UNIVERSITAT POLITÈCNICA DE CATALUNYA

Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial

Mètode d'Extracció Multiparamètrica de Característiques de Textura Orientat a la Segmentació d'lmatges

Autor: Antoni Grau i Saldes
Directora: Alícia Casals i Gelpí

Barcelona, maig de 1997

CAPÍTOL 6. ARQUITECTURA ESPECÍFICA.

La tècnica de segmentació presentada s'ha elaborat amb els criteris de poder operar a velocitat de vídeo. Per aquest motiu, s'ha dissenyat l'algorisme d'extracció de paràmetres de textura de forma que pugui ser implementat sobre un *hardware* específic de complexitat raonable. En aquest capítol es presentarà l'arquitectura específica dissenyada per al càlcul dels paràmetres de textura. Es presenta també la implementació de l'algorisme de la classificació de les característiques de textura per tal d'obtenir els grups de textura.

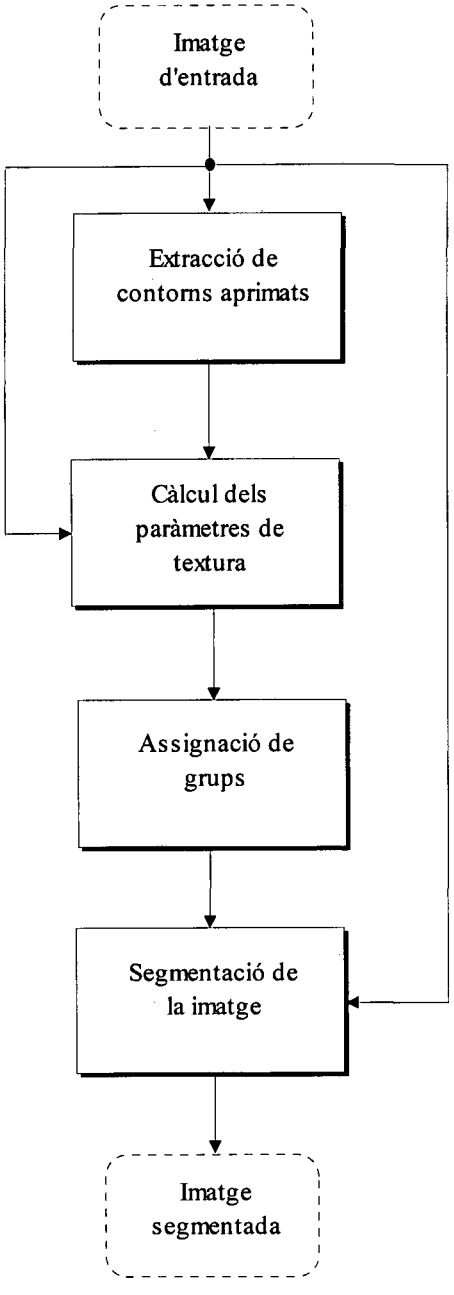

Figura 6.1. Diagrama del procés de segmentació d'imatges per textures.

En el diagrama de la figura 6.1 es pot observar els diferents mòduls que composen l'arquitectura dissenyada específicament per aquest problema. El mòdul d'extracció de contorns aprimats no s'inclou en el disseny ja que no és l'objectiu d'aquest treball i, a més, existeixen diversos dispositius comercials que efectuen aquesta operació.

El fet d'implementar amb una arquitectura específica l'algorisme d'extracció de característiques és clar: no hi ha temps suficient amb un algorisme programat (*software*) per tractar les dades a *video rate*. Amb l'algorisme d'assignació de grups de textura serà diferent: és una etapa de més alt nivell en la qual no hi ha tantes dades a tractar per unitat de temps i per aquest motiu podem aprofitar la potència dels processadors actuals comercials i desenvolupar un algorisme optimitzat per realitzar aquesta tasca d'assignació. Tot i que les dades arriben més espaiades en aquesta etapa, el temps continua sent un factor crític i en l'apartat 6.2 es detallen aquests conceptes i s'estudia la viabilitat d'un algorisme per mantenir el flux de dades a *video rate*. L'algorisme s'ha desenvolupat pensant en un entorn PC, però és fàcilment exportable a qualsevol altre sistema i el mateix succeeix amb l'arquitectura: és independent del bus al qual es connecti la placa dissenyada.

Després d'obtenir els paràmetres de textura i generar els tèxels, cal assignar a quin grup pertany cada punt de l'espai de característiques. Aquesta tasca la du a terme el mòdul d'assignació de grups a partir de l'aprenentatge que s'ha realitzat amb anterioritat. Recordem que hi ha dues fases d'operació: aprenentatge i treball. Aquesta arquitectura s'ha dissenyat específicament per a l'operació en la fase de treball (fase *on-line*) ja que la fase d'aprenentatge no requereix en cap moment operar en temps real; és una fase que es realitza *off-line*.

Cal afegir un darrer mòdul per a la generació dels resultats. Hi ha diverses alternatives per presentar la imatge segmentada: i) imatge en nivells de gris on cada nivell és una classe; ii) número de la classe com a corrent de bits sincronitzat amb la posició del tèxel; iii) imatge original superposant fronteres entre les regions de textura.

A continuació es desenvolupa amb més detall cadascun dels mòduls que formen el procés d'extracció de característiques de textura i la segmentació de la imatge, així com la interfície que existeix entre ells. Aquesta implementació ha estat presentada en [GRA97d].

6.1. Mòdul de càlcul dels paràmetres de textura.

En aquest apartat tractarem el problema del càlcul dels paràmetres de textura en temps real. Partint dels contorns aprimats generarem els tèxels corresponents a cada característica de textura per poder ser llegits pel mòdul d'assignació de grups. En aquest apartat veurem com es du a terme la tasca encomanada, així com també analitzarem el temps emprat per al càlcul dels paràmetres.

El mètode per calcular els paràmetres serà el mateix per quatre d'ells els quals utilitzen com a dades d'entrada els contorns aprimats. Recordant-los, són la *linealitat*, la

granulositat, la *discontinuïtat* i l'*abruptitat*. El tractament del paràmetre *difuminat* serà especial donat que necessita com a entrada la imatge en nivells de gris.
El procediment per calcular els paràmetres de textura el podem dividir en dos mòduls clarament separats:

- Detecció dels píxels que pertanyen o caracteritzen els paràmetres de textura.
- Comptatge d'aquests píxels dins un tèxel, per calcular cada paràmetre.

En la figura 6.2 podem veure els dos mòduls esquemàticament.

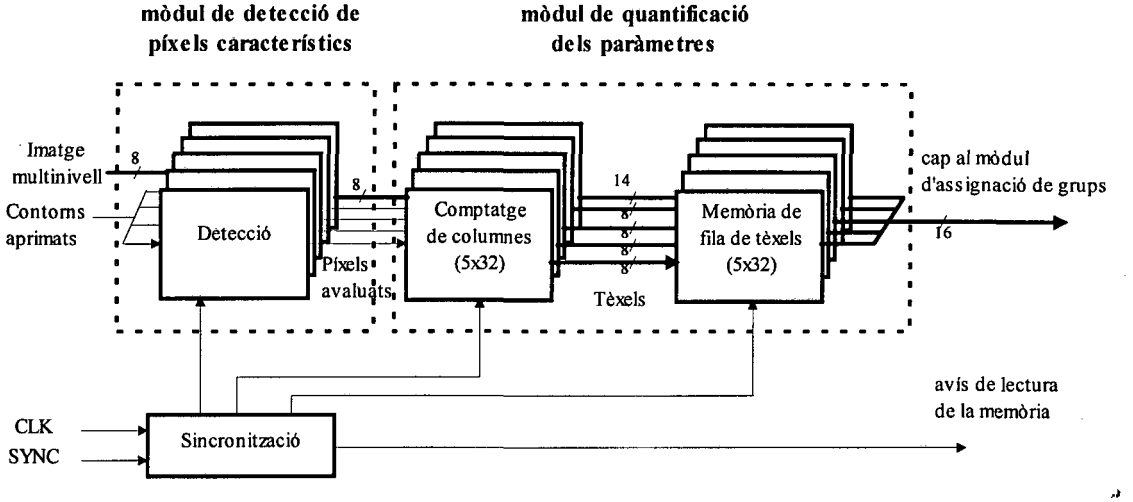

Figura 6.2. Mòdul de càlcul dels paràmetres de textura.

En el **mòdul de detecció de píxels característics**, cada submòdul de detecció genera com a sortida un senyal serialitzat síncron que corresponen al valor del píxel avaluat per a cada paràmetre. Aquest píxels avaluats van al **mòdul de quantificació dels paràmetres** on cada submòdul de comptatge recull aquest corrent de bits i genera simultàniament una fila de tèxels (32 tèxels) que es guarden síncronament en una memòria de sortida per ser llegits pel mòdul d'assignació de grups.

El submòdul de sincronització generarà els senyals necessaris per saber quin píxel s'està analitzant en cada instant de temps. A més de disposar del rellotge de píxel i sincronismes de línia i quadre, es generaran 16 bits de control, 8 per la selecció de les files (adreces *f0, ..., f7)* i 8 per les columnes (*c0, ..., c7)*.

A continuació detallarem cadascun dels submòduls anteriors i explicarem el seu funcionament a partir del disseny.

6.1.1. Mòdul de detecció de píxels característics.

Cada submòdul determinarà quins píxels pertanyen o caracteritzen el paràmetre de textura corresponent. El nombre de píxels a la sortida de cada submòdul serà el mateix nombre de píxels que hi entren però amb valor zero (0) si no pertanyen o caracteritzen la característica en curs i amb valor ú (1) si hi pertanyen. El píxel de la imatge

d'entrada que s'està avaluant en cada instant de temps l'anomenarem element actiu **EA** i només hi haurà un únic element actiu per unitat de temps i per cada paràmetre. Els píxels surten amb una cadència marcada per la freqüència de lectura de cada nou píxel de la imatge (*video rate*).

Cada característica es calcula per separat però seguint en tots els casos la mateixa tècnica: comparació de les màscares sobre els píxels d'entrada. El cas del paràmetre del difuminat és especial i requerirà un tractament diferent.

6.1.1.1. Càlcul del paràmetre linealitat.

Per saber si un píxel de la imatge d'entrada caracteritza el paràmetre *linealitat,* cal comparar les diferents màscares de *linealitat* amb la regió que conté aquest píxel. Al ser un procés síncron només hi haurà un únic element actiu en cada instant per la qual cosa les màscares es compararan sobre aquest píxel sigui quina sigui la seva posició dins la màscara. Per tant, la màscara s'haurà d'adaptar a la posició de l'element actiu i no a l'inrevés. Això vol dir que no sempre el píxel amb el qual es comparen les màscares serà el de la cantonada superior esquerra sinó que, segons la màscara, aquest píxel a avaluar pot ser l'element de la cantonada inferior dreta com a pitjor cas. És a partir d'aquest raonament quan neix la necessitat de guardar tant píxels, o línies de píxels, com requereixin les màscares.

Per aquest motiu recordem les màscares definides en l'apartat 3.3.1.2. En les màscares es veu que apareix un element marcat en color gris. Aquesta posició indica l'element actiu sobre el qual qüestionem si caracteritza o no la característica de la *linealitat.*

En la figura 6.3 etiquetem els elements de les màscares per poder fer referència a ells més endavant.

x_{11}	x_{12}	x_{13}	x_{14}
x_{21}	x_{22}	x_{23}	x_{24}
x_{31}	x_{32}	x_{33}	x_{34}
x_{41}	x_{42}	x_{43}	x_{44}

Figura 6.3. Etiquetatge dels elements dins una màscara.

En la taula 6.1 podem veure precisament la variació en l'element actiu. Segons la seva posició, els requeriments de memòria per emmagatzemar els valors dels píxels poden variar considerablement.

Posició de l'element actiu **EA**	Nom de la màscara
x_{11}	L11, L21, L31, L12, L13, L14, L15, L16, L17, L22, L23, L24, L25, L26, L27,
x_{14}	L41, L32, L33, L34, L35, L36, L37, L42, L43, L44, L45, L46, L47

Taula 6.1. Localització de l'element actiu dins les màscares de linealitat.

Com tots els elements actius estan en la primera fila de la màscara, el nombre de píxels a guardar correspon a tres línies de la imatge completes tal com veiem en la figura 6.4.

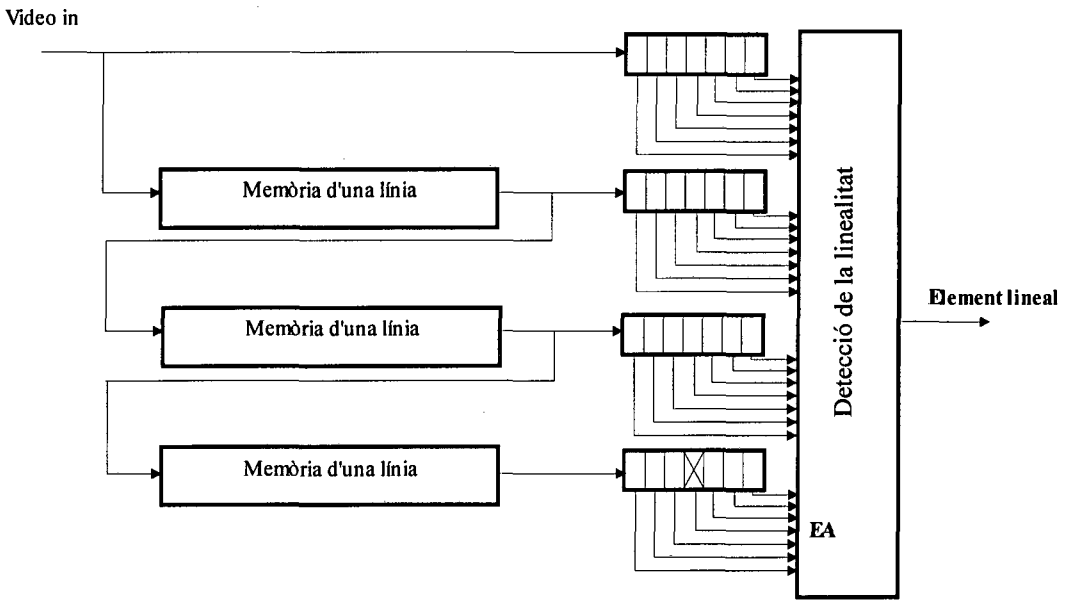

Figura 6.4. Conversió sèrie/paral·lel per a la detecció de la linealitat.

L'element actiu és la casella marcada com **EA** en la figura 6.4 (posició marcada amb una creu). Si prenem com a referència les màscares que tenen com a element actiu x11, el fet d'analitzar aquest element serà situar la màscara sobre la posició que indica la figura, mentre que si es comparen les màscares que tenen com a element actiu x14, les màscares se situen desplaçades 3 posicions o píxels més endarrerits. Per aquest motiu, per fer la conversió sèrie/paral·lel es necessiten registres de 7 posicions.

Cada casella representa un píxel i s'implementen amb registres de desplaçament, així cada vegada que arriba un nou element tots els valors es desplacen una posició. No s'indica el senyal de rellotge ni els senyals de control dels registres per simplicitat en el dibuix.

Després de preparar en forma paral·lela els valors dels píxels, cal realitzar el càlcul de la *linealitat* el qual s'implementa amb les operacions lògiques que cada màscara necessita. Així doncs, les sortides dels registres aniran a unes portes lògiques AND per forçar que a la sortida es compleixin tots els requisits de la màscara. Després del càlcul de cada màscara cal sumar lògicament les sortides per veure si existeix alguna màscara que hagi avaluat positivament els elements que li corresponien. D'aquesta manera es genera l'**Element lineal**. Això es pot veure en la figura 6.5.

Per senzillesa en el dibuix no s'han representat totes les màscares de linealitat només 12 de 28, però s'implementen de manera semblant. Damunt de cada porta AND hi ha indicat el nom de la màscara de *linealitat* que s'implementa.

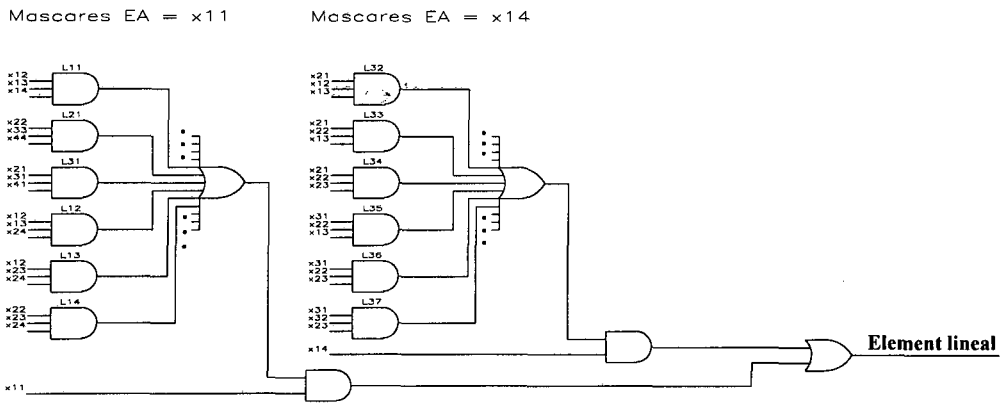

Figura 6.5. Lògica de les màscares de linealitat.

La sortida **Element lineal** formarà una de les cinc línies de sortida síncrones que correspondran a l'entrada del següent submòdul de comptatge.

6.1.1.2. Càlcul del paràmetre granulositat.

El procediment de detecció de la granulositat és semblant al pas de les màscares de linealitat. L'únic que varia és la disposició de les màscares, el nombre de línies a guardar i les entrades de les portes que calcularan l'element granulós.

Posició de l'element actiu **EA**	Nom de la màscara
x_{22}	G1, G2_1, G3_1, G4_1
x_{23}	G2_2, G3_2
x_{32}	G2_3, G4_2
x_{33}	G2_4

Taula 6.2. Localització de l'element actiu dins les màscares de granulositat.

Després de l'etiquetatge dels elements de les màscares segons la figura 6.3, construïm la següent taula per veure quin és l'element actiu dins de cada màscara per determinar la quantitat de memòria que es necessitarà per realitzar satisfactòriament la comparació amb les màscares de granulositat. Recordem, apartat 3.2.2, que per al càlcul d'aquest paràmetre necessitem comparar 4 màscares (3 d'elles amb relaxació) però, com només hi pot haver un únic element actiu, les desglossem en nou màscares.

En la figura 6.6 es pot veure l'esquema del procés de conversió sèrie/paral·lel dels píxels per a la detecció de la granulositat.

Aquí l'element actiu està a una distància d'una fila d'una màscara a una altra però començant per la segona fila, per tant els requeriments de memòria seran superiors que no pas si haguessin estat tots sobre la mateixa fila.

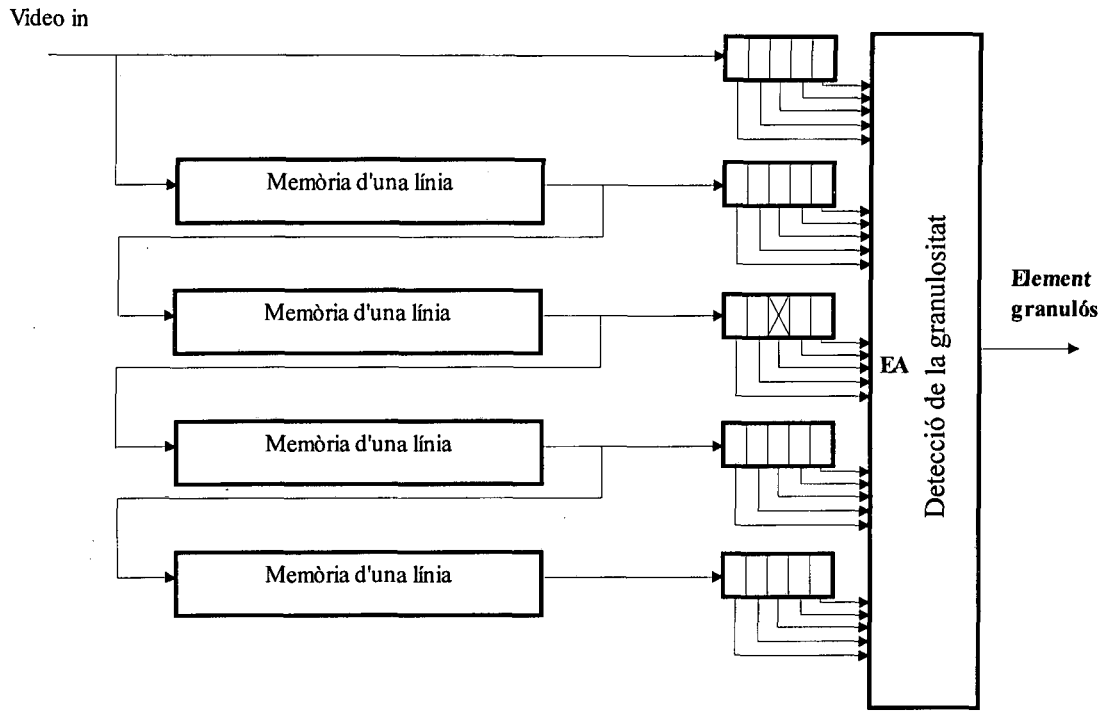

Figura 6.6. Conversió sèrie/paral·lel per a la detecció de la granulositat.

L'element actiu **EA** és la casella marcada amb una creu. El registre de desplaçament serà de 5 posicions per poder tenir preparats a l'entrada del mòdul de càlcul de la granulositat tots els píxels necessaris per avaluar totes les màscares. També serà necessari guardar quatre línies de vídeo.

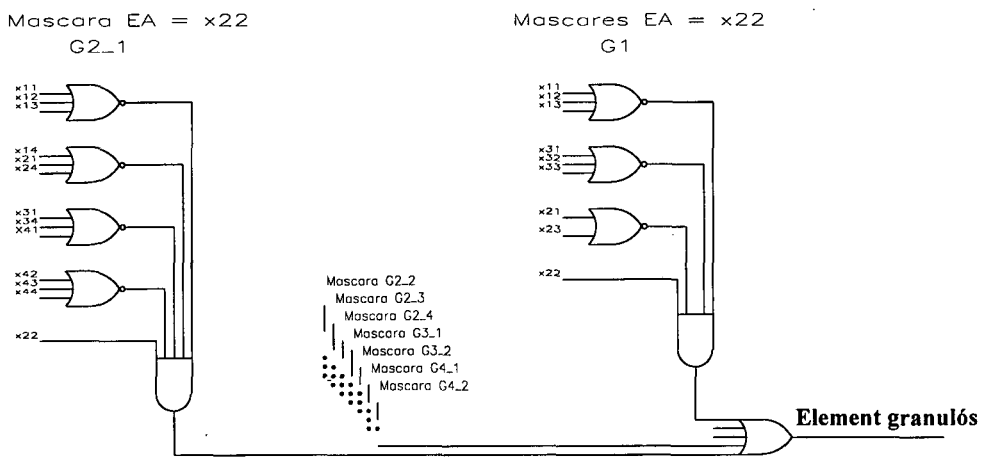

Figura 6.7. Lògica de les màscares de granulositat.

En la figura 6.7 s'implementa l'algorisme per a la detecció de la granulositat però no es representen totes les màscares per donar més senzillesa al dibuix. Cada màscara processarà els píxels que necessiti segons la posició de l'element actiu. Per crear l'**Element granulós** cal recollir el resultat de totes les màscares i, només que una d'elles hagi avaluat positivament a l'element actiu, la sortida serà positiva.

6.1.1.3. Càlcul del paràmetre discontinuïtat.

Seguint la línia dels anterior paràmetres, construirem la taula 6.3 per situar l'element actiu dins les màscares i fer així el disseny de l'arquitectura.

Posició de l'element actiu **EA**	Nom de la màscara
x_{22}	F17, F18, F21, F23
x_{23}	F11, F12, F22, F27
x_{32}	F15, F16, F24, F25
x_{33}	F13, F14, F26, F28

Taula 6.3. Localització de l'element actiu dins les màscares de discontinuïtat.

Donat que la posició de l'element actiu en les màscares és la mateixa que apareixia en el paràmetre de la granulositat, els requeriments de memòria i la conversió sèrie/paral·lel dels elements d'entrada és idèntica al paràmetre anterior, per la qual cosa no repetirem l'esquema. El que realment els fa diferents és la lògica per a la detecció de l'element discontinu, donat que les màscares dels paràmetres són diferents.

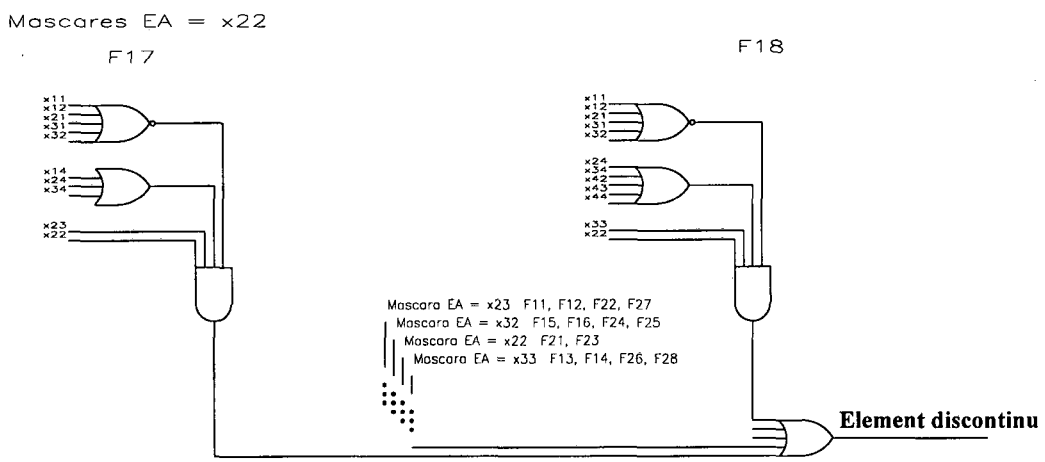

Figura 6.8. Lògica de les màscares de discontinuïtat.

Per la disposició de l'element actiu dins les màscares usarem l'esquema presentat en la figura 6.6, guardant quatre línies completes de la imatge juntament amb registres de desplaçament de 5 posicions. Els píxels paral·lelitzats serviran d'entrada a l'algorisme que detecta la discontinuïtat.

Per calcular l'**Element discontinu**, usarem el circuit de la figura 6.8. Cada màscara obtindrà la seva entrada correcta en funció de la posició de l'element actiu dins la màscara. Només s'han representat dues màscares (F17 i F18). La resta de les màscares es realitzen de manera semblant per la qual cosa no es presenten donant més senzillesa a l'esquema.

6.1.1.4. Càlcul del paràmetre abruptitat.

En la taula 6.4 es mostra on se situa l'element actiu dins les màscares i permetre així fer el disseny correctament de l'arquitectura.

Posició de l'element actiu **EA**	Nom de la màscara
x_{22}	A11, A24, A31, A32, A41, A42, A43, A44, A45, A46, A47, A48
x_{23}	A22, A33, A34
x_{32}	A21, A35, A36
x_{33}	A23, A37, A38

Taula 6.4. Localització de l'element actiu dins les màscares d'abruptitat.

Els requeriments de memòria i l'arquitectura pel càlcul de l'abruptitat és idèntica al que hem vist de la granulositat donat que la distància entre elements actius és la mateixa. El que canvia és la manera de calcular l'element abrupte. En la figura 6.9 podem veure l'esquema per calcular-lo. Els noms que apareixen sota les portes AND són els noms de les màscares que s'implementen amb cada porta.

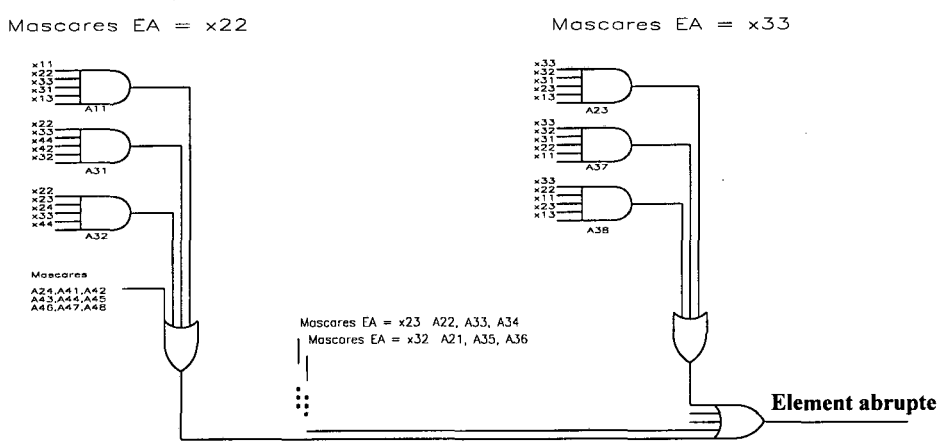

Figura 6.9. Lògica de les màscares d'abruptitat.

6.1.1.5. Esquema de la detecció dels píxels característics.

Tal com acabem de veure en els subapartats anteriors, cada submòdul de detecció necessita guardar unes certes línies de la imatge per poder detectar els paràmetres de textura. Per senzillesa i comprensió dels esquemes, s'ha dibuixat la detecció de cada paràmetre per separat, però en realitat tots els submòduls de detecció anteriors comparteixen les mateixes memòries de línies de vídeo binari. El requisit imprescindible per assegurar la sincronització de tot el procés és que el píxel a detectar (**element actiu**) sigui el mateix per tots els submòduls, per tant, el retard que existeix des de que un píxel entra en els submòduls fins que es detecta ha de ser el

mateix. Analitzem el retard que existeix en cada submòdul mirant el retard amb el qual l'element actiu és detectat.

Retard en l'element actiu de *linealitat*: 3 línies + 4 CLK
Retard en l'element actiu de *granulositat*: 2 línies + 3 CLK
Retard en l'element actiu de *discontinuïtat*: 2 línies + 3 CLK
Retard en l'element actiu d'*abruptitat*: 2 línies + 3 CLK

Es pot apreciar que no tots els retards són iguals, per la qual cosa el retard total ve marcat pel submòdul que té un major retard (linealitat). Per tant, els submòduls de la granulositat, discontinuïtat i abruptitat tindran un retard addicional d'una línia de vídeo. Pel que fa als retards en els períodes de píxel, tant el submòdul de la discontinuïtat com en el submòdul de l'abruptitat i el de granulositat s'hauran de retardar un període de píxel.

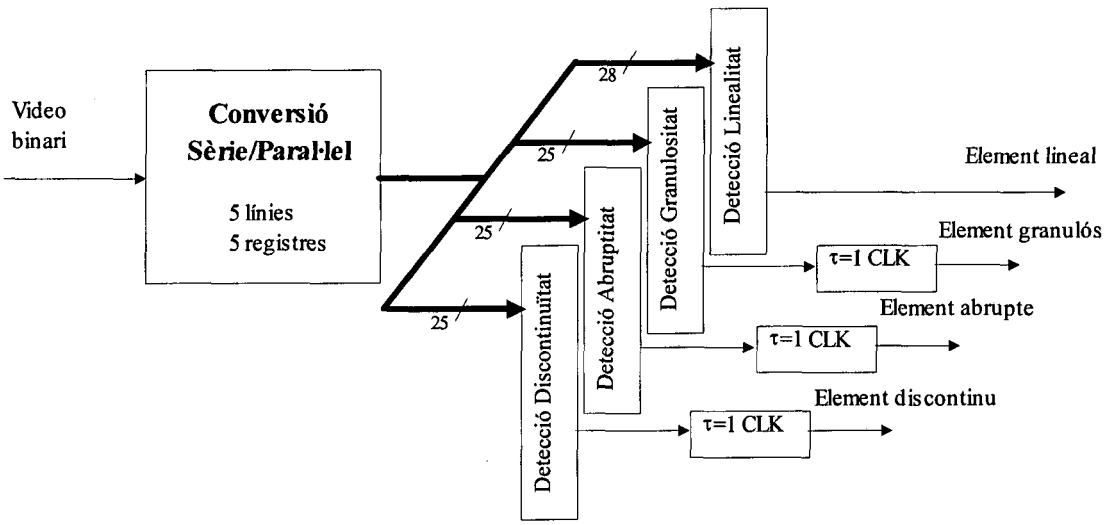

Figura 6.10. Conversió sèrie/paral·lel única per a la detecció dels paràmetres de textura.

Després d'aquesta avaluació dels retards, el submòdul de conversió sèrie/paral·lel contindrà una memòria corresponent a 5 línies de vídeo (2 prèvies per la linealitat, l'actual i 2 posteriors per a la resta dels paràmetres) més 5 registres de desplaçament de 7 posicions (el màxim de tots els paràmetres). En la figura 6.10 es pot veure la configuració de la memòria de vídeo necessària per a la conversió sèrie/paral·lel del senyal de vídeo binari de l'entrada.

6.1.1.6. Càlcul del paràmetre difuminat.

Per al càlcul del paràmetre del difuminat s'ha de canviar la tècnica ja que no podem usar un sistema de portes, com amb la resta de paràmetres, per calcular l'**Element difós** i tampoc no hi ha cap màscara a comparar. En aquest cas, recordant l'apartat 3.3.5, s'ha de trobar la màxima diferència entre un píxel **X** i els seus veïns (*a, b, c, d, e, f, g,* i *h*) sempre que aquesta diferència estigui dins un interval de valors [*I*1, *I*2]. L'**Element difós** *D* s'ha definit a (3.17) com

$$D = màx\{|X-a|, |X-b|, ..., |X-h| \in [I1, I2]\}$$

Per disposar de tots els veïns del píxel **X** al mateix temps, s'han de guardar dues línies de vídeo multinivell completes.

Es plantegen dos dissenys: iteratiu i paral·lel. En el disseny iteratiu per a cada període de píxel cal realitzar 8 comparacions amb el veïnatge i escollir la màxima diferència, treballant a 9 vegades la freqüència de píxel (45MHz). En el disseny paral·lel no cal incrementar la freqüència de treball donat que les comparacions es fan en un sol període amb el conseqüent increment de components. Existeix un compromís entre la quantitat de components a utilitzar pel disseny i la freqüència a la qual han de treballar aquests components.

6.1.1.6.1. Disseny iteratiu.

En el procés iteratiu, es trobarà la màxima diferència entre el píxel **X** i els seus vuit veïns. Per realitzar aquest càlcul usarem un comparador de 8 bits més un multiplexor. En un pas posterior es comprova que aquesta màxima diferència estigui entre els dos límits de l'interval [*I1*, *I2*] i es dóna aquesta sortida com a **Element difós**. Si la diferència no està dins l'interval, la sortida serà zero.

En la figura 6.12 es mostra l'arquitectura per al càlcul de l'element difós a partir dels veïns de l'element actiu **X**. Recordem que cada píxel es representa amb 8 bits i cada bus té el mateix ample.

El multiplexor i el comparador sempre mantenen el màxim valor de la resta entre el píxel **X** i els seus veïns. Aquesta resta serà en valor absolut. El mòdul ML serà el que comprovarà que el màxim estigui entre els dos límits de l'interval, donant un '1' en cas afirmatiu i un '0' en cas negatiu.

Per deixar passar cada veí cal un seqüenciament en subfases i es treballa amb un rellotge a una freqüència 9 vegades la freqüència d'un píxel, figura 6.11.

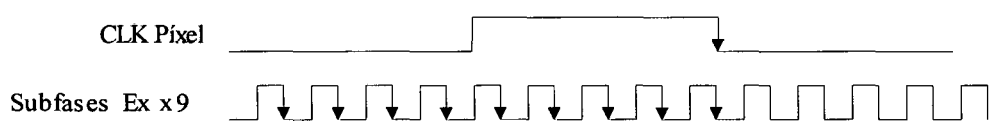

Figura 6.11. Rellotge per al seqüenciament dels veïns del píxel **X**.

Es necessita una subfase per deixar passar cada veí (Ea, Eb, Ec, ..., Eh) cap al restador més una novena subfase (El) que serà per: i) guardar el valor del màxim de la diferència durant el període d'un píxel sobre el registre de sortida; ii) posar a zero el llaç que manté el valor màxim cada vegada que es canvia d'element actiu. Aquesta subfase anirà endarrerida τ respecte la fase del registre per tal que no es guardi el valor del zero inicial. A la freqüència de treball, cada subfase ocupa 22nseg. Amb aquest

temps tant reduït, la tecnologia dels components ha de ser capaç de treballar a alta velocitat per a que es puguin efectuar totes les operacions necessàries en cada període.

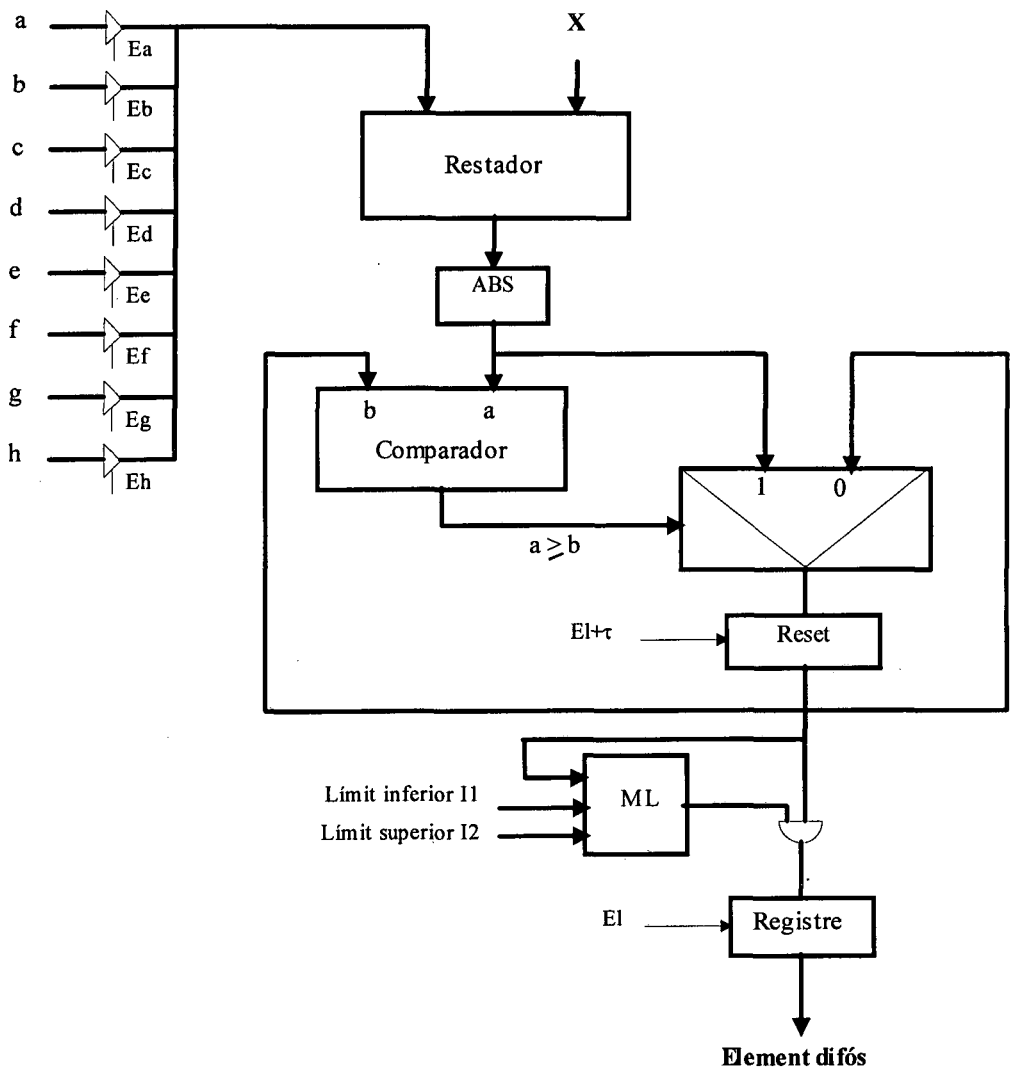

Figura 6.12. Càlcul de l'element difós.

A causa de l'alta freqüència a la es necessita treballar, a continuació es presenta el disseny paral·lel del mateix algorisme, en el qual no existeix una restricció tant forta en el temps d'operació al treballar a una freqüència molt més baixa.

6.1.1.6.2. Disseny paral·lel.

En aquest disseny, en lloc de comparar en un sol període el píxel central amb tot el veïnatge, se segueix la següent tècnica. Només es resten del píxel central el valor màxim i el valor mínim del veïnatge i s'escull la màxima diferència. Per tant, sempre s'ha de guardar el valor màxim i el valor mínim del veïnatge i, per als tres nous píxels que entren, es calculen i s'actualitzen el valor màxim i mínim. En la figura 6.13 es mostra l'esquema global del disseny paral·lel. En totes les figures, els busos són de vuit bits per la qual cosa no s'indicarà l'ample de cadascun sobre la figura.

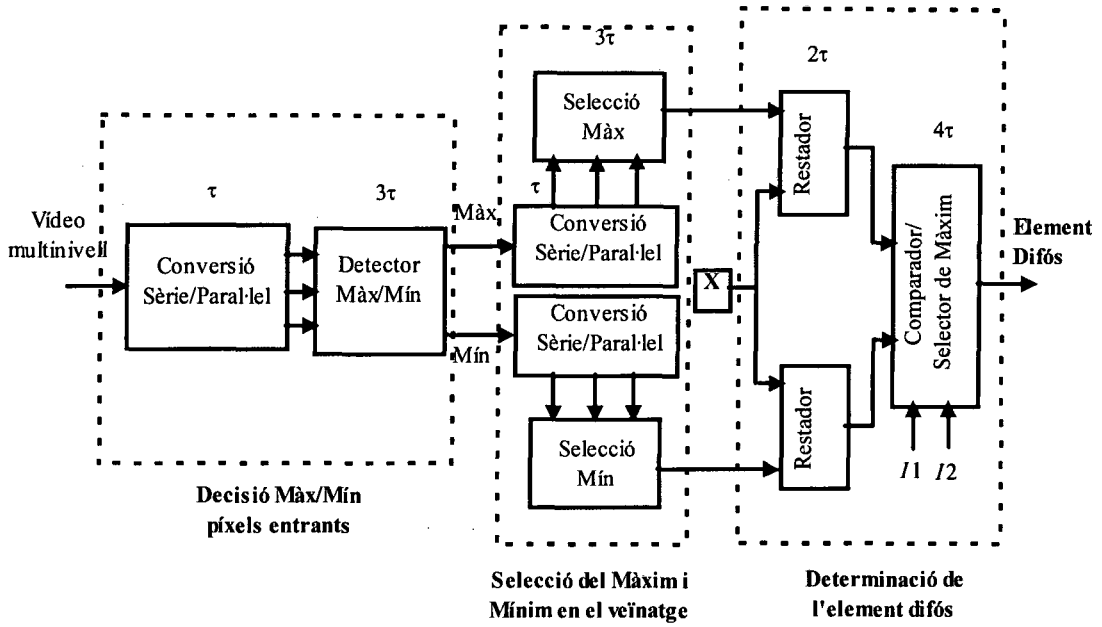

Figura 6.13. Esquema del càlcul de l'element difós de forma paral·lela.

En cada bloc hi ha el retard estimat que s'introdueix i cal tenir-lo en compte per saber si la sortida de l'element difós estarà disponible dins el període de píxel o caldrà esperar al següent període. Al final d'aquest apartat es calcularà quin és el retard total afegit.

En la figura 6.14 es mostra el veïnatge del píxel **X** sobre el qual es busca la màxima diferència. Els nous píxels que entren a cada rellotge de píxel són *a*, *d* i *f*, tots els de vuit bits.

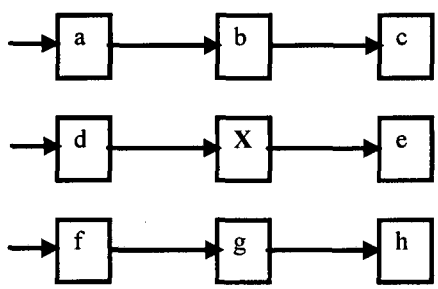

Figura 6.14. Veïnatge del píxel **X**.

El primer pas serà decidir quin dels tres nous píxels tindrà el valor màxim i el valor mínim (figura 6.15). La següent lògica permet generar uns senyals (**Màx** i **Mín**) que seleccionaran aquests valors per guardar-los com a màxim i mínim.

Després de seleccionar el màxim i el mínim dels nous píxels, cal guardar-los juntament amb els dos màxims i els dos mínims que ja teníem dels píxels anteriors i s'han de recalcular els nous valors màxim i mínim per restar-los del píxel central, figura 6.16. L'estat actual l'hem anomenat *i*.

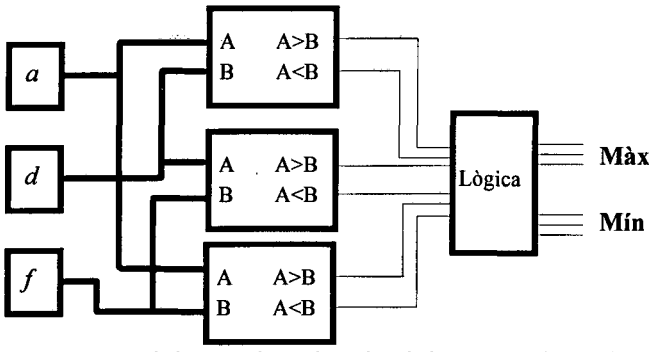

Figura 6.15. Decisió per la selecció del màxim i mínim per als píxels entrants de 3 files.

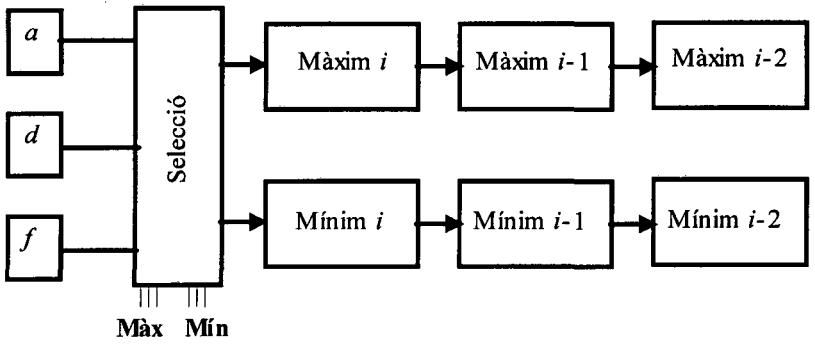

Figura 6.16. Selecció del màxim i mínim per als píxels entrants.

Després d'haver guardat els valors màxims i mínims per al veïnatge, cal decidir quin és el valor màxim dels màxims i quin és el valor mínim dels mínims. Per realitzar aquesta tasca caldria repetir un circuit igual al de la figura 6.15, on tres comparadors més una lògica generen els senyals necessaris per prendre la decisió. Quan ja s'han seleccionat aquests dos nous valors, cal restar-los del píxel central i escollir el valor màxim, el qual serà el valor del píxel difuminat sempre que estigui dins l'interval de difuminat, en cas contrari serà zero. Aquest procés el poden veure en la figura 6.17.

Els senyals **MMàx** i **MMín** indiquen quin és el màxim dels valors màxims i el mínim dels valors mínims respectivament. Els senyals $I1$ i $I2$ són els límits de l'interval de difuminat. Si el valor màxim de la resta està fora dels límits, la sortida és zero.

L'avantatge d'aquest disseny és que treballa a freqüència de píxel (5MHz), valor pel qual els dispositius FPGA funcionen correctament.

Pel que fa als retards introduïts per la pròpia lògica, en total, aquests retards sumen 14τ, on aproximadament un retard unitari són 8 nseg. Així doncs, el retard introduït per tot el procés és 112 nseg. Al treballar a una freqüència de 5MHz, el període d'un píxel són 200 nseg, per la qual cosa la sortida estarà dins el mateix període de píxel.

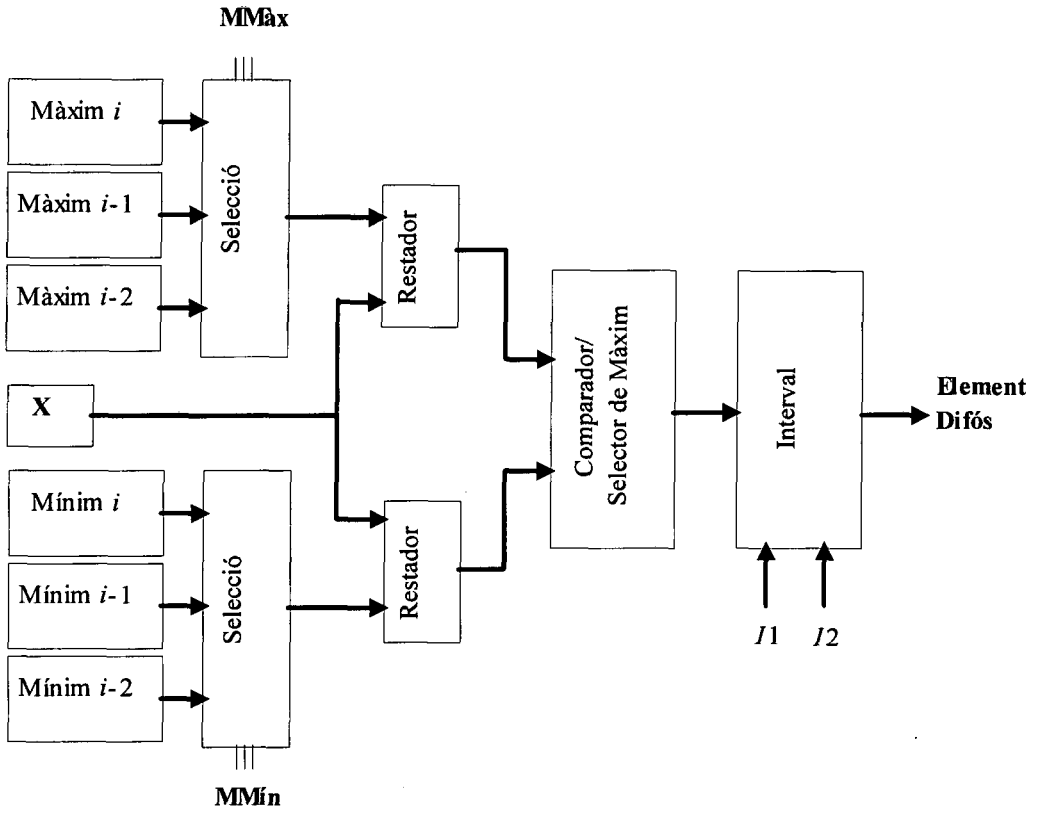

Figura 6.17. Determinació de l'**Element difós**.

Un altre aspecte important és la sincronització a la sortida de l'element difós amb la resta d'elements dels altres paràmetres. Cal ajustar els retards de manera que siguin iguals per a tots els paràmetres. Si recordem el màxim retard amb els quatre paràmetres anteriors aquest valor era de 3 línies de vídeo més 4 períodes de píxel. Com que aquí ja tenim 2 línies de vídeo de retard més 2 períodes de píxel, en la conversió sèrie/paral·lel, caldria afegir el retard d'una línia més de vídeo més dos períodes de píxel.

6.1.2. Mòdul de quantificació dels paràmetres.

Aquest mòdul calcularà els paràmetres de cada tèxel a partir dels píxels avaluats en el mòdul de detecció. L'entrada de cada submòdul són les respectives línies provinents del càlcul de cada paràmetre juntament amb la posició que ocupen aquells píxels en la imatge. A partir d'aquesta coordenada, els submòduls de comptatge sabran de quin tèxel es tracta i on afegir els valors dels elements de textura calculats (**Element lineal**, **Element granulós**, ...) que arriben. Precisament per aquesta serialització de píxels síncrona, es generen tots els tèxels d'una fila (32 en total) i se serveixen simultàniament als registres de sortida. Per realitzar aquesta operació ens fixarem en l'esquema de la figura 6.18.

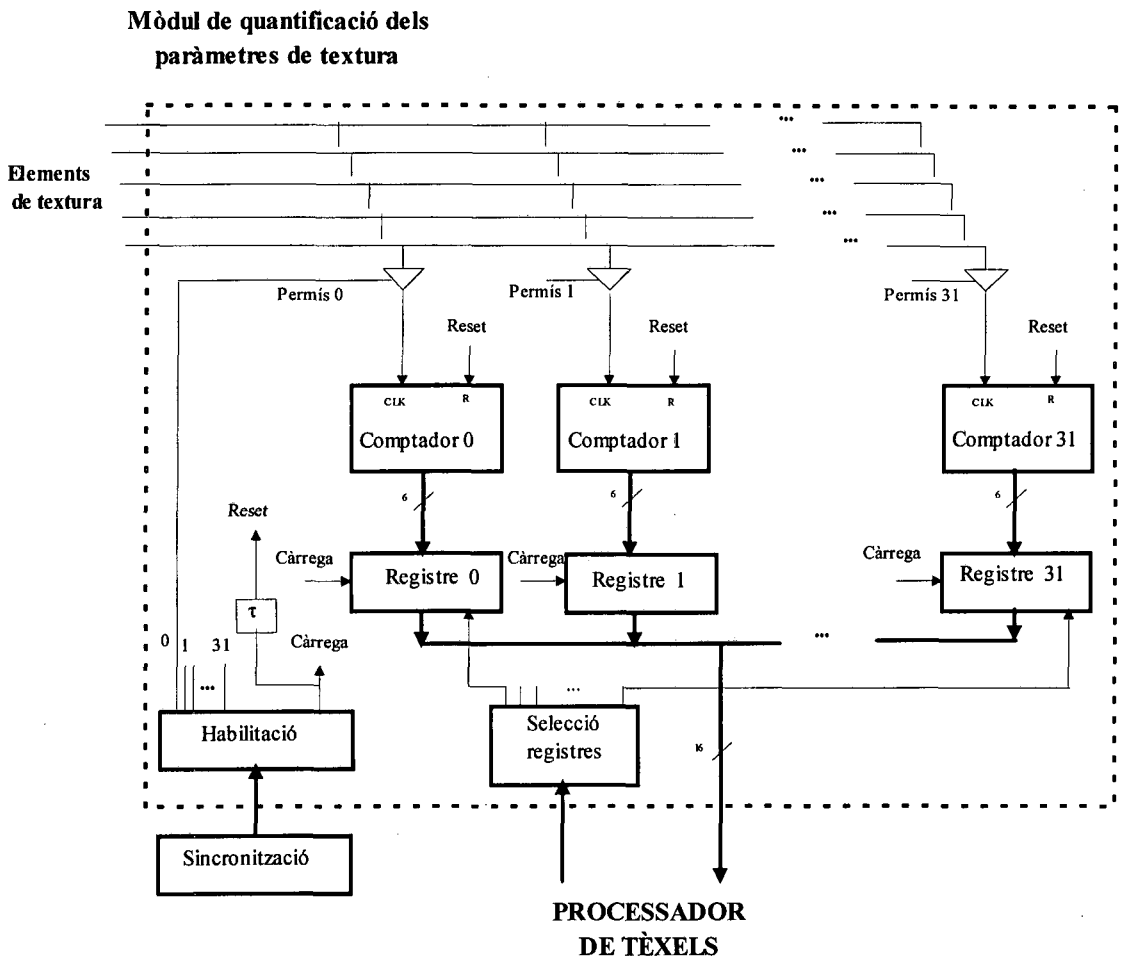

Figura 6.18. Esquema del comptatge de les característiques de textura.

Cada element de textura dels detectats en els submòduls anteriors arriba a un bloc com el de la figura, per tant, hi ha tants blocs de comptatge com paràmetres. Segons la posició que ocupi l'element de textura en la columna de la imatge, que va de 0 a 255, es permetrà el comptatge en un comptador o en un altre. Llavors, la porta *tri-state* s'activarà segons una lògica de decodificació (Habilitació) amb les adreces generades en el mòdul de sincronització. Aquest senyal d'activació serà el senyal **Permís i**.

En la taula 6.5 es pot veure la correspondència entre la posició que ocupa l'element actiu en la imatge, el comptador activat i les adreces que permeten l'activació. Els tres primers bits d'adreces que indiquen la columna en la imatge (**c2, c1 i c0**) no s'usen en la decodificació perquè cada tèxel ocupa vuit elements de la imatge.

Posició de l'element actiu **EA**	Nº de compt. activat	**c7**	**c6**	**c5**	**c4**	**c3**
0, ..., 7	0	0	0	0	0	0
8, ..., 15	1	0	0	0	0	1
...	...	...	...	...	...	...
248, ... 255	31	1	1	1	1	1

Taula 6.5. Relació entre el número de comptador i l'adreça de la posició de l'element de textura actiu.

Aquest procés de comptatge per columnes ha de finalitzar cada vuit files. Quan s'arribi a una fila múltiple de vuit, el valor del comptador tindrà acumulat el valor dels tèxels. Per aquest motiu el procés serà: i) guardar en el registre el valor del comptador; ii) fer un *reset* als comptadors doncs ens arribarà el primer element dels nous tèxels.

Els dos senyals (*Reset* i *Càrrega*) es generaran a partir de les adreces de les files que són les que porten el compte de la fila actual. Al ser 8 la mida dels tèxels pel que fa a les files, caldrà generar aquests senyals quan les adreces **f2, f1 i f0** prenguin el valor 0. Això correspondrà al principi de la línia 0, de la línia 8, de la línia 16 i així successivament per totes les files, mòdul 8. Una altra consideració serà retardar un cert temps la generació del senyal *Reset* respecte el senyal *Càrrega* doncs podria provocar l'esborrat del comptador i la conseqüent càrrega del valor zero al registre. Per tant, el retard haurà de ser inferior al període de píxel per no esborrar els comptadors quan ja hagi arribat el primer element de textura del nou tèxel ($\tau <$ CLK Píxel).

Per a cada paràmetre de textura (excepte el *difuminat*) disposem d'un bloc de comptadors exactament igual al de la figura 6.18 on l'únic que varia és l'element actiu d'entrada. La sortida de tots els blocs són 32 registres per 4 paràmetres que contenen tots els tèxels d'una fila de tèxels. Aquestes seran les dades que ha de llegir el mòdul d'assignació de grups per classificar els 32 tèxels que hi ha disponibles.

El comptador pel paràmetre *difuminat* serà un sumador acumulador de 14 bits per cada tèxel. Donat que l'element difós és un número de 8 bits, cal acumular aquest valor per les 8 files i les 8 columnes que ocupa un tèxel de difuminat. La sortida del sumador serà el valor que es guardarà per ser llegit en el següent mòdul d'assignació de grups. A l'haver-hi en cada tèxel 64 elements a sumar, de 8 bits cadascun, la sortida normalitzada serà com a màxim de 14 bits. Tot i que el valor del tèxel del *difuminat* s'ha de normalitzar i acabarà sent de 8 bits, no podem suprimir cap bit de la suma, doncs perdríem resolució del valor. No es pot realitzar l'operació de l'equació (6.1), sinó que s'haurà de calcular mitjançant (6.2). La normalització s'ha de realitzar en el mòdul d'assignació de grups després d'haver llegit el registre de 14 bits.

$$T_{D(i,j)} = \frac{t_{D(i,j)}}{64} * \frac{N_{màx}}{I2} \tag{6.1}$$

$$T_{D(i,j)} = t_{D(i,j)} * \frac{N_{màx}}{64 * I2} \tag{6.2}$$

El principal problema que apareix ara és saber si el mòdul d'assignació de grups dissenyat podrà suportar la cadència amb la qual es recarregarà un nou valor sobre els registres de sortida, els quals han de ser llegits abans no es recarreguin amb la nova fila de tèxels. Aquest serà el temps que ocupen 8 línies del senyal de vídeo i és el següent

8 línies de vídeo * 64 μseg /línia = 512 μseg.

En aquest temps s'han de llegir 32 registres, on cadascun d'ells forma un punt de l'espai de característiques. D'això es desprèn que el temps que tenim per llegir cada tèxel és de

$$512\ \mu\text{seg} / 32\ \text{tèxels} = 16\ \mu\text{seg per tèxel.}$$

Per obtenir cada tèxel s'han de llegir cinc registres que es carreguen simultàniament (un per cada component del punt).

Tot el procés de detecció i quantificació dels paràmetres de textura que s'ha dissenyat en aquests darrers subapartats té una característica principal: no consumeix temps. Amb això es vol dir que el circuit és transparent i la cadència dels elements a l'entrada és la mateixa que a la sortida. Aquesta característica es pot aconseguir mitjançant l'estructura *pipe-line* que té tot el procés de generació d'elements de textura.

El mòdul d'assignació de grups ha de ser capaç de llegir síncronament i tractar cada registre en el temps especificat. No és una tasca senzilla doncs s'ha de calcular el grup al qual pertany el tèxel i escriure'l a la sortida. Veiem aquest mòdul a continuació.

6.2. Mòdul d'assignació de grups.

Per implementar aquest mòdul no usarem una arquitectura específica sinó que ens servirem d'un ordinador estàndard (tipus PC). Les raons de l'ús d'una màquina estàndard són, primer la dificultat de la tasca a realitzar sense un dispositiu microprocessador i, segon, que el temps que es necessita per executar aquesta tasca la pot dur a terme una màquina estàndard.

Sobre aquesta màquina executarem les dues fases de la classificació:

1. Fase d'aprenentatge: a partir d'unes imatges inicials es calculen les seves característiques de textura per *software* i s'agrupen tal com es veia en el capítol 4.

2. Fase de treball: es connecta el *hardware* amb l'arquitectura específica sobre un *slot* d'expansió de la màquina i s'executa el programa que llegirà dels registres de la placa específica assignant cada mostra a un grup determinat anteriorment en la fase d'aprenentatge.

La tasca que ha de realitzar la màquina es pot resumir en el següent algorisme:

1. Lectura i normalització dels cinc registres que formen una mostra.
2. Càlcul de les distàncies a tots els centroïds dels grups.
3. Generació com a sortida del número de grup de distància mínima.

El diagrama de la figura 6.19 ens mostra les tasques a realitzar segons l'algorisme anterior amb el màxim de temps disponible.

Quan aquest algorisme es tradueixi a un llenguatge de programació algunes fases es solaparan o es reduiran, com per exemple la lectura dels registre o el càlcul del mínim a mida que es van calculant les distàncies.

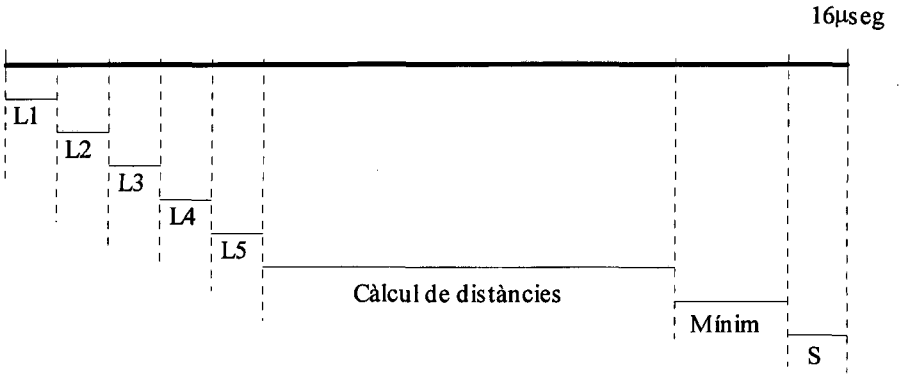

Figura 6.19. Diagrama de la disposició en el temps de les accions a realitzar.

Per demostrar la viabilitat de l'algorisme dins aquest estret marge de temps, a continuació detallarem l'algorisme.

6.2.1. Algorisme del càlcul de distància mínima.

Aquest algorisme té per objectiu determinar quin és el grup que està a distància mínima d'un tèxel en forma de cinc components. Com a entrada tenim els cinc valors dels paràmetres de textura del tèxel i com a sortida tenim un número que correspon al grup més proper al tèxel. El càlcul de la distància es converteix en la distància al centroïd del grup tal com s'ha explicat en el capítol de la classificació.

Com a estructura de dades, tenim els grups sobre la variable **Grups** on hi ha cada component del centre en posicions consecutives. Els tèxels es guarden també en forma consecutiva i en el mateix ordre de components que els grups per poder ser comparats amb la mateixa variable d'índex. Aquests tèxels han estat prèviament normalitzats mitjançant una taula en la posició de memòria **Taula** (actua de *Look-up-table*).

Hi ha també unes constants que indiquen les adreces dels ports de lectura i escriptura. El valor del mínim de la distància es guarda sobre una variable que s'ha inicialitzat al màxim valor que pot prendre, donat que així segur que s'actualitzarà juntament amb el número de grup (variable **Punter**).

Si recordem la regla d'assignació que havíem definit per (4.2), es busca la mínima distància euclidiana entre la mostra i tots els centroïds dels grups:

$$\text{Regla d'assignació} = \{ \mathbf{X} \in i \mid \mathit{mín}\{d(\mathbf{X},C_i)\}$$

L'algorisme en pseudocodi per seguir la regla d'assignació és el següent:

```
inici
Llegir_texel(Tèxel)
normalitzar(Tèxel)
Mínim = Màxim;

per  i = 1 fins a  i = Nb_grups de 1 en 1

        per  j = 1 fins a  j = 5 de 1 en 1
             aux = aux + (grup[i][j] - Tèxel[j])^2;
             aux = sqrt (aux);
        fper
        si   Mínim > aux  llavors        Mínim = aux;
                                         grup = i
        fsi
fper
escriure (grup)
final
```

Mirant aquest codi no és immediat saber si es podrà executar en el tèmps limitat del qual disposem. Per això, també he fet la seva implementació en llenguatge assamblador on es té la precisió tant del temps d'execució com de les instruccions a executar, ja que no hi ha cap compilador entre mig i no s'afegeixen instruccions no desitjades i incontrolables.

En el següent codi hi ha més detall donat que en calcularem el temps, però la funció que desenvolupa és la mateixa que en el programa en pseudocodi.

Algorisme de càlcul de la mínima distància entre un tèxel i els centres dels grups.

```
; Inicialització de registres
        mov   dx, Ports              (1)     ; base dels ports
        mov   bx, Tèxel              (1)     ; base dels tèxels
        mov   si, Taula              (1)     ; base de la taula de normalització
        mov   Mínim, Valor_Màxim (1)         ; inicialitzo la variable amb el màxim
                                             ; valor
; Lectura de registres
        in    ax, dx                 (20); primer i segon registres (AH+AL)
        mov   cx, [ah+si]            (1)  ; primer tèxel
        mov   [bx], cx               (1)
        add   ax, 256                (1)
        mov   cx, [al+si]            (1)  ; segon tèxel
        mov   [bx+2], cx             (1)
        inc   dx                     (1)
        add   si,'256                (1)
        in    ax, dx                 (20); tercer i quart registres (AH+AL)
        mov   cx, [ah+si]            (1)  ; tercer tèxel
```

```
        mov   [bx+4], cx          (1)
        add   si, 256             (1)
        mov   cx, [al+si]         (1)  ; quart tèxel
        mov   [bx+6], cx          (1)
        inc   dx                  (1)
        in    ax, dx              (20); cinquè registre (AX: difuminat)
        mul   ax, norm_dif        (26); normalització del tèxel
        mov   [bx+8], ax          (1)

; preparació i salt al càlcul de distàncies
        mov   cx, Nb_grups        (1)  ; número de grups
B1:     jmp   distància           (3)
B2:     cmp   Mínim, ax           (2)  ; busco el mínim
        jge   FI_B1               (3)  ; salto si no és menor que Mínim
        mov   Mínim , ax          (1)  ; actualitzo el mínim
        mov   Punter, cx          (1)  ; guardo la posició del mínim
FI_B1:loop    B1                  (7)  ; bucle per tots els grups

; sortida
        mov   dx, Port_out        (1)
        out   dx, Punter          (20); escric a un port de sortida el número
                                       ; del grup que està a distància mínima
                                       ;del tèxel que s'ha llegit

; càlcul de la distància
distància:  mov   aux, 0          (1)  ; variable on s'acumularà la suma
            mov   bp, Grups       (1)  ; base als grups
            mov   ax, cx          (1)
            mul   ax, 5           (26)
            add   bp, ax          (1)  ; base al grup que es tracta
            push  cx              (4)
            mov   cx, 5           (1)
B3:         mov   ax, [bx + si]   (1)  ; primer registre
            mov   dx, [bp + si]   (1)  ; primera component del grup
            sub   ax, dx          (1)  ; x_i -y_i
            mul   ax, ax          (26)
            add   aux, ax         (3)  ; Σ
            add   si, 2           (1)
            loop  B3              (7)

; sortida de distància
            mov   ax, aux         (1)
            fsqrt ax              (87) ; distància euclidiana
            jmp   B2              (3)
```

Falten certes directives de control per a l'assamblat del codi però que no generen instruccions significatives per a l'increment del temps d'execució.

Suposant que treballem amb un microprocessador Pentium a 150MHz, cal saber el cicles de rellotge per a cada instrucció per fer un càlcul del temps d'execució del codi. Mirant el manual del microprocessador sabem la correspondència entre el nombre de cicles i les instruccions. Aquests cicles venen indicats en el codi pel número entre parèntesi al final de cada línia de codi.

Part del codi	**Cicles de rellotge**
Inicialització	4
Lectura de registres	73
Preparació i salt	1+ (17+cicle *distància*)*número de grups
Sortida	21
Distància	235
Total	99+(252*número de grups)

Taula 6.6. Relació del codi amb els cicles de rellotge que ocupen.

Suposant que tot el codi, les variables i els centres dels grups estan en la memòria *cache* que té internament el microprocessador, els cicles que necessita l'algorisme es recullen en la taula 6.6, desglossant cada rutina del codi. Tal com es pot veure, el temps serà funció del nombre de grups que s'hagin definit, donat que s'ha de calcular la distància a tots els centres dels grups.

La relació entre els cicle de rellotge i el temps real d'execució compleix la següent relació, segons [INT95],

Temps d'un cicle = 1/freqüència de treball del microprocessador

Si el nostre processador treballa a 150 MHz, llavors el temps de cicle és de 6.66nseg. Per saber quants cicles tenim disponibles en els 16μseg hem de calcular

Nb. de cicles = 16μseg / 6.66nseg per cicle = 2400 cicles.

Llavors, si volem saber quin serà el nombre màxim de grups amb els quals podem trobar totes les distàncies dins el límit de temps, cal fer

2400 cicles = 99+(252*número de grups) + cicles de seguretat

Si donen uns 200 cicles de seguretat per instruccions no controlades i possibles instruccions que estiguin en la memòria *cache* del processador, queda que

2200 cicles = 99+(252*número de grups)

número de grups = 2101/252 = 8.33 ≥ **8 grups**

D'aquí es desprèn que el màxim nombre de grups és 8 amb aquest suport informàtic. Aquest nombre de grups és suficient per crear les fronteres entre textures en diverses tipologies d'imatges.

Tota aquesta rutina pot ser cridada de dues maneres: primer, quan s'han carregat els registres de sortida dels comptadors es pot generar una interrupció per llegir-los i executar l'anterior codi, pel fet de la interrupció no hi ha despesa de temps; segon, si

no cal executar cap més procés podem fer enquesta sobre un registre que ens indiqui quan s'han carregat els registres, tampoc aquí cal restar temps pel fet de fer l'enquesta, ja que aquesta es fa quan encara no s'han carregat els registres.

6.3. Generació de la imatge segmentada.

Tal com s'ha vist en l'algorisme anterior, els resultats de la classificació per textures es donen en forma numèrica on per cada mostra a l'entrada s'indica el grup de textura al qual pertany. Una forma de visualització de la imatge segmentada es combinar la imatge original i una imatge formada per les fronteres entre els grups de textura de sortida per cada mostra. La figura 6.20 representa esquemàticament tot el procés de generació dels grups de textura més aquest concepte de barrejar la imatge original amb la imatge de les fronteres entre regions de textura.

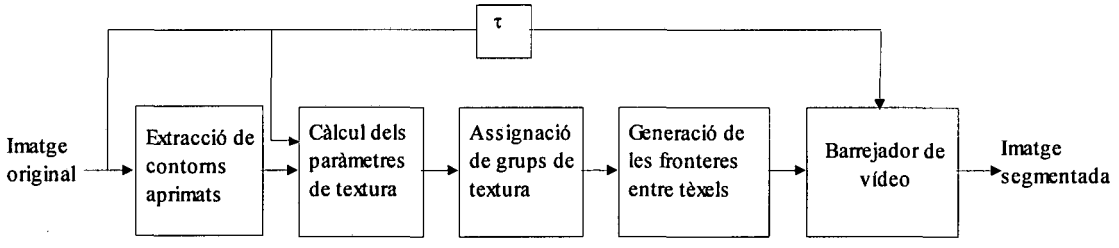

Figura 6.20. Procés de segmentació amb visualització de les regions per fronteres.

Per aquest motiu cal generar la imatge de fronteres i per això s'ha de saber per cada tèxel el seu valor de textura en relació amb els seus veïns. Per comparar el veïnatge només caldrà guardar la última línia de tèxels segmentada i decidir si hi ha o no hi ha frontera entre mostres.

Fila guardada	...	Grup i	**Grup j**	Grup k	...
Fila actual	...	**Grup m**	**X**		...

Taula 6.7. Generació de fronteres.

Suposem que l'element actual que acaba de sortir del mòdul d'assignació de grups és l'element **X**. Els seus veïns implicats en el càlcul de les fronteres serà la mostra **Grup j** i la mostra **Grup m**. Si el número de grup de la mostra **X** i el número de grup de la mostra **Grup j** són diferents, cal generar una línia que separi aquells dos grups. El mateix passarà entre l'element actual i el seu veí immediatament anterior. Si, pel contrari, el número de grup és el mateix no cal generar cap línia doncs pertanyen al mateix grup i no cal que aparegui una segmentació entre ells. La actualització de les mostres guardades és automàtica per cada nova mostra que surt, guardant sempre només 33 elements (1 fila més un element). La taula 6.7 indica la disposició dels elements de la imatge de sortida.

6.4. Implementació.

L'estudi d'implementabilitat requereix avaluar la complexitat del *hardware* necessari utilitzant la tecnologia actual.

La taula 6.8 ens resum les necessitats tant de memòria com de portes lògiques que es necessiten en el mòdul de càlcul dels paràmetres de textura, és a dir, en el mòdul de detecció de píxels característics juntament amb el mòdul de quantificació dels paràmetres.

Paràmetre	**Memòria**	**Portes i biestables**
Linealitat	-	28 màscares *2 portes/màsc.
Granulositat	-	9 màscares * 5 portes/màsc.
Discontinuïtat	-	16 màscares * 3 portes/màsc.
Abruptitat	-	21 màscares * 1 portes/màsc.
Difuminat	Versió iterativa: 2 línies vídeo multinivell 10 registres	1 restador, 1 selector, 3 comparadors
	Versió paral·lela: 2 línies vídeo multinivell 9 registres	2 restadors, 4 selectors, 12 comparadors
Parcial	5 línies vídeo binari 3 línies vídeo multinivell	150 biestables (registres) 32x4 comptadors (8 bits), 32 comptadors (14 bits)
Total	29 línies vídeo: 29 x 256 x1 bit < 8Kbits.	≈ 1800 Biestables ≈ 4500 portes

Taula 6.8. Requeriments de la nostra arquitectura.

Després d'observar els requeriments dels components que es necessiten en aquest disseny, arribem a la conclusió que els dispositius FPGA són suficients per cobrir aquestes necessitats. Un muntatge amb components discrets MSI no té sentit donat el gran nombre de portes i biestables. Pel que fa a la freqüència de treball, a l'escollir un dispositiu FPGA s'ha de descartar el disseny del càlcul del paràmetre difuminat amb la versió iterativa, donat que no poden treballar a aquella freqüència. Aleshores, l'increment que resulta d'escollir la versió paral·lela és perfectament absorbible pel dispositiu d'implementació.

Actualment existeixen productes al mercat dels FPGAs que superen àmpliament els requeriments del nostre disseny. La taula 6.9 ens mostra alguns dels productes que actualment comercialitza ALTERA.

Model	Portes lògiques	Biestables	Memòria	Pins I/O
EPF10K10	10.000	576	6.144	150
EPF10K20	20.000	1152	12.288	198
EPF10K30	30.000	1728	12.288	246
EPF10K40	40.000	2304	16.384	278
EPF10K50	50.000	2880	20.480	310
EPF10K70	70.000	3744	18.432	358

Taula 6.9. Alguns dispositius FPGA comercialitzats.

Tot i que qualsevol dispositiu conté suficient nombre de portes lògiques, no tots inclouen el mínim necessari pel que respecta a biestables. Es podria escollir entre el model EPF10K50 o, fins i tot, el model superior, ja que el grau d'ocupació, pel que fa als biestables, és excessivament elevat en model inferiors, afectant al correcte funcionament de la FPGA.

Com a conclusió del disseny de l'arquitectura, podem dir que és possible tractar sincronitzadament i a velocitat de *video rate* les dades que es generen en l'extractor de característiques de textura. Aquestes dades poden ser classificades també sincronitzadament, creant una sortida ja sigui amb el número de grup on està cada tèxel o bé ja sigui una imatge superposant la imatge original amb les fronteres que es generen entre regions de textura. Tot aquest disseny és possible implementar-lo amb un dispositiu FPGA mantenint els requeriments de funcionament a temps real.