través de l'histograma obtenim una gràfica on es mostren el nombre de pixels per cada nivell de gris que apareixen a la imatge. S'ha de triar un llindar de forma que l'histograma formi una vall en aquest nivell.

2) Un altre mètode semblant a l'anterior, consisteix en partir de la gràfica de l'histograma per determinar el nivell de gris tal que el nombre de pixels amb colors més clars i més foscos a aquest siguin iguals. Amb aquest mètode la imatge binària resultant té aproximadament igual nombre de pixels blancs i negres.

3) El mètode més senzill i més emprat en sistemes hardware de petites dimensions, consisteix en considerar com a llindar el valor mitjà entre el valor del pixel més clar i més fosc de la imatge.

La Figura 4.2 mostra un exemple d'histograma i la ubicació del llindar calculat pels tres mètodes.

En el nostre cas, i donat que la implementació del primer mètode requereix bastants recursos hardware, treballarem amb els altres dos mètodes.

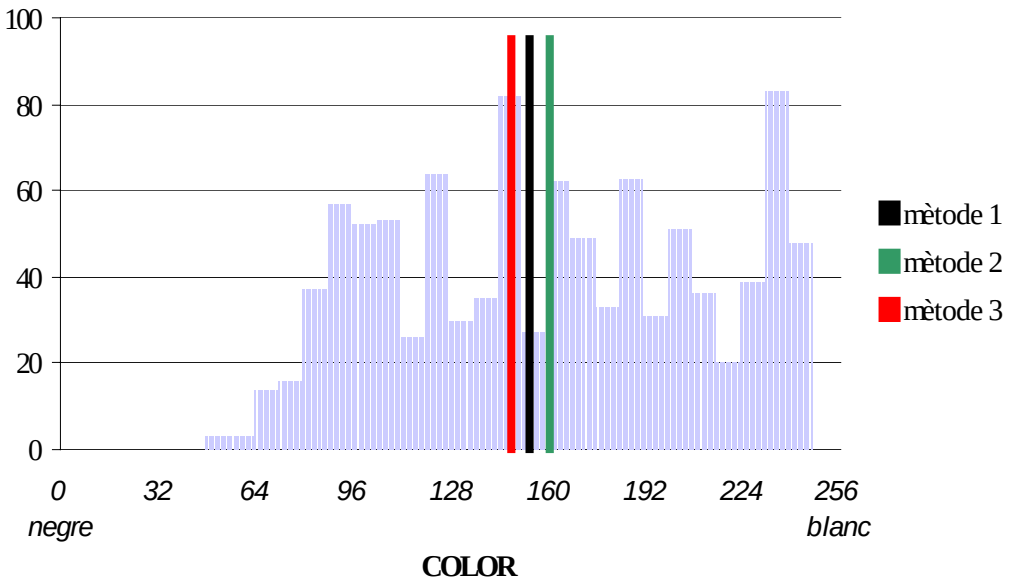


Figura 4.2. Càlcul del pixel llindar segons mètode

Filtrat de la imatge.

El procés de filtrat de la imatge, s'utilitza per la modificació d'aquesta per l'eliminació de soroll. Hi ha molts tipus de filtres aplicables en el processat d'imatges [Pro96].

En el nostre cas el possible filtre a utilitzar que més s'escau a les nostres necessitats és l'anomenat Filtre de Mitjana [Sch].

El Filtre de Mitjana és un filtre no lineal que té el següent funcionament. Per cada pixel es prenen els valors dels 8 pixels adjacents a ell. Aquests pixels més el propi, formen un vector de 9 pixels. Es prenen aquests valors i s'ordenen de menor a major. El filtre assigna al pixel en qüestió, el valor del pixel mitjà, és a dir el del pixel numero 4 del vector de pixels (veure exemple a la Figura 4.4). En aquest cas el vector és [1,2,3,4,5,6,7,8], per tant el valor del pixel mitjà és 4 i no 5 com seria si és calculés el valor promig.

Aquesta tasca s'ha de realitzar per tota la imatge abans de modificar cap pixel, de forma que els valors dels pixels filtrats s'han d'emmagatzemar apart i un cop finalitzat el procés, actualitzar la imatge sencera. Els pixels dels extrems queden exclosos d'aquest procés de filtrat. La Figura 4.3 mostra per un pixel (x, y), els pixels considerats en el procés del Filtre de Mitjana.

Aquest procés té un efecte de difuminat de la imatge i permet obtenir una eliminació de soroll de forma eficaç.

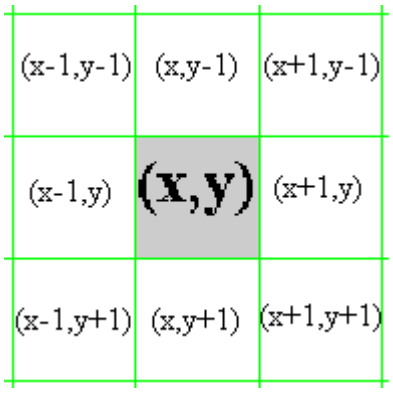


Figura 4.3. Pixels considerats en el procés del Filtre de Mitjana

5	3	6		x	x	x
2	**1**	9	→	x	**4**	x
8	4	7		x	x	x

Figura 4.4. Exemple d'aplicació del Filtre de Mitjana

Aprimament del caràcter.

Una vegada realitzats els processos de binarització i filtrat, es realitza un procés d'aprimament de cadascuna de les components connexes de la imatge. Una component connexa és un conjunt de pixels negres tal que per a qualsevol parell de pixels del conjunt, existeix un camí digital (de pixels negres) que els uneix.

Com es troba detallat a [GTC], el procediment d'aprimament consisteix en anar esborrant successivament els punts de l'extrem exterior de cada component connexa, de manera que es preservi la seva topologia.

Les condicions que determinen si un punt es pot esborrar estan relacionades amb el concepte de punt simple (un pixel negre P_X de la vora de la imatge es considera simple si el conjunt dels seus veïns negres formen part de la mateixa component connexa de la que forma part P_X) i punt final (un punt és final si té un únic veí negre, o el que és el mateix, un punt extrem de la imatge). És a dir, un punt de l'extrem exterior de cada component es pot eliminar si és simple i no és final. La definició de veïnatge es pot prendre com a relativa a 4ª o 8ª adjacència en el cas, com el present, de pixels modelats com a quadrats. La diferència d'aquestes dues definicions és el nombre de

pixels que es consideren veïns d'un pixel determinat. La Figura 4.5 mostra els pixels considerats veïns del pixel (x, y) en cas de 4ª o 8ª adjacència. La 4ª adjacència considera veïns aquells que tenen un costat comú mentre que la 8ª considera als corresponents a la 4ª i a més a més els que tenen un vèrtex en comú.

L'esborrat de punts ha de seguir un esquema d'escombrats successius perquè la imatge segueixi tenint les mateixes proporcions que l'original i no quedi deformada. L'esborrat en cada iteració s'ha de fer en paral·lel, és a dir, assenyalar els pixels esborrables i posteriorment eliminar-los tots alhora. Aquest procés s'ha de fer per cada extrem de la imatge, és a dir, prenent les vores de la figura en les direccions nord, est, sud i oest. Un cop determinats els pixels que s'esborraran es realitza la seva eliminació alhora, i es torna a començar el procés. El procés acaba quan en una iteració complerta, no s'ha detectat cap pixel esborrable. A la Figura 4.6 podem veure el resultat d'un procés complert d'aprimament d'una imatge, i la Figura 4.7 mostra una iteració d'un procés d'aprimament parcial pel nord d'una imatge.

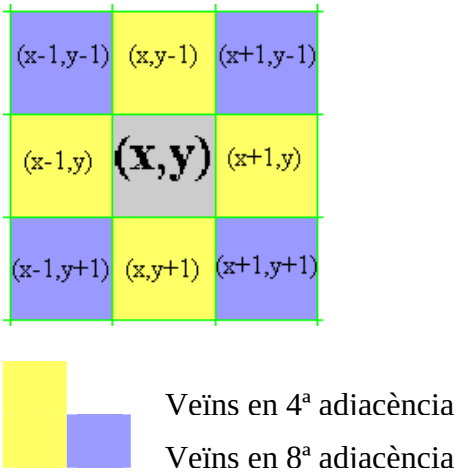


Figura 4.5. Veïnatge digital segons adjacència considerada

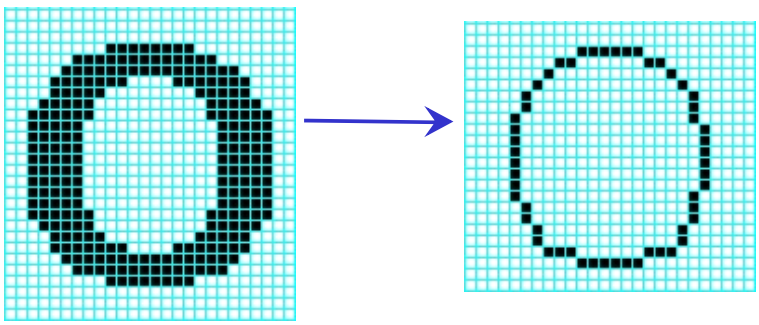


Figura 4.6. Resultat de l'aprimament complert d'una imatge

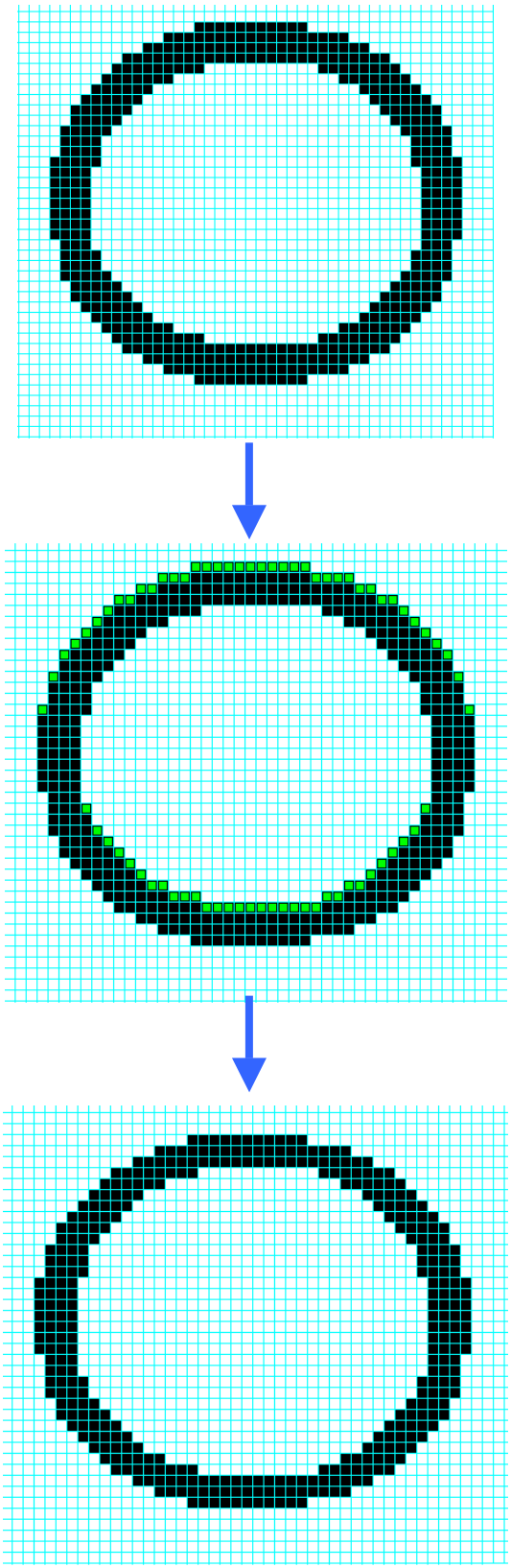


Figura 4.7. Procés d'aprimament nord d'una imatge

4.2.4 Obtenció de característiques del caràcter.

Una vegada realitzat l'aprimament de la imatge, s'ha de realitzar l'obtenció de característiques que posteriorment seran comparades amb patrons o plantilles.

Existeixen diversos mètodes per l'obtenció de característiques d'un caràcter:

- **Mètodes geomètrics o estadístics:** Consisteixen en obtenir dels caràcters extrets de la imatge, un conjunt de *m* mesures que constitueixen la composició d'un vector d'un espai de representació *Rm* de dimensió *m*. Aquestes mesures solen ser elevades en nombre ($m>100$). Els seus components fonamentals són mesures topològiques i mètriques, com superfícies, regions, perfils, concavitats, bucles, interseccions, etc.

- **Mètodes estructurals:** Aquests mètodes passen per una esqueletització dels caràcters extrets de la imatge, de manera que passen per una detecció de contorns interiors i exteriors per a detectar una sèrie de punts singulars, d'aquests punts s'extraurà una informació topològica. Posteriorment, es passarà a una vectorització, que permetrà representar una descripció en forma de cadena de símbols o de gràfics.

- **Mètodes Neuro-Mimètics:** Estan basats en la utilització de xarxes neuronals de tot tipus. La mecànica general és partir d'una imatge amb caràcters normalitzats o de símbols. Els resultats obtinguts, són bons amb la condició de disposar de grans bases de dades d'aprenentatge. Com a contrapartida, els temps d'aprenentatge seran elevats i a més seria molt difícil determinar les causes d'errors d'aquests mètodes.

- **Mètodes Markovians:** Es basen a fer taules de recerca de seqüències de senyals de caràcters però variables en el transcurs del temps. És una mica semblat a una memòria "caché" de reconeixement de caràcters. Això quant a emmagatzematge, però el procés de reconeixement, combina l'ús d'un Graf i un procés aleatori en les transicions del Graf així com en la seva re-alimentació.

- **Mètodes basats en IA:** Aquests mètodes apliquen sistemes de decisió a base de regles al procés de reconeixement. Són algorismes de recerca d'arbres que se sustenten la base d'algorismes clàssics.

- **Mètodes de Zadeh:** Aquests mètodes estan basats en la lògica difusa. Estan molt bé adaptats al reconeixement de caràcters amb un tipus de dades imprecís. Utilitzen Màquines d'Estats o regles de “Zadeh”.

- **Mètodes Mixtos:** Consisteixen en una combinació d'alguns dels mètodes anteriors, com per exemple un sistema CTRBF, que combina un arbre de decisió amb una xarxa neuronal.

4.2.5 Comparació de característiques amb patrons.

Una vegada realitzada l’obtenció de característiques, s'haurà de realitzar la comparació amb els patrons.

En aquesta etapa s'haurà de comparar els caràcters obtinguts amb uns caràcters teòrics emmagatzemats en una base de dades.

Aquest pas és bastant important, ja que el bon funcionament del OCR es deu en gran mesura a una bona definició d'aquesta etapa. Per a poder realitzar aquest pas, s'ha de disposar a l’OCR d'una base de dades en la qual s'han d'emmagatzemar tots els caràcters que han de ser reconeguts per l’esmentat OCR. En aquesta base de dades s'emmagatzemarà, per tant, tots els patrons o plantilles.

4.3 Mètodes d’obtenció de característiques de caràcters.

4.3.1 Mètode de Crossings.

El mètode de “Crossings” o encreuaments consisteix en calcular el nombre de vegades que hi ha un pas de BLANC a NEGRE per cada fila i per cada columna de la imatge, obtenint així un vector de (n + m) elements que descriu la imatge, on n és el nombre de columnes i m el nombre de files. Per tal de realitzar aquest procés la imatge ha d’haver sigut prèviament binaritzada.

La Figura 4.8 il·lustra el resultat de l’obtenció d’encreuaments horitzontals (1 x fila) i verticals (1 x columna) d’una imatge amb la figura d’un dígit ‘1’.

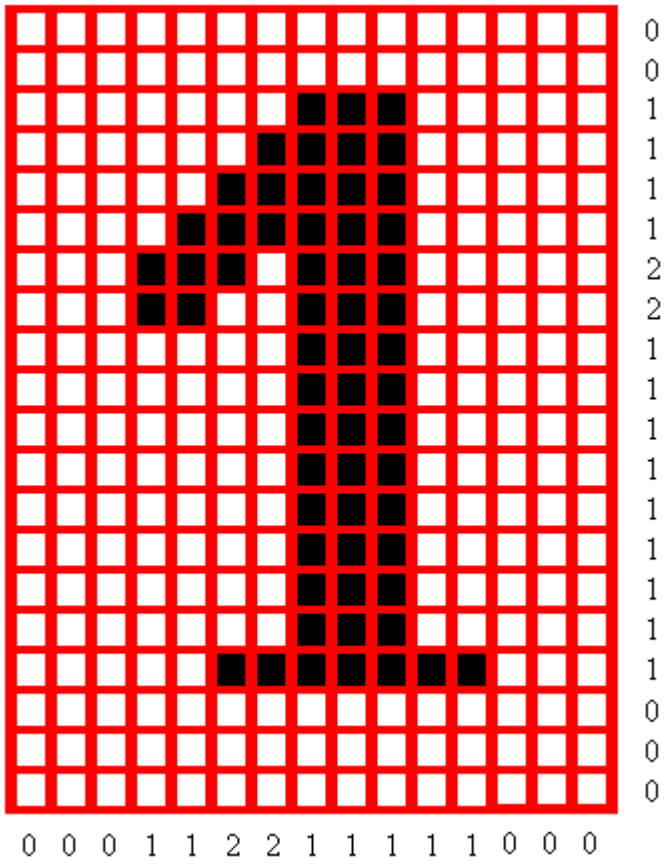


Figura 4.8. Exemple d'aplicació del mètode de "Crossings"

4.3.2 Mètode de Zoning.

El mètode de "Zoning" o zonificació de la imatge, que vol dir delimitació de la imatge per zones, consisteix en el següent. Es descompon la imatge en zones de més d'un pixel. A cada zona se li assigna un valor en funció del nivell gris mitjà, o el que és el mateix, en funció del color promig de la zona. El nombre d'elements del vector resultant depèn del nombre de zones de la imatge, que depèn alhora del tamany d'aquestes.

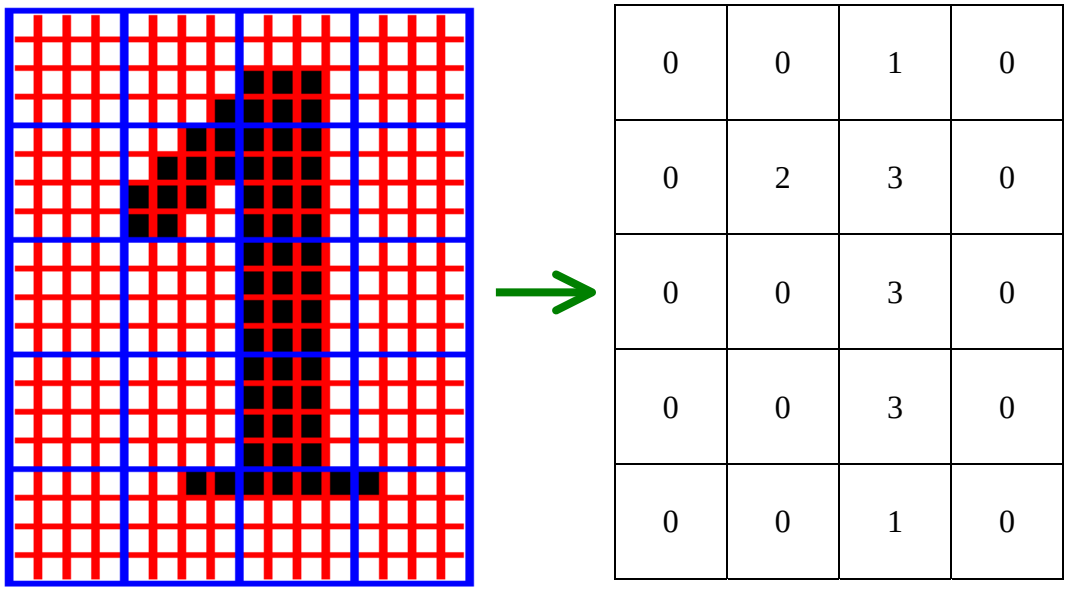
El rang dels valors assignats a cada zona determinarà la resolució, o el que és el mateix, la quantitat d'informació que es perd. La Figura 4.9 mostra un exemple d'imatge binària en la que s'ha aplicat la tècnica de "Zoning". A cada zona, de 4 x 4 pixels, se li assignen valors en funció del nombre de pixels negres que hi ha, segons la següent funció.

$$f_Z = 0 \quad si \quad n_{NEGRES} < 4$$

$$f_Z = 1 \quad si \quad 4 \leq n_{NEGRES} < 8$$

$$f_Z = 2 \quad si \quad 8 \leq n_{NEGRES} < 12$$

$$f_Z = 3 \quad si \quad n_{NEGRES} \geq 12$$



0	0	1	0
0	2	3	0
0	0	3	0
0	0	3	0
0	0	1	0

Figura 4.9. Exemple d'aplicació del mètode de "Zoning"

4.3.3 Altres mètodes.

Hi ha altres mètodes per obtenir característiques dels caràcters que es vol identificar. Un d'ells és el mètode de moments geomètrics, que es basa en definir un centre a la imatge i trobar els eixos principals dels moments d'inèrcia, la longitud del caràcter i la orientació d'aquest. Altres es basen en transformacions matemàtiques, com per exemple el mètode Karhunen-Loeve o el de la Transformada de Hough.

Aquests mètodes son força complexos i estan orientats principalment al reconeixement de caràcters escrits a mà. Donat que en el nostre cas els caràcters per reconèixer estan molt acotats, son impresos i amb una font concreta i invariable, en el present projecte es treballarà amb els mètodes de "Crossings" i "Zoning".

5 Disseny i implementació

5.1 Disseny software i test d'algoritmes de reconeixement de caràcters.

La primera fase del disseny consisteix en la implementació de diferents algoritmes OCR en llenguatge C i la generació d'arxius executables. Amb aquests executables es processaran imatges obtingudes amb la càmera, de les que disposem en format d'arxius *.*bmp* (mapa de bits). Els programes en C llegiran aquests arxius, n'extrauran la informació de la imatge, per la qual cosa és necessari conèixer el format *bmp*, i realitzaran el corresponent processament. També generaran un altre arxiu *.*bmp* amb la imatge processada per poder observar el resultat del procés.

5.1.1 Algoritmes basats en el Mètode de Crossings.

S'han implementat 5 diferents algoritmes basats en el mètode d'encreuaments ("Crossings") per realitzar la identificació del dígit. A continuació s'exposa una breu descripció de les funcions que realitzen, així com els diagrames de flux.

Algoritme OCR_crossings.

L'algoritme *OCR_crossings* és el més simple. El seu funcionament es pot resumir en els següents punts:

- Obté el valor del pixel llindar de la imatge com a valor mitjà entre el valor dels pixels més fosc i més clar de la imatge.
- Binaritza la imatge amb el llindar calculat.
- Per identificar el dígit aplica la tècnica de "Crossings". Obté el nombre de vegades que hi ha un pas de BLANC a NEGRE per cada fila i per cada columna, obtenint així un vector de (26+39) elements que descriu la imatge.
- Compara aquest vector amb els corresponents a cada imatge patró (dígits 0 al 9) i obté l'error corresponent per cada patró.
- Determina que el dígit és el corresponent al patró amb menys error al comparar-lo amb la imatge.

La Figura 5.1 en mostra el diagrama de flux.

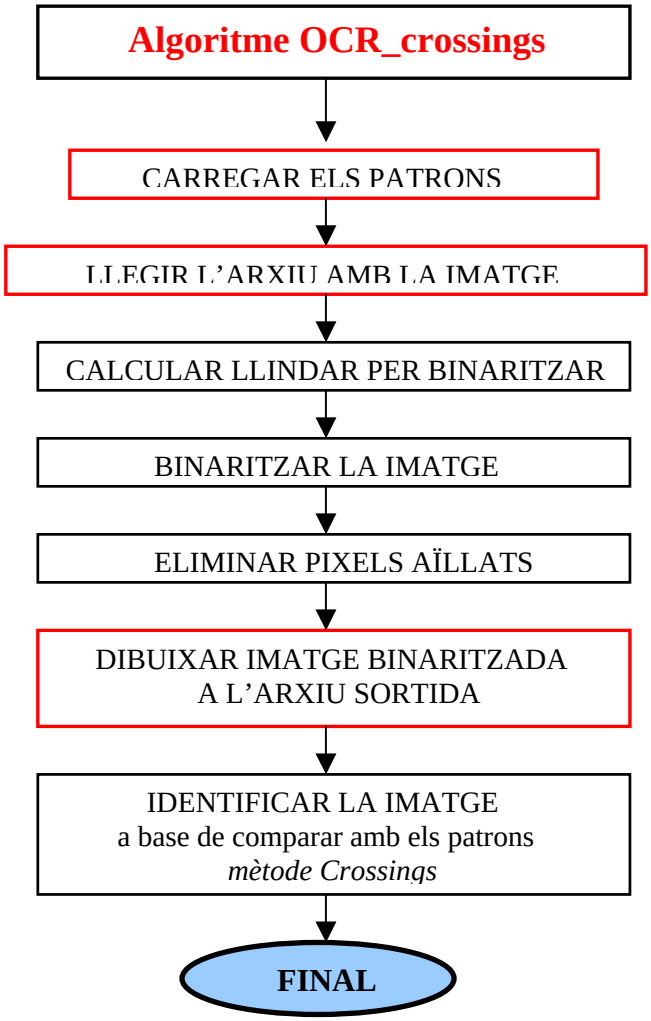


Figura 5.1. Diagrama de flux de l'algoritme OCR_crossings

Algoritme OCR_crossings_filtre.

L'algoritme *OCR_crossings_filtre* realitza el mateix que l'anterior, però aplica un Filtre de Mitjana a la imatge abans de processar-la. Resumint, el funcionament de l'algoritme seria el següent:

- Aplica a la imatge un Filtre de Mitjana, consistent a assignar a cada pixel (excepte els situats als extrems de la imatge) el valor mitjà dels vuit pixels que l'envolten i el propi pixel.
- Obté el valor del pixel llindar de la imatge com a valor mitjà entre el valor dels pixels més fosc i més clar de la imatge filtrada.
- Binaritza la imatge amb el llindar calculat.

- Per identificar el dígit aplica la tècnica de “Crossings”. Obté el nombre de vegades que hi ha un pas de BLANC a NEGRE per cada fila i per cada columna, obtenint així un vector de (26+39) elements que descriu la imatge.
- Compara aquest vector amb els corresponents a cada imatge patró (dígits 0 al 9) i obté l’error corresponent per cada patró.
- Determina que el dígit és el corresponent al patró amb menys error al comparar-lo amb la imatge.

La Figura 5.2 en mostra el diagrama de flux.

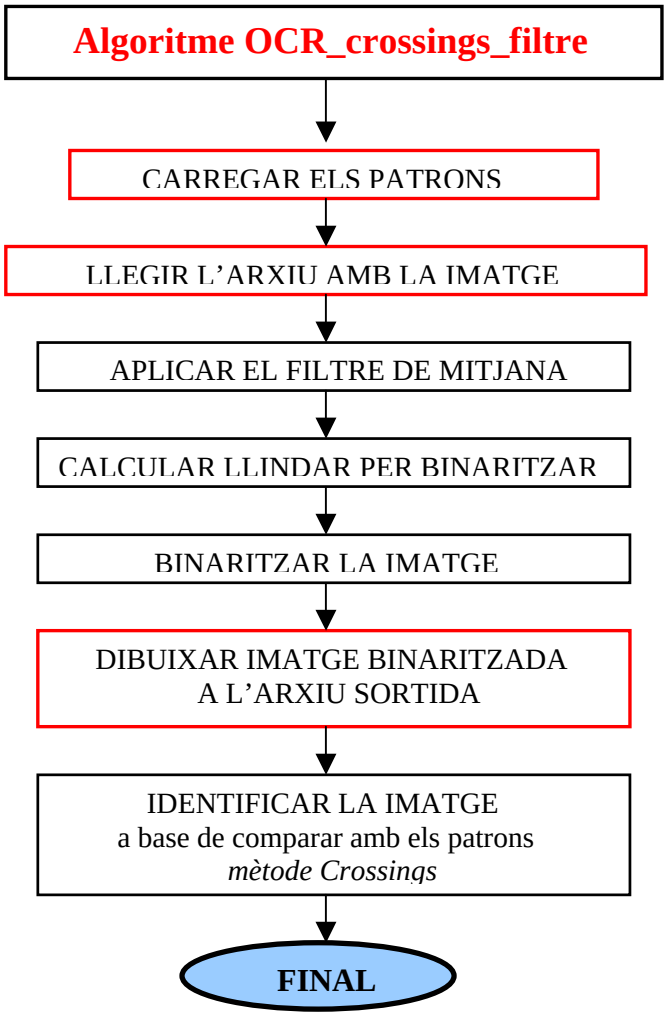


Figura 5.2. Diagrama de flux de l'algoritme OCR_crossings_filtre

Algoritme OCR_crossings_filtre_bin.

L'algoritme *OCR_crossings_filtre_bin* és similar a l'anterior. La principal diferència és que aquest aplica el Filtre de Mitjana només als pixels negres i un cop la imatge ja ha sigut binaritzada. Això equival a esborrar els pixels negres que no tinguin, com a mínim, 4 veïns negres, entenent com a definició de veïnatge la corresponent a la 8ª adjacència. Les funcions realitzades per l'algoritme *OCR_crossings_filtre_bin* son les següents:

- Obté el valor del pixel llindar de la imatge com a valor mitjà entre el valor dels pixels més fosc i més clar de la imatge filtrada.
- Binaritza la imatge amb el llindar calculat.
- Aplica a la imatge un Filtre de Mitjana, considerant només els pixels negres.
- Per identificar el dígit aplica la tècnica de "Crossings". Obté el nombre de vegades que hi ha un pas de BLANC a NEGRE per cada fila i per cada columna, obtenint així un vector de (26+39) elements que descriu la imatge.
- Compara aquest vector amb els corresponents a cada imatge patró (dígits 0 al 9) i obté l'error corresponent per cada patró.
- Determina que el dígit és el corresponent al patró amb menys error al comparar-lo amb la imatge.

La Figura 5.3 en mostra el diagrama de flux.

Si observem les imatges resultants del procés d'aquest algoritme, i les comparem amb les resultants de l'anterior, podem comprovar que els resultats son pràcticament idèntics. Això ens permet concloure que obtenint els mateixos resultats, i essent el present algoritme més fàcilment implementable en hardware, la millor opció, en cas d'haver d'incloure filtrat al disseny, és la del filtrat post-binaritzador en lloc de pre-binartitzador.

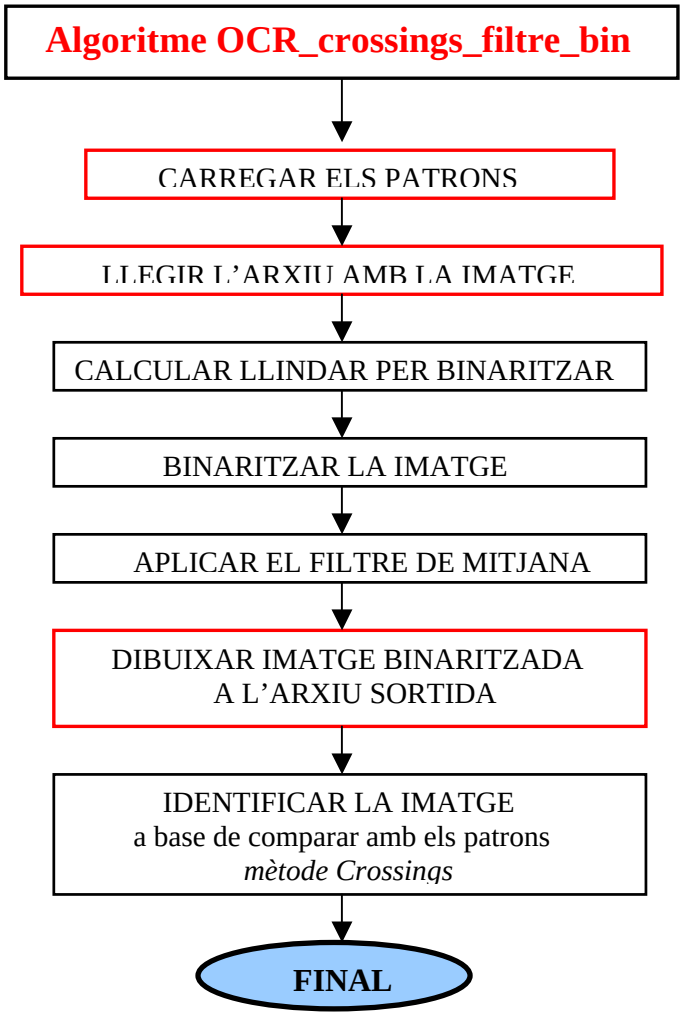


Figura 5.3. Diagrama de flux de l'algoritme OCR_crossings_filtre_bin

Algoritme OCR_crossings_histograma.

L'algoritme *OCR_crossings_histograma* calcula el valor del llindar a partir de l'histograma de la imatge. El procediment consisteix en comptar el nombre de vegades que un color concret dels 256 possibles es repeteix a la imatge. A partir d'aquí s'obté l'histograma de la imatge. Determina el llindar tal que la freqüència acumulada dels pixels de valor inferior al llindar sigui igual a la dels pixels de valor superior.

Les funcions realitzades per l'algoritme son les següents:

- Obté el valor del pixel llindar de la imatge a partir de l'histograma.
- Binaritza la imatge amb el llindar calculat.

- Per identificar el dígit aplica la tècnica de “Crossings”. Obté el nombre de vegades que hi ha un pas de BLANC a NEGRE per cada fila i per cada columna, obtenint així un vector de (26+39) elements que descriu la imatge.
- Compara aquest vector amb els corresponents a cada imatge patró (dígits 0 al 9) i obté l’error corresponent per cada patró.
- Determina que el dígit és el corresponent al patró amb menys error al comparar-lo amb la imatge.

La Figura 5.4 en mostra el diagrama de flux.

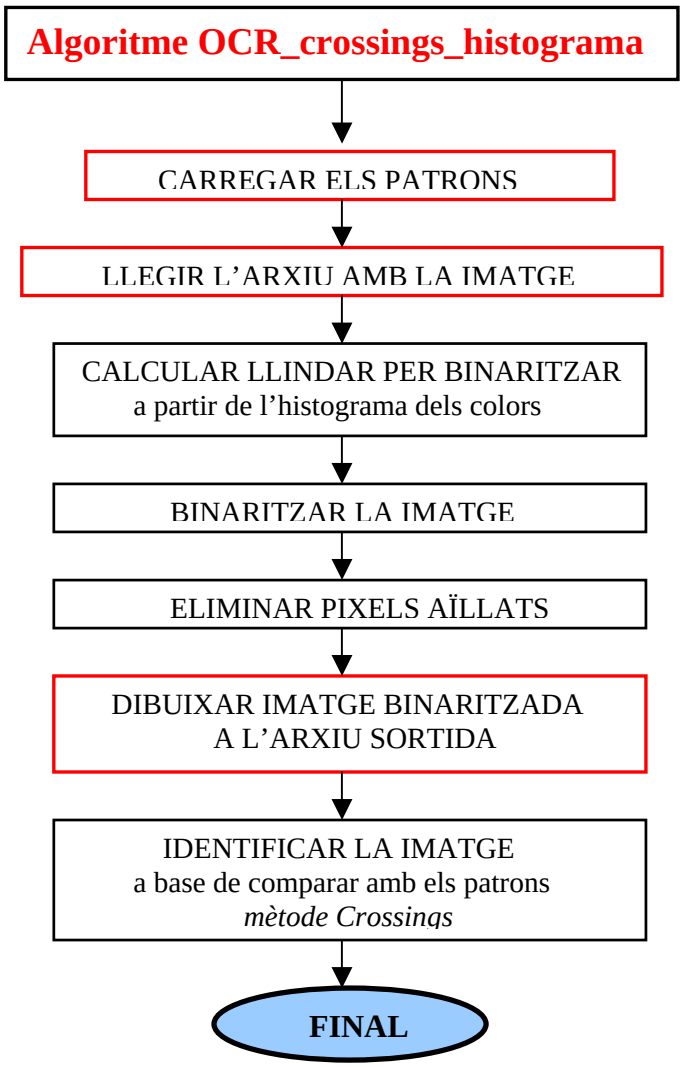


Figura 5.4. Diagrama de flux de l'algoritme OCR_crossings_histograma

Aquest algoritme també ens permet observar l'histograma de les imatges de cadascun dels dígits i comparar el valor del llindar calculat amb aquest mètode i el calculat amb el mètode del pixel promig. Observant el resultat apreciem la gran similitud entre els valors llindars calculats pels dos mètodes, el que ens fa decantar-nos per treballar amb el mètode promig en el càlcul de llindar, de cara al disseny en hardware.

Algoritme OCR_crossings_aprimament.

L'algoritme *OCR_crossings_aprimament* realitza un procés d'aprimament a la imatge, posteriorment al procés de binarització. D'aquesta manera, al obtenir el nombre d'encreuaments (horitzontals i verticals), aquests tindran uns valors molt més fiables i propers a la figura ideal corresponent al dígit en qüestió. En resum, l'algoritme realitza les següents funcions:

- Obté el valor del pixel llindar de la imatge com a valor mitjà entre el valor dels pixels més fosc i més clar de la imatge filtrada
- Binaritza la imatge amb el llindar calculat.
- Aprima la imatge de forma que només en quedi l'esquelet. L'aprimament es realitza amb 4ª adjacència.
- Per identificar el dígit aplica la tècnica de "Crossings". Obté el nombre de vegades que hi ha un pas de BLANC a NEGRE per cada fila i per cada columna, obtenint així un vector de (26+39) elements que descriu la imatge.
- Compara aquest vector amb els corresponents a cada imatge patró (dígits 0 al 9) i obté l'error corresponent per cada patró.
- Determina que el dígit és el corresponent al patró amb menys error al comparar-lo amb la imatge.

La Figura 5.5 en mostra el diagrama de flux.

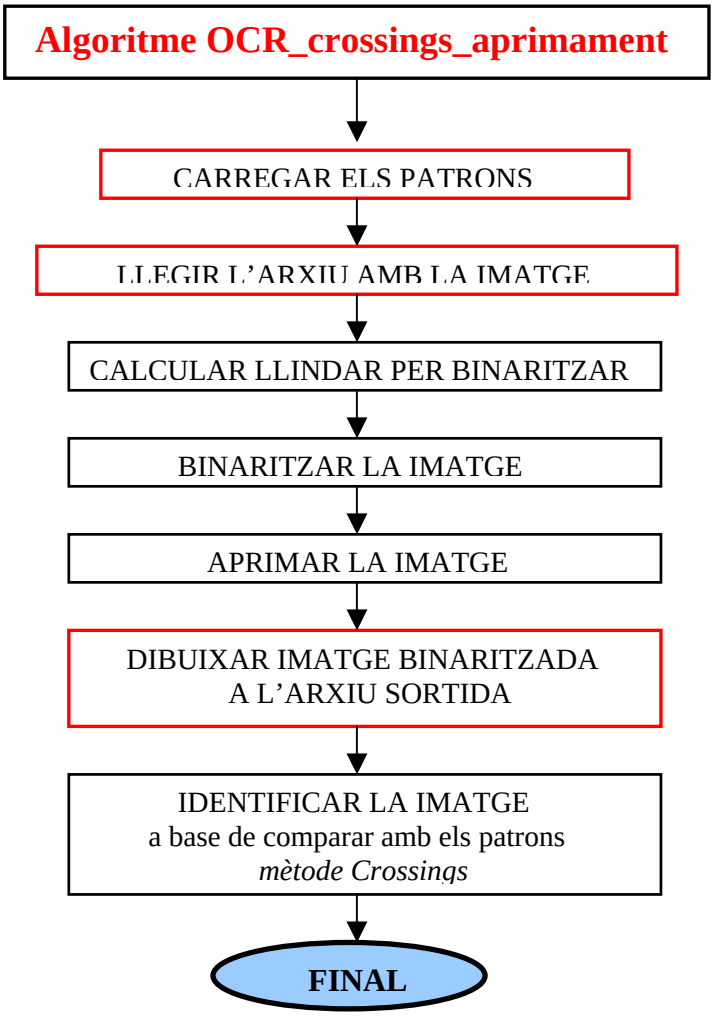


Figura 5.5. Diagrama de flux de l'algoritme OCR_crossings_aprimament

5.1.2 Algoritmes basats en el Mètode de Zoning.

S'han implementat 2 algoritmes basats en el mètode de zonificació de la imatge ("Zoning") per realitzar la identificació del dígit. A continuació s'exposa una breu descripció de les funcions que realitzen, així com els diagrames de flux.

Algoritme OCR_zoning.

L'algoritme *OCR_zoning* simplement zonifica la imatge, prenent com a trams de colors els delimitats pels valors 000h, 17Fh, 2FEh, 47Dh i 5FAh, que corresponen als valors 00h, 40h, 80h, C0h i FFh augmentats un factor 6.

Les funcions que realitza, es resumeixen a continuació:

- Per identificar el dígit aplica la tècnica de “Zoning”.
- Descompon la imatge en zones de 6 pixels (3 files i 2 columnes).
- A cada zona li assigna un valor de 0 a 3 segons el valor de la suma dels seus 6 pixels, emprant com a llindars els valors 17Fh, 2FEh i 47Dh. D’aquesta manera s’obté un vector de 169 elements (13 files * 13 columnes) de 4 bits.
- Compara aquest vector amb els corresponents a cada imatge patró (dígits 0 al 9) i obté l’error corresponent per cada patró.
- Determina que el dígit és el corresponent al patró amb menys error al comparar-lo amb la imatge.

La Figura 5.6 en mostra el diagrama de flux.

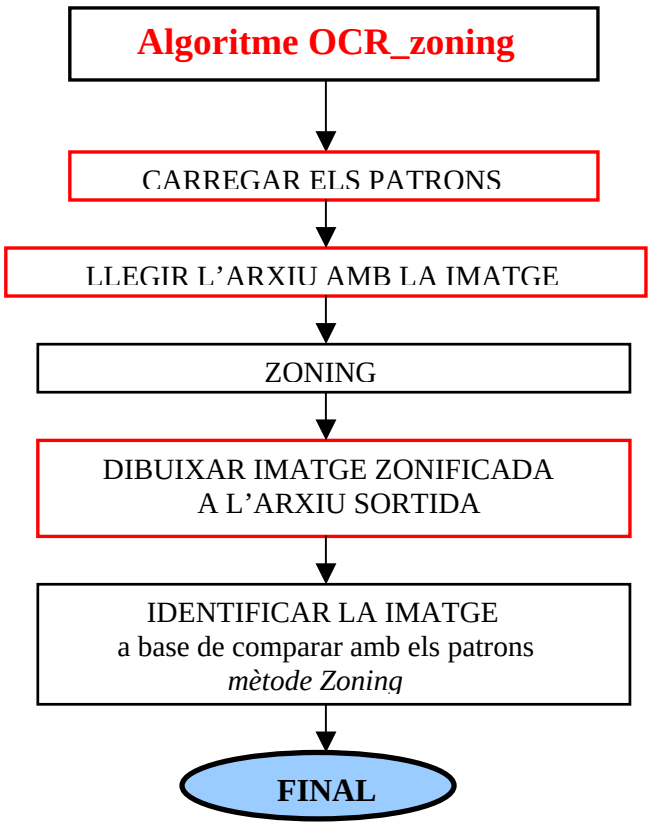


Figura 5.6. Diagrama de flux de l'algoritme OCR_zoning

Algoritme OCR_zoning_llindar.

L'algoritme *OCR_zoning_llindar* afegeix a l'anterior el càlcul dels tres nivells llindars pels colors de les zones de 6 pixels. Aquests valors llindars es calculen, tal que siguin equidistants entre ells i amb els valors dels pixels més clar i més fosc de la imatge, augmentats un factor 6. És a dir, a diferència de l'anterior, té en compte els valors dels pixels més clar i més fosc de la imatge que es processa.

Les formules per obtenir els valors llindars son les següents.

$$llindar_1 = 6 * \left(PIXEL_{\min} + 1 * \frac{PIXEL_{MAX} - PIXEL_{\min}}{4} \right)$$

$$llindar_2 = 6 * \left(PIXEL_{\min} + 2 * \frac{PIXEL_{MAX} - PIXEL_{\min}}{4} \right)$$

$$llindar_3 = 6 * \left(PIXEL_{\min} + 3 * \frac{PIXEL_{MAX} - PIXEL_{\min}}{4} \right)$$

Les funcions que realitza, es resumeixen a continuació:

- Per identificar el dígit aplica la tècnica de "Zoning".
- Descompon la imatge en zones de 6 pixels (3 files i 2 columnes).
- Calcula els valors dels llindars de color tal que siguin equidistants entre ells i amb els valors dels colors de zona més clar i més fosc de la imatge.
- A cada zona li assigna un valor de 0 a 3 segons el valor de la suma dels seus 6 pixels, emprant com a llindars els calculats. D'aquesta manera s'obté un vector de 169 elements (13 files * 13 columnes) de 4 bits.
- Compara aquest vector amb els corresponents a cada imatge patró (dígits 0 al 9) i obté l'error corresponent per cada patró.
- Determina que el dígit és el corresponent al patró amb menys error al comparar-lo amb la imatge.

La Figura 5.7 en mostra el diagrama de flux.

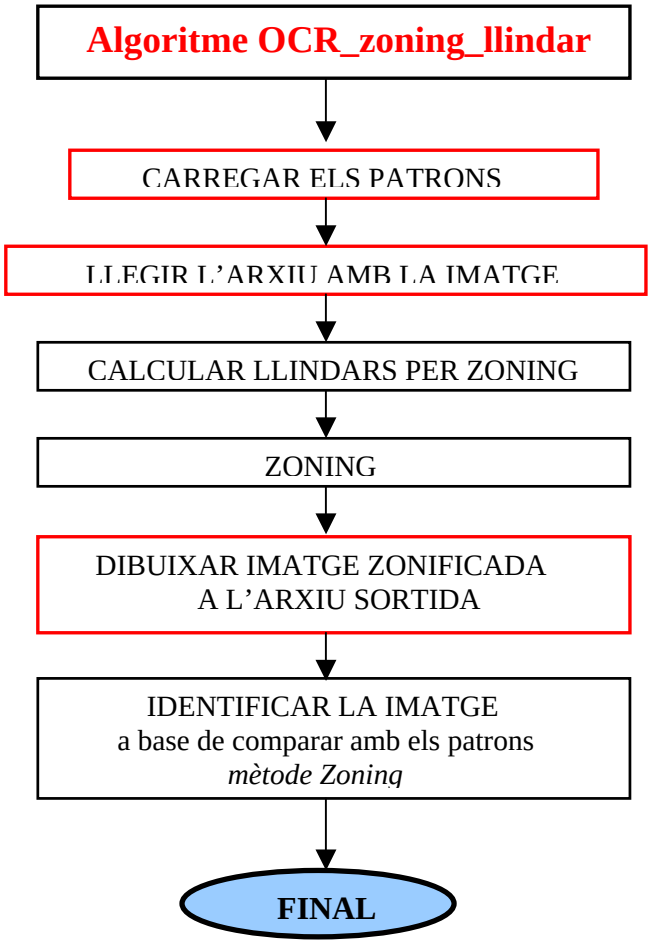


Figura 5.7. Diagrama de flux de l'algoritme OCR_zoning_llindar

5.1.3 Comparació dels algoritmes.

Les següents taules mostren, de forma comparativa, els 7 algoritmes implementats en Software i alguns dels seus principals paràmetres.

La Taula 5.1 mostra les prestacions dels diferents algoritmes implementats en Software.

La Taula 5.2 mostra, els recursos mínims en bytes, aproximats, de memòria necessaris per implementar-los en hardware. Aquests càlculs es fan considerant un únic patró per cada dígit possible, emmagatzemat a priori a la memòria del sistema. Serien necessaris bastants més recursos de memòria per implementar un algoritme OCR amb aprenentatge automàtic, que realitza un aprenentatge continu a partir dels dígits que va identificant.

La Taula 5.3 mostra els ràtios de compressió corresponent als diferents algoritmes. El ràtio de compressió indica la relació entre el nombre de bits que ocupa una imatge abans del processament i el nombre d'aquests un cop executat l'algoritme. Com s'observa, els ràtios son els mateixos per algoritmes que identifiquen el dígit amb el mateix mètode. No obstant això, la descomposició dels algoritmes en funcions permetrà la implementació de parts d'aquests, de forma que els ràtios de compressió dels diferents dissenys possibles s'hauran de mesurar prenent com a sortida les dades resultants de les diferents funcions i no del total com aquí.

Finalment, a la Figura 5.8 es mostren diferents imatges capturades per la nostra càmera i la imatge resultant del processament per cada algoritme. En aquestes imatges, no s'aprecia el resultat de la identificació, sinó que únicament es veu la imatge un cop realitzades les diferents fases de pre-processament.

Per tots els algoritmes implementats, son necessàries una imatge patró per cada dígit, és a dir 10 imatges. En la fase de reconeixement els algoritmes comparen les imatges capturades amb cadascuna de les patró per identificar el dígit. Les imatges patró han d'haver sigut preses en les millors condicions possibles donat que representen les versions més perfectes de totes les possibles.

En el nostre cas, no disposem d'imatges d'alta qualitat preses amb la càmera, sinó que únicament disposem d'una imatge de qualitat mitjana per dígit. De manera que dels diferents algoritmes no podrem corroborar com son de fiables en la fase de reconeixement. No obstant, el que si que podem comprovar és el procés de tractament de la imatge, i la quantitat d'informació requerida en el procés d'identificació per cada algoritme.

ALGORITME OCR	Binarització	Càlcul llindar	Filtre Mitjana	Filtre Mitjana imatge binaritzada	Càlcul llindar amb histograma	Càlcul llindars zoning	Aprimament imatge	Mètode de Reconeixement
Crossings	SI	SI	NO	NO	NO	NO	NO	**Crossings**
Crossings_filtre	SI	SI	SI	NO	NO	NO	NO	**Crossings**
Crossings_filtre_bin	SI	SI	NO	SI	NO	NO	NO	**Crossings**
Crossings_histograma	SI	NO	NO	NO	SI	NO	NO	**Crossings**
Crossings_aprimament	SI	SI	NO	NO	NO	NO	SI	**Crossings**
Zoning	NO	NO	NO	NO	NO	NO	NO	**Zoning**
Zoning_llindar	NO	NO	NO	NO	NO	SI	NO	**Zoning**

Taula 5.1. Prestacions dels algoritmes OCR implementats

ALGORITME OCR	Memòria mínima necessària sense aprenentatge automàtic (aprox.) [bytes]
Crossings	10 x 65 encreuaments patrons + 65 encreuaments mostra + 10 x 2 errors per dígit entre enc. i enc. patró + (39 x 26)/8 pixels binaris ≈ **862**
Crossings_filtre	10 x 65 encreuaments patrons + 65 encreuaments mostra + 10 x 2 errors per dígit entre enc. i enc. patró + (39 x 26)/8 pixels binaris + 39 x 26 pixels abans filtre ≈ **1,876**
Crossings_filtre_bin	10 x 65 encreuaments patrons + 65 encreuaments mostra + 10 x 2 errors per dígit entre enc. i enc. patró + (39 x 26)/8 pixels binaris + (39 x 26)/8 registre pixels eliminables per filtre ≈ **989**
Crossings_histograma	10 x 65 encreuaments patrons + 65 encreuaments mostra + 10 x 2 errors per dígit entre enc. i enc. patró + (39 x 26)/8 pixels binaris +256 x 1 freqüència de cada color ≈ **1,118**
Crossings_aprimament	10 x 65 encreuaments patrons + 65 encreuaments mostra + 10 x 2 errors per dígit entre enc. i enc. patró + (39 x 26)/8 pixels binaris + (39 x 26)/8 registre pixels eliminables al aprimar ≈ **989**
Zoning	10 x (13 x 13)/2 imatge zonificada per cada patró + (13 x 13)/2 imatge zonificada mostra + 10 x 2 errors per dígit entre mostra i patró ≈ **955**
Zoning_llindar	10 x (13 x 13)/2 imatge zonificada per cada patró + (13 x 13)/2 imatge zonificada mostra + 10 x 2 errors per dígit entre mostra i patró ≈ **955**

Taula 5.2. Recursos de memòria necessaris per implementar els algoritmes OCR

ALGORITME OCR	bits entrada (bits imatge)	bits sortida	Ràtio de compressió [%] (bits sortida / bits entrada)
Crossings	8,112	286	**3.526 %**
Crossings_filtre	8,112	286	**3.526 %**
Crossings_filtre_bin	8,112	286	**3.526 %**
Crossings_histograma	8,112	286	**3.526 %**
Crossings_aprimament	8,112	286	**3.526 %**
Zoning	8,112	338	**4.167 %**
Zoning_llindar	8,112	338	**4.167 %**

Taula 5.3. Ràtios de compressió de dades dels diferents algoritmes OCR

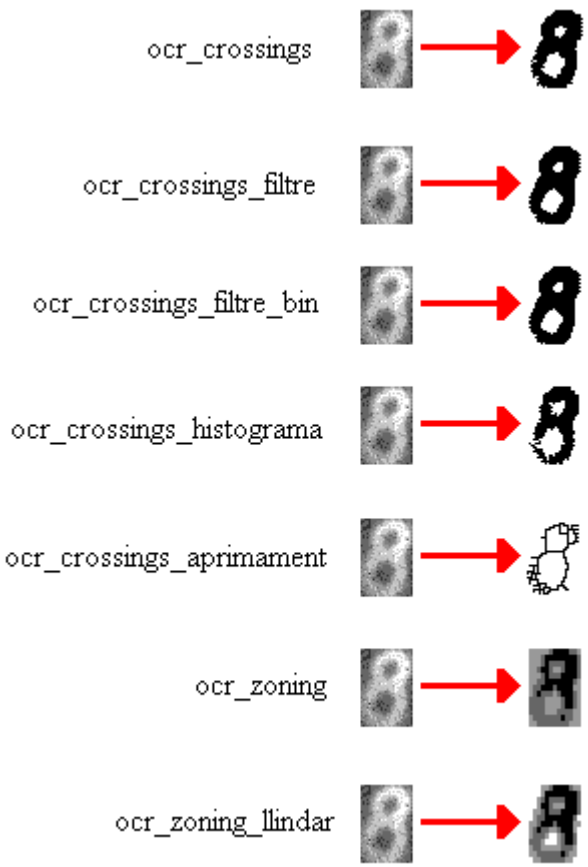


Figura 5.8. Imatges resultats dels algoritmes OCR

El fet que sigui molt més eficient i fiable un sistema d'identificació amb aprenentatge automàtic, i tenint en compte les grans quantitats de recursos hardware que un sistema així requereix, es descarta la implementació en hardware d'un algoritme OCR sencer.

El disseny es basarà, doncs, en la implementació de parts d'aquests algoritmes, per tal de reduir el nombre de bits per transmetre i executar una part del procés. La fase d'identificació es farà al punt de recepció de dades, on els recursos d'espai i consum, permeten la realització d'un procés software de reconeixement molt més fiable.

5.1.4 Descomposició dels algoritmes en funcions.

Per tot lo exposat anteriorment, el pas previ al disseny Hardware, és la descomposició dels algoritmes implementats en software, en les diferents funcions que realitzen.

A partir dels diversos algoritmes, podem identificar les següents funcions.

1) **Determinació del llindar de binarització.** Consisteix en determinar els pixels més clar i més fosc de la imatge, i calcular el promig, que serà el llindar per binaritzar.

2) **Binarització de la imatge.** Consisteix en transformar la imatge, de nivells de gris a blanc i negre. Tots els nivells de grisos més foscos que el nivell llindar calculat es convertiran en negre i tots els més clars en blanc.

3) **Filtrat (Filtre de Mitjana)**. Consisteix en filtrar la imatge amb un Filtre de Mitjana.

4) **Aprimament de la imatge.** Consisteix en esborrar pixels negres dels extrems, de forma que al final només quedi l'esquelet de la figura de la imatge.

5) **Càlcul dels encreuaments horitzontals i verticals de la imatge.** Consisteix en calcular el nombre de vegades que hi ha un pas de BLANC a NEGRE per cada fila i per cada columna, obtenint així un vector de (26+39) elements que descriu la imatge.

6) **Determinació dels llindars per zonificar la imatge.** És anàloga a la funció de Determinació del llindar de binarització. En aquest cas els llindars resultants son 3, ja que els valors possibles s'agrupen en 4 trams a diferència dels 2 del primer cas. Els valors possibles no son els d'un pixel sinó els de 6 d'ells sumats.

7) **Zonificació de la imatge.** Consisteix en sumar els valors dels pixels de cada zona (3 files x 2 columnes), i assignar un valor (de 0 a 3) a cadascuna en funció del valor total dels seus pixels.

5.2 Implementació en hardware de blocs funcionals d'algoritmes OCR.

En aquesta fase, es procedirà a la implementació en hardware de blocs que realitzen diverses de les funcions enumerades anteriorment. Cadascun d'aquests blocs serà provat per separat, i un cop comprovat el seu correcte funcionament, es procedirà al disseny de diferents arquitectures de mòduls de processament d'imatge, a partir de diferents combinacions. Aquests mòduls, a diferència dels algoritmes implementats en Software, no realitzaran tot el procés de reconeixement del dígit, sinó que n'executaran diverses fases i donaran com a sortida una quantitat d'informació per dígit menor a la proporcionada per la càmera.

La implementació dels blocs funcionals es realitzarà en llenguatge de descripció de hardware (VHDL) [Zha]. Posteriorment es comprovarà el seu correcte funcionament simulant el seu comportament a través de *Modelsim 5.7*. La simulació de la càmera es realitzarà convertint els arxius **.bmp* emprats en la simulació dels algoritmes implementats en software a arxius **.dua*. Aquest format és utilitzat en VHDL per simular memòries ROM [Sav], i donat que la càmera es comporta en un espai de temps com a tal, aquest arxiu representarà els pixels de la matriu.

A continuació es detalla una descripció dels diferents blocs funcionals implementats.

5.2.1 Bloc que realitza el procés de binarització.

El bloc *binaritzador* funciona de forma seqüencial, sincronitzat a través de la senyal de rellotge global *[clk]*. La seva implementació s'ha realitzat amb una arquitectura de màquina d'estats.

La funció que realitza és la de donada una adreça de la càmera (fila i columna) i un valor llindar, proporciona el valor del pixel com un bit. És a dir, segons el valor del pixel i el del llindar, el pixel de la càmera (8 bits) és representat com a BLANC o NEGRE (1 bit).

La seva implementació hardware es troba a l'arxiu *binaritzador.vhd*.

5.2.2 Bloc que calcula el llindar per al procés de binarització.

El bloc *calcul_llindar* funciona de forma seqüencial, sincronitzat a través de la senyal de rellotge global *[clk]*. La seva implementació s'ha realitzat amb una arquitectura de màquina d'estats.

La funció que realitza és la de calcular el valor del pixel llindar amb el mètode del promig entre els valors del pixel més clar i més fosc de la imatge. El procés que realitza consisteix en demanar els valors dels pixels de cada adreça de la matriu de la càmera, obtenir els valors màxim i mínim, sumar-los i posteriorment dividir el resultat per 2, obtenint així el valor llindar, que és la seva sortida. Aquest valor és una de les entrades del bloc *binaritzador*.

La seva implementació hardware es troba a l'arxiu *calcul_llindar.vhd*.

5.2.3 Blocs que implementen un Filtre de Mitjana.

Hi ha dos blocs que implementen un Filtre de Mitjana: els blocs *control_filtre_9_pixels* i *filtre_9_pixels*. Tots dos funcionen de forma seqüencial, sincronitzats a través de la senyal de rellotge global *[clk]* i amb estructura de màquines d'estats.

El primer, *control_filtre_9_pixels* realitza la funció de Master en el procés de filtrat. Quan s'ha d'iniciar el procés de filtrat, activa el bloc *filtre_9_pixels* , que comprova pixel per pixel de la càmera si aquests s'han d'eliminar o no i envia la dada de si el pixel és o no esborrable a un mòdul de memòria. Posteriorment, quan el bloc *filtre_9_pixels* ha acabat, realitza un escombrat a la imatge, i si el pixel en qüestió és eliminable (determinat prèviament per l'altre bloc) procedeix a la seva eliminació, és a dir a fixar-lo de color blanc.

La implementació hardware d'aquests dos blocs es troba als arxius *control_filtre_9_pixels.vhd* i *filtre_9_pixels.vhd* , respectivament.

5.2.4 Bloc del Compressor de dades.

El bloc *compressor* funciona de forma seqüencial, sincronitzat a través de la senyal de rellotge global *[clk]* i amb estructura de màquina d'estats.

Aquest bloc, realitza la funció de comprimir una imatge binaritzada, que es troba en memòria. Per tal de realitzar el procés de compressió, es realitza un escombrat que segueix el següent camí. L'inici és la posició de memòria que correspon al pixel de l'extrem sud-oest, és a dir a baix a l'esquerra, de la imatge. A partir d'aquí segueix cap a la dreta fins al final de la fila i llavors comença a la següent columna superior. Les dades de la imatge comprimida s'agrupen en paraules de 4 bits. El bit de major pes de la paraula indica el color del pixel (0 → blanc, 1 → negre) i els tres de menor pes indiquen quants bits iguals a aquest hi ha a continuació. S'entén per continuació els pixels que corresponen a les següents posicions de la ruta descrita. La Figura 5.9 mostra un exemple de resultat de compressió de les primeres dades de la imatge.

Pixels imatge

..																												
0	0	0	0	0	0	0	0	0	0	0	0	1	1	...	...													
0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	0			

Valors comprimits

0	0	0	0	Representa →	0
1	0	0	0	Representa →	1
0	0	1	0	Representa →	000
1	1	1	1	Representa →	11111111
1	1	1	0	Representa →	1111111
0	0	1	0	Representa →	000
1	0	0	1	Representa →	11
0	1	1	1	Representa →	00000000
0	1	0	0	Representa →	00000
1	0	0	1	Representa →	11

Figura 5.9. Exemple de compressió de dades d'una imatge

La seva implementació hardware es troba a l'arxiu *compresor.vhd*.

5.2.5 Blocs per obtenir els encreuaments ("Crossings").

Hi ha dos blocs que calculen els encreuaments.

Els encreuaments horitzontals (un per cadascuna de les files) son calculats pel bloc *calcul_encreuaments_horitzontals* i els verticals (un per cadascuna de les columnes) pel bloc *calcul_encreuaments_verticals*.

Aquests blocs contenen registres interns que emmagatzemen els valors dels encreuaments, i donada una fila o columna, segons el tipus d'encreuament, retornen el valor corresponent. De forma que no necessita memòria exterior per emmagatzemar les dades calculades.

Els dos blocs funcionen de forma seqüencial, sincronitzats a través de la senyal de rellotge global *[clk]* i amb estructura de màquina d'estats.

La seva implementació es troba als arxius *calcul_encreuaments_horitzontals.vhd* i *calcul_encreuaments_verticals.vhd*, respectivament.

5.2.6 Blocs que realitzen l'aprimament de la imatge.

Per realitzar el procés d'aprimament de la imatge s'han implementat cinc blocs funcionals. El principal i Master en el procés, és el bloc *aprimament_imatge*. Hi ha quatre altres blocs en el procés: *aprimament_nord*, *aprimament_est*, *aprimament_sud* i *aprimament_oest*. Cadascun d'aquests 4 blocs realitzen les funcions de determinar per cada pixel de la imatge si és esborrable o no segons la direcció de l'aprimament, i en cas afirmatiu enviar a un mòdul de memòria la dada de si el pixel s'ha d'eliminar o no.

Quan s'ha d'iniciar el procés d'aprimament, el bloc *aprimament_imatge* activa els altres 4 blocs de forma seqüencial. L'ordre de crida comença pel nord i segueix un sentit horari (nord, est, sud, oest). Un cop el darrer bloc (*aprimament_oest*) ha acabat, realitza l'esborrat dels pixels marcats com a tal en aquesta iteració.

Un cop esborrats els pixels corresponents, el procés es repeteix fins que en una iteració complerta, cap dels 4 blocs Slaves detecta cap pixel eliminable i per tant la imatge ja no es pot aprimar més.

Resumidament, els blocs que intervenen en el procés d'aprimament i els arxius on es troben implementats en VHDL es relacionen a continuació.

- *aprimament_imatge* → *aprimament_imatge.vhd*
- *aprimament_nord* → *aprimament_nord.vhd*
- *aprimament_est* → *aprimament_est.vhd*
- *aprimament_sud* → *aprimament_sud.vhd*
- *aprimament_oest* → *aprimament_oest.vhd*

La Figura 5.10 mostra el diagrama de flux del procés d'aprimament de la imatge.

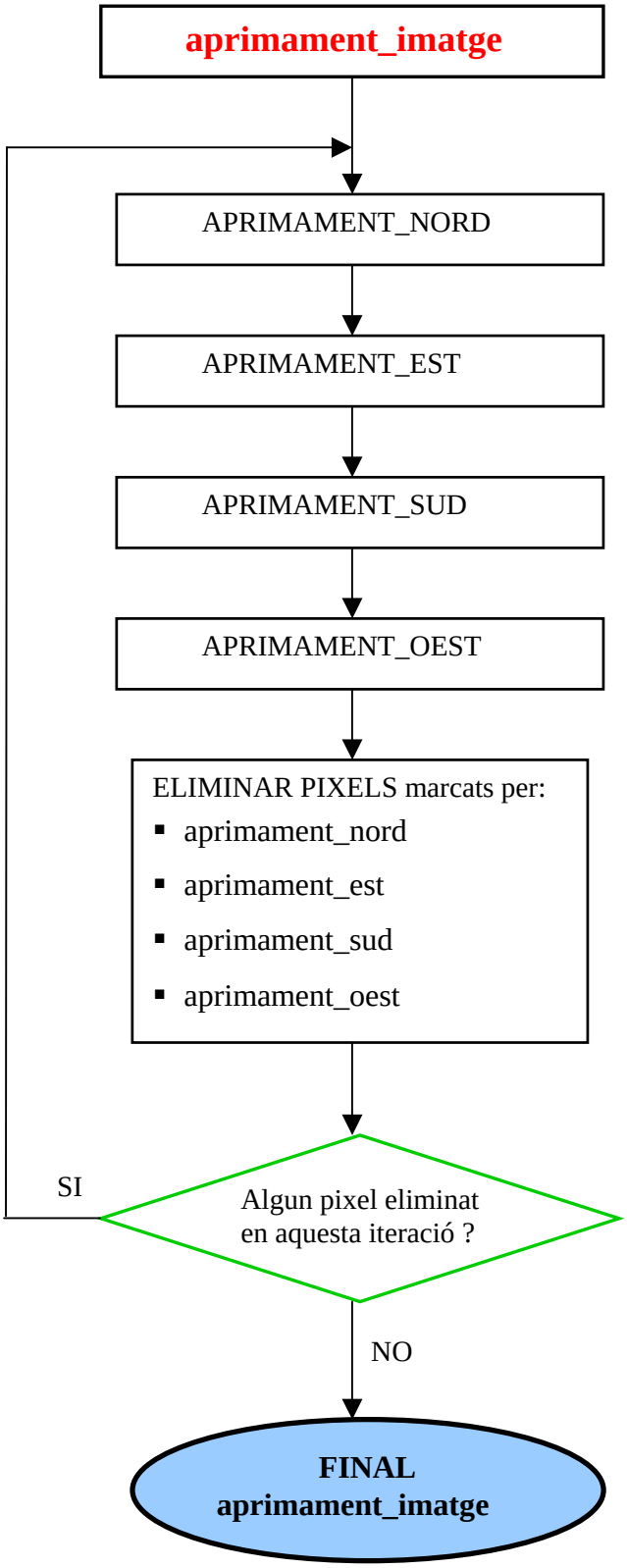


Figura 5.10. Diagrama de flux del bloc aprimament_imatge

5.2.7 Bloc que realitza la zonificació de la imatge ("Zoning").

El bloc *zonning* funciona de forma seqüencial, sincronitzat a través de la senyal de rellotge global *[clk]*. La seva implementació s'ha realitzat amb una arquitectura de màquina d'estats.

La funció que realitza és la següent. Donada una adreça de la càmera (fila i columna) i tres valors llindars, realitza la tasca de sumar els valors dels pixels corresponent a la zona, el primer pixel de les quals és l'indicat, i retorna una paraula de 2 bits en funció del resultat de la suma. La Figura 5.11 mostra els pixels considerats en el procés, per una entrada (x, y).

La seva implementació hardware es troba a l'arxiu *zonning.vhd*.

5.2.8 Bloc que calcula els llindars per al procés de zonificació.

El bloc *calcul_llindar_zonning* funciona de forma seqüencial, sincronitzat a través de la senyal de rellotge global *[clk]*. La seva implementació s'ha realitzat amb una arquitectura de màquina d'estats.

La seva funció és anàloga al bloc *calcul_llindar*, amb la diferència que aquest bloc ha d'obtenir tres valors llindars en lloc d'un. Aquests valors son entrades del bloc *zonning*.

El procés que realitza consisteix en demanar els valors dels pixels de cada adreça de la matriu de la càmera i obtenir els valors màxim i mínim d'aquests. Un cop obtinguts, aplica un factor sis a aquests valors donat que les zones estan formades per 6 pixels i per tant els possibles valors dels pixels d'una zona sumats corresponen a sis vegades els pixels més clar i més fosc de la imatge. Finalment obté els valors dels llindars seguint les següents formules.

$$llindar_1 = 6*\left(PIXEL_{\min} + 1*\frac{PIXEL_{MAX} - PIXEL_{\min}}{4} \right) = 6*\left(\frac{3*PIXEL_{\min} + PIXEL_{MAX}}{4} \right)$$

$$llindar_2 = 6*\left(PIXEL_{\min} + 2*\frac{PIXEL_{MAX} - PIXEL_{\min}}{4} \right) = 6*\left(\frac{PIXEL_{\min} + PIXEL_{MAX}}{2} \right)$$

$$llindar_3 = 6*\left(PIXEL_{\min} + 3*\frac{PIXEL_{MAX} - PIXEL_{\min}}{4} \right) = 6*\left(\frac{PIXEL_{\min} + 3*PIXEL_{MAX}}{4} \right)$$

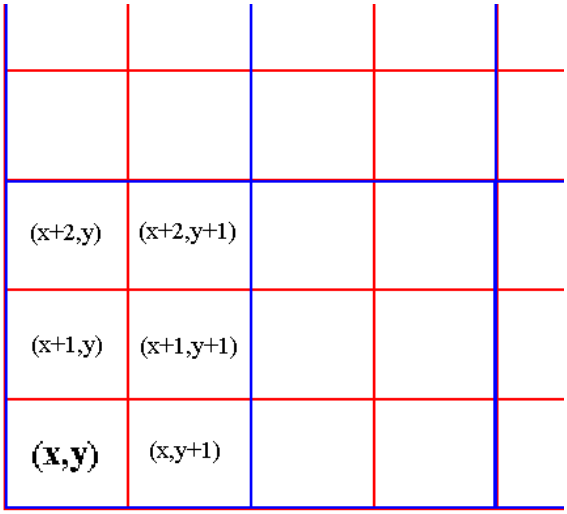


Figura 5.11. Pixels considerats en el procés de Zoning

Aquestes formules s'han adaptat per poder implementar-les més fàcilment en hardware, donat que les divisions realitzades corresponen a valors amb logaritme binari sencer.

La seva implementació hardware es troba a l'arxiu *calcul_llindar_zonning.vhd*.

5.2.9 Blocs combinacionals de control d'accés.

S'han implementat cinc blocs combinacionals que tenen la funció de controlar l'accés a mòduls de memòria o a la càmera quan en una determinada arquitectura, aquests mòduls han de ser accedits en el mateix mode (lectura o escriptura) per diversos blocs en instants diferents. Aquests blocs discriminen en funció de l'origen de l'accés i d'una entrada de selecció. Les diferències entre ells estan en el mòdul o mode del mateix, l'accés del qual controlen i en el nombre d'entrades. La Taula 5.4 mostra els diferents blocs i el mòdul per al que arbitren l'accés.

El seu comportament és el d'un multiplexor en el cas de senyals que son llegides pel mòdul de destí, i de demultiplexor, en cas de senyals que son escrites pel mateix. La Figura 5.12 il·lustra el comportament d'un controlador o àrbitre d'accés.

CONTROLADOR D'ACCÉS	Controla l'accés al mòdul	Entrades	Senyal selecció (bits)
arbitre_camara_read	*camara*	2	1
arbitre_mem_pixels_read	*mem_pixels* (mode lectura)	2	1
arbitre_mem_pixels_write	*mem_pixels* (mode escriptura)	2	1
arbitre_mem_pixels_read_4	*mem_pixels* (mode lectura)	4	2
arbitre_mem_pixels_write_4	*mem_pixels* (mode escriptura)	4	2

Taula 5.4. Relació de blocs combinacionals de control d'accés

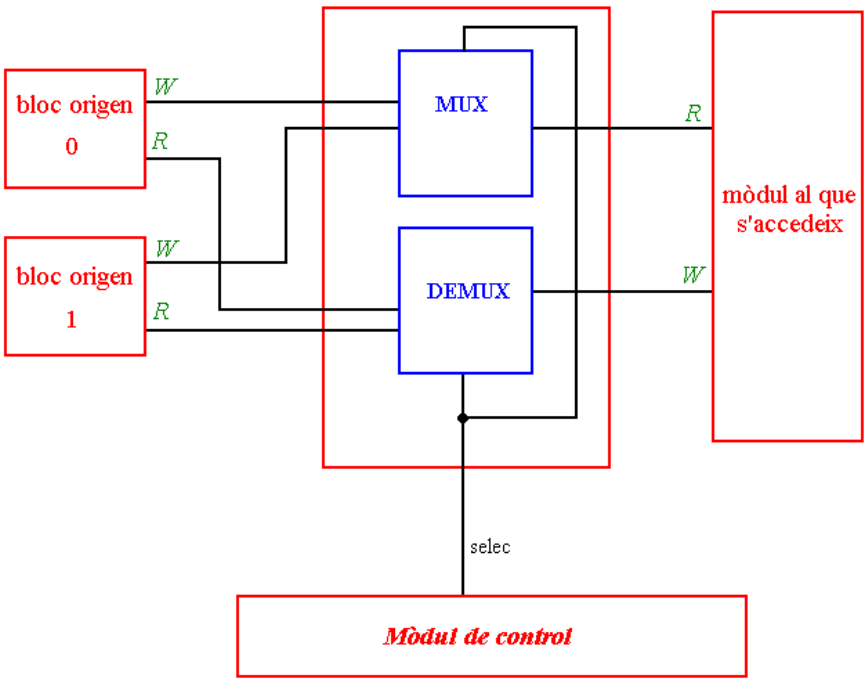


Figura 5.12. Esquema d'un controlador d'accés

5.2.10 Mòduls no sintetitzables: Càmera.

En el procés de disseny, serà requisit, per tal de poder realitzar les corresponents proves amb els dissenys del mòdul-OCR, de disposar d'una implementació que simuli el comportament de la càmera i interaccioni amb els blocs dissenyats. Aquest bloc virtual també haurà de ser capaç de llegir arxius **.bmp*, donat que les imatges preses amb la càmera real estan en aquest format.

El mòdul *camara* realitza aquesta funció. La seva arquitectura és semblant als models de memòries ROM implementades en VHDL. Les dades que els altres blocs llegiran es troben amb un arxiu **.dua* que el mòdul carregarà quan s'inicií el procés de simulació. Per tal de convertir les dades de les imatges, que es troben en format **.bmp* al format **dua* s'ha implementat un programa *camara.exe*, l'arxiu font del qual és *camara.cpp*, que realitza aquesta funció. La Figura 5.13 il·lustra aquest procés.

La implementació en VHDL del mòdul *camara* es troba a l'arxiu *camara.vhd*.

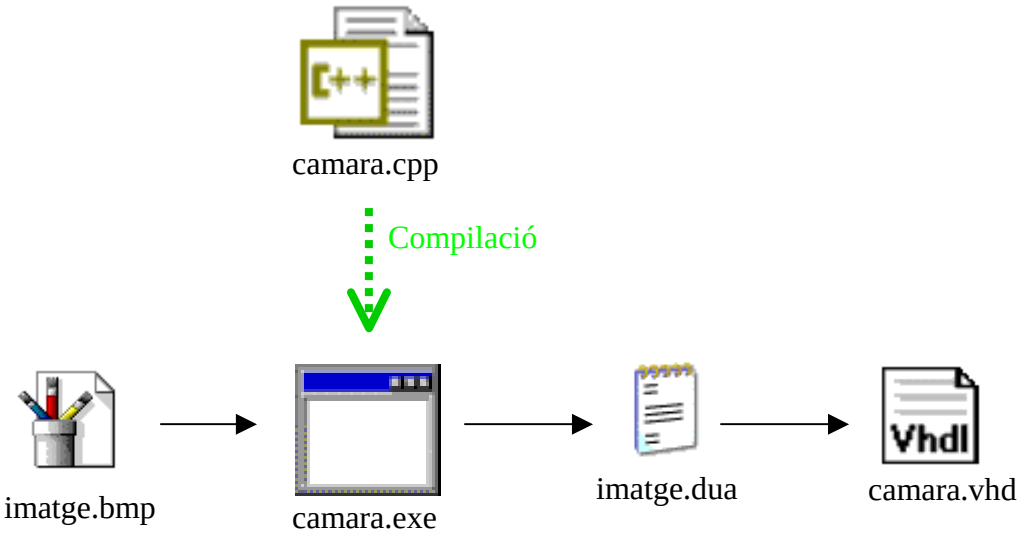


Figura 5.13. Procés de simulació de la càmera

5.2.11 Mòduls no sintetitzables: Memòries.

Pel funcionament del mòdul-OCR és necessària la utilització de memòria. S'han implementat mòduls de memòria que no seran sintetitzats en el procés, donat que l'ASIC on s'integrarà el mòdul dissenyat disposarà de mòduls de memòria.

Per realitzar les simulacions, es treballarà amb mòduls de memòria implementats en VHDL, i al final del procés, en cas d'integrar-se el disseny, s'utilitzaran mòduls de memòria SRAM sintetitzables amb la tecnologia C35B4. En aquest projecte s'estimarà l'àrea requerida pels mòduls SRAM sintetitzable, en funció de la memòria necessària per implementar el disseny definitiu.

Els mòduls de memòria no sintetitzables, seran models de memòries dissenyats per les necessitats dels blocs que hi interaccions i el format de les dades que s'hi emmagatzemen. En cas de realitzar-se la integració, es requerirà, únicament, afegir adreçadors de memòria a les entrades de la SRAM per tal d'adaptar el format de dades i adreces utilitzat en els models VHDL al format real de la memòria.

La Taula 5.5 mostra els diferents mòduls de memòria implementats en VHDL i les funcions que realitzen en el marc del nostre sistema.

MÒDUL DE MEMÒRIA	**Capacitat (bits)**	Tamany paraules (bits)	[#]@ (bits)	funció
mem_pixels	1 kbits	1	[6,5]	▪ Emmagatzemar la imatge binaritzada. ▪ Emmagatzemar la dada per cada pixel si és o no eliminable (processos de filtrat i aprimament).
mem_imatge_comprimida	4 kbits	4	10	▪ Emmagatzemar les dades de la imatge comprimida.
mem_pixels_gris	512	2	[4,4]	▪ Emmagatzemar les dades dels valors de les zones (procés de Zoning).

Taula 5.5. Mòduls de memòria implementats en el disseny

[#] En els mòduls *mem_pixels* i *mem_pixels_gris*, les adreces son dos paraules que representen la fila i la columna de la imatge. Internament els mòduls calculen l'adreça real correlativa.

5.2.12 Mòduls no sintetitzables: *Imatge_bmp* i *Imatge_gris_bmp*.

Aquests dos mòduls, no seran sintetitzats ni representen cap element de l'ASIC amb el que interacciona el mòdul-OCR. La seva utilitat és, únicament permetre comprovar el correcte funcionament dels respectius dissenys.

Aquests mòduls, tenen una estructura semblant als mòduls de memòria *mem_pixels* i *mem_pixels_gris*. Aquests mòduls realitzen les mateixes tasques que les memòries, amb la diferència de que per una banda no es pot llegir d'ells, i per l'altra disposen de dos senyals addicionals, que serveixen perquè el procés de simulació els cridi al final. Quan son cridats generen amb les dades emmagatzemades (les mateixes que les respectives memòries) arxius **.dua* amb les corresponents dades. Posteriorment, aquests arxius son convertits a arxius **.bmp* mitjançant el programa *imatge_bmp.exe*, l'arxiu font del qual és *imatge_bmp.cpp*. Amb el que obtenim un arxiu **.bmp* amb la imatge resultant del procés. La Figura 5.14 il·lustra aquest procés.

Per tal de realitzar la seva funció, aquests mòduls han de tenir les seves entrades de dades i adreces connectades als mateixos punts que les dels mòduls de memòria respectius.

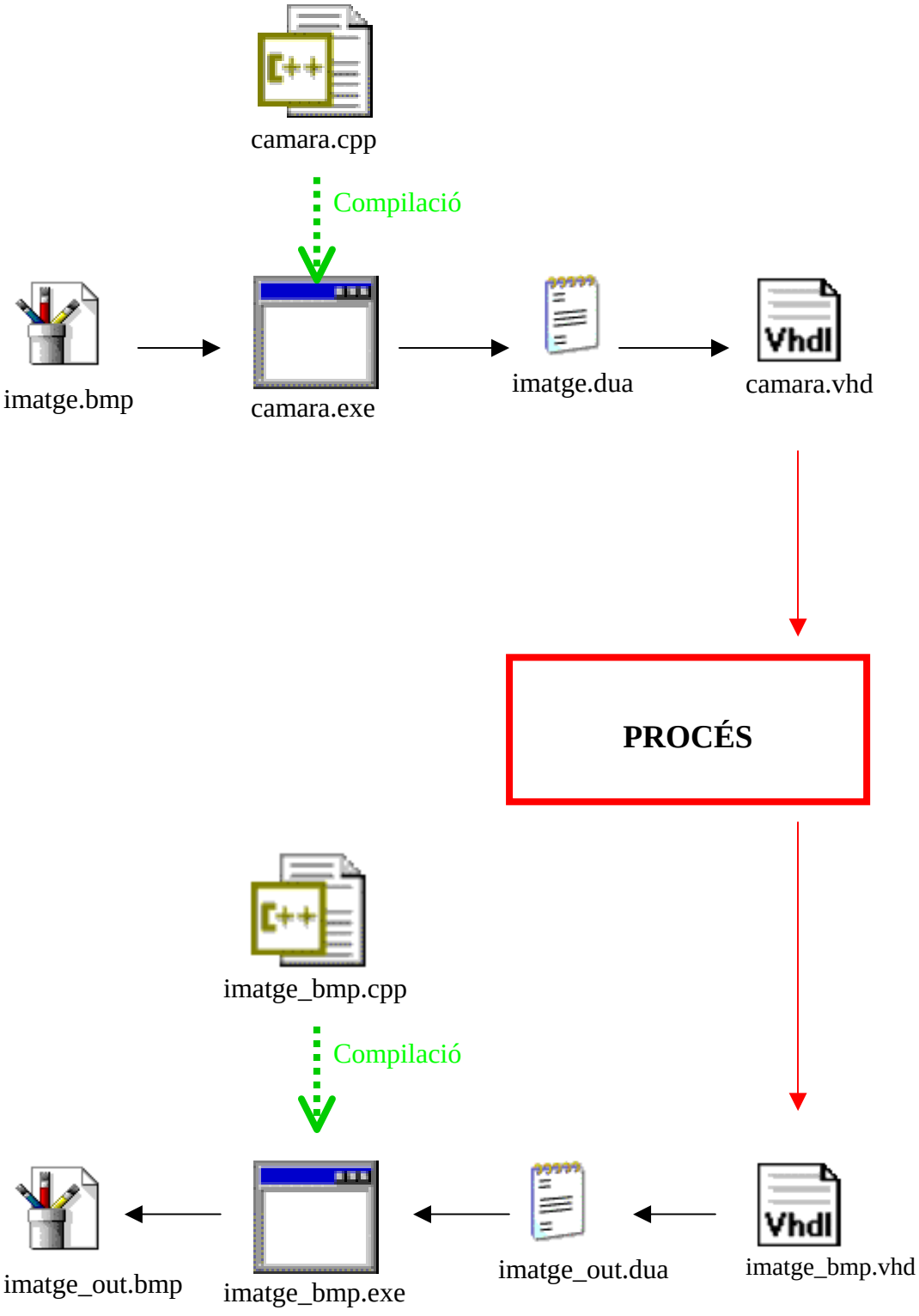


Figura 5.14. Procés per obtenir la imatge resultant del processament

5.3 Síntesis i test dels blocs funcionals.

Un cop comprovat el correcte funcionament dels blocs implementats amb l'entorn *Modelsim 5.7*, es procedeix a la seva síntesis amb la tecnologia C35B4, i a la verificació del seu bon funcionament a aquest nivell.

El procés de síntesis és el que ens situa en un nivell d'abstracció més baix en el procés de disseny. Consisteix en convertir els arxius **.vhd* , que descriuen els blocs des d'un punt de vista funcional, a altres que els implementin amb models d'elements lògics concrets. Les característiques i prestacions d'aquests elements lògics depenen de la tecnologia amb què es realitzi el procés de síntesis. Les característiques La fase de síntesis és la prèvia a les de *placement* i *routing*, en un procés de disseny d'un IC. La Figura 5.15 il·lustra el concepte de síntesis d'un sistema.

En el nostre cas la tecnologia serà la Austriamicrosystems 0.35 μm C-MOS, amb 4 capes de metalls i 2 de poli-silicis (C35B4), compatible amb TSMC. Les característiques estan disponibles a [AMS01].

En el procés de síntesis utilitzem l'entorn *Synopsys Design Analizer*. Per cada bloc que es sintetitza es genera un altre arxiu **.vhd* amb la implementació tecnològica del sistema. A l'Apèndix B.2 es mostra un fragment d'un arxiu resultant de la síntesis. La seva interconnexió amb els altres blocs és exactament igual, i només canvia la implementació interna. Els arxius resultants del procés no descriuen processos, ni contenen instruccions d'execució iterativa, sinó que descriuen unitats lògiques de la tecnologia corresponent i les connexions per cada instància d'aquestes entre elles i amb els ports I/O del bloc.

Pels principals blocs realitzats, implementarem arxius VHDL d'aplicació o test. En aquests arxius es generen els senyals de RESET, que mantindrem en estat baix els primers moments de la simulació, i de CLK, que simula una senyal de rellotge. Aquest rellotge és una senyal quadrada a freqüència de 20MHz (T = 50 ns, T/2 = 25 ns).

La Taula 5.6 mostra els arxius resultants del procés i els corresponents arxius de test.

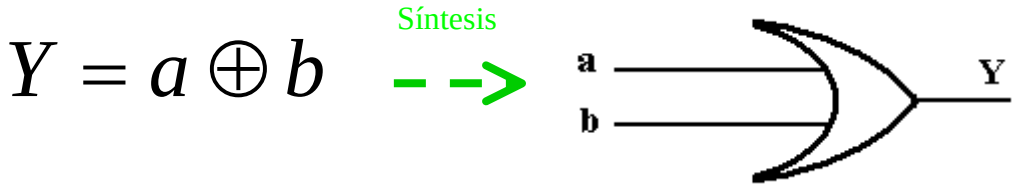


Figura 5.15. Concepte de síntesis

Arxiu bloc dissenyat	Arxiu bloc sintetitzat	Arxiu de test
binaritzador.vhd	*binaritzador_sin.vhd*	*aplicacio_ binaritzador.vhd*
calcul_llindar.vhd	*calcul_llindar_sin.vhd*	*aplicacio_calcul_llindar.vhd*
zonning.vhd	*zonning_sin.vhd*	*aplicacio_zonning.vhd*
calcul_llindar_zonning.vhd	*calcul_llindar_ zonning_sin.vhd*	*aplicacio_calcul_llindar _zonning.vhd*
calcul_encreuaments_ horitzontals.vhd	*calcul_encreuaments_ horitzontals_sin.vhd*	*aplicacio_calcul_ encreuaments_horitzontals.vhd*
calcul_encreuaments_ verticals.vhd	*calcul_encreuaments_ verticals_sin.vhd*	*aplicacio_calcul_encreuaments_ verticals.vhd*
filtre_9_pixels.vhd	*filtre_9_pixels_sin.vhd*	*aplicacio_filtre_9_pixels.vhd*
control_filtre_9_pixels.vhd	*control_filtre_9_ pixels_sin.vhd*	*aplicacio_control_filtre_9_ pixels.vhd*
aprimament_imatge.vhd	*aprimament_imatge_sin.vhd*	*aplicacio_aprimament_ imatge.vhd*
aprimament_nord.vhd	*aprimament_nord_sin.vhd*	*aplicacio_aprimament_nord.vhd*
aprimament_est.vhd	*aprimament_est_sin.vhd*	*aplicacio_aprimament_est.vhd*
aprimament_sud.vhd	*aprimament_sud_sin.vhd*	*aplicacio_aprimament_sud.vhd*
aprimament_oest.vhd	*aprimament_oest_sin.vhd*	*aplicacio_aprimament_oest.vhd*

Taula 5.6. Arxius resultants del procés de síntesis dels blocs funcionals

Amb els arxius sintetitzats i les aplicacions implementades, es pot, mitjançant *Modelsim*, comprovar el comportament del sistema, observant les evolucions temporals dels diferents senyals. Per tal de realitzar aquesta tasca, juntament amb els arxius sintetitzats i el de l'aplicació (també anomenat "Testbench"), s'han d'incloure les llibreries de la tecnologia amb la que hem realitzat el procés, en aquest cas, C35B4. La Figura 5.16 mostra el resultat de simular el comportament d'un dels blocs implementats, un cop sintetitzat.

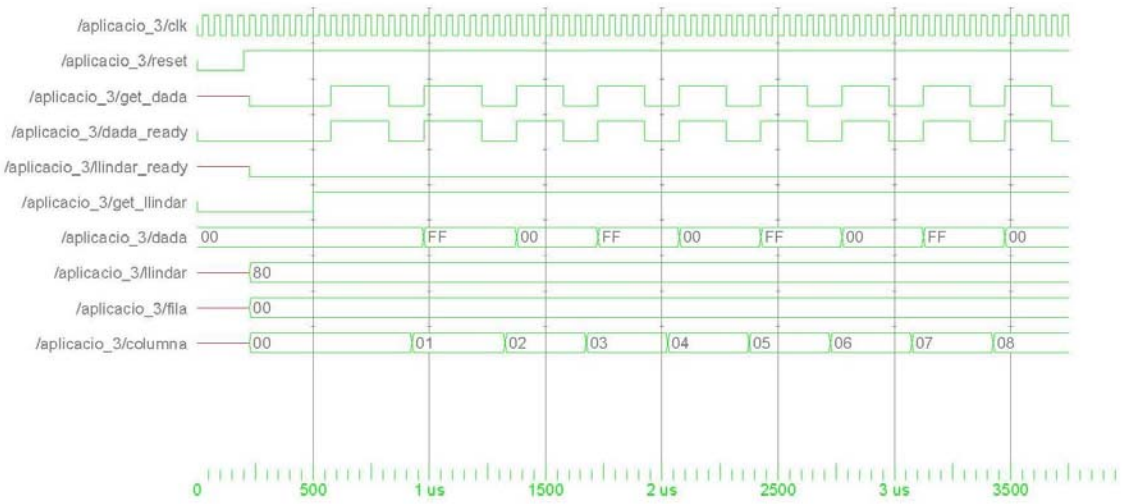


Figura 5.16. Resultats de la simulació de blocs funcionals amb *Modelsim*

5.4 Disseny hardware de mòduls de processament d'imatges.

Un cop comprovat el correcte funcionament dels blocs funcionals sintetitzats amb tecnologia C35B4, es procedirà al disseny d'un mòdul de processament d'imatges. Es dissenyaran diferents arquitectures per a aquest mòdul de processament d'imatges, cadascuna d'elles a partir de diferents combinacions de blocs funcionals. Aquests mòduls executaran diverses fases dels algoritmes OCR, implementats anteriorment en software i proporcionaran com a sortida una quantitat d'informació per dígit menor a la necessària per representar les imatges sense processament.

Cada arquitectura disposarà d'un mòdul intern de control, que realitzarà les funcions de Master i cridarà als altres blocs funcionals (Slaves) en el procés d'execució.

A continuació es detallen de les diferents arquitectures implementades, el seu funcionament i els blocs funcionals que les integren.

5.4.1 Arquitectura 1.

L'arquitectura 1 és la més simple. Amb la imatge realitza, únicament, el procés de binarització i comprimeix les dades resultants, abans d'enviar-les per emmagatzemar a un mòdul de memòria, des d'on seran demanades pel mòdul-RF de l'ASIC quan hagin de ser enviades al punt de recepció de la companyia d'aigües. El diagrama de flux de l'algoritme que implementa es mostra a la Figura 5.17 .

La Taula 5.7 mostra els blocs funcionals que integren l'arquitectura 1.

BLOCS FUNCIONALS SEQÜENCIALS sintetitzats	**BLOCS COMBINACIONALS DE CONTROL D'ACCÉS**	**MÒDULS DE MEMÒRIA necessaris**
▪ **control_algoritme_1** ▪ calcul_llindar ▪ binaritzador ▪ compresor	▪ arbitre_camara_read	▪ mem_pixels ▪ mem_imatge_comprimida

Taula 5.7. Recursos hardware de l'arquitectura 1

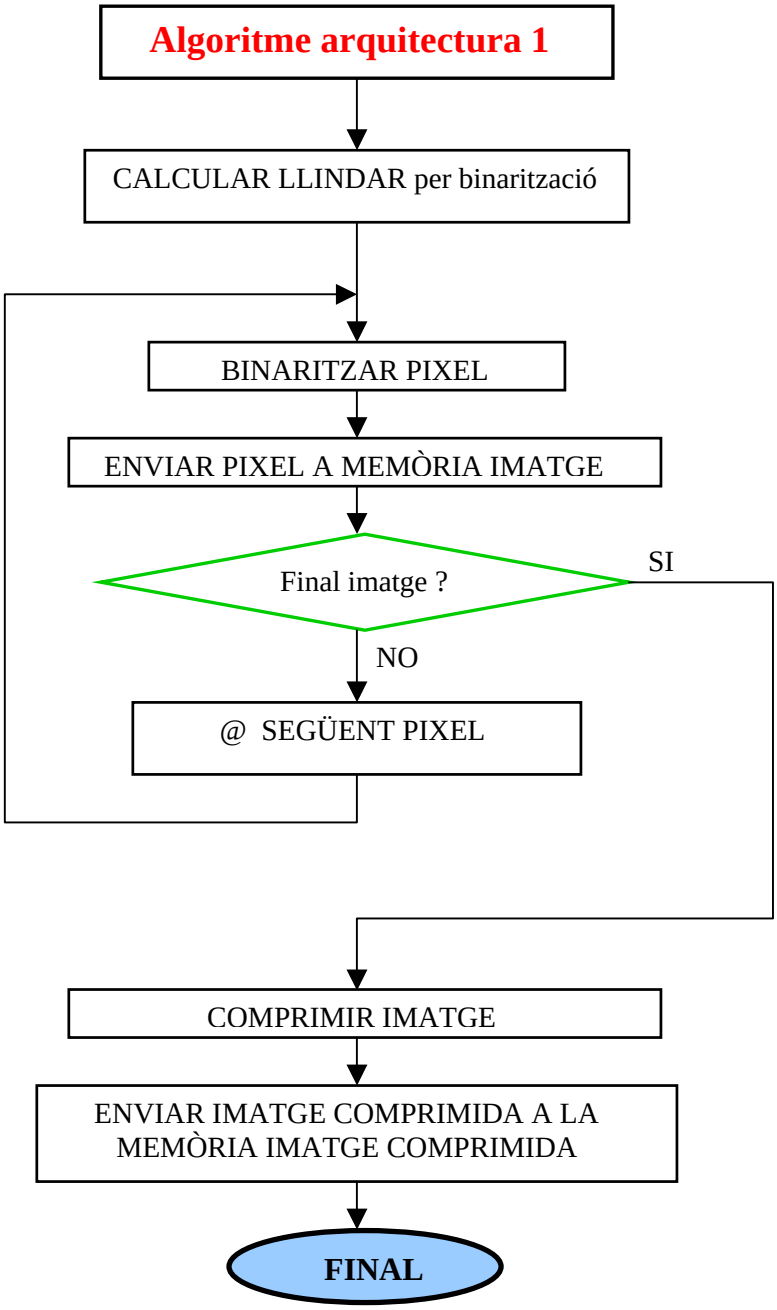


Figura 5.17. Diagrama de flux de l'algoritme de l'arquitectura 1

La implementació del mòdul-OCR amb l'arquitectura 1 es troba a l'arxiu *algoritme_1.vhd*, i la del controlador corresponent a *control_algoritme_1.vhd*.

5.4.2 Arquitectura 2.

L'arquitectura 2 afegeix a l'anterior un filtre de mitjana que tracta la imatge un cop aquesta ha estat binaritzada i enregistrada a la memòria *mem_imatge*. El diagrama de flux de l'algoritme que implementa es mostra a la Figura 5.18.

La implementació del mòdul-OCR amb l'arquitectura 2 es troba a l'arxiu *algoritme_2.vhd*, i la del controlador corresponent a *control_algoritme_2.vhd*.

La Taula 5.8 mostra els blocs funcionals que integren l'arquitectura 2.

BLOCS FUNCIONALS SEQÜENCIALS sintetitzats	**BLOCS COMBINACIONALS DE CONTROL D'ACCÉS**	**MÒDULS DE MEMÒRIA necessaris**
▪ **control_algoritme_2** ▪ calcul_llindar ▪ binaritzador ▪ compresor ▪ filtre_9_pixels ▪ control_filtre_9_pixels	▪ arbitre_camara_read ▪ arbitre_mem_pixels_write ▪ arbitre_mem_pixels_read	▪ mem_pixels ▪ mem_imatge_comprimida

Taula 5.8. Recursos hardware de l'arquitectura 2

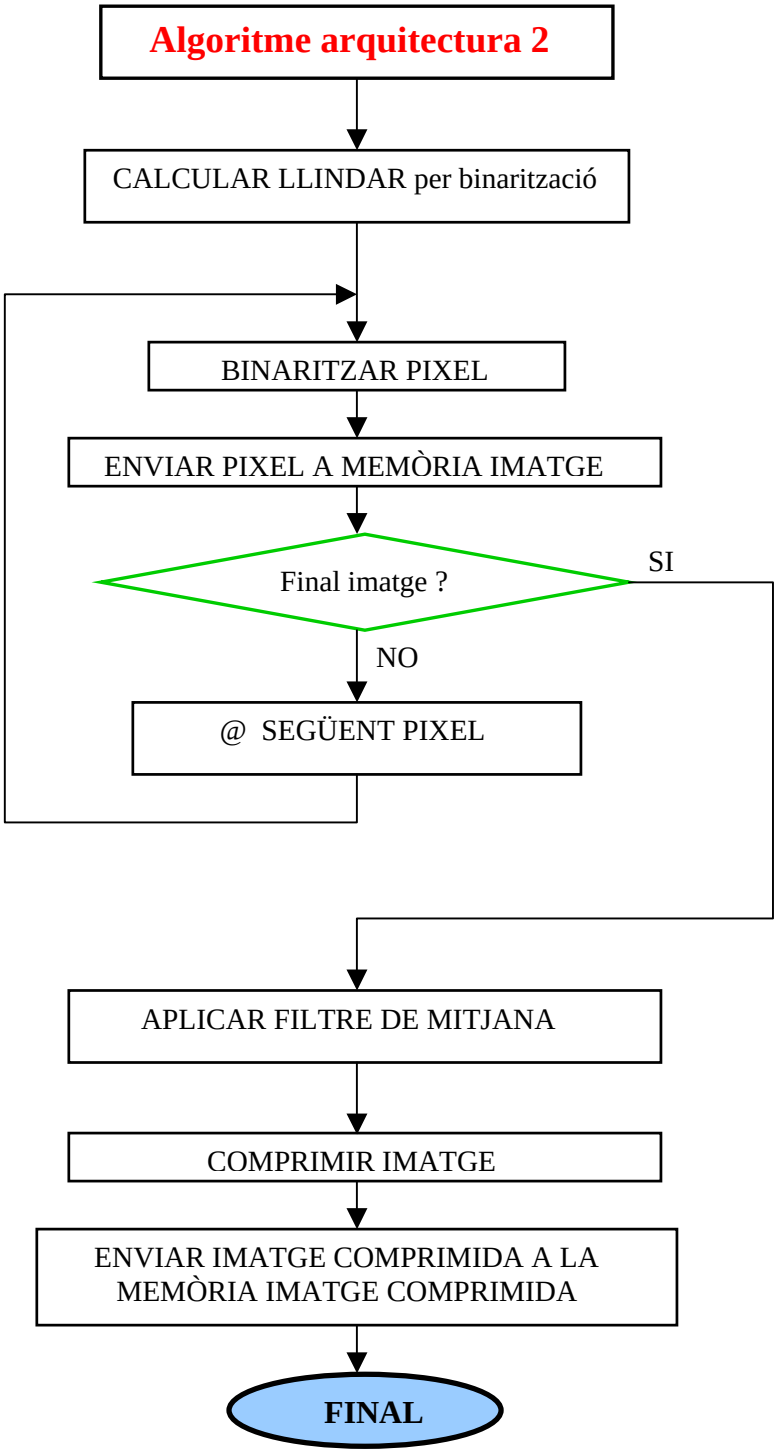


Figura 5.18. Diagrama de flux de l'algoritme de l'arquitectura 2

5.4.3 Arquitectura 3.

L'arquitectura 3 realitza un aprimament a la imatge després de binaritzar-la i enregistrar-la a la memòria *mem_imatge*. La imatge aprimada s'enregistra a les mateixes posicions de memòria que l'original, substituint aquesta. El diagrama de flux de l'algoritme que implementa es mostra a la Figura 5.19.

La implementació del mòdul-OCR amb l'arquitectura 3 es troba a l'arxiu *algoritme_3.vhd*, i la del controlador corresponent a *control_algoritme_3.vhd*.

La Taula 5.9 mostra els blocs funcionals que integren l'arquitectura 3.

BLOCS FUNCIONALS SEQÜENCIALS sintetitzats	**BLOCS COMBINACIONALS DE CONTROL D'ACCÉS**	**MÒDULS DE MEMÒRIA necessaris**
▪ **control_algoritme_3** ▪ calcul_llindar ▪ binaritzador ▪ compresor ▪ aprimament_imatge ▪ aprimament_nord ▪ aprimament_est ▪ aprimament_sud ▪ aprimament_oest	▪ arbitre_camara_read ▪ arbitre_mem_pixels_write ▪ arbitre_mem_pixels_read ▪ arbitre_mem_pixels_write_4 ▪ arbitre_mem_pixels_read_4	▪ mem_pixels ▪ mem_imatge_comprimida

Taula 5.9. Recursos hardware de l'arquitectura 3

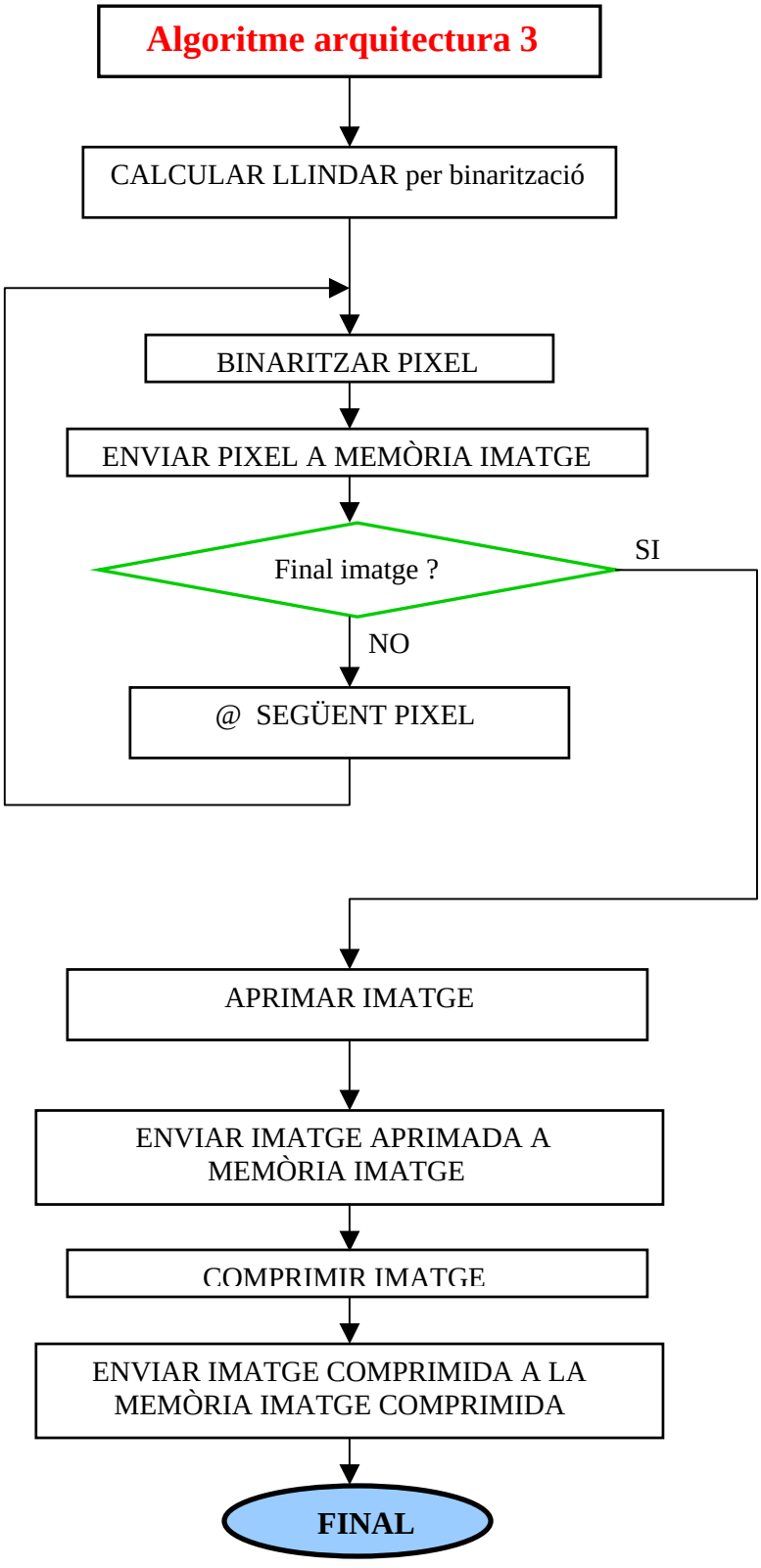


Figura 5.19. Diagrama de flux de l'algoritme de l'arquitectura 3

5.4.4 Arquitectura 4.

L'arquitectura 4 realitza les mateixes funcions que l'anterior excepte que al final del procés, en lloc de comprimir la imatge, en calcula els encreuaments horitzontals i verticals. Els valors calculats son emmagatzemats en registres interns dels blocs *calcul_encreuaments_horitzontals* i *calcul_encreuaments_verticals*. El diagrama de flux de l'algoritme que implementa es mostra a la Figura 5.20.

La implementació del mòdul-OCR amb l'arquitectura 4 es troba a l'arxiu *algoritme_4.vhd*, i la del controlador corresponent a *control_algoritme_4.vhd*.

La Taula 5.10 mostra els blocs funcionals que integren l'arquitectura 4.

BLOCS FUNCIONALS SEQÜENCIALS sintetitzats	**BLOCS COMBINACIONALS DE CONTROL D'ACCÉS**	**MÒDULS DE MEMÒRIA necessaris**
▪ **control_algoritme_4** ▪ calcul_llindar ▪ binaritzador ▪ compresor ▪ calcul_encreuaments_horitzontals ▪ calcul_encreuaments_verticals ▪ aprimament_imatge ▪ aprimament_nord ▪ aprimament_est ▪ aprimament_sud ▪ aprimament_oest	▪ arbitre_camara_read ▪ arbitre_mem_pixels_write ▪ arbitre_mem_pixels_read ▪ arbitre_mem_pixels_write_4 ▪ arbitre_mem_pixels_read_4	▪ mem_pixels

Taula 5.10. Recursos hardware de l'arquitectura 4

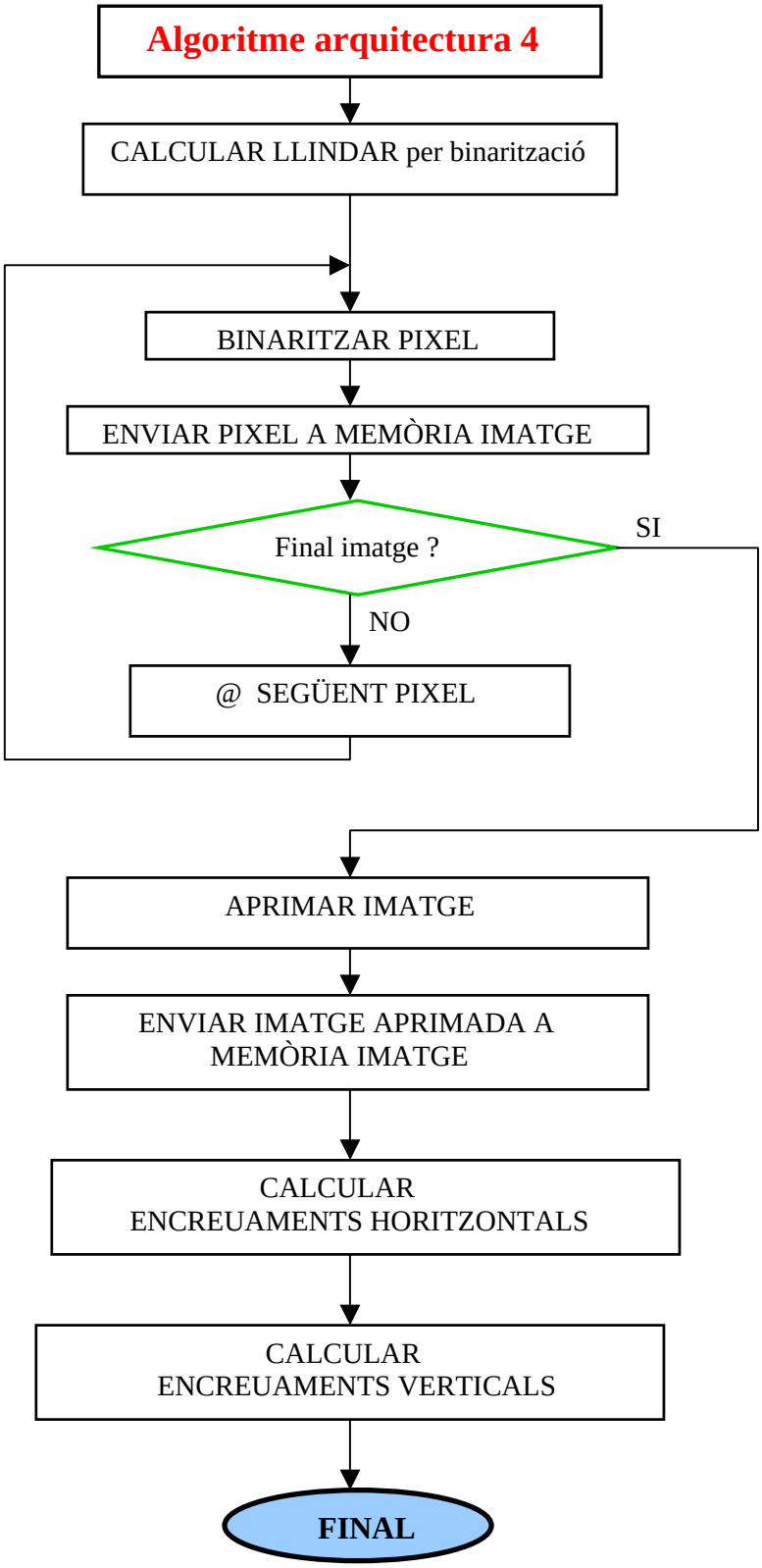


Figura 5.20. Diagrama de flux de l'algoritme de l'arquitectura 4

5.4.5 Arquitectura 5.

L'arquitectura 5 implementa un algoritme de "zoning". Amb la imatge procedent de la càmera, calcula els valors llindars, en funció dels pixels més clar i més fosc. Posteriorment, obté els valors corresponents a les zones delimitades, que corresponen a grups de 6 pixels, obtenint així un vector de (13 x 13) elements de 2 bits cadascun. Els 338 bits que representen la imatge processada, son enregistrats a la memòria *mem_imatge_gris*. El diagrama de flux de l'algoritme que implementa es mostra a la Figura 5.21.

La implementació del mòdul-OCR amb l'arquitectura 5 es troba a l'arxiu *algoritme_5.vhd*, i la del controlador corresponent a *control_algoritme_5.vhd*.

La Taula 5.11 mostra els blocs funcionals que integren l'arquitectura 5.

BLOCS FUNCIONALS SEQÜENCIALS sintetitzats	**BLOCS COMBINACIONALS DE CONTROL D'ACCÉS**	**MÒDULS DE MEMÒRIA necessaris**
▪ **control_algoritme_5** ▪ calcul_llindar_zonning ▪ zoning	▪ arbitre_camara_read	▪ mem_pixels_gris

Taula 5.11. Recursos hardware de l'arquitectura 5

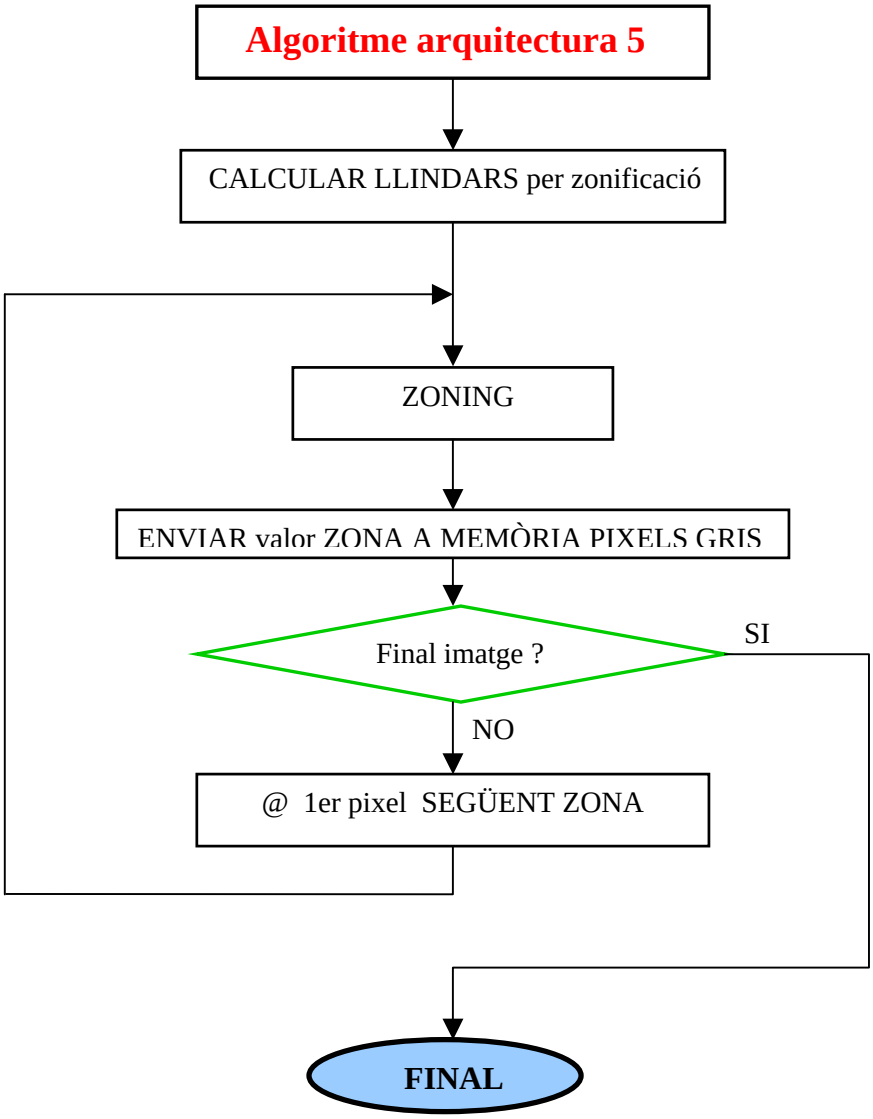


Figura 5.21. Diagrama de flux de l'algoritme de l'arquitectura 5

5.5 Síntesis i test dels mòduls de processament d'imatges.

5.5.1 Test dels mòduls sintetitzats.

Un cop comprovat el correcte funcionament de les diferents arquitectures del mòdul-OCR implementades, es procedeix a la seva síntesis amb la tecnologia C35B4, i a la verificació del seu bon funcionament a aquest nivell.

El procés és anàleg al seguit per la síntesis i test dels blocs funcionals per separat, afegint al procés de test, els retards produïts per les portes lògiques del mòdul sintetitzat. Aquests retards es simulen amb uns arxius proporcionats per l'entorn *Synopsys Design Analizer*, a partir d'especificacions de la tecnologia. Aquests arxius tenen el format **.sdf*, i permeten, en el procés de simulació, observar el comportament amb els retards produïts pels components tecnològics que integren el mòdul sintetitzat. Per tal de realitzar la simulació considerant retards, s'han de seguir les especificacions indicades a [Rub05].

A continuació es mostra el resultat d'una mateixa simulació sense considerar (Figura 5.22) i considerant retards (Figura 5.23). En ella s'observa com en el cas de no considerar retards de portes, *Modelsim* considera les entitats lògiques com ideals, i això es pot comprovar en el fet que les senyals *[get_dada_cam]*, *[dada_ready_cam]* i el bus *[dada_cam]* commuten al mateix instant que el flanc de pujada del rellotge *[clk]*. Mentre que en cas de simular considerant retards, aquestes mateixes senyals en el mateix instant de temps commuten un ns després del flanc.

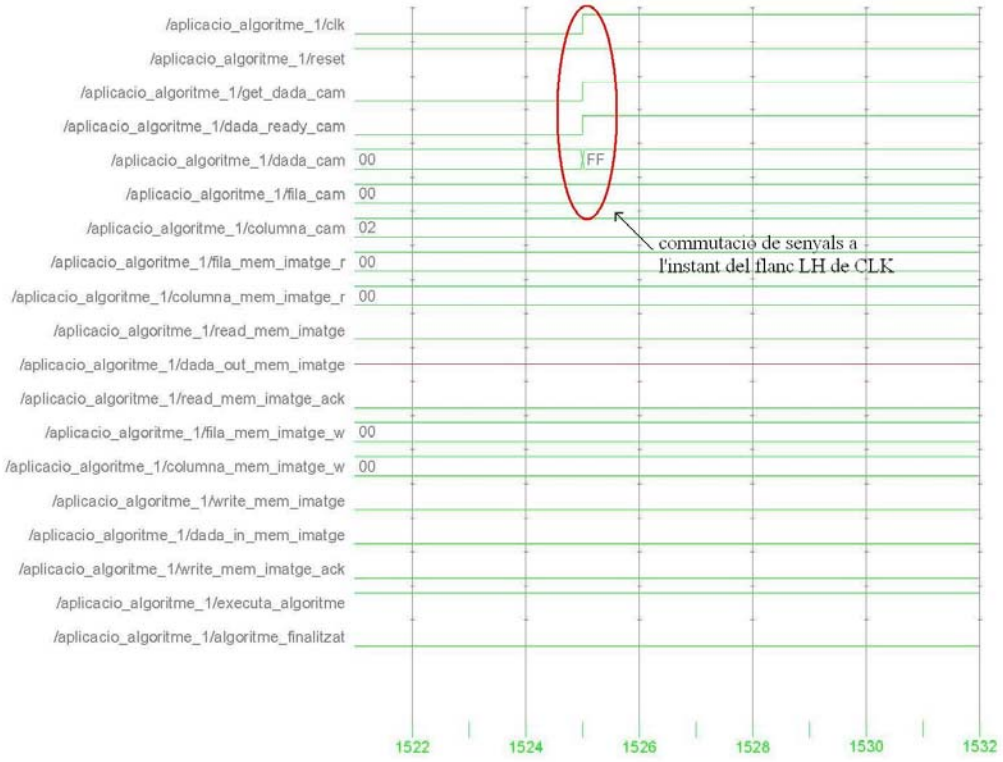


Figura 5.22. Exemple de simulació sense considerar retards de portes

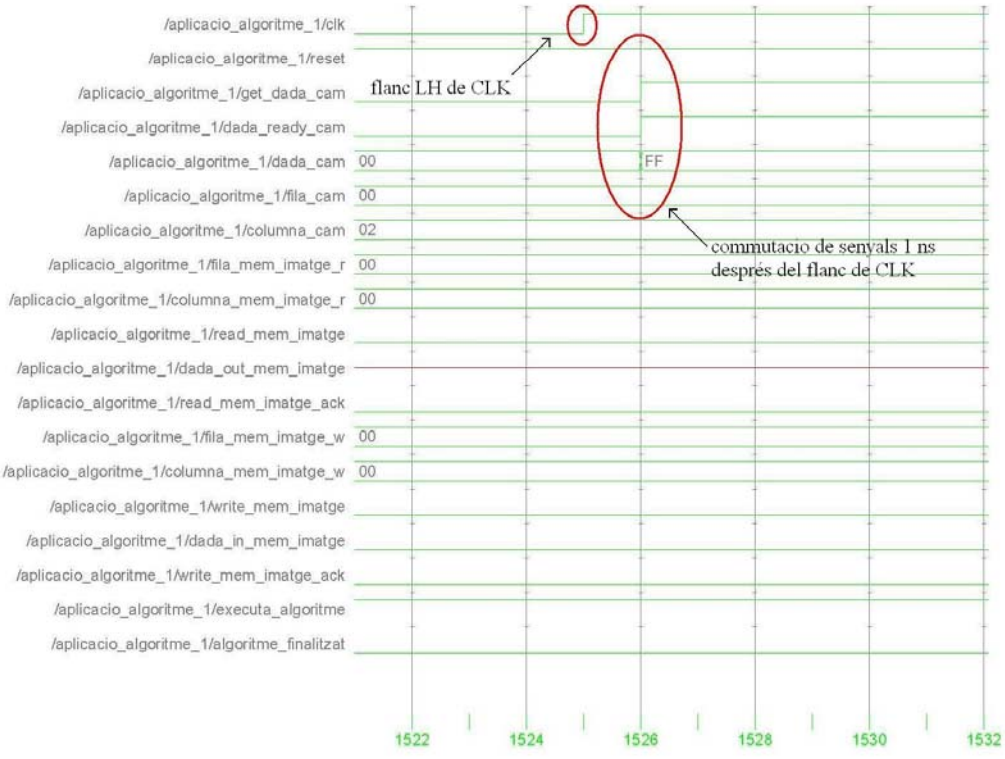


Figura 5.23. Exemple de simulació considerant retards de portes

5.5.2 Test sobre FPGA.

Un cop finalitzat el procés de test dels mòduls sintetitzats, es procedeix a fer la simulació dels diferents algoritmes sintetitzats a una FPGA [Car00], per a provar el seu funcionament. S'utilitza una plataforma anomenada *PAC*, que es basa en una *APEX20K* d' *Altera*. Aquesta plataforma disposa de 4 ports d'expansió per connectar la majoria de pins I/O de la FPGA a l'exterior. Altres característiques de la *PAC* són un regulador de tensió a 3,3 Volts i un generador de senyal de rellotge a 33 MHz. Així mateix també disposa d'un connector ByteBlaster per a programar la FPGA a partir d'una connexió al port paral·lel d'un ordinador. La Figura 5.25 mostra aquest entorn de test.

Aquesta plataforma és la que s'ha utilitzat també per caracteritzar la càmera de l'ASIC en el seu procés de disseny. Per tal de poder realitzar la simulació, hem de fer unes petites variacions.

Els arxius que implementen les respectives aplicacions del mòdul-OCR (arxius *aplicacio_algoritme_n.vhd*) son substituïts pels arxius *aplicacio_algoritme_n_fpga.vhd*. En aquests, a diferència del disseny original, es sintetitzen tots els recursos hardware emprats i els senyals de rellotge *[clk]* i reset global *[reset]* son entrades a la FPGA i per tant seran generats de forma real.

Per tal de simular la càmera, es disposa d'un arxiu *fotos.exe*, el codi font del qual és *bmpafpga.cpp*, que a partir d'una imatge en format **.bmp*, i d'un arxiu **.vhdl* patró (*FPGAbasic.vhd*), genera un arxiu *FPGAcamara.vhd*, que implementa una memòria ROM amb els valors dels pixels de la imatge a les corresponents adreces.

Aquesta memòria ROM, juntament amb les altres memòries emprades en el disseny i la implementació del mòdul-OCR, seran sintetitzades sobre la FPGA, amb l'entorn *Quartus II* versió *4.0*.

Per tal de poder observar els resultats, l'entorn *Quartus* ens permet de reservar memòria a la FPGA, que durant la simulació, emmagatzema els valors dels senyals que indiquem, ens els instants de temps consignats. En el nostre cas utilitzarem aquest recurs per emmagatzemar els valors de la imatge després del processament, d'una manera similar al bloc *imatge_bmp*. Al final del procés es genera un arxiu **.txt*, amb aquests valors. Posteriorment, l'arxiu *fpga2bmp.exe*, el codi font del qual és *fpga2bmp.cpp*, converteix aquest arxiu **.txt*, en el corresponent **.bmp* per poder observar els resultats.

La Figura 5.24 mostra aquest procés.

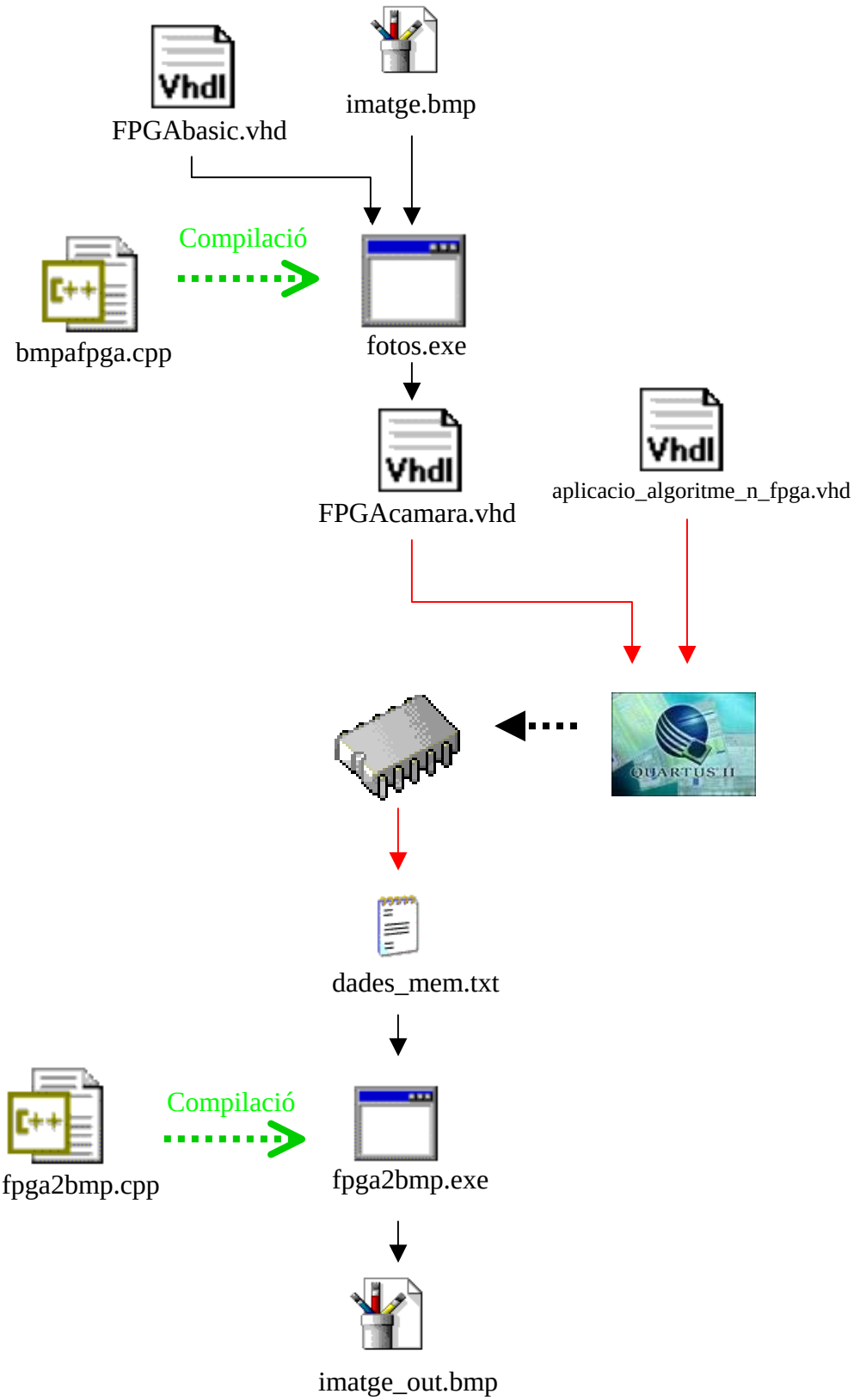


Figura 5.24. Interacció amb la FPGA

El procés ha estat un èxit en el cas de l'arquitectura 1, però no ha sigut possible de realitzar aquest test amb les altres arquitectures, perquè els recursos de la FPGA no eren suficients.

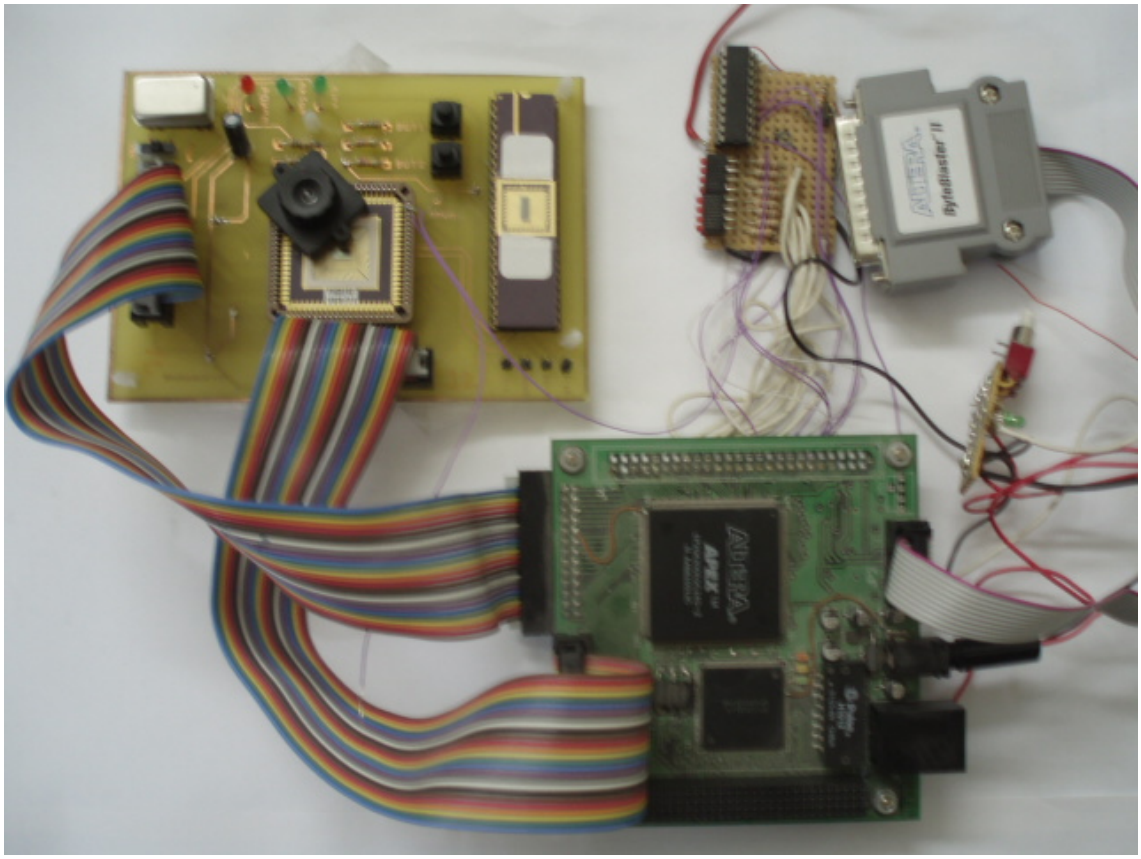


Figura 5.25. Entorn de test FPGA i càmera

5.6 Comparació dels diferents dissenys i determinació del més òptim.

Finalment, un cop realitzats els processos de síntesis de les diferents arquitectures, i comprovat el seu correcte funcionament, procedim a la seva comparació per determinar la més adient a les nostres necessitats.

Per tal de realitzar la comparativa, tindrem en compte els següents paràmetres:

- Àrea i consum de potència del mòdul-OCR sintetitzat en C35B4.
- Àrea del mòdul sintetitzat de memòria.
- Ràtios de compressió de dades.
- Resultats gràfics amb les imatges disponibles.

Les següents taules mostren, de forma comparativa, les 5 arquitectures de mòdul-OCR implementades en Hardware i els seus principals paràmetres.

5.6.1 Comparació de l'àrea i consum de potència.

A la Taula 5.12 podem veure els resultats de la síntesis del mòdul-OCR per cadascuna de les 5 possibles arquitectures, i la Taula 5.13 mostra quins del blocs funcionals implementats formen part del mòdul-OCR per cada arquitectura.

ARQ.	descripció	ports	nets	cells	àrea [mm^2]	P [mW]
1	binarització + compressió	69	126	5	0.086	24.47
2	binarització + filtre + compressió	97	205	10	0.149	35.63
3	binarització + aprimament + compressió	97	295	20	0.320	81.96
4	binarització + aprimament + calcul encreuaments	105	334	22	0.543	139.73
5	zonificació	37	128	4	0.148	39.54

Taula 5.12. Resultats de la síntesis del mòdul-OCR segons arquitectura

ARQUITECTURA	1	2	3	4	5
Calcul_llindar	☑	☑	☑	☑	
Binaritzador	☑	☑	☑	☑	
Filtre_9_pixels		☑			
Control_filtre_9_pixels		☑			
Compresor	☑	☑	☑		
Calcul_encreuaments_horitzontals				☑	
Calcul_encreuaments_verticals				☑	
Aprimament_imatge			☑	☑	
Aprimament_nord			☑	☑	
Aprimament_est			☑	☑	
Aprimament_sud			☑	☑	
Aprimament_oest			☑	☑	
Zoning					☑
Calcul_llindar_zoning					☑
arbitre_camara_read	☑	☑	☑	☑	☑
arbitre_mem_pixels_read		☑	☑	☑	
arbitre_mem_pixels_write		☑	☑	☑	
arbitre_mem_pixels_read_4			☑	☑	
arbitre_mem_pixels_write_4			☑	☑	

Taula 5.13. Blocs funcionals integrants del mòdul-OCR segons arquitectura

5.6.2 Comparació de l'àrea del mòdul de memòria sintetitzat.

A la Taula 5.14 podem veure quins recursos de memòria necessiten els dissenys de cadascuna de les cinc arquitectures.

Els mòduls de memòria que s'han implementat per realitzar la simulació dels dissenys, no corresponen a cap mòdul de memòria disponible comercialment. De manera que serà necessari obtenir una relació de mòduls de memòria sintetitzables i integrables amb la tecnologia C35B4 i a partir d'ella determinar les capacitats de memòria que cada arquitectura necessita. Aquesta relació ens permetrà determinar l'àrea ocupada per part dels mòduls de memòria sintetitzables.

La Taula 5.15 mostra una relació d'àrees de diferents mòduls sintetitzables de memòria RAM estàtica (SRAM), en funció de la quantitat de memòria (en kbits), obtinguda de [AMS02]. Aquestes memòries son del tipus DPRAM, és a dir que es pot llegir i escriure alhora a diferents adreces.

ARQUITECTURA	Recursos de memòria requerits (Mòduls de memòria utilitzats)
1	1 x Mem_pixels (1,024 bits) + 1 x Mem_imatge_comprimida (4,096 bits) **= 5,120 bits**
2	2 x Mem_pixels (1,024 bits) + 1 x Mem_imatge_comprimida (4,096 bits) **= 6,144 bits**
3	2 x Mem_pixels (1,024 bits) + 1 x Mem_imatge_comprimida (4,096 bits) **= 6,144 bits**
4	2 x Mem_pixels (1,024 bits) **= 2,048 bits**
5	Mem_pixels_gris (512 bits) **= 512 bits**

Taula 5.14. Recursos de memòria requerits pel mòdul-OCR segons arquitectura

Àrea [mm^2]							
Procés	1k bit	2k bit	4k bit	8k bit	16k bit	32k bit	64k bit
C35	0.34	0.48	0.75	1.26	2.23	3.78	7.15

Taula 5.15. Àrees de mòduls de memòria SRAM sintetitzables; font: [AMS02].

Com que els valors de capacitat de memòria només poden prendre valors sencers de logaritmes binaris i la capacitat mínima és de 1kbit, les capacitats requerides per les arquitectures son les que es mostren a la Taula 5.16. Les àrees dels mòduls de SRAM de AMS necessaris per cada arquitectura es relaciones a la Taula 5.17.

El mòdul-OCR en la seva implementació es comunica, tant per llegir com per escriure, amb diferents mòduls virtuals de memòria, i en els accessos a aquests les adreces tenen en alguns casos el format de (fila, columna). De manera que a l'hora d'integrar el mòdul físic de memòria serà necessari integrar, entre el mòdul-OCR i els punts d'accés al mòdul de memòria per part d'aquest, mòduls adreçadors de memòria. L'existència d'aquests també permetrà que el mòdul de memòria pugui ser utilitzat per altres mòduls de l'ASIC i siguin els mòduls adreçadors els que, en funció de l'origen de l'accés, apuntin a una o altra adreça.

ARQUITECTURA	**1**	**2**	**3**	**4**	**5**
mínims bits memòria	5,120	6,144	6,144	2,048	512
$\log_2$ (mín. bits mem.)	12.32	12.58	12.58	11.00	9.00
bits @ mòdul SRAM AMS	13	13	13	11	9
mínims bits mòdul SRAM AMS	8,192	8,192	8,192	2,048	512
mínims kbits mòdul SRAM AMS	**8**	**8**	**8**	**2**	**1**

Taula 5.16. Capacitat de memòria sintetitzable requerida segons arquitectura

ARQUITECTURA	Recursos de memòria requerits (Àrea mòduls SRAM AMS)
1	8 kbits → 1.26 mm^2
2	8 kbits → 1.26 mm^2
3	8 kbits → 1.26 mm^2
4	2 kbits → 0.48 mm^2
5	1 kbits → 0.34 mm^2

Taula 5.17. Àrees de memòria sintetitzable requerides per les arquitectures

5.6.3 Comparació dels ràtios de compressió de dades.

Un cop obtinguts els valors de les àrees requerides per integrar el mòdul de memòria sintetitzable necessària per cada arquitectura, procedim a comparar els ràtios de compressió de dades. A la Taula 5.18 es mostren aquests ràtios per cada arquitectura.

Es pot veure com amb les arquitectures 4 i 5 és amb les s'obtenen els millors resultats.

ARQUITECTURA	bits entrada (bits imatge)	[*] bits sortida	Ràtio de compressió [%] (bits sortida / bits entrada)
1	8,112	760	**9.374 %**
2	8,112	732	**9.029 %**
3	8,112	886	**10.917 %**
4	8,112	286	**3.526 %**
5	8,112	338	**4.167 %**

Taula 5.18. Ràtios de compressió de dades de les diferents arquitectures

[*] En les arquitectures 1, 2 i 3 les dades corresponen a valors promig entre les deu imatges dels dígits. En aquestes arquitectures el nombre de bits de sortida és variable depenent de la imatge.

5.6.4 Comparació de resultats gràfics amb imatges disponibles.

Aquest paràmetre, tot i no ser analític, te una rellevància donat que el procés consisteix en processar imatges que posteriorment han de ser identificades. No és un paràmetre fonamental, però en cas de dubte entre dos o més dissenys en els quals les altres prestacions siguin comparables, aquesta comparació gràfica pot ser un bon element de decisió.

A la Figura 5.26 es pot veure, per cada arquitectura, la imatge resultant per en cas que la imatge d'entrada sigui la del dígit '7'.

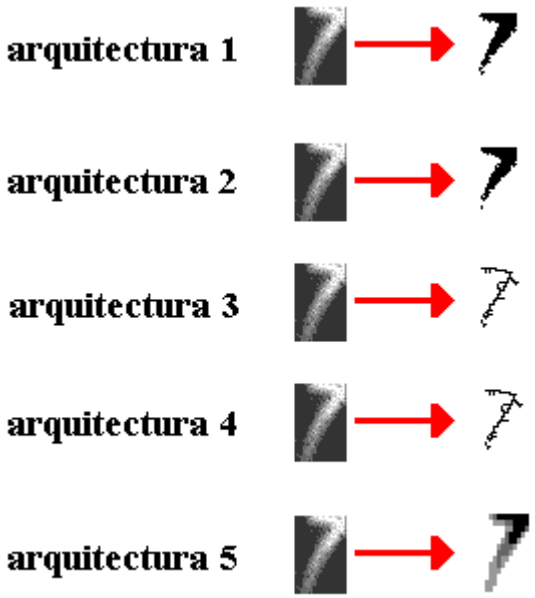


Figura 5.26. Imatges resultants del processat, segons arquitectura

5.6.5 Determinació del disseny més òptim.

Finalment, a la Taula 5.19 podem veure els diferents paràmetres tinguts en compte a l'hora de realitzar la comparativa definitiva entre les diferents arquitectures possibles.

ARQUITECTURA	Ràtio de compressió de dades	Àrea mòduls sintetitzats	Àrea mòduls Memòria	Àrea total
	[%]	*[mm^2]*	*[mm^2]*	*[mm^2]*
1	9.374%	0.086	1.26	1.346
2	9.029%	0.149	1.26	1.409
3	10.917%	0.320	1.26	1.580
4	3.526%	0.543	0.48	1.023
5	4.167%	0.148	0.34	0.488

Taula 5.19. Resum de característiques de les arquitectures

Aquests paràmetres, juntament amb el consum de potència del mòdul i la comparació dels resultats gràfics, constitueixen els criteris que determinaran l'arquitectura per la qual s'opta.

Per cada criteri, s'ordenaran les alternatives en funció dels valors dels respectius paràmetres, considerant com a 5, el disseny amb resultats més satisfactoris i com a 1 el contrari. Posteriorment, les puntuacions es ponderaran per cada criteri, segons la importància d'aquest en la decisió final, i es sumaran les puntuacions per cada alternativa. L'alternativa amb millor puntuació, serà l'escollida.

Criteri del ràtio de compressió de dades.

Considerarem millor, com menor sigui, és a dir com menor bits hagi d'enviar. Aquest criteri rebrà una importància relativa del 40 % donat que és un dels principals condicionants en el desenvolupament del projecte.

Criteri de l'àrea necessària per integrar el disseny (mòdul + memòria).

Considerarem millor, com menor sigui. Aquest criteri rebrà una importància relativa del 30 %.

Criteri del consum de potència del mòdul.

Considerarem millor, com menor sigui. Aquest criteri rebrà una importància relativa del 10 %.

Criteri de la comparació del resultat gràfic.

Considerem que a la vista dels resultats observats, una mostra dels quals s'il·lustra a la Figura 5.26, l'arquitectura que introdueix una major distorsió en la imatge son la 4 i la 3, i la que menor distorsió introdueix és la 1. Les arquitectures 2 i 5 introdueixen una distorsió moderada, essent major en el darrer cas.

Aquest criteri rebrà una importància relativa del 20 %.

Comparació final.

ARQUITECTURA	ponderació criteri	1	2	3	4	5
Ràtio de compressió de dades	*40%*	2	3	1	5	4
Àrea (mòdul + memòria)	*30%*	3	2	1	4	5
Consum de potència mòdul-OCR	*10%*	5	4	2	1	3
Resultat gràfic	*20%*	5	4	1	2	3
TOTAL	*100%*	3.20	3.00	1.10	3.70	**4.00**

Taula 5.20. Comparativa de les diferents arquitectures

Com podem observar, la millor opció sembla ser l'arquitectura 5.

L'arquitectura 4 està a una puntuació bastant propera. Si considerem únicament els paràmetres analítics, la millor opció semblaria ser la que correspon a l'arquitectura 4. És la que dona un millor ràtio de compressió de dades, i els recursos hardware que requereix (Àrea i Potència) estan dintre del permès per les especificacions de disseny.

No obstant, i tenint en compte que l'objectiu final no deixa de ser reduir la quantitat d'informació transmesa, procurant en qualsevol cas que el dígit sigui identificable al punt de recepció, la distorsió produïda pels processos de les arquitectures 3 i 4 deixen el dígit amb una qualitat visual notablement baixa. Aquesta conclusió es raona considerant únicament les imatges de les que disposem, donat que en cas de disposar de més mostres, les conclusions al respecte serien molt més consistents.

Per tot el que s'ha exposat, l'arquitectura més apropiada resulta ser la 5, que implementa un algoritme que zonifica la imatge, i al final del procés de la qual, la imatge ha patit variacions que permeten, a simple vista identificar el dígit corresponent. Per la qual cosa considerem que el sistema d'identificació al punt de recepció podrà realitzar aquesta tasca amb una mínima seguretat.

6 Resultats

6.1 Descripció del disseny escollit.

En el present capítol s'explica detalladament l'arquitectura escollida pel mòdul-OCR que estem dissenyant. També s'il·lustra l'encaix d'aquest mòdul amb aquesta arquitectura, a l'ASIC del sistema. L'opció escollida correspon a l'arquitectura 5. A la Taula 6.1 es mostren els principals paràmetres de l'arquitectura.

PARÀMETRE	Valor per arquitectura 5
Ports	37
Nets	128
Cells (blocs funcionals)	4
Àrea [mm^2]	0.148
P [mW]	39.54
Memòria necessària [bits]	512
Àrea de memòria necessària [mm^2]	0.34
Ràtio de compressió de dades [%]	4.167 %
Blocs funcionals integrants	▪ **Control_algoritme_5** ▪ Zonning ▪ Calcul_llindar_zonning ▪ Arbitre_camara_read

Taula 6.1. Paràmetres del mòdul-OCR

La Taula 6.2 ens mostra la relació dels ports d'entrada i sortida del mòdul hardware de l'arquitectura 5, així com una explicació de les funcions de tots ells.

port	I/O	funció
clk	I	Senyal de rellotge
reset	I	Reset global
executa_algoritme	I	Senyal per iniciar l'execució de l'algoritme
algoritme_finalitzat	O	Activa quan l'algoritme ha acabat i les dades estan a la memòria corresponent
get_dada_cam	O	Petició de pixel a la càmera
dada_ready_cam	I	Activa quan la dada de la càmera està a [dada_cam]
dada_cam [8]	I	Dada d'entrada de la càmera
fila_cam [6]	O	@ del pixel de la càmera
columna_cam [5]	O	@ del pixel de la càmera
fila_mem_pixels_gris_w [4]	O	@ fila de la memòria dels valors dels colors de les zones de la imatge, mode escriptura
columna_mem_pixels_gris_w [4]	O	@ columna de la memòria dels valors dels colors de les zones de la imatge, mode escriptura
write_mem_pixels_gris	O	Senyal d'escriptura de la memòria de la zonificació
dada_in_mem_pixels_gris [2]	O	Dada per escriure a la memòria de la zonificació
write_mem_pixels_gris_ack	I	Activa quan la dada per escriure ja està registrada

Taula 6.2. Mapa de ports I/O del mòdul-OCR

A continuació es mostren els resultats del processament de les imatges de les que disposem, per part del mòdul dissenyat.

El que s'enviarà per RF un cop realitzat el processat de cada imatge serà un vector de 338 bits. Si els bits d'aquest vector es prenen de dos en dos, i a cada grup de 6 pixels corresponents a les zones que s'han pres per realitzar el "zoning" a la imatge original, se li assigna un valor de blanc, gris clar, gris fosc o negre (els 6 pixels iguals) segons el valor de la zona, les imatges resultants serien les mostrades a la Figura 6.1.

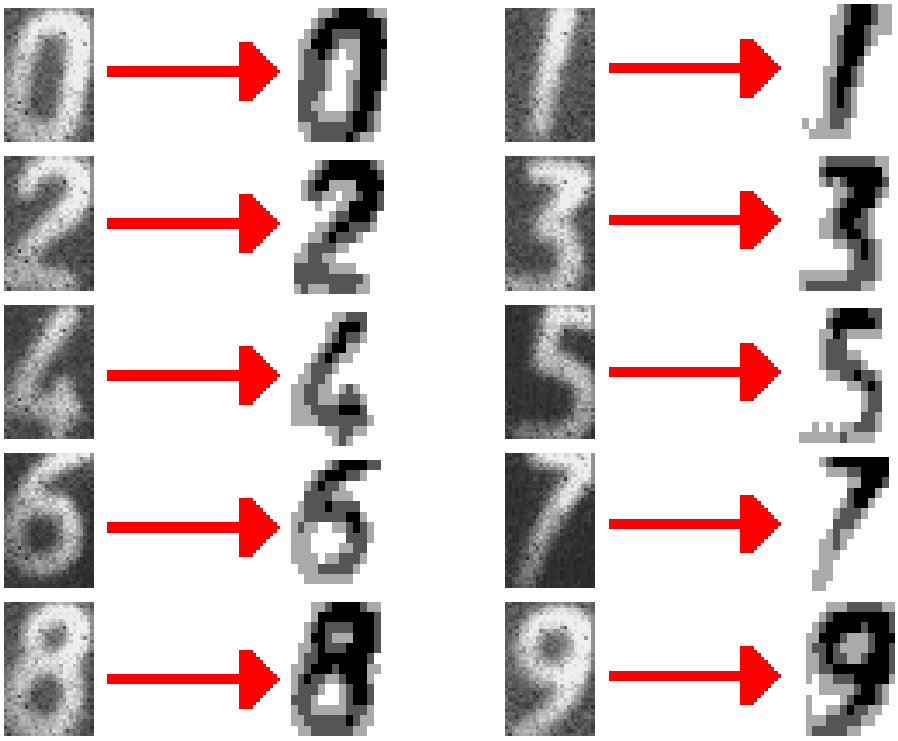


Figura 6.1. Imatges resultants del processament del mòdul-OCR

La Figura 6.2 ens mostra l'encaix del mòdul-OCR amb l'arquitectura escollida a l'ASIC del sistema intel·ligent de captura d'imatges.

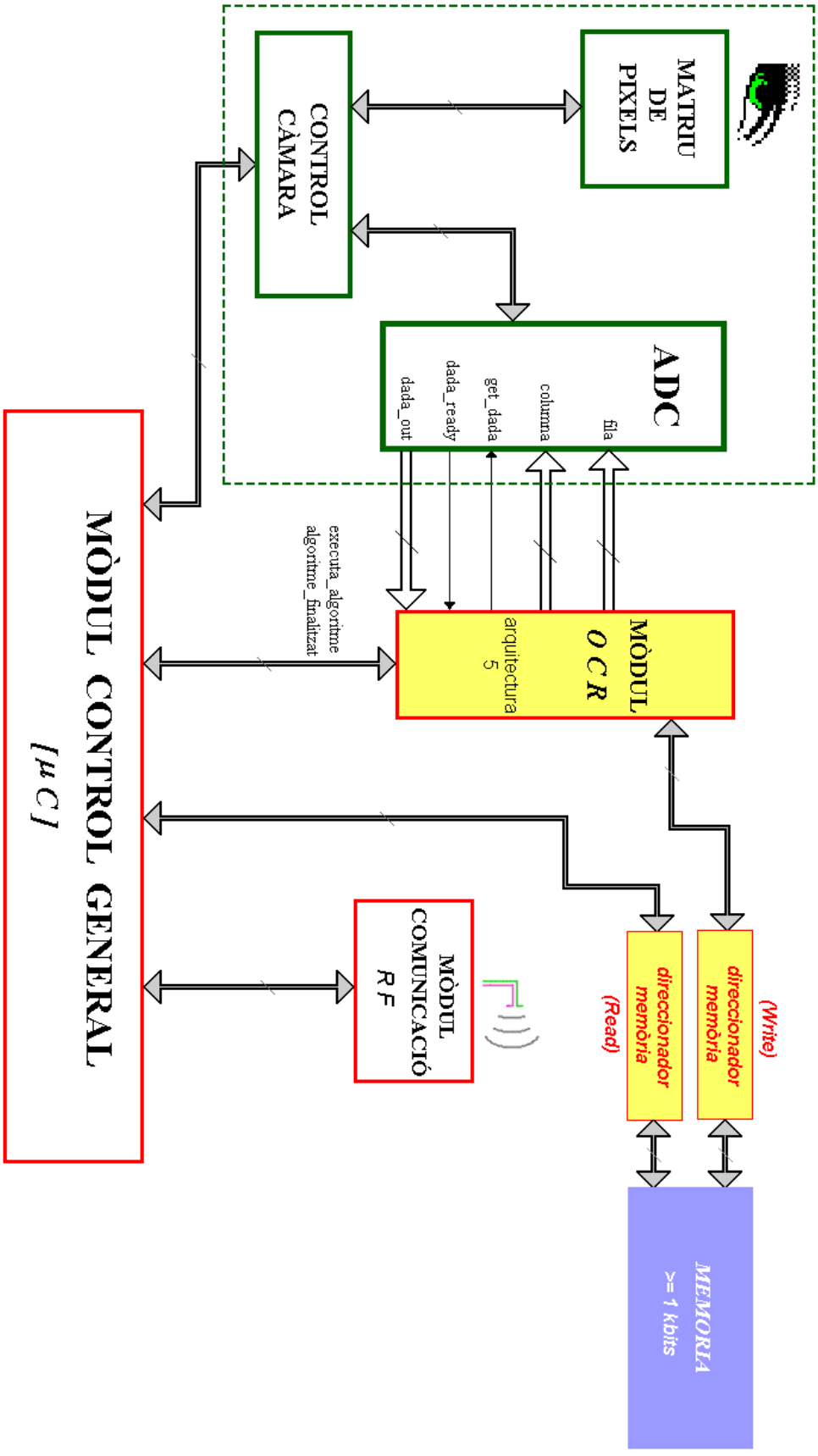


Figura 6.2. Mòdul-OCR integrat a l'ASIC

6.2 Limitacions del disseny escollit.

La principal limitació del disseny escollit, és que requereix que el procés d'identificació dels dígits al punt de recepció de la companyia, es faci emprant el mètode de "zoning" i amb el tamany de zones fixats en el nostre disseny.

En cas que aquest reconeixement empri un altre mètode, el nostre disseny no aportaria grans beneficis al disseny final del sistema.

No obstant, els altres dissenys estudiats s'adaptarien bastant bé a altres mètodes d'identificació. El disseny corresponent a l'arquitectura 4, per exemple, aniria bé en cas que es fes el reconeixement pel mètode de "crossings", sempre i quan se li afegís el Filtre de Mitjana o algun altre que assegurés una millor qualitat en les imatges aprimades i per tant una major eficiència en el càlcul dels encreuaments.

Les altres arquitectures, tot i no realitzar funcions importants del procés de reconeixement o d'extracció de característiques, tenen, precisament per això, l'avantatge de poder-se adaptar a diferents processos de reconeixement.

7 Conclusions

7.1 Valoració dels resultats obtinguts.

Dels objectius fixats inicialment s'ha aconseguit dissenyar un mòdul hardware, integrable a l'ASIC del sistema sobre el que s'ha treballat, que redueix substancialment la quantitat d'informació a transmetre per RF entre el comptador d'aigua i el punt de recepció de dades de la companyia. No s'ha pogut implementar en hardware un algoritme OCR sencer, que identifiqués els dígits del comptador completament, tot i que el mòdul dissenyat n'executa una part. Tanmateix, la reducció en el volum de dades a enviar és molt notable i es pot deixar pel node de recepció de dades de la companyia la tasca final del reconeixement de la lectura.

A més de comprovar el funcionament de l'algoritme un cop sintetitzat amb les portes pròpies de la tecnologia de destí, la C35B4 d'*Austriamicroystems*, també s'ha comprovat el correcte funcionament del mòdul sobre una FPGA.

La valoració en conjunt dels resultats obtinguts és positiva ja que s'ha aconseguit reduir la quantitat d'informació per transmetre a un 4 % de la que seria en cas de no realitzar-se el processament, tot això amb un increment d'àrea inferior al 3 % respecte a l'àrea original ocupada per la càmera a l'ASIC.

En el desenvolupament d'aquest projecte, s'ha treballat en diversos nivells d'abstracció. S'han implementat en llenguatge d'alt nivell (C++) algoritmes OCR. Posteriorment, s'ha treballat amb llenguatge de descripció de hardware (VHDL), implementant blocs funcionals que executen parts dels algoritmes OCR. En la següent fase, s'han dissenyat i provat diferents arquitectures possibles del mòdul de processament d'imatges, a partir dels blocs funcionals prèviament implementats. Aquestes diferents arquitectures, s'han sintetitzat utilitzant la tecnologia C35B4, i s'han realitzat processos de test, tant a nivell de simulació, considerant les especificacions tecnològiques, com amb una FPGA.

El no disposar de més mostres d'imatges ha limitat la capacitat a l'hora de realitzar la comparació de les possibles arquitectures i de decidir-se per una d'elles. En cas d'haver disposat de més mostres es podria haver realitzat una decisió més fonamentada. El criteri de l'aparença de la imatge resultant, tot i no ser analític, ha tingut una rellevància a l'hora de prendre la decisió de la millor arquitectura. Això està motivat en el fet que una condició *sine qua non* és que la informació que es transmeti per RF permeti realitzar el reconeixement del dígit.

En cas de comprovar, amb més mostres d'imatges, que aquest disseny o algun altre dels implementats en aquest projecte, no donen una certesa alta en el procés de reconeixement, cap la possibilitat de dissenyar el protocol de comunicació entre l'ASIC i el punt de recepció de dades de la companyia, perquè en cas de produir-se un cert grau d'incertesa en el reconeixement, el punt de recepció li ho comuniqui a l'ASIC. En aquest cas, aquest darrer podria enviar la imatge amb un altre processament, que tingui un menor ràtio de compressió, però una major probabilitat de certesa en el procés d'identificació.

No obstant, la valoració global és positiva, donat que s'ha implementat un mòdul de processament de la imatge, que redueix la quantitat d'informació per transmetre, com era l'objectiu, i realitza una part d'un algoritme OCR complert. Això alhora condiciona la seva viabilitat final al fet que el punt de recepció realitzi el procés de reconeixement amb el mateix mètode, que l'emprat en el procés del mòdul dissenyat.

7.2 Propostes d'ampliació.

Aquest projecte pot ampliar-se en diverses direccions. La principal ampliació seria el perfeccionament del disseny realitzat tan bon punt es disposi de més mostres d'imatges.

Una altra ampliació consistiria en completar el disseny de l'ASIC, programant el mòdul de control, així com també el desenvolupament de protocols de comunicació entre els diferents mòduls, i entre aquests i el dissenyat en aquest projecte.

També es podria treballar sobre la part relacionada amb el mòdul de comunicació per RF, estudiant possibles protocols adequats per la nostra aplicació.

7.3 Valoració personal.

Amb el desenvolupament del projecte l'experiència personal ha sigut molt positiva. Desenvolupar el projecte al Centre Nacional de Microelectrònica – Institut de Microelectrònica de Barcelona, m'ha servit per comprovar l'ambient de treball d'un centre de recerca i desenvolupament en el camp de l'electrònica.

Addicionalment, amb la realització d'aquest projecte, he pogut comprovar sobre el terreny, com és el procés de desenvolupament d'un sistema amb la finalitat de convertir-se en producte comercial.

Un altre camp en el que he adquirit experiència ha sigut el relacionat amb les eines de disseny de sistemes digitals integrats, com les utilitzades en el present projecte.

Els constants problemes, sobretot amb les processos de síntesis i test, i la cerca de solucions als mateixos, han servit per adquirir hàbits de treball, que seran molt útils per enfrontar-se a les situacions en les que, de ben segur, qualsevol enginyer es troba, sobretot a les primeres etapes de la seva vida laboral.

Concloent, la valoració personal és positiva, i crec que ha representat un esforç molt ben invertit, tant a nivell acadèmic, com a nivell d'experiència personal.

8 Referències

[Car00]. Carrabina, J.; Lisa, F; Velasco,A.J. *Implementación con FPGAs* Pg281-318. 2000. Capítol del llibre *Sistemas Digitales*.

[Des93]. Deschamps, Jean P. Diseño de Circuitos Integrados de Aplicación Especifica, ASIC. Editorial Paraninfo, 1993.

[Pon03]. Pons i Montserrat, Enric. *Disseny d'un sistema HW/SW de lectura de comptadors amb FPGA.* (Treball Final de Carrera, Enginyeria Electrònica). Universitat Autònoma de Barcelona, 2003.

[PRE96]. Comité PRENDA. *PRENDA: Metodología para el diseño de ASICs*. Universidad Politécnica de Madrid, ETSII, 1996.

[Pro96]. Proakis, J.G.; Manolakis, D.G. *Digital Signal Processing: Principles, algorithms and applications*. Prentice-Hall, Inc. 1996

[Rub05]. Rubio, Leire. *The ASIC Design Process With CNM Resources*. CNM-CSIC (Barcelona). Document intern CSIC.

[Sav]. Savas, Erkay. *VHDL basic I/O* [en línia]. Sabanci University. Adreça URL: <http://people.sabanciuniv.edu/erkays/el310/io_10.pdf>. [Consulta: octubre 2006].

[Sch]. Schulze, Mark A. [en línia]. Adreça URL: <http://www.markschulze.net/java/meanmed.html>. [Consulta: novembre 2006].

[Zha]. Zhang, Weijun. *VHDL Tutorial: Learn by Example* [en línia]. Adreça URL: <http://esd.cs.ucr.edu/labs/tutorial/>. [Consulta: juliol 2006].

[AMS01]. http://www.austriamicrosystems.com/. Web d'Austriamicrosystems. [Consulta: novembre 2006].

[AMS02]. http://asic.austriamicrosystems.com/databooks/digital/mc_dpram_c35.html. Web d'Austriamicrosystems; secció de memòries RAM bi-port. [Consulta: desembre 2006].

[GTC]. http://alojamientos.us.es/gtocoma/. Grupo de Topología Computacional y Matemática Aplicada. Universidad de Sevilla. [Consulta: octubre 2006].

9 Glossari

AD. Analògic-digital.

ADC. Analog-to-digital converter. *(Conversor d'analògic a digital).*

AMS. Austriamicrosystems.

APS. Active pixel sensor. *(Sensor de pixels actiu).*

ASIC. Application-specific integrated circuit.
(Circuit integrat per aplicacions especifiques).

BMP. Bit map. *(Mapa de bits).*

CMOS. Complementary metal–oxide–semiconductor.
(metall-òxid-semiconductor complementari).

C35B4. *Tecnologia Austriamicrosystems 0.35 μm C-MOS, amb 4 capes de metalls i 2 de poli-silicis, compatible amb tecnologia TSMC.*

DEMUX. Demultiplexor.

DPRAM. Dual-ported RAM. *(RAM de dos ports).*

FPGA. Field programmable gate array. *(Matriu de portes amb camp programable).*

HDL. Hardware description language. *(Llenguatge de descripció de hardware).*

IC. Integrated circuit. *(Circuit integrat monolític).*

LED. Light-emitting diode. *(díode emissor de llum).*

MUX. Multiplexor.

OCR. Optical character recognition. *(Reconeixement òptic de caràcters).*

Pixel. Picture element. *(element de la imatge).*

RAM. Random access memory. *(Memòria d'accés aleatori).*

RF. Ràdio-Freqüència.

ROM. Read-only memory. *(Memòria de només lectura).*

SRAM. Static random access memory. *(Memòria d'accés aleatori estàtica).*

TSMC. Taiwan Semiconductor Manufacturing Company.

VHDL. VHSIC + HDL.

VHSIC. Very High Speed Integrated Circuit. *(Circuit integrat de molt alta velocitat).*

Apèndixs

Apèndix A. Contingut del CD adjunt.

En aquest apèndix, es detalla el contingut del CD adjunt, i l'estructuració dels arxius en directoris.

L'estructuració del CD consisteix en els següents directoris:

ARXIUS

- ARXIUS_CPP
- ARXIUS_EXE
- ARXIUS_TEST_FPGA
- ARXIUS_VHDL
  - ARQUITECTURES_MODUL
  - LLIBRERIES_C35
  - SINTESIS
    - ARQUITECTURES_MODUL
  - TESTBENCH
- COMANDES_MODELSIM
- INFORMES_SINTESIS
- RETARDS

MEMORIA

I el seu contingut és el següent:

\MEMORIA.

- Memòria del projecte en format **.pdf*.

\ARXIUS\ARXIUS_CPP.

- Arxius font en C++ (**.cpp*).
  Algoritmes OCR i programes per convertir formats.

\ARXIUS\ARXIUS_EXE.

- Arxius executables (**.exe*).
  Algoritmes OCR i programes per convertir formats.

\ARXIUS\ARXIUS_TEST_FPGA.

- Arxius implicats en el test amb la FPGA (**.cpp, *.exe, *.vhd, *.bmp*).
  - aplicacio_algoritme_1_fpga.vhd.
  - FPGAbasic.vhd.
  - bmpafpga.cpp.
  - fpga2bmp.cpp.
  - fotos.exe.
  - fpga2bmp.exe.
  - imatge_patro.bmp.

\ARXIUS\ARXIUS_VHDL\.

- Arxius font en llenguatge VHDL (**.vhd*).
  Blocs funcionals, camara, mòduls de memòria i controladors algoritmes.

\ARXIUS\ARXIUS_VHDL\ARQUITECTURES_MODUL.

- Arxius font en llenguatge VHDL (**.vhd*).
  Implementació de les 5 arquitectures del mòdul OCR.

\ARXIUS\ARXIUS_VHDL\LLIBRERIES_C35.

- Arxius font en llenguatge VHDL (**.vhd*).
  Llibreries de la tecnologia C35B4.

\ARXIUS\ARXIUS_VHDL\SINTESIS.

- Arxius font en llenguatge VHDL (**.vhd*).
  Blocs funcionals sintetitzats en tecnologia C35B4.

\ARXIUS\ARXIUS_VHDL\SINTESIS \ARQUITECTURES_MODUL.

- Arxius font en llenguatge VHDL (**.vhd*).
  Mòdul OCR (per cada arquitectura) sintetitzat en tecnologia C35B4.

\ARXIUS\ARXIUS_VHDL\TESTBENCH.

- Arxius font en llenguatge VHDL (**.vhd*).
  Aplicacions per provar funcionament del Mòdul OCR (per cada arquitectura) i blocs funcionals.

\ARXIUS\COMANDES_MODELSIM.

- Arxius *.*txt*.
  Comandes per introduir a *Modelsim* per realitzar certes funcions.

\ARXIUS\INFORMES_SINTESIS.

- Arxius *.*txt*.
  Informes dels resultats dels processos de síntesis de cadascuna de les 5 arquitectures del mòdul OCR.

\ARXIUS\RETARDS.

- Arxius *.*sdf*.
  Arxius per simular considerant retards de portes.

Apèndix B. Exemple de Codis font.

Apèndix B.1. Exemple d'arxiu font VHDL : arxiu *binaritzador.vhd*.

A continuació es mostra, a mode d'exemple, l'arxiu font *binaritzador.vhd*, que implementa una màquina d'estats.

```
library IEEE;
        use IEEE.Std_logic_1164.all; -- llibreries utilitzades

entity binaritzador is  -- nom del bloc implementat

  port (clk              : in  std_logic;                -- ports I/O
        reset            : in  std_logic;
        get_dada         : in  std_logic;
        fila             : in  std_logic_vector(5 downto 0);
        columna          : in  std_logic_vector(4 downto 0);
        dada_ready       : out std_logic;
        dada_binaritzada : out std_logic;

        llindar          : in  std_logic_vector(7 downto 0);
        fila_cam         : out std_logic_vector(5 downto 0);
        columna_cam      : out std_logic_vector(4 downto 0);
        get_dada_cam     : out std_logic;
        dada_ready_cam   : in  std_logic;
        dada_cam         : in  std_logic_vector(7 downto 0));


end binaritzador;

architecture sim of binaritzador is

   -- estats de la ME que implementem
   type   Estat is (esperant,demanar_pixel,binaritzar,final);
   signal estat_actual : Estat;


begin

  fila_cam <= fila;             -- assignacio concurrent de senyals
  columna_cam <= columna;       -- assignacio concurrent de senyals
```

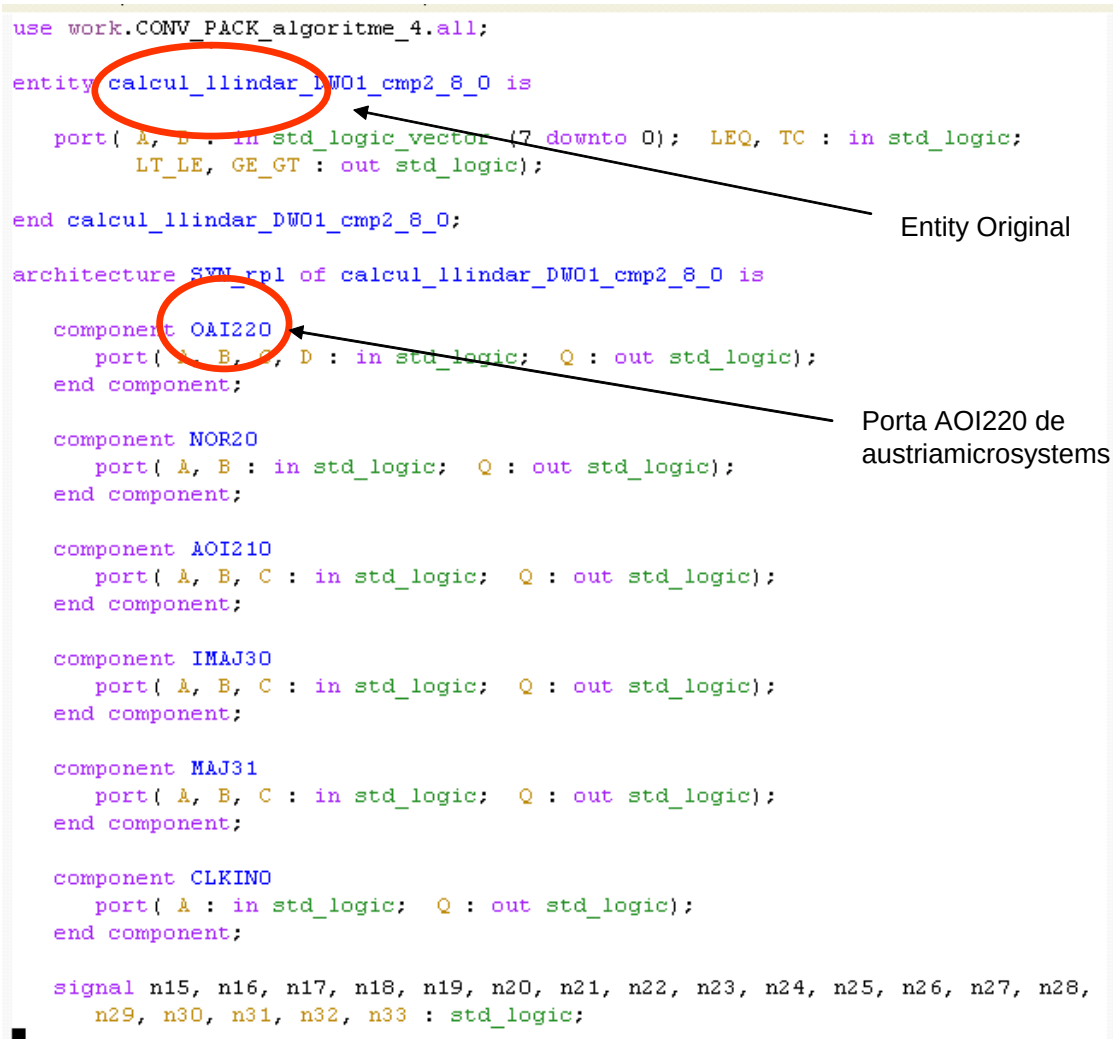
```
  logica_control : process (clk,reset)
  begin

        if (reset='0') then     -- quan RESET, posa totes les senyals a 0

                get_dada_cam <= '0';
                dada_ready <= '0';
                dada_binaritzada <= '0';

                estat_actual <= esperant; -- i l'estat es l'inicial

        -- nomes actualitza sortides i canvia d'estat
        -- quan es produeix un flanc de pujada al rellotge
        elsif clk'event and clk= '1' then

                case estat_actual is

                        when esperant =>
                                -- actualitza les sortides
                                get_dada_cam <= '0';
                                dada_ready <= '0';
                                dada_binaritzada <= '0';

                                -- actualitza els estats
                                if (get_dada = '1') then
                                        estat_actual <= demanar_pixel;
                                else
                                        estat_actual <= esperant;
                                end if;

                        when demanar_pixel =>
                                -- actualitza les sortides
                                get_dada_cam <= '1';

                                -- actualitza els estats
                                if (dada_ready_cam = '1') then
                                        estat_actual <= binaritzar;
                                else
                                        estat_actual <= demanar_pixel;
                                end if;

                        when binaritzar =>
                                -- actualitza les sortides
                                if ( dada_cam > llindar) then
                                        dada_binaritzada <= '1';
                                else
                                        dada_binaritzada <= '0';
                                end if;

                                -- actualitza els estats
                                estat_actual <= final;

                        when final =>
                                -- actualitza les sortides
                                get_dada_cam <= '0';
                                dada_ready <= '1';

                                -- actualitza els estats
                                if (get_dada = '0') then
                                        estat_actual <= esperant;
                                else
                                        estat_actual <= final;
                                end if;

                end case;
        end if;
    end process logica_control;

end sim;
```

A partir de la capçalera de l'arxiu, podem saber quin és el mapa de ports I/O de l'entitat. En el cas de *binaritzador*, el següent:

port	I/O	funció
clk	I	Senyal de rellotge
reset	I	Reset global
get_dada	I	Senyal per demanar que el mòdul converteixi el pixel de la [fila] i [columna] a bit
fila [6]	I	@ del pixel per binaritzar
columna [5]	I	@ del pixel per binaritzar
dada_ready	O	S'activa quan la dada binaritzada està a [dada_binaritzada]
dada_binaritzada	O	Dada resultant del procés
llindar [8]	I	Valor llindar per binaritzar. Ha d'estar el seu valor en aquesta entrada mentre es binaritza una dada.
fila_cam [6]	O	@ del pixel de la càmera
columna_cam [5]	O	@ del pixel de la càmera
get_dada_cam	O	Petició de pixel a la càmera. Les @ estan a [fila_cam] i [columna_cam]
dada_ready_cam	I	Activa quan la dada de la càmera està a [dada_cam]
dada_cam [8]	I	Dada d'entrada de la càmera

Apèndix B.2. Exemple d'arxiu font VHDL sintetitzat.

A la següent imatge podem veure com queda un codi sintetitzat. La jerarquia és la mateixa que al sistema original, amb totes les entitats amb els seus ports corresponents. La diferència és la arquitectura interna de cada entitat, que està formada exclusivament per portes lògiques de la tecnologia, en aquest cas la C35B4 d'Austriamicrosystems.



Apèndix C. Dades tecnològiques.

Apèndix C.1. Exemple de datasheet d'un component C35B4.

A la següent imatge podem veure, a mode d'exemple, el datasheet d'una de les portes sintetizade, concretament la AOI220.

austriamicrosystems

c35 Cells: AOI220

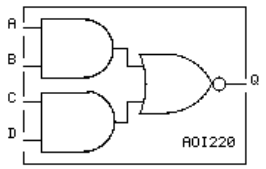
ASIC HOME | WWW HOME | SEARCH | SITE MAP | NEWS | HELP | FEEDBACK | REGISTER

AOI220 is an AND / NOR circuit with 0.5x drive strength providing the logical function Q = NOT (A.B+C.D).

Truth table

A	B	C	D	Q
0	X	0	X	1
0	X	X	0	1
X	0	0	X	1
X	0	X	0	1
X	X	1	1	0
1	1	X	X	0

Symbol



Capacitance

Pin	Cap [pF]
A	0.004
B	0.005
C	0.004
D	0.005

Area

0.141 mil²
91 µm²

Power

0.33 µW/MHz

AC Characteristics

Delay[ns] = f(SL, L), Output Slope [ns] = f(SL, L) with SL = Input Slope [ns] and L = Output Load [pF]

AC Characteristics: Tj = 25°C VDD=3.3V Typical Process

	Rise				Fall			
Slope [ns]	**0.05**		**2**		**0.05**		**2**	
Load [pF]	**0.0005**	**0.16**	**0.0005**	**0.16**	**0.0005**	**0.16**	**0.0005**	**0.16**
Delay A => Q	0.18	1.80	0.40	2.01	0.12	0.97	0.20	1.21
Delay B => Q	0.20	1.81	0.46	2.03	0.12	0.97	0.12	1.07
Delay C => Q	0.10	1.72	0.44	2.05	0.06	0.89	0.04	1.14
Delay D => Q	0.12	1.73	0.50	2.08	0.07	0.90	-0.02	0.99
Slew A => Q	0.29	4.33	0.61	4.34	0.23	2.21	0.70	2.31
Slew B => Q	0.34	4.37	0.61	4.38	0.23	2.21	0.63	2.29
Slew C => Q	0.27	4.34	0.52	4.33	0.12	2.09	0.49	2.21
Slew D => Q	0.32	4.39	0.55	4.38	0.12	2.09	0.47	2.17

Apèndix C.2. Dades de memòries RAM sintetitzables de C35B4.

austriamicrosystems

ASIC HOME | WWW HOME | SEARCH | SITE MAP | NEWS | HELP | FEEDBACK | REGISTER

Memory Compiler for Dual Port RAM in 0.35µm CMOS (C35)

Key Features

- Memory Compiler for Dual Port RAM in 0.35µm CMOS Process (C35/S35)
- triple metal layout data of memory
- full read/write capability on each port
- 65536 bit maximum memory size
- 8 .. 32 bits per word
- 128 .. 8192 words per DPRAM
- Simulation models for 3.3V nominal supply
- 2 separated or 1 common datain/dataout buses per port
- tri-state dataout bus

Deliverables

- Frontend services (available via Internet)
  - CADENCE
    - cell library with symbol, functional, abstract and msps view
    - TLF 3.0 & TLF 4.3 timing data file
    - LEF file for silicon ensemble
    - SDF annotable Verilog model
  - VHDL
    - VITAL95 compliant simulation model
    - TLF timing data file for SDF generation
    - LEF file for silicon ensemble
  - SYNOPSYS
    - cell timing model (interface model)
  - MENTOR
    - Design Architect symbol
    - QSIM II simulation model & black box desription
    - black box desription
- Backend Services (on order)
  - CADENCE
    - cell library with additional layout view with reduced layout data (*)
  - gds2 data
    - reduced layout data in gds2 format (*)

(*) reduced layout data does not contain the following layers:

- Diffusion
- Poly1
- N+ Implant
- P+ Implant
- Contact

The reduced layout data will be replaced with full layout data by *austriamicrosystems* before production.

Area

Area [mm2]							
Process	**1k bit**	**2k bit**	**4k bit**	**8k bit**	**16k bit**	**32k bit**	**64k bit**
C35	0.34	0.48	0.75	1.26	2.23	3.78	7.15

NOTE:
For many configurations different height/width rations are available. The aspect ratio influences area and timing data. The specified data are for the configuration, which gives the maximum density of the memory.

Timing

Access Time [ns]							
Process	**1k bit**	**2k bit**	**4k bit**	**8k bit**	**16k bit**	**32k bit**	**64k bit**
C35	2.93	3.07	3.33	3.42	3.54	4.41	4.69

Power consumption

Supply Current [mA/MHz]							
Process	**1k bit**	**2k bit**	**4k bit**	**8k bit**	**16k bit**	**32k bit**	**64k bit**
C35	0.175	0.196	0.239	0.255	0.279	0.362	0.402

Power and Timing data conditions:

- typical process parameters
- VDD = 3.3V
- Tj = 27°C
- Cload = 1pF

Description

The DPRAM memory compiler system enables automatic generation of dual port RAM blocks for the configurations shown above. The compiler system ist not included in a HIT-Kit shipment. Memory Simulation models are distributed via Internet (see beyond), layout data are provided on request.

Documentation

All data shown in this document are related to Revision "D" of the dual port RAM compiler for 0.35um process..

- Timing Diagram - Rev A - 01/00

Simulation Model Generation

Registered Customers can generate their required simulation model(s) directly via Internet by clicking on the model generation link below.

- Simulation Model generation - Compiler Rev. NC

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

L'autor del document, **Daniel Castillo Hand**

Signat: .....

Bellaterra, 29 de gener de 2007.