AVALUACIÓ DELS ALGORISMES D'ENCAMINAMENT PER A UNA NOC AMB TOPOLOGIA MESH 2D

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Sergi Risueño Ruiz
i dirigit per
Jordi Carrabina Bordoll
Jaume Joven Murillo
Bellaterra 17 de Setembre de 2007.

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, Jordi Carrabina Bordoll i Jaume Joven Murillo
Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Sergi Risueño Ruiz

I per tal que consti firma la present.

Signat: Jordi Carrabina Bordoll Jaume Joven Murillo

Bellaterra, 17 de Setembre de 2007

Índex

Capítol 1

1 Introducció

En aquest primer capítol s'introduirà ell projecte, les seves motivacions i els objectius als quals s'intentarà arribar.

1.1 Introducció i Motivacions

En l'actualitat, l'increment del nombre de transistors en cada xip, seguint la llei de Moore que diu que cada 18 mesos l'increment de portes es duplica, ens porta a desenvolupar, cada cop amb mes efectivitat, noves eines i metodologies per aprofitar aquesta tecnologia en constant progrés. Les metodologies que actualment es proposen son la utilització dels anomenats IP cores, els quals disposen d'una qualitat molt important en aquest mon de modularització al que portem i es la reusabilitat, aprofitar dissenys ja realitzats per portar-los a d'altres projectes.

Així es com es crea el concepte de SoC, on un conjunt d'IP cores estan intercomunicats utilitzant busos, creant així tot un sistema dintre d'un únic xip.

En aquest projecte anirem encara un pas mes endavant, utilitzant, com ja fèiem al SoC, aquest IP cores com a part d'un mateix xip i afegint una comunicació factible entre ells, es el sistema de les NoC, la comunicació d'aquest elements es una de les parts mes importants i el seu algorisme d'encaminament és el que tractarem en profunditat.

El sistema de comunicació de la NoC es completa sabent la topologia es la qual s'engloba tot aquest sistema, per aquest projecte utilitzarem una topologia de xarxa Mesh 2D.

Els algorismes d'encaminament son els que permetran la comunicació entre nodes, ja que son els encarregats de definir el camí que recorrerà el paquet des de el node origen fins el node destinació. Hi ha diferents tipus d'algorismes, nosaltres ens centrarem principalment en quatre, l'algorisme determinista XY i tres algorismes parcialment adaptatius el West First, el North Last i el Negative First.

Aquest tres algorismes tenen la propietat de en certes situacions no fer esperar el paquet perquè algun altre esta transmeten sinó que s'adapten i l'encaminen cap a un altre camí fent així el tràfic més fluid.

Així doncs aquest projecte constarà de l'avaluació d'aquests algorismes dintre del sistema de comunicació de la NoC, el seu router.

Les motivacions pel que fa a l'elecció del projecte, son aprofundir mes en l'arquitectura de les NoCs, ja que si be aquesta arquitectura es vista durant algunes assignatures de la carrera, no es tant plausible com quan es fa un estudi mes exhaustiu de la mateixa i ens endinsem en el mon de la microelectrònica amb ella.

A mes un altre motivació és veure la profunditat i el paper que aquests algorismes parcialment adaptatius prenen en el sistema de comunicació de la NoC i com cadascun d'ells actua enfront els mateixos estímuls, així tindrem una idea de quin algorisme es mes òptim depenent de les condicions de la nostra NoC. Amb aquesta informació es pot millorar el sistema de comunicació de tota la NoC permeten així que el tràfic sigui mes fluid, ràpid o fiable.

A nivell personal, les motivacions que em van portar a escollir aquest projecte son ben clares, doncs he cursat la majoria de les assignatures del departament de microelectrònica, i l'interès per l'àrea es mes que evident. Pel que respecte a aquest projecte en concret, té un gran incentiu, ja que es veu l'actualitat de la microelectrònica reflectida en les NoCs, aquestes donen una idea de fins aquin grau d'evolució portem i son el futur de la mateixa. Així doncs a part d'escollir la tecnologia amb la que es treballa avui en dia, també es veia en profunditat el sistema de comunicacions empleat, ja que s'avaluaven alguns dels diferents algorismes d'aquesta arquitectura.

1.2 Objectius del projecte

L'objectiu d'aquest projecte es l'estudi i realització d'un router adaptatiu per topologies mesh 2D, aquest router ha de ser capaç d'encaminar adaptativament segons l'estat del canal (ocupat / lliure) a cada moment.

Un cop s'ha fet l'estudi del router amb algorisme d'encaminament XY, es crearan tres router similars amb cadascun dels algorismes parcialment

adaptatius, West First, North Last i Negative First, per tal de poder comparar els diferents algorismes dintre de les mateixes condicions i així poder obtenir resultats fiables sobre certs aspectes dels mateixos com són, la freqüència de rellotge o els costos en tamany dintre d'una FPGA.

Així doncs l'objectiu principal d'aquest projecte es tracta d'una avaluació de les diferents alternatives pel que fa a algorismes d'encaminament dintre d'una NoC amb topologia mesh 2D, on es comparen els mateixos i on s'obtenen resultats experimentals dels mateixos.

Capítol 2

2 Visió Global de les NoCs

En aquest segon capítol s'explicarà l'evolució de la microelectrònica fins a les NoCs, els conceptes basics de la mateixa i certs aspectes que faran mes entendible l'estructura i composició de les NoCs.

2.1 Evolució de la microelectrònica: d'ASICs fins a NoCs en SoCs

El primers 20 anys després de la invenció del circuit integrat, el disseny microelectrònic es va mantenir molt lligat a cada fabrica i tecnologia pròpia. Així doncs els dissenyadors eren especialistes en la tecnologia en la que treballaven mes que en el propi disseny. La tecnologia anava avançant, però els processos de disseny estaven estancats en les tècniques manuals i no hi havia cap suport metodològic ni cap eina per poder dissenyar, es a partir d'aquí quan comencen a sorgir algunes eines per al disseny i la simulació de circuits elèctrics, que va fer d'estímul perquè el procés de disseny no anés lligat a la fabrica. Es aquí on realment es pren consciencia del gran buit que existeix entre la tecnologia que s'és capaç de fer i el poc que es pot desenvolupar donada la inexistència de metodologies i eines dedicades específicament.
Es van dur a terme doncs dues estratègies, desenvolupar metodologies i eines que permetessin controlar l'enorme complexitat del disseny microelectrònic i facilitar la reutilització tecnològica a traves de proporcionar estructures precaracteritzades per facilitar el procés de disseny aquest últim va donar lloc a les gate-arrays i les standard cells seguint dos conceptes diferents respectivament, un de estructures regulars que ja contenien dispositius actius i l'altre de disseny de biblioteques de cel·les.
La combinació d'aquestes dues estratègies va donar lloc a un salt qualitatius a un nivell superior d'abstracció, el nivell estructural. S'implanta la metodologia de disseny bottom-up on un cop realitzada de forma manual la descomposició arquitectural per blocs de disseny, es seleccionava la tecnologia i la seva biblioteca de cel·les corresponent i s'iniciava un procés de composició ascendent a partir d'aquestes cel·les fins a completar el circuit total.
Després d'això sorgeixen els dispositius reconfigurables per hardware FPGAs (***Field Programable Gate-Array***) que afegiren una alternativa al disseny microelectrònic digital I van reduir dràsticament el temps I els costos de desenvolupament de prototips I petites series al no requerir cap procés de fabricació específic.
Amb la introducció dels HDL es facilita el salt fins al nivell funcional i s'implanten metodologies top-down on, a partir d'unes certes especificacions inicials del disseny es plasmen en descripcions d'alt nivell d'abstracció en HDL, que

inclueixen tant el model funcional del circuit com un banc de proves del mateix, facilitant d'aquesta manera la simulació des de el principi del disseny. En els últims anys s'han consolidat a nivell industrial les tecnologies submicròniques i submicròniques profundes i s'estan desenvolupant i posant a punt les anomenades tecnologies nanomètriques.

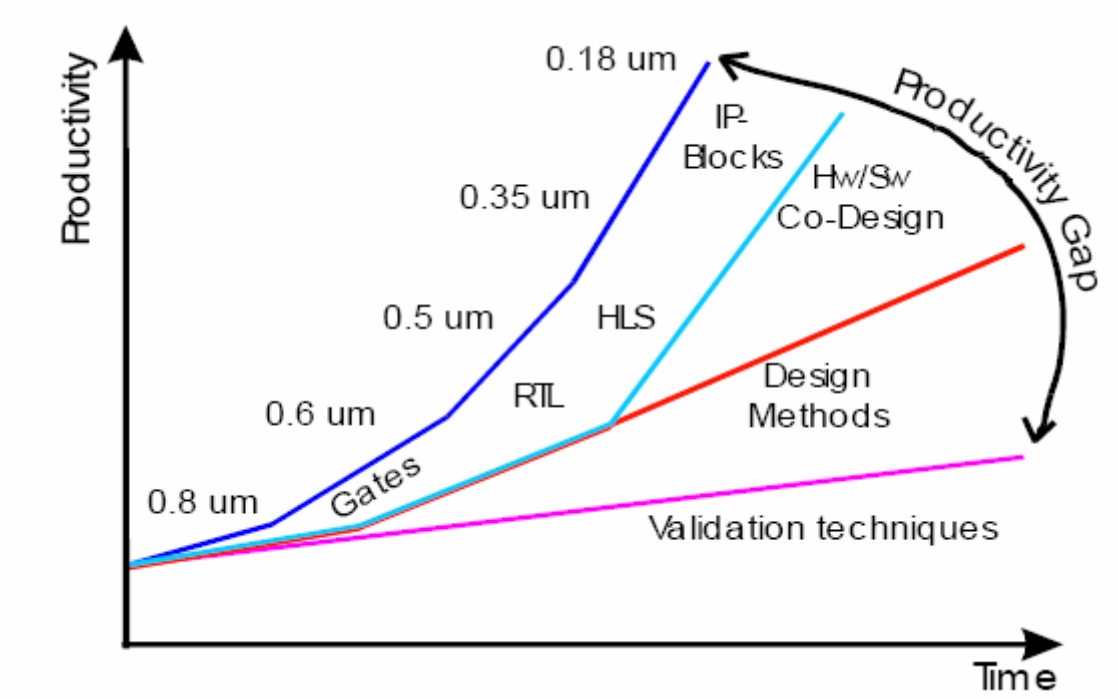


Figura 2.1 Desfase de productivitat entre tecnologia y disseny microelectrònic [LluísTeres06]

Tot i que es segueix mantenint la llei de Moore, que diu que cada 18 mesos s'ha de doblar la densitat de dispositius per unitat de superfície de silici, i malgrat els enormes avenços en les metodologies i eines de disseny, segueix augmentant el vuit relatiu de productivitat entre tecnologia i disseny tal i com veiem a la figura 2.1.

Per tal de reduir aquest forat entre la productivitat i la tecnologia, s'opta per fer dissenys reusables mitjançant la utilització d'IP cores. Aquesta és la solució escollida per tal d'evitar infrautilitzar la tecnologia de que es disposa.

Un IP core (Intelectual Property core), es un bloc o macro de lògica, dissenyat i verificat per qui te els drets, que es pot integrar en el nostre sistema ja sigui sintetitzant-lo dins d'una FPGA, o en un ASIC, alguns exemples d'IP cores poden ser processadors, controladores d'Ethernet, etc.

Tot i aquesta mesura avui en dia no hi ha cap metodologia estàndard que faci que aquest forat entre productivitat i tecnologia desaparegui. Malgrat això s'han creat unes best practices, que juntament amb un nou paradigma de disseny i eines mes eficients intenten apropar els disseny a la tecnologia disponible dins del xip. Aquestes best practices son:

a) Reusabilitat : Utilitzant els IP Cores. Existeixen tres tipis d'IP Cores: el Soft Cores, el Hard Cores i els Firm Cores.

   - Soft Cores : *netlist* o codi HDL sintetitzable cap a diferents tecnologies. Són els més portables, i per tant els més flexibles a l'hora d'incorporar-los al disseny del SoC i NoC.

   - Hard Cores : *layout* específic cap a una tecnologia concreta (el *place & route* ja està fixat), normalment ja estan optimitzats per rendiment, àrea i/o consum. Són els menys portables i flexibles, però són els millors per aplicacions de *plug-and-play*.

   - Firm Cores : *netlist* genèrica, relativament independent de la tecnologia, ja que tenim unes restriccions de *place & route*. Són molt similars als hard cores, però un mica més portables cap altres tecnologies.

Per aconseguir que aquests IP Cores tinguin un major grau de reusabilitat, aquests han de tenir les següents característiques :

- ✓ Dissenyats per solucionar un problema de manera general.
- ✓ Dissenyats perquè es pugui utilitzar cap a diferents tecnologies.
- ✓ Dissenyats per poder-los simular en diversos simuladors.
- ✓ Verificats independentment del xip en el qual serà utilitzat (*complete suite*).
- ✓ Verificats amb un alt nivell de confiança.
- ✓ Completament documentats, tant a nivell d'aplicacions permeses (paràmetres que són configurables), com de restriccions, i addicionalment cal adjuntar els requeriments d'interconnexió.

b) Realitzar el disseny software i hardware paral·lelament. Això fa que als grups de treball hi hagi dissenyadors software, hardware i també d'altres que ho siguin de tot el sistema.
c) Utilitzar tècniques de validació i disseny.

Tenir en compte aquestes practiques fa que el nostre temps i esforç s'incrementi considerablement però a la llarga tindrem moltes avantatges pel fet que l'IP core estigui ben dissenyat.

Aquesta continua evolució tecnològica ens ha permès incloure milions de transistors en un cub de silici materialitzant d'aquesta manera sistemes complets en un sol xip, es el que coneixem amb el nom de SoC (System on Chip) [Jantsch03]. Aquests SoC inclouen tot tipus de blocs (analògics, digitals, memòries, processadors, DSPs, etc) que abans es distribuïen en xips especialitzats. Hi ha però una sèrie de limitacions tecnològiques que dificulten el desenvolupament del SoCs i limiten la seva aplicació i evolució a pesar de la seva gran flexibilitat funcional ja que incorporen molts blocs programables o configurables, entre aquestes limitacions trobem la integritat e les senyals, asincronismes entre punts allunyats del mateix xip, i d'altres.

Així doncs, una de les característiques dels SoC, que son dispositius de prestacions molt altes i molt flexibles, tenen fixada i tancada la connectivitat entre els diferents blocs que el composen, això no nomes suposa un clar punt en contra sinó que produeix el coll d'ampolla del disseny en les comunicacions globals dintre del propi xip.

Durant els 90 l'element central de comunicació en el SoCs és el bus al qual es connecten la resta d'elements del sistema. Aquesta topologia es difícilment escalable, ja que eficientment poden escalar de 3 a 10 components, però es fa molt costos incrementar el nombre de components[Jantsch03].

Fent un resum direm que els SoCs es caracteritzen per una funcionalitat flexible i una connectivitat i comunicacions fixades i inamovibles, que dificulten el seu escalat a nivells de complexitat majors.

Partint del problema de comunicació dels SoCs sorgeix el concepte de NoC (Network-on-Chip) on l'idea global es proporcionar flexibilitat a les comunicacions intraxip adoptant, esquemes, conceptes i protocols de les tant

esteses xarxes informàtiques[Jantsch03]. Entrarem amb mes profunditat al següent punt on explicarem els conceptes de les NoCs.
Tota aquesta evolució es pot veure detalladament en la figura 2.2 on es mostra gràficament l'evolució de la tecnologia i de les eines de disseny i producció (CAD, síntesi i simulació), passant de l'era ASIC fins a la NoC, on la tecnologia ha passat dels 20K portes fins als 2.5M.

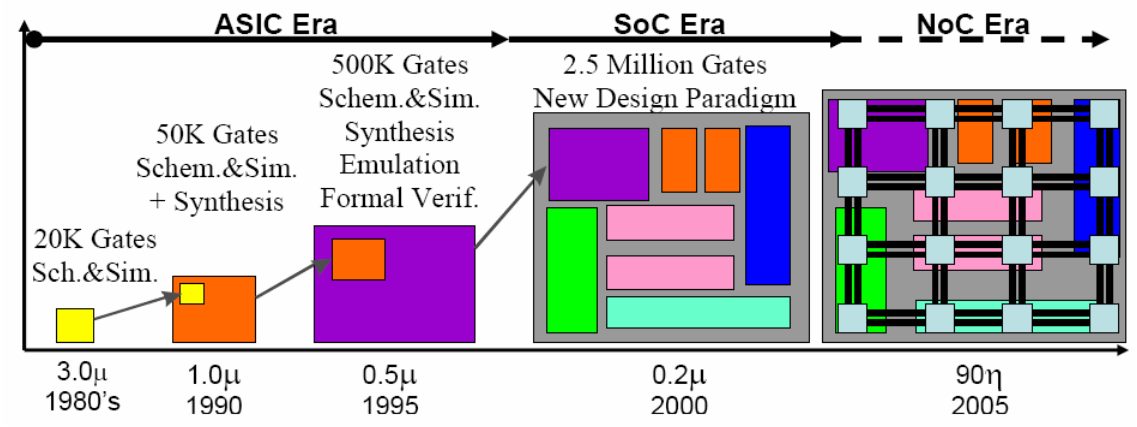


Figura 2.2 Evolució de la tecnologia i les eines en els xips. [LluísTeres06]

2.2 Conceptes basics de les NoCs

El concepte de NoC sorgeix de la necessitat d'aprofitar la gran quantitat de transistors dins del xip, amb el pas de la tecnologia submicrònica cap a les nanotecnologies. Donat que en un SoC, les estructures de bus no son eficients, ja que si parlem en termes d'escalabilitat, i si volguéssim que fossin escalables estaríem fent un ús molt poc eficient de l'energia que utilitzaríem ja que la comunicació global dintre del xip sempre incrementa en gran mesura el consum d'energia total. Així doncs la NoC, aconsegueix una escalabilitat molt alta, obté un nivell de consum molt bo i alhora disposa de una gran reconfigurabilitat.

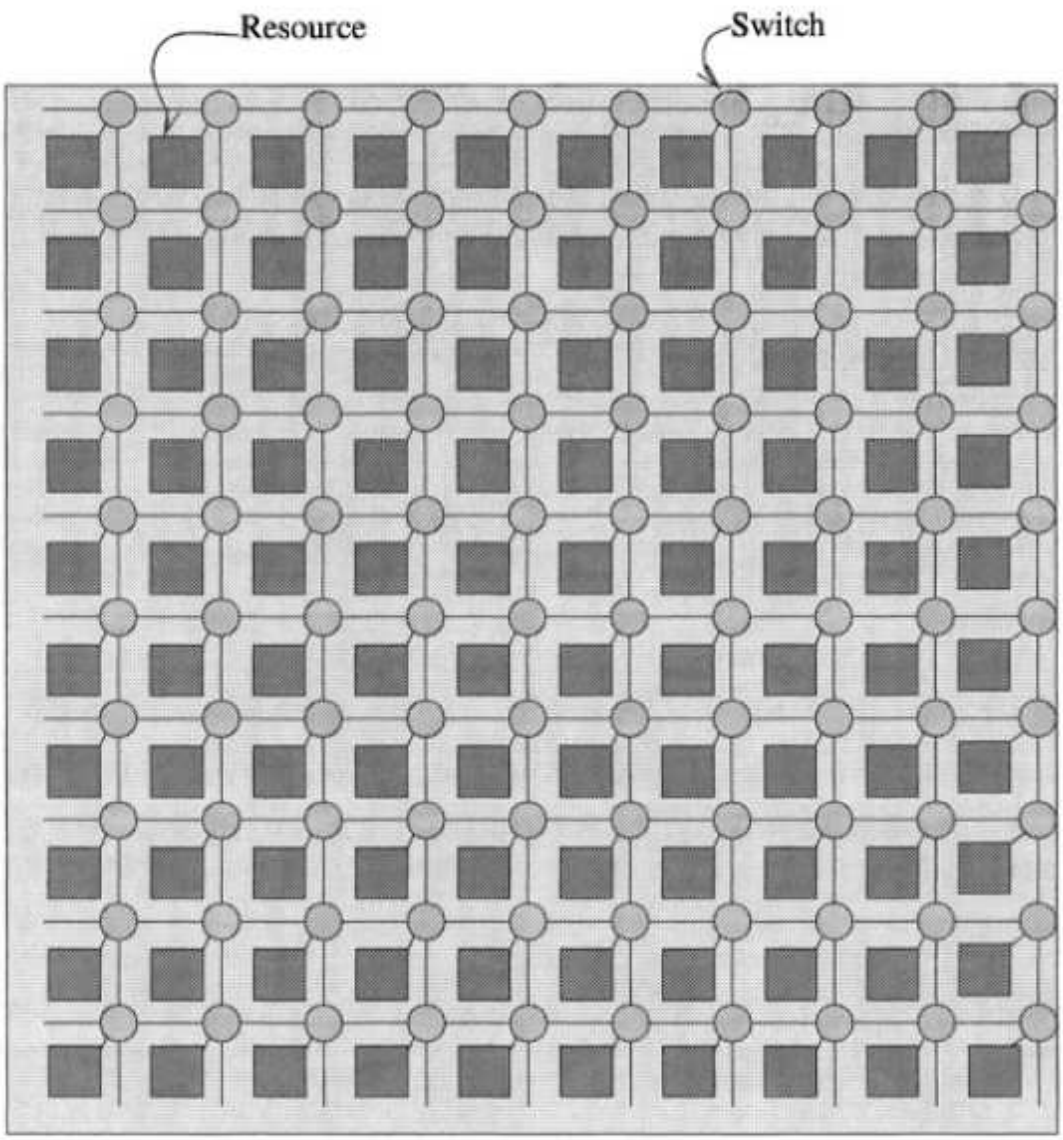


Figura 2.3 Vista global d'una NoC [Jantsch03].

Com podem veure en la figura 2.3, en la NoC podem distingir dos grans infraestructures:

1) Infraestructura de còmput: Un conjunt de nodes funcionals que constitueixen els recursos de còmput, que poden ser homogenis (un mateix processador elemental a cada node) o heterogenis (un node pot ser un recurs senzill o fins i tot una SoC completa) , amb major o menor grau de complexitat (configurabilitat o programació) i independents entre si.
2) Infraestructura de connexionat i comunicació: La qual segueix l'esquema de xarxa que proporciona flexibilitat en aquest nivell.

Podem veure un exemple de la NoC amb recursos diferents i el connexionat i els elements que el formen en la figura 2.4

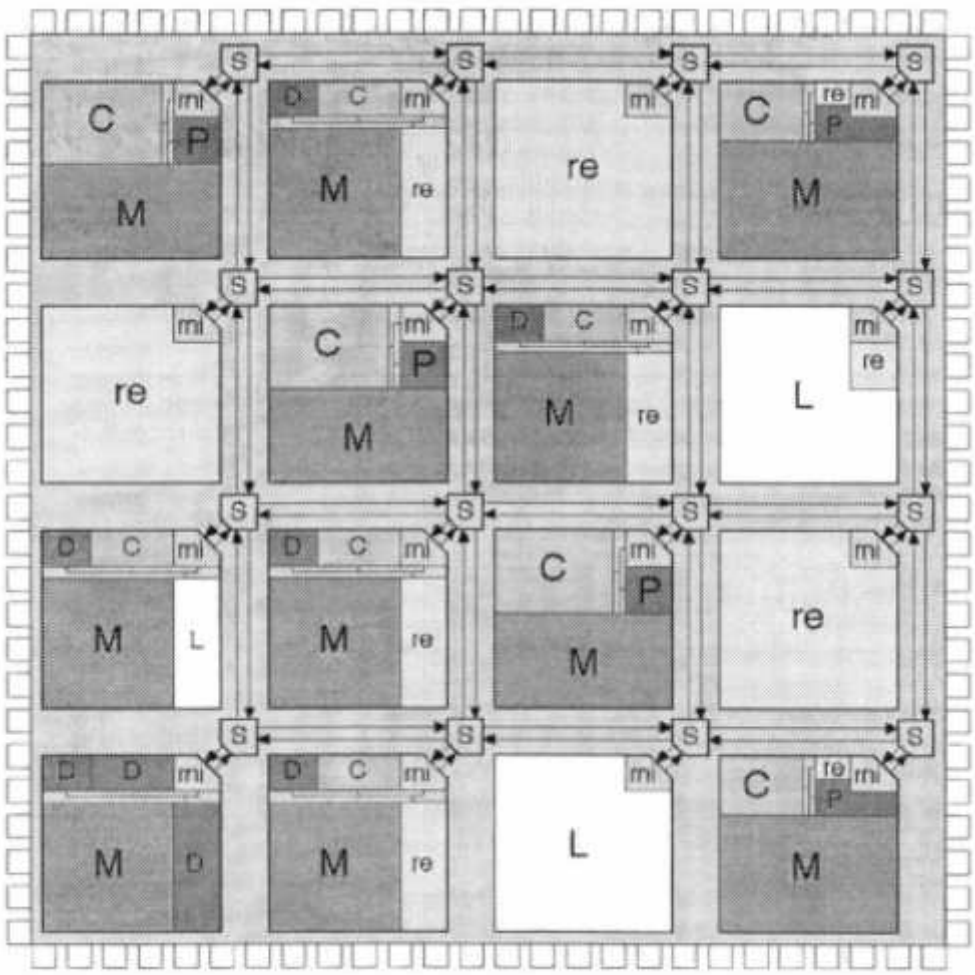


Figura 2.4 Exemple d'una arquitectura NoC, on S= Switch, rni=recurs d'interface de xarxa, P = processador, C = cache, M= memòria, D = DSPcore, re = lògica reconfigurable, L = Hardware dedicat [Jantsch03].

Aquesta NoC en comptes de tenir una comunicació global síncrona, que provocaria un gran consum al tenir una distribució molt elevada, utilitza una comunicació local síncrona eficient i una interconnexió entre aquest asíncrona, així doncs les comunicacions locals dintre de cada node tendeixen a utilitzar línees dedicades i esquemes clàssics de busos i sincronismes, les comunicacions globals es realitzaran a traves de la xarxa i tendiran a utilitzar protocols asíncrons, fent així constar l'estratègia de GALS (Globally Asynchronous & Locally Synchronous). Així doncs veiem que les NoC incorporen flexibilitat tant a nivell computacional com a nivell de

comunicacions intraxip i alhora faciliten el seu escalatge gradual a nivells superiors de complexitat.

Amb tot el comentat, podem dir amb certesa que la comunicació es un punt molt important en la infraestructura de la NoC, es per això que existeixen diverses topologies d'interconnexió. Una topologia de xarxa es la disposició física en la que es connecten els nodes d'una xarxa mitjançant la combinació d'estàndards i protocols. Aquestes xarxes poden ser de dos tipus directes i indirectes, la primera es aquella on els switch o router esta connectat a cada node terminal, en canvi a la segona el switch o router pot esta connectat a un altre switch o a un node terminal. Nosaltres ens centrarem en les directes i en presentarem d'algunes explicades breument ja que nosaltres ens centrarem majoritàriament en una.

- Topologia 2D Torus: Aquesta topologia redueix el nombre de routers o switchs, consisteix en que cada router de cada fila i columna esta connectat amb una estructura d'anell, ho podem veure a la figura 2.5. Es diferencia de la mesh 2D en que els elements externs tenen les mateixes connexions que els elements interns.

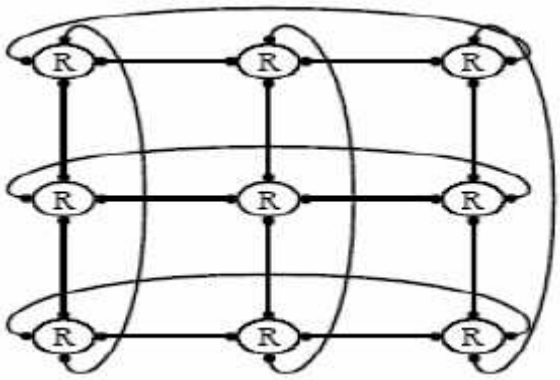


Figura 2.5 Topologia de xarxa 2D Torus

- Topologia hipercub : Aquesta topologia incrementa el grau de complexitat segons el numero de nodes. Consisteix en anar afegint cubs i unir-los entre ells, com es pot veure a la figura 2.6.

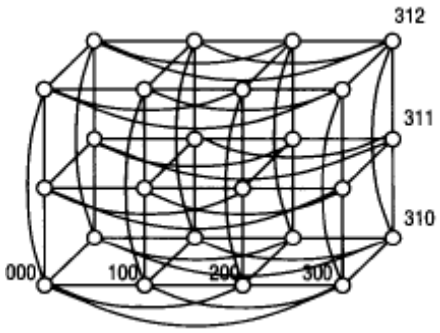


Figura 2.6 Topologia de xarxa Hipercub [Gaghan93]

- Topologia Mesh 2D: Aquesta topologia defineix que cada switch esta connectat amb 4 switchs mes, com es mostra a la figura 2.7. Aprofundirem més en aquesta topologia ja que es la topologia escollida per la nostra NoC.

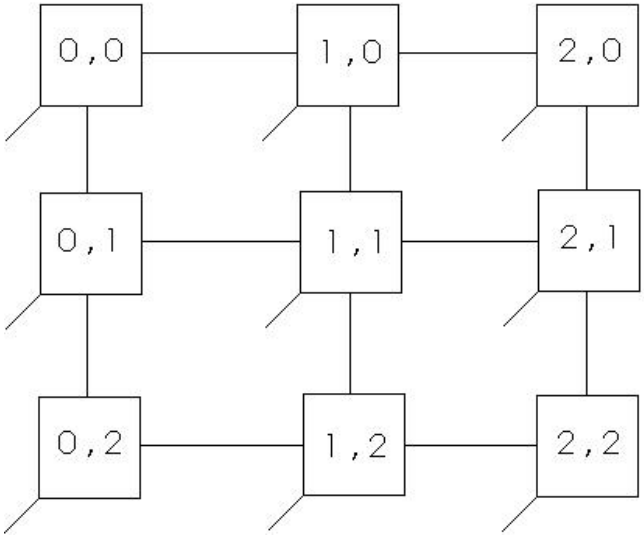


Figura 2.7 Topologia de xarxa Mesh 2D

2.2.1 Topologia Mesh 2D

La topologia Mesh 2D ens diu que cada swtich esta connectat a direccions diferents o veïns, al nord, al sud, al est i al oest i clarament també esta conectat al seu node que li direm local.
Una mesh 2D, es predecible i regular a nivell de layout físic ja que la seva estructura els nodes sempre tenen els mateixos veïns, i es diu que és predecible ja que es pot predir el comportament d'un paquet.
Per aquest tipus de topologia hi ha dos variants segons el sistema de eixos que es vulgui agafar, un és agafar el sistema d'eixos matemàtic, el qual parteix que

el punt 0,0 esta al sud-oest de la malla i l'altre es partint dels eixos com si d'una matriu es tractes d'aquesta manera el 0,0 estaria al nord-oest, com es mostra a la figura 2.8. Qualsevol d'aquestes dues maneres es bona sempre que es tingui en compte sempre quina d'elles estem seguint.

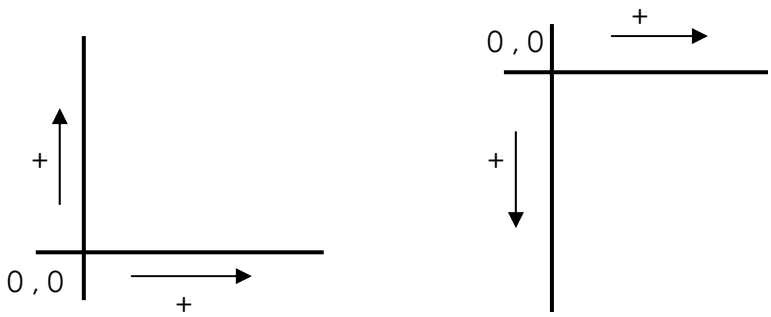


Figura 2.8 Diferents maneres d'agafar els eixos

2.2.2 Sistema de comunicació en una NoC

Hi trobem una altra tipus de classificació referent a les connexions dels diferents components en el nostre cas els diferents routers i switchs. Principalment podem dir que existeixen dos tipus de mecanismes:

- Conexion-less : En aquesta comunicació, cada paquet es preparat amb una capçalera que conte l'adreça de destinació que és suficient per fer que l'entrega sigui independent del paquet. Aquesta es una comunicació molt mes dinàmica sense haver una connexió prèvia entre l'origen i el destinació [Jantsch03].

- Conexion-oriented :En aquest tipus de connexió el primer que ens transmet es una connexió entre el origen i el destinació i un cop aquesta connexió esta fixada es comencen a transmetre els paquets d'informació pel mateix canal. En aquest tipus de connexions la relació entre les fonts d'origen i destinació se'n diu que es una relació lògica (logical relationship) [Jantsch03].

Algunes tècniques de comunicació que utilitzen tots dos sistemes de comunicació son :

- Store and Forward : El missatge es descompost en unitats mes petites que són els paquets, cadascun d'ells conte l'informació sobre el seu destinació a la capçalera. Quan es reserva un canal, el paquet es transmet complet emmagatzemant-se en la memòria del següent node i fins que no ha arribat el paquet complet no es pot encaminar pel següent canal.

- Wormhole : El missatge es descompost en unitats de flux (flits), en el primer flit s'encamina reservant el canal de comunicació, els flits de dades segueixen al flit de capçalera fins que arriba l'últim flit, el flit de cua. A cada pas per node es poden emmagatzemar uns quants flits a les cues.

- Cut through : Aquesta tècnica es molt semblant al wormhole explicat anteriorment, l'única diferencia és que el paquet sencer s'emmagatzema a memòria quan la capçalera no pot avançar.

Hi ha d'altres que només utilitzen un d'ells com son, circuit switching i ephimeral circuit switching que utilitzen conection-oriented :

- Circuit switching : Aquesta tècnica estableix un circuit o canal entre nodes i terminals abans que s'enviïn els paquets. Cada circuit no es pot utilitzar per altre paquets fins que no sigui alliberat i una nova connexió s'hagi creat. Als canals que no estan en un se'ls diu que estan es estat idle o d'espera.

- Ephimeral circuit switching : Aquesta tècnica podríem dir que es una evolució del circuit switching ja que a mesura que intenta establir la connexió entre l'origen i el destinació, també envia el paquet amb ella fent així que el paquet contingui l'adreça.

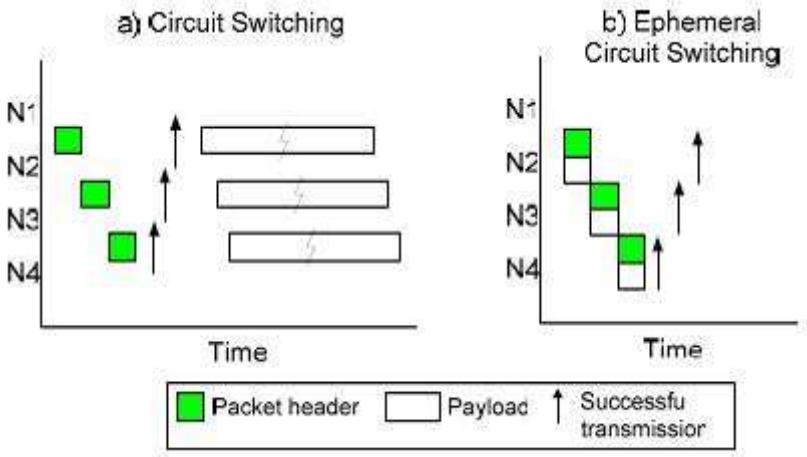


Figura 2.9 Comparació entre circuit switching i ephimeral circuit switching [Castells06]

Capítol 3

3 Algorismes d'encaminament

En aquest tercer capítol s'analitzaran els diferents algorismes d'encaminament tant els deterministes com els adaptatius.

Els Algorismes d'encaminament ens defineixen el camí a recorre per un paquet entre els *switches* d'origen i destinació. Han de poder prevenir el *deadlock* i el *livelock*. El Deadlock és una dependència entre nodes que requereixen accés a un conjunt de recursos determinats i no poden arribar-hi, no importa el conjunt d'estats que es duguin a terme que mai podran avançar. El livelock es refereix als paquets circulant per una xarxa que intenten arribar al seu destinació però mai hi poden arribar ja que són encaminats de manera cíclica. Els mètodes per evitar el deadlock i el livelock es poden aplicar localment a cada un dels nodes amb el suport de primitives o be globalment assegurant la separació lògica de les dades aplicant mecanismes de control end-to-end [MELLOt][Gaghan93].
Podem veure exemples de deadlock i livelock en la figura 3.1.

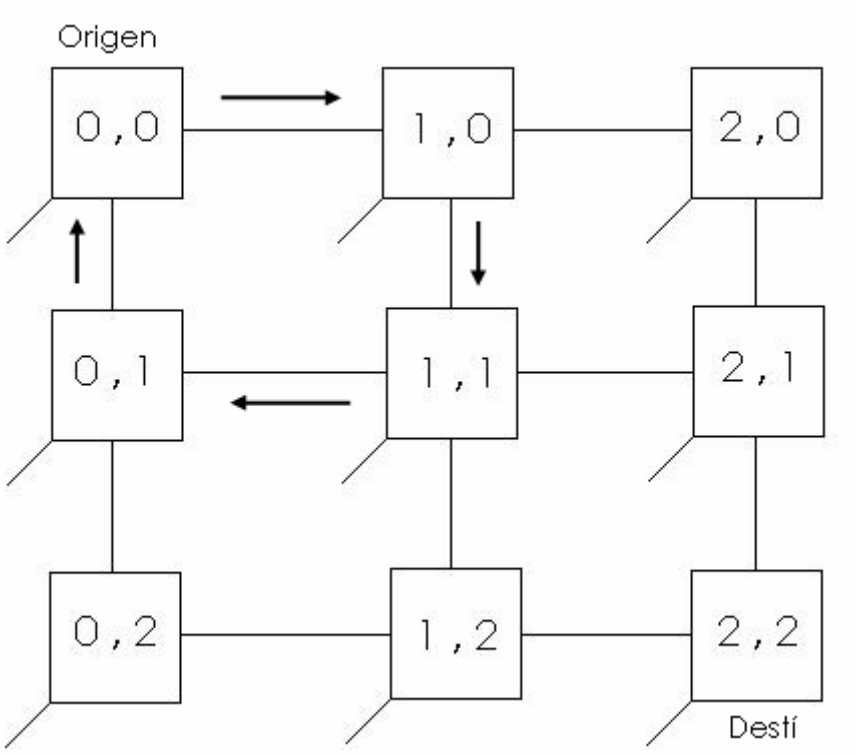


Figura 3.1 Exemple de deadlock i livelock

Com podem veure a la figura 3.1 el paquet mai arriba al seu destinació i queda voltant o fa cercles per tota la mesh.
Els algorismes d'encaminament es poden classificar conforme a tres criteris, per la longitud del camí, per on es prenen les decisions i per com es defineix el camí.

Si considerem el criteri de la longitud del camí, l'algorisme d'encaminament pot ser mínim o no-mínim. Un algorisme mínim es aquell que sempre escull el camí mes curt possible per arribar al seu destinació, si no es així es diu que es no-mínim.
Els algorismes mínims estan lliures de deadlocks i livelocks i entreguen el paquet amb una latència i un consum energètic baix, per contra es menys flexible quan el router esta congestionat.
Els algorismes no-mínims ofereixen una gran flexibilitat en termes de possibles camins alternatius quan el router està congestionat però poden arribar a estats de livelock i la latència per entregar el paquet augmenta.

Considerant el criteri de la presa de decisions, l'algorisme d'encaminament pot ser distribuït o centrat a l'origen. Pel que fa a l'algorisme centrat a l'origen, tot el camí es decideix al router origen, els paquets han de tenir una capçalera amb tota la informació d'encaminament incrementant així el tamany del paquet.
L'algorisme distribuït, cada router rep el paquet i decideix la direcció per enviar-lo en funció de les condicions del tràfic d'aquell instant per contra pot caure en camins erronis.

Com a últim criteri de encaminament, direm que un algorisme és determinista o adaptatiu segons com es defineix el camí.
Un algorisme determinista és aquell que la seva estratègia d'encaminament està determinada per l'origen i el destinació [MELLOt][Hu04a]. Per contra en una estratègia d'encaminament adaptativa o parcialment adaptativa [Gaghan93] , el camí es decideix en els diferents salts i intervenen uns mecanismes arbitraris dinàmics. Això fa que els algorismes adaptatius ofereixin millores respecte als deterministes però també guanyin en complexitat en termes d'implementació. Això es degut a que un algorisme adaptatiu pot caure en situacions de deadlock i livelock i tots dos problemes han d'estar solucionats per tal que siguin correctes [Schafer05][Hu04][Gaghan93].

Podem veure doncs en la següent figura 3.1, les possibles combinacions que es poden fer seguint els criteris exposats anteriorment i les tècniques de connexió explicades al punt 2.2.4, on s'explicava les tècniques conection-less i conection-oriented.

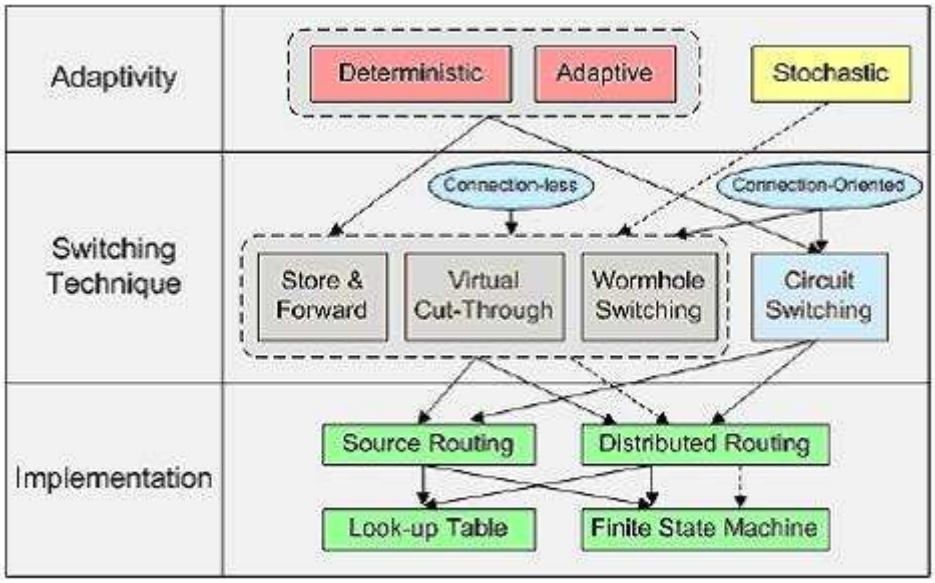


Figura 3.2 Combinacions possibles entre les tècniques de commutació , els algorismes de routing i les tècniques de connexió [Ogras05]

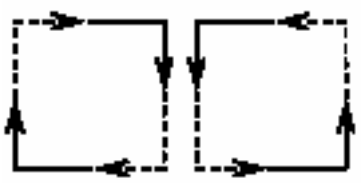
Veiem a la figura anterior com agafant un criteri d'encaminament, una tècnica de commutació i una implementació, obtenim un dels nostres algorismes de routing.

3.1 Algoritmes Deterministes

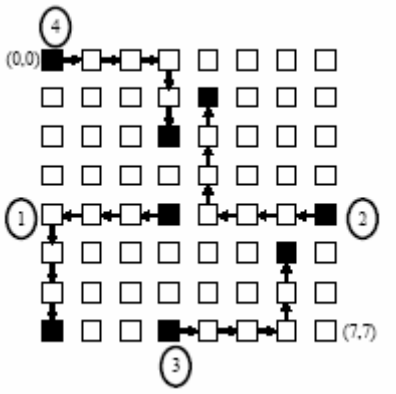
3.1.1 Algoritme X-Y

L'algorisme XY és un algorisme determinista, mínim i lliure de Ideadlocks i livelocks.

Els paquets son encaminats primer en la direcció de les X fins que arriben a la coordenada de destinació de les X (X_T) (veure figura 3.2b), on després són encaminats en la direcció de les Y fins que arriben a la coordenada de destinació de les Y (Y_T). Si cap de les direccions en les que ha d'encaminar esta sent utilitzada per un altre paquet, aquest romandrà bloquejat al router fins que el camí sigui alliberat. Per aquest algorisme, els girs on el paquet ve de la direcció de la coordenada Y estan prohibits[MELLOt].



a) Les línies continues representen els girs permesos. Només 4 girs estan permisos.



b) Exemples Algorisme XY

Fig 3.3 Algorisme XY [MELLOt]

Com podem observar en la figura anterior, en els camins 3(des de 3,7 fins a 6,5) i 4 (des de 0,0 fins a 3,2) s'encamina primer en direcció a l'est i després al eix de les Y, al nord o al sud respectivament, en canvi pels camins 1 (des de 3,4) fins a 0,7) i 2(des de 7,4 fins a 4,1) s'encamina primer cap a l'oest i després al sud o al nord.

3.1.2 Algoritme Y-X

L'algorisme YX es un algorisme determinista, mínim i lliure de Ideadlocks i livelocks.

Aquest algorisme és molt semblant al algorisme XY, l'única diferencia radica en que aquest encamina primer els paquets per l'eix de les Y fins al destinació Y_T i

tot seguit encamina els paquets per l'eix de les X fins que arriba al seu destinació XT. Per tant els camins que tindrà prohibits seran aquells on el gir ve de la direcció de la coordenada X.

3.2 Algoritmes Adaptatius

3.2.1 Algoritme West First

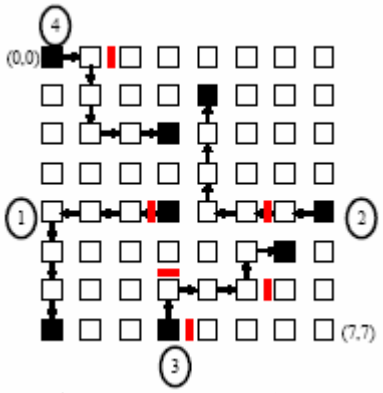
L'algorisme West First és un algorisme parcialment adaptatiu, mínim i distribuït. En aquest algorisme s'ha de fer una comprovació, si en la coordenada de la X, el destinació es mes petit o igual que l'origen els paquets s'encaminaran deterministament com a l'algorisme XY. Si per contra el destinació es mes gran que l'origen els paquets s'encaminaran adaptativament cap a l'est, nord o sud. El temps total per entregar un paquet es pot reduir utilitzant algoritmes adaptatius, ja que aquest pot fer girs en algunes situacions que poden evitar-li algunes condicions de bloqueig [MELLOt]. Es pot veure millor en la següent formula:

X(destinació) ≤ X (origen) → Forma determinista

X(destinació) ≥ X (origen) → Forma adaptativa

Així els dos encaminaments prohibits seran els dos que van cap a l'oest [MELLOt]

c) Els camins prohibits els dos cap al oest



b) Exemples Algorisme West First

Fig 3.4 Algorisme West First [MELLOt]

Com podem veure a la figura anterior, els camins bloquejats són aquells que tenen un línia en vermell, això significa que no podran passar per allà, així doncs tant en el camí 1 com en el 2 al complir-se la condició esmentada

anteriorment s'aplica l'algorisme determinista i per tant s'ha d'esperar a que s'alliberi el camí, per contra en els camins 3 i 4 al bloquejar-se els camins típics de l'algorisme XY l'encaminament s'adapta per altres camins fins arribar al seu destinació.

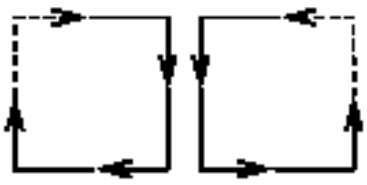
3.2.2 Algoritme North Last

L'algorisme North Last és un algorisme parcialment adaptatiu, mínim i distribuït. En aquest algorisme s'ha de fer una comprovació, si en la coordenada de la Y, el destinació és mes petit o igual que l'origen els paquets s'encaminaran deterministicament utilitzant l'algorisme XY. Si per contra el destinació es mes gran que l'origen els paquets s'encaminaran adaptativament cap al oest, est o sud. El temps total per entregar un paquet es pot reduir utilitzant algoritmes adaptatius, ja que aquest pot fer girs en algunes situacions que poden evitar-li algunes condicions de bloqueig [MELLOt]. Es pot veure millor en la següent formula:

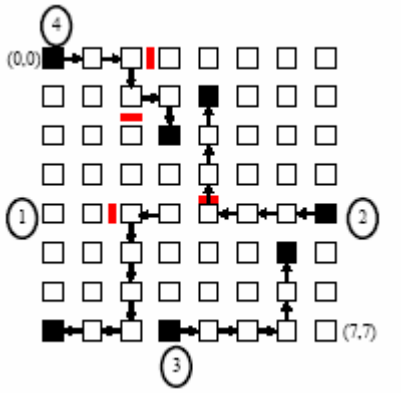
Y(destinació) ≤ Y (origen) → Forma determinista

Y(destinació) ≥ Y (origen) → Forma adaptativa

Així els dos encaminaments prohibits seran els dos que van cap al nord [MELLOt].



b) Els camins prohibits els dos cap al nord



b) Exemples Algorisme North Last

Fig 3.5 Algorisme North Last [MELLOt]

Com podem veure a la figura anterior, els camins on l'algorisme no adapta són el 3 i el 2, en canvi pels camins 2 i 4 el router adapta al estar bloquejats el seu camí natural i no complir la condició esmentada.

3.2.3 Algoritme Negative First

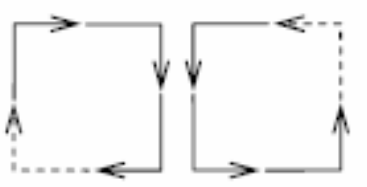
L'algorisme Negative First és un algorisme parcialment adaptatiu, mínim i distribuït.

En aquest algorisme els paquets són encaminats primer a les direccions negatives, com el nord o l'oest. Es fa una comprovació, si en la coordenada X el destinació es mes petit o igual que l'origen i en la coordenada Y el destinació és mes gran o igual que l'origen ó en la coordenada X el destinació és mes gran o igual que l'origen i en la coordenada Y el destinació es mes petit o igual que l'origen, els paquets s'encaminaran deterministicament utilitzant l'algorisme XY. Si per contra no es donen aquest casos, els paquets s'encaminaran de forma adaptativa. El temps total per entregar un paquet es pot reduir utilitzant algoritmes adaptatius, ja que aquest pot fer girs en algunes situacions que poden evitar-li algunes condicions de bloqueig [MELLOt]. Es pot veure millor en la següent formula:

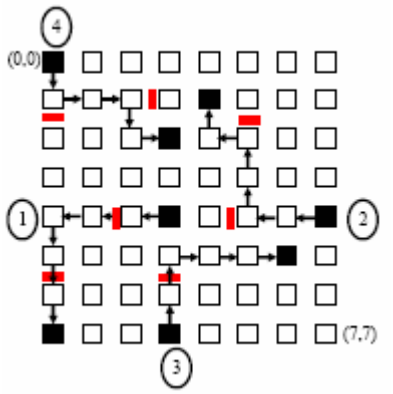
(X (destinació) ≤ X (origen) and Y (destinació) ≥ Y (origen)) or (X (destinació) ≥ X (origen) and Y (destinació) ≤ Y (origen)) → Forma determinista

(X (destinació) ≥ X (origen) and Y (destinació) ≤ Y (origen)) or (X (destinació) ≤ X (origen) and Y (destinació) ≥ Y (origen)) → Forma adaptativa

Així els dos encaminaments prohibits seran els que van d'una direcció positiva a una direcció negativa [MELLOt].



a) Els camins prohibits els dos de positiu a negatiu



b) Exemples Algorisme Negative first

Fig 3.6 Algorisme Negative First [MELLOt]

En aquesta figura podem veure com els camins 1 i 3 es compleixen les condicions esmentades en aquest algorisme i per tant s'encamina seguint l'algorisme XY i s'espera quan el camí està bloquejat, per contra als camins 2 i 4 l'algorisme s'adapta i quan el camí esta bloquejat canvia de direcció per arribar al seu destinació sense haver d'esperar a que s'alliberin.

Capítol 4

4 Disseny i metodologia del treball

En aquest quart capítol analitzarem ell disseny i la metodologia de treball que seguirem durant tot el projecte i les eines necessàries per a la realització del mateix.

Per aquest capítol, veurem les eines que hem utilitzat per el disseny i el desenvolupament del nostre projecte i el flux de treball que ens ha portat a la realització del mateix.

Pel que respecte a les eines emprades per la creació dels fitxers teníem la possibilitats d'utilitzar dos llenguatges descriptius de hardware, el verilog o el VHDL, mes endavant veurem les seves característiques i el criteri seguit per avaluar les dos opcions i escollir la que millor s'adequava a les nostres característiques. Per tal de crear aquest fitxers, es podrien fer en el format d'un text pla, on es descriuríem el comportament de cada un dels mòduls i després verificar el seu funcionament mitjançant les eines de verificació i síntesi com són el ModelSim i el Synplicity-Synplify Pro® respectivament, però el problema ve en quan és molt mes difícil observar les paraules claus i els errors comesos en un text pla que no pas fent servir el mateix ModelSim el qual té un editor de texts d'aquest llenguatges descriptius que fa la seva lectura mes agradable i fàcil d'entendre.

4.1 Etapes del disseny del treball

En la realització del nostre treball, partíem d'un router realitzat amb l'algorisme determinista XY, el qual havíem d'adaptar a les nostres necessitats de disseny.

Pel que respecte a les etapes en les qual s'ha realitzat el treball, o el que és el mateix el flux de treball que hem seguit durant la creació del projecte, se'ns diferencien clarament quatre etapes: etapa d'aprenentatge, etapa d'implementació, etapa de verificació i etapa d'obtenció de resultats.

En l'etapa d'aprenentatge s'han estudiat tant l'arquitectura de la NoC amb una topologia Mesh amb circuit switching com els diferents algorismes adaptatius i les seves característiques, a mes a mes s'ha après a utilitzar les diverses eines que es requerien per a la implementació i posterior verificació del treball.
En l'etapa d'implementació es va realitzar la creació dels tres algorismes adaptatius: West First, North Last i Negative First, i els seus respectius fitxers de

comprovació amb la seva posterior unió amb totes les parts que composen el mòdul global, per crear així el router amb cada un dels tres algorismes.

En l'etapa de verificació s'han realitzat totes les probes del treball realitzat en l'etapa anterior, s'han simulat tots els mòduls creats amb els seus respectius fitxers de comprovació o benchmarks amb l'eina ModelSim, per veure el seu correcte funcionament. Un cop aquest fitxers es comportaven correctament passàvem a utilitzar l'eina de sinterització el Synplicity-Synplify Pro® el qual ens permetia veure l'esquema RTL del nostre disseny.

En l'etapa d'obtenció de resultats, s'han extret els mateixos fent servir la informació obtinguda en l'etapa de verificació, així doncs s'ha pogut fer una comparativa entre els quatre algorismes simulats i sintetitzats, per extreuen diverses dades com són per exemple l'àrea de desenvolupament, les prestacions, etc.

4.2 Eines emprades

4.2.1 El Llenguatge de Descripció de Hardware: Verilog

En aquest punt, explicarem el llenguatge utilitzat per tal de definir el comportament dels mòduls creats. El llenguatge que utilitzarem és un llenguatge HDL (Hardware Description Language) el qual ens permet modelar el comportament de circuits digitals, aquest llenguatges hereten moltes característiques dels llenguatges de programació de software però tenen també certes diferencies com veurem a continuació [LluísTeres98].

Les principals similituds entre HDL i SDL son:

- Capacitat per descriure funcionalitat (tipus de dades i sentencies)
- Modularitat: estructuració de codi
- Us de biblioteques: reutilització de codi

Les principals diferencies entre HDL i SDL son:

- El software s'executa en un processador
- El Hardware es simula i/o es materialitza (síntesi)
- El Hardware requereix el maneig d'estructures, concurrències i temps.

Els dos llenguatges descriptius de hardware mes estesos son el VHDL i el Verilog, per tal de realitzar el nostre treball, hem d'escollir una de les dues alternatives. Degut a la complexitat de disseny d'ASICs i FPGAs han incrementat el numero de dissenyadors especialistes en una eina especifica, per tal de crear macros, llibreries i mega cel·les descrites en Verilog o en VHDL. Per tant es molt important que els dissenyadors coneguin tots dos llenguatges tant el Verilog com el VHDL, i coneguin també les Electronic Design Automation (EDA) que proporcionen les companyies, les quals poden funcionar normalment per ambdos llenguatges o fins i tot en alguns casos una mescla de tots dos.

Doncs procedirem a explicar les característiques d'ambdós llenguatges fent una comparativa general d'ells.

El llenguatge VHDL, va convertir-se en l'estàndard IEEE 1076 a l'any 1987. En el 1993 es va fer una revisió i es va actualitzar, tot i així s'han anat fent mes revisions de mica en mica fins a l'actualitat. D'una altra banda Verilog, és un llenguatge que ja es feia servir molt abans que el llenguatge VHDL, mes concretament, Gateway Design Automation el feia servir l'any 1984. Al 1989 Cadence Design Systems, va adquirir Gateway i amb ella els drets sobre Verilog, fent-lo de domini públic l'any 1990. Uns anys mes tard, es va formar una organització, Open Verilog International (OVI) , per tal de convertir, al 1995, el llenguatge en l'estàndard IEEE 1364.
Un llenguatge de descripció hardware ha de tenir una certa abstracció del comportament i estructura a nivell hardware. Això fa que l'estructura hardware ha de poder ser dissenyada independentment del disseny comportamental, al mateix temps que el comportament del hardware modelat no s'ha de veure perjudicat per el disseny estructural.

Pel que fa a la capacitat de modelatge d'estructures hardware, el dos llenguatges son suficientment bons perquè no tinguem problemes en poder triar un o l'altre. Així doncs farem la nostra elecció basant-se en les especificacions de màrqueting de les empreses o simplement per elecció personal. Tot i així es pot percebre alguna diferencia com podem veure a la següent figura.

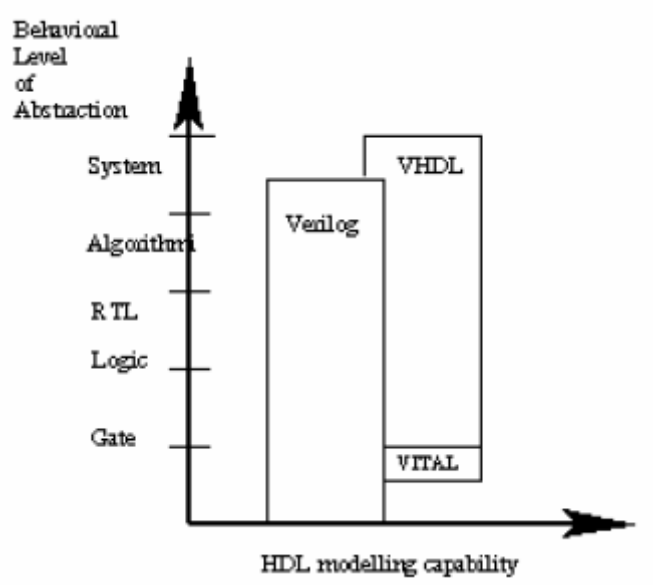


Figura 4.1: Capacitats de modelatge (VHDL i Verilog)

Com podem veure, Verilog dona certes facilitats alhora de definir nivells d'abstracció mes baixos, com a nivell de portes, i de l'altre banda VHDL ens permet descriure comportaments d'algorismes i nivell de sistema amb mes eficàcia. Però la diferencia es mínima.

Pel que respecte als tipus de dades, Verilog te uns tipus de dades senzills, definits especialment per al model hardware, això fa que les estructures siguin mes simples però mes concretes, en VHDL es al contrari, els tipus de dades son mes genèrics i fan que les estructures puguin ser definides per l'usuari, el que el fa molt mes genèric que el verilog però també li genera mes complexitat ja que s'ha de seleccionar de forma correcta els tipus de dades.

Pel que respecte al modelat d'alt nivell, en Verilog no podem fer packages com fem a VHDL sinó que l'unica eina que tenim a les nostres mans, es la de parametritzar els models mitjançant constants, en canvi VHDL te els packages

que li permet tenir una alta reusabilitat dels mòduls, podent esser utilitzats per configuracions o per models genèrics.

Per contra, al modelat de baix nivell, Verilog es molt mes competent ja que originalment va ser dissenyat per ser utilitzat a nivell de porta, podem modelar llibreries per ASICs o FPGAs sense cap problema, en canvi VHDL no te aquesta avantatge ja fa servir operadors simples i nomes pot definir timings fent servir *after*.

Si aprofundim a nivell de facilitat d'aprenentatge, verilog es segurament el mes senzill, ja que es un llenguatge mes compacte, però realment la diferencia entre tots dos es mínima.

Un cop havent vist les característiques d'aquest dos llenguatges i havent comparat ambdós en una sèrie de camps, ens podem decantar ja per un d'ells per a la realització del nostre projecte. Així doncs s'ha escollit Verilog pel fet que al mon empresarial, el verilog esta més estes que el VHDL. A més a més s'afegeix l'interes per aprendre una altre llenguatge descriptiu de hardware ja que el llenguatge VHDL es l'utilitzat en l'entorn universitari i així obtenim un requeriment mes pel mon laboral.

4.2.2 Eina per la simulació: Modelsim

El compilador i simulador ModelSim, es un simulador hardware que permet a partir de fitxers Verilog o VHDL generats, simular tot el sistema complet. Permet accedir a totes les senyals de cada mòdul que s'ha integrat en el sistema, i simular el seu comportament tal i com s'indicarà als fitxers de validació que aniran incorporats amb els mòduls a testejar. Aquest programa també ens serveix per compilar els fitxers amb el llenguatge verilog i es l'eina mes utilitzada pels dissenyadors d'avui en dia ja que és molt genèrica i la seva funcionalitat és molt amplia.

4.2.3 Eina per la sintesi: Synplicity-Synplify Pro®

El Synplicity-Synplify Pro® és una eina desenvolupada per Synplicity que hem utilitzat per sintetitzar els fitxers de Verilog. Aquesta eina ens mostra si l'arquitectura especificada és correcta. Per examinar l'arquitectura, aquesta eina ens mostra per pantalla un diagrama de blocs jeràrquic a partir dels fitxers Verilog. En el capítol 6 es poden observar captures d'aquest diagrames de blocs, representant una arquitectura determinada.

Capítol 5

5 Implementació del router adaptatiu

En aquest cinquè capítol explicarem el procés de implementació del nostre router, denotant les dues part mes importants del mateix, el mòdul d'encaminament i el mòdul de recepció/ transmissió de dades.

5.1 Disseny del hardware

Per a la implementació del nostre hardware, un router adaptatiu amb cadascun dels algorismes adaptatius explicats en el punt 3, partirem com ja hem dit al punt 4, d'un router amb algorisme determinista XY implementat per l'enginyer Jaume Joven.

Aquest router, com podem veure a la figura 5.1, es composa de 5 sortides /entrades de informació. Això es degut a que pot tenir un màxim de 4 veïns, nord, sud, est i oest i a mes esta connectat al seu node local. Aquestes direccions es composen del mòdul Path Switch Matrix, el qual fa les funcions d'encaminament i de transmissió de dades, com podem veure a la figura que representa el path del est, li entren les senyals request de les altres direccions, això es replicaria per cada una de les direccions que composen el router.

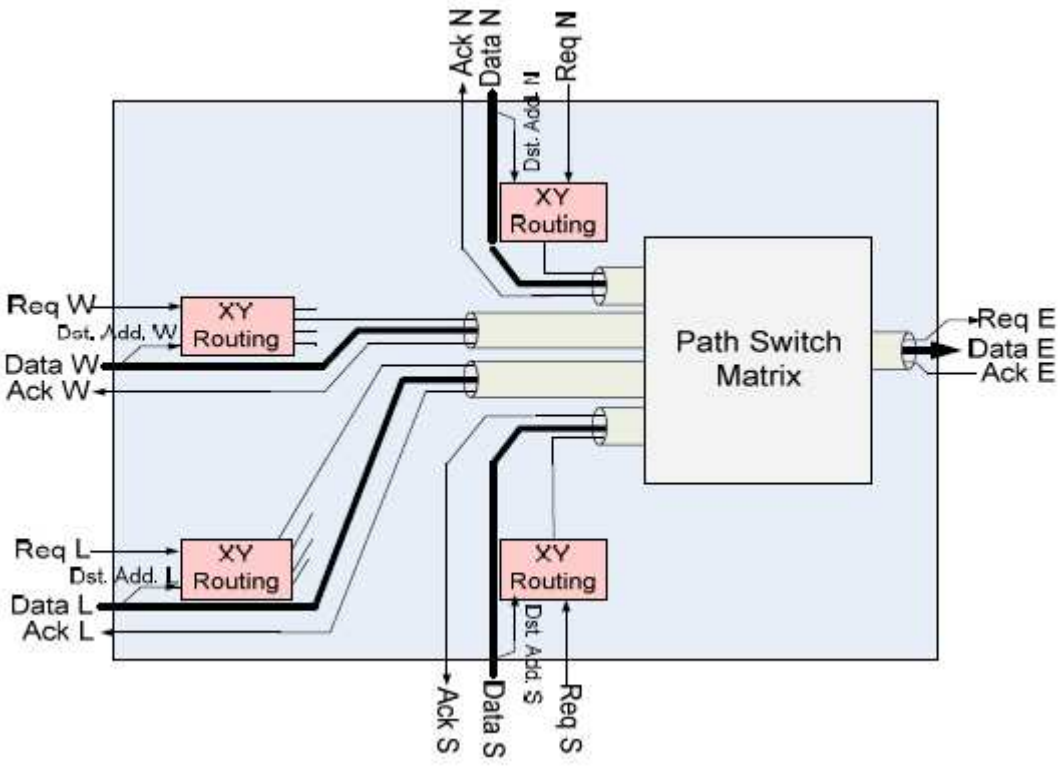


Figura 5.1 Router

Cada una d'aquestes direccions o Path Switch Matrix es composa de dos mòduls diferents, un mòdul que l'anomenarem d'encaminament (Mesh XY Routing) i un altre de commutació en la recepció/transmissió de dades (Mesh Switch Matrix).

Les senyals d'entrada i sortida d'aquest mòdul serveixen perquè es comuniqui amb els altres routers, així doncs cada path switch matrix estarà connectat amb un router diferent o ve amb el node al qual pertoca el router. Així doncs la comunicació de router funciona seguint les següents etapes i per fer-ho mes fàcil d'entendre ho farem amb un exemple, anirem des de l'origen 0,0 al destinació 1,1 fent servir l'algorisme XY:

1. El node origen activa la senyal request per tal de demanar entrada al path switch matrix on tots els path switch matrix.
2. El Mesh XY Routing rep les adreces i encamina segons l'algorisme implementat a un dels path switch matrix (en el nostre exemple al est).
3. El path switch matrix on s'ha encaminat el Mesh XY routing (el del est) activa la senyal d'aquella direcció, ja que es a on anirà i passa les senyals de dades.
4. Aquest path switch matrix envia la senyal de request i les dades al path switch matrix de l'altre router (el del oest) i espera respostes.
5. Aquest farà els passos 2, 3 i 4 anteriors fins arribar al node destinació(en l'exemple encaminarà al sud i activarà la direcció del sud i després encaminarà cap al local perquè haurà arribar al destinació).
6. Un cop ens han arribat les dades al node, aquest enviarà la senyal Acknowledge la qual s'anirà retransmeten fins al router origen (0,0) que podrà o be enviar el següent paquet o be finalitzar la connexió.

A la següent figura podrem veure com avança el paquet en la nostra mesh.

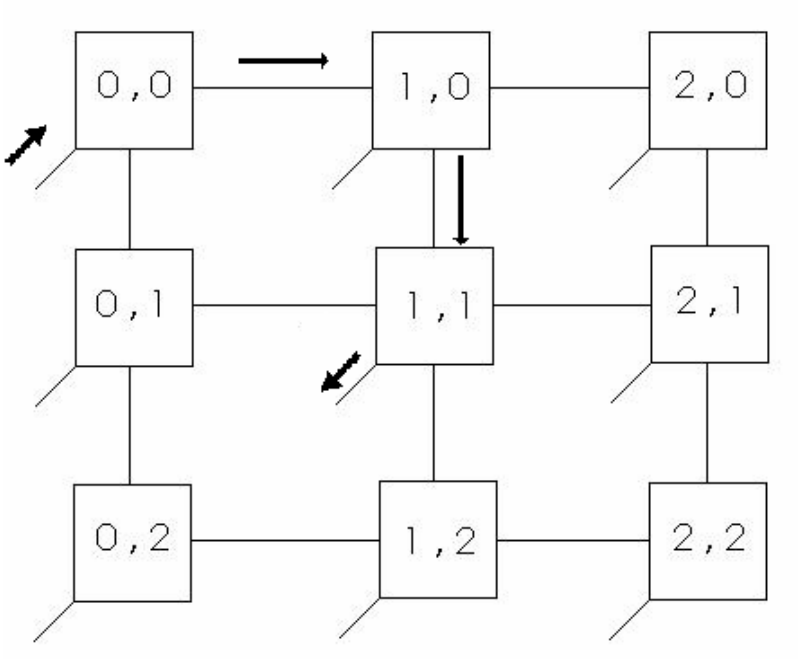


Figura 5.2 Exemple d'encaminament des de l'origen fins al destinació

Així doncs, un cop hem vist el funcionament del router, introduirem el format del paquet, que com ja hem vist al punt 2, al ser una conection-oriented amb ephimeral circuit switching, el paquet tindrà una part que s'utilitzarà per poder fer l'encaminament i l'altre part que serà el cos del paquet i es on anirà la informació a transmetre.

Així doncs el paquet estarà format per 3 parts diferenciades, la direcció d'origen, la direcció de destinació i el cos o body del missatge.

- Direcció d'origen: aquesta està dividida en dos, la direcció de les X i la direcció de les Y, com podem veure a la figura 5.4, ens indica l'origen del paquet, en l'exemple anterior seria el 0, 0.
- Direcció de destinació: aquesta esta dividida en dos parts que corresponen a cada una de les coordenades, ens indica la direcció de destinació del paquet i es la informació que s'utilitza per tal d'encaminar el missatge, en l'exemple anterior correspondria al 1, 1.
- El cos o body: es la part mes important del missatge ja que són les dades que es volen transmetre des de l'origen fins al destinació.

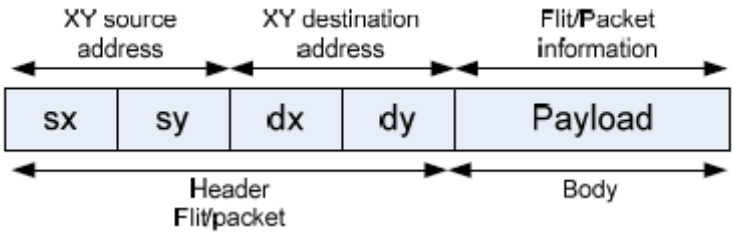


Figura 5.3 Format d'un paquet.

En l'ephimeral circuit switching els paquets no tenen una mida fixada sinó que com mes gran sigui la mesh, mes gran serà el tamany del paquet.
Partim que el body del paquet té un tamany fixat, aquesta part del paquet no canvia mai de tamany sempre són 32 bits, un tamany estàndard del paquet. Així doncs la part del paquet que canvia de tamany son les direccions. Per la mesh mes senzilla de totes, una mesh de 2x2 sabem que tenim un tamany de direcció per coordenada d'un bit. Així doncs el tamany mínim d'un paquet es de 36 bits. Al anar augmentant el numero de routers que conformen la mesh, hem d'anar augmentant el numero de bits que disposem per tal d'identificar tots els routers que la composen, així el numero de bits de direcció per cada coordenada augmenta i això fa augmentar la mida del paquet.

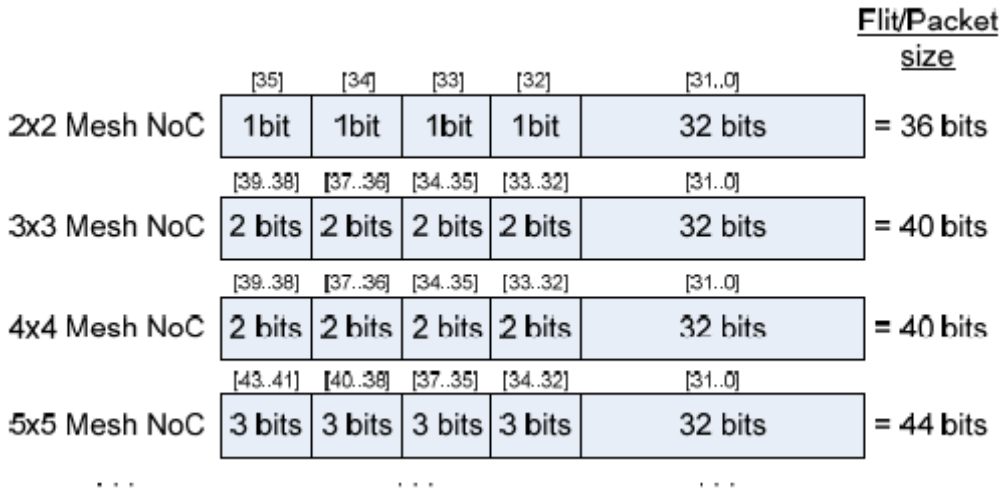


Figura 5.4 Tipus de paquets segons la mesh

Com podem veure a la figura 5.5 el tamany de paquet va augmentant considerablement a mesura que augmentem el numero de routers en la mesh, això fa que l'escalabilitat amb aquest tipus de formats de paquets sigui molt costos en termes de tamany de paquet.

La comunicació entre router es realitza a traves d'un protocol handshake de 4 fases no ambigu el qual recordarem que ens permet enviar les dades sabent si les hem rebudes o no. Funciona de forma general seguint els passos següents:

- Primer es realitza un request i es proporcionen les dades
- El receptor rep el request i les dades i envia un acknowledge
- El transmissor rep l'acknowledge i baixa la senyal request
- El receptor al baixar el transmissor la senyal request baixa també la senyal acknowledge.

Pel nostre projecte el mòdul de recepció/transmissió (Mesh Switch Matrix) de dades no l'hem de tocar, ja que el funcionament es el mateix tant per un algorisme determinista com per un algorisme adaptatiu o parcialment adaptatiu com els que hem d'implementar. Així doncs ens centrarem en el mòdul d'encaminament (Mesh XY Routing), tot i que també explicarem l'altre mòdul.

5.1.1 Mòdul d'encaminament

El mòdul d'encaminament es composa com podem veure a la figura 5.5, de dos mòduls comparadors, un per l'eix de les X i l'altre per l'eix de les Y, i una certa lògica addicional que ens servirà per encaminar el paquet cap a una direcció o cap a una altre.

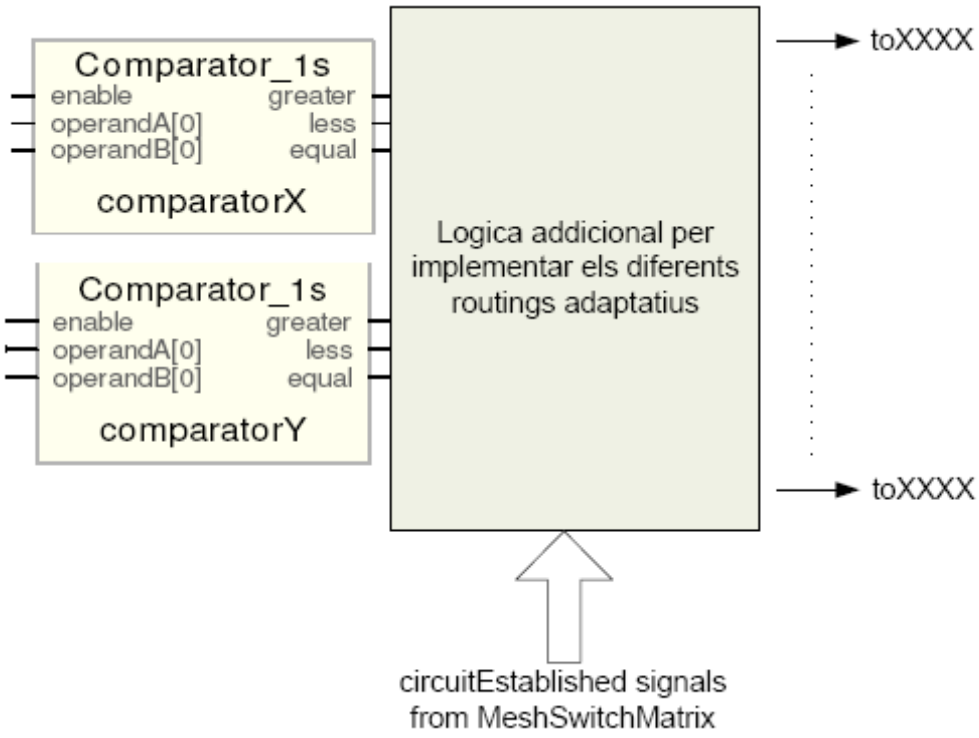


Figura 5.5 Disseny del mòdul d'encaminament

Els comparadors ens proporcionen 3 senyals, greater, equal i less, segons les direccions d'entrada que li proporciona el paquet rebut.

- Greater : S'activa quan la direcció de destinació es mes gran que la direcció del router.

- Equal : S'activa quan la direcció de destinació es igual a la direcció del router.

- Less : S'activa quan la direcció de destinació es mes petita que la direcció del router.

Aquestes senyals de sortida que ens proporcionen els comparadors, ens donen la posició cap a on hem de continuar l'encaminament ja que si recordem el punt 2.2.3 on s'explicava la topologia mesh i seguint l'ordre de les matrius que diu que el 0,0 esta al punt mes al nord-oest de la mesh 2D, podem saber cap a on s' encaminarà segons l'algorisme escollit. Així doncs per exemple, si s'activa la senyal greaterX i la senyal greaterY sabrem que el router de destinació esta

al sud-est del nostre router i actuarem en conseqüència amb el nostre algorisme.

Per tal d'incorporar els algorismes adaptatius, s'han d'incorporar a la lògica addicional del mateix. Disposem per aquest mòdul de 11 senyals, les senyals que ens arriben dels comparadors (greater, equal, less) i les senyals que ens arriben de les altres direccions i que ens indiquen si aquestes estan ocupades (1 lógic) o no (northBusy, eastBusy, westBusy, southBusy, localBusy). Amb aquestes senyals d'entrada s'ha creat una taula de la veritat que verifica el comportament de cadascun dels algorismes i que ens dona com a resultat les senyals de sortida per saber cap a on encaminarà el router (toNorth, toEast, toWest, toSouth, toLocal).

Un cop tenim clares les senyals utilitzades, i la topologia on s'implementaran aquests algorismes, hem de tenir en compte les consideracions exposades en el punt 3, on dèiem que un algorisme adaptatiu o parcialment adaptatiu ha d'estar lliure de deadlock i livelock. En una mesh 2D tenim 8 possibles girs, com podem veure a la figura 5.6, així doncs per tal que els algorismes estiguin lliures d'aquestes situacions, hem de evitar almenys dos girs d'aquest vuit possibles [MELLOt].

Figura 5.6 Tots els camins possibles i possibles camins a evitar [MELLOt]

Així doncs seguint les explicacions proporcionades al punt 3, on expliquem el funcionament dels algorismes adaptatius utilitzats en la implementació del nostre router, hem fet les taules de la veritat que es mostren a les figures següents per tal de reduir les possibilitats en les possibles combinacions, ja que si no féssim aquesta reducció tindríem 2^{11} possibles combinacions, això són 2048 combinacions que farien molt difícil la seva implementació.
Com podrem recordar en l'algorisme West First, s'encamina deterministicament utilitzant l'algorisme XY quan la coordenada X de destinació és mes petita o igual que la coordenada del router on estem situats,

en els altres casos es fa adaptativament, així doncs la taula de la veritat queda com podem veure a la figura 5.7.

	toWest	toSouth	toSouth	toEast	toNorth	toNorth	toLocal
NorthBusy	x	x	x	x	0	0	x
SouthBusy	x	0	0	x	x	x	x
EastBusy	x	x	1	0	x	1	x
WestBusy	0	x	x	x	x	x	x
LocalBusy	x	x	x	x	x	x	0
X Greater	0	0	1	1	0	1	0
X Equal	0	1	0	0	1	0	1
X Less	1	0	0	0	0	0	0
Y Greater	x	1	1	x	0	0	0
Y Equal	x	0	0	x	0	0	1
Y Less	x	0	0	x	1	1	0

X GREATER					**X EQUAL**			**X LESS**		
Y GREATER		**Y EQUAL**	**Y LESS**		**Y GREATER**	**Y EQUAL**	**Y LESS**	**Y GREATER**	**Y EQUAL**	**Y LESS**
eastBusy	!eastBusy	eastBusy	eastBusy	!eastBusy	southBusy	localBusy	northBusy	westBusy	westBusy	westBusy
toEast	southBusy	toEast	toEast	nortBusy	toSouth	toLocal	toNorth	toWest	toWest	toWest
	toSouth			toNorth						

Figura 5.7 Taula de la veritat algorisme West First i arbre de decisions

En l'algorisme North Last, utilitzem l'encaminament determinista XY quan la coordenada de destinació de l'eix de les Y és mes petita o igual i per contra quan no sigui aquest cas ho farem de forma adaptativa, així doncs la taula de la veritat quedarà segons la figura 5.8

	toWest	toSouth	toSouth	toSouth	toEast	toNorth	toLocal
NorthBusy	x	x	x	x	x	0	x
SouthBusy	x	0	0	0	x	x	x
EastBusy	x	x	1	x	0	x	x
WestBusy	0	x	x	1	x	x	x
LocalBusy	x	x	x	x	x	x	0
X Greater	0	0	1	0	1	0	0
X Equal	0	1	0	0	0	1	1
X Less	1	0	0	1	0	0	0
Y Greater	x	1	1	1	x	0	0
Y Equal	x	0	0	0	x	0	1
Y Less	x	0	0	0	x	1	0

Y GREATER					Y EQUAL			Y LESS		
X GREATER		X EQUAL	X LESS		X GREATER	X EQUAL	X LESS	X GREATER	X EQUAL	X LESS
eastBusy	!eastBusy	southBusy	westBusy	!westBusy	eastBusy	localBusy	westBusy	eastBusy	northBusy	westBusy
toEast	southBusy	toSouth	towest	toSouth	toEast	toLocal	toWest	toEast	toNorth	toWest
	toSouth									

Figura 5.8 Taula de la veritat algorisme North Last i arbre de decisions

En l'algorisme Negative First, encaminarem de forma determinista i utilitzant l'algorisme XY quan es compleixin un d'aquest dos casos, be la coordenada de les X és mes petita o igual i la coordenada de les Y es mes gran, o be quan la coordenada de les X és mes gran i la coordenada de les Y es mes petita o igual, per altres casos utilitzarem un encaminament adaptatiu, com podem veure a la taula de la veritat de la figura 5.9

	toWest	toSouth	toSouth	toEast	toNorth	toNorth	toLocal
NorthBusy	x	x	x	x	0	0	x
SouthBusy	x	0	0	x	x	x	x
EastBusy	x	x	1	0	x	x	x
WestBusy	0	x	x	x	x	1	x
LocalBusy	x	x	x	x	x	x	0
X Greater	0	0	1	1	0	0	0
X Equal	0	1	0	0	1	0	1
X Less	1	0	0	0	0	1	0
Y Greater	x	1	1	x	0	0	0
Y Equal	x	0	0	x	0	0	1
Y Less	x	0	0	x	1	1	0

X GREATER				X EQUAL			X LESS			
	Y GREATER	Y EQUAL	Y LESS	Y GREATER	Y EQUAL	Y LESS	Y GREATER	Y EQUAL	Y LESS	
eastBusy	!eastBusy	eastBusy	eastBusy	southBusy	localBusy	northBusy	westBusy	westBusy	westBusy	!westBusy
toEast	southBusy	toEast	toEast	toSouth	toLocal	toNorth	toWest	toWest	toWest	northBusy
	toSouth									toNorth

Figura 5.9 Taula de la veritat algorisme Negative First i arbre de decisions

Un cop tenim les taules de la veritat acabades només hem de dissenyar utilitzant el Verilog perquè faci el que la nostra taula de la veritat disposa, així doncs només tenint en compte que només podrem encaminar sempre i quan tinguem la senyal enable activa ja que ens la que ens indica si les nostres

dades a l'adreça són les correctes i quan hi ha una petició (request) per tal de poder encaminar a la direcció que pertoqui.

Un cop dissenyat l'algorisme d'encaminament, s'ha de verificar el seu correcte funcionament i per tant hem de crear testbenchs o fitxers d'estímuls que intentin simular el comportament de l'algorisme. Utilitzant una de les eines esmentades anteriorment, el Modelsim, verifiquem el funcionament dels fitxers a base de waveforms els quals ens donen els valors de les senyals de sortida davant els estímuls que li introduïm a traves de les senyals d'entrada. Això ens permet acabar de testejar els fitxers i saber que el seu funcionament davant totes les situacions serà l'adequat.

5.1.2 Modul recepcio/transmissio de dades

El mòdul de transmissió/recepció de dades o Mesh Switch Matrix, és el que escull la direcció, que té un request actiu, que passarà de la seva direcció tot seguint un procés. Aquest mòdul es composa de 4 mòduls diferents:

- Priority Encoder : Aquest mòdul activa la direcció amb major prioritat de totes, les prioritats venen donades segons les direccions d'un rellotge. Això és nord, est, sud, oest i com a menys prioritari el local.

- Register with Clear : Aquest mòdul, guarda al seu registre la direcció que està activa durant l'enviament de les dades.

- Mux4to1 : Selecciona una de les quatre senyals d'entrada a partir del codificador de prioritats.

- DeMux1to4 : activa una senyal de sortida en base a les 4 d'entrada a partir de l'entrada del codificador de prioritats.

Aquest són els quatre mòduls bàsics que composen el Mesh Switch Matrix, a part de la lògica que hi ha dintre. Aquest mòdul és el responsable de l'elecció de deixar passar la direcció que te prioritat mes alta, així doncs tal i com hem

explicat anteriorment, el mòdul priority encoder rep les senyals de request que provenen del mòdul Mesh XY Routing de les direccions que volen encaminar cap a ell i decideix quina ha de passar. Un cop la senyal esta establerta, passa la seva elecció als altres mòduls fent així que el Mux enviï la data i permeten al mòdul register guardar la direcció d'on ha vingut la senyal perquè el demux pugui enviar la senyal d'acknowledge. Aquí es on es fa la comunicació handshake comentada anteriorment.

Capítol 6

6 Resultats Experimentals

En aquest sisè capítol ens centrarem en els resultats del projecte, exposant tant els models RTL dels algorismes, com algunes captures gràfiques del funcionament dels mateixos, així com una comparativa entre ells.

Un cop s'han implementat i testejat els mòduls corresponents al routing adaptatius per NoCs. Podem extreure'n resultats per a les posteriors conclusions. Així doncs, per cada un dels punts següents farem una comparativa entre els tres algorismes adaptatius implementats i l'algorisme base XY.

Primerament, compararem els algoritmes en base al seus respectius esquemes RTL, aquests han estat extrets de l'eina esmentada anteriorment en el punt 4.2.3 Synplicity-Synplify Pro®, la qual ens donava un esquema de com quedarien els mòduls sintetitzats, un cop els algorismes han estat sintetitzats per la mateixa, els quals podrem veure a les següents figures: algorisme XY (figura 6.1), algorisme West First (figura 6.2), algorisme North Last (figura 6.3) i algorisme Negative First (figura 6.4).

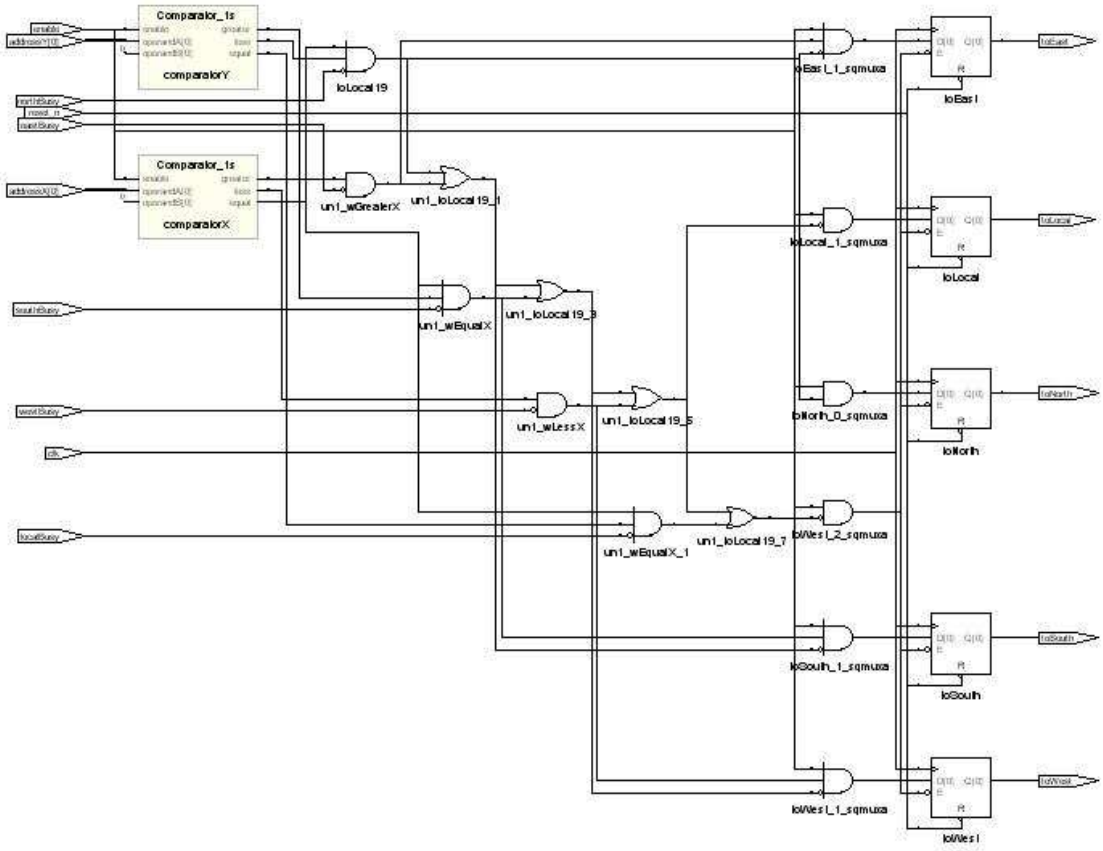


Figura 6.1 Esquema RTL de l'algorisme XY

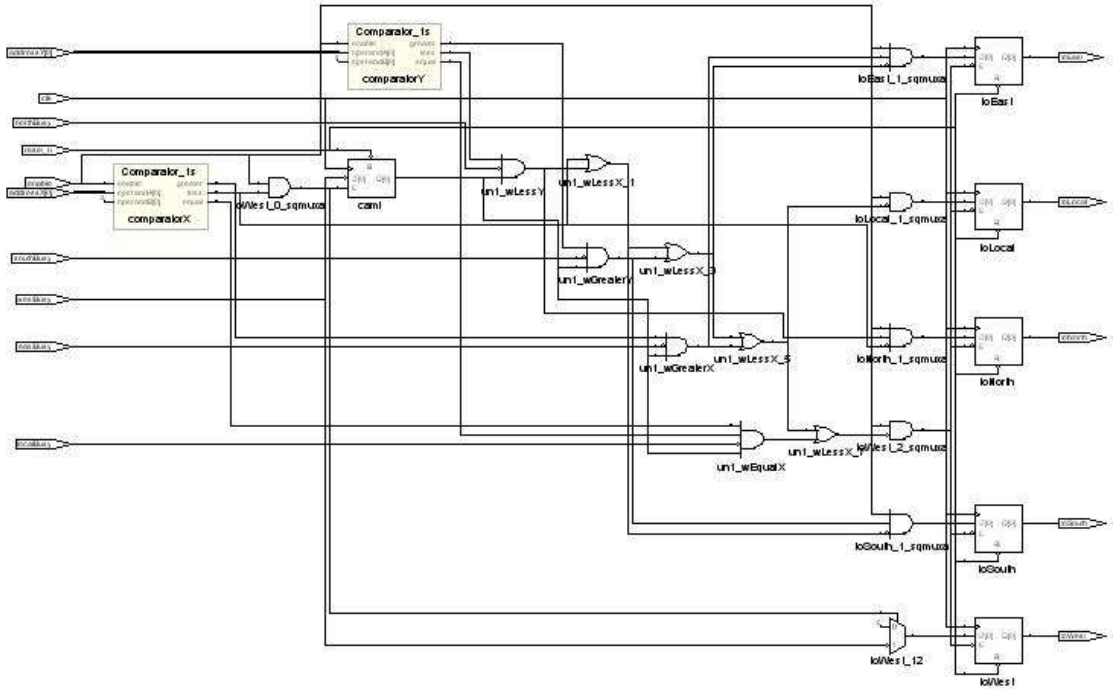


Figura 6.2 Esquema RTL de l'algorisme West First

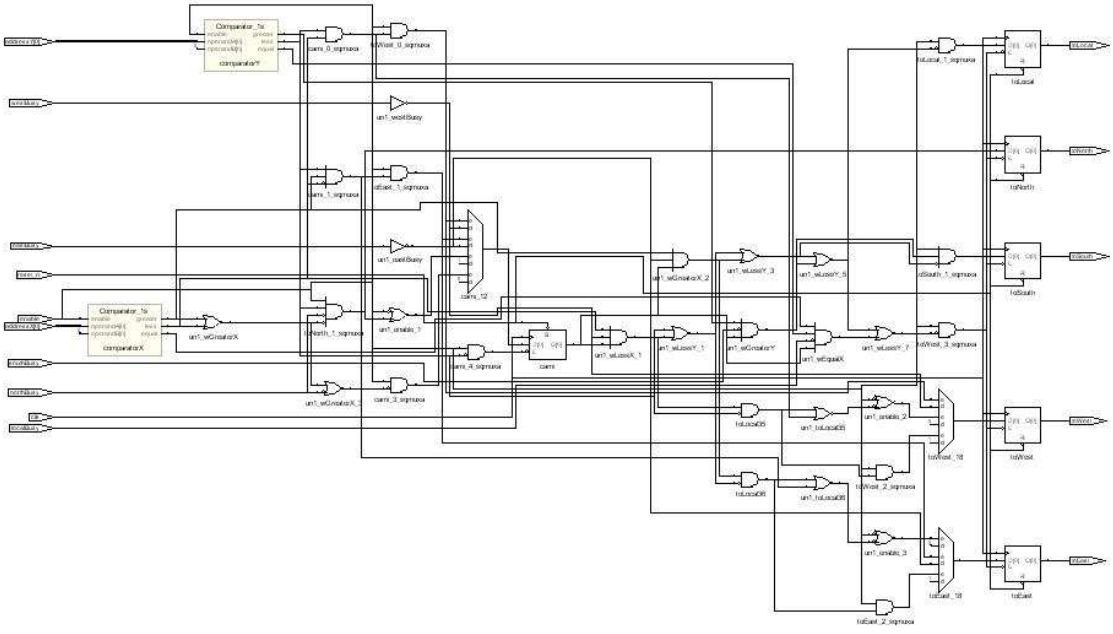


Figura 6.3 Esquema RTL de l'algorisme North Last

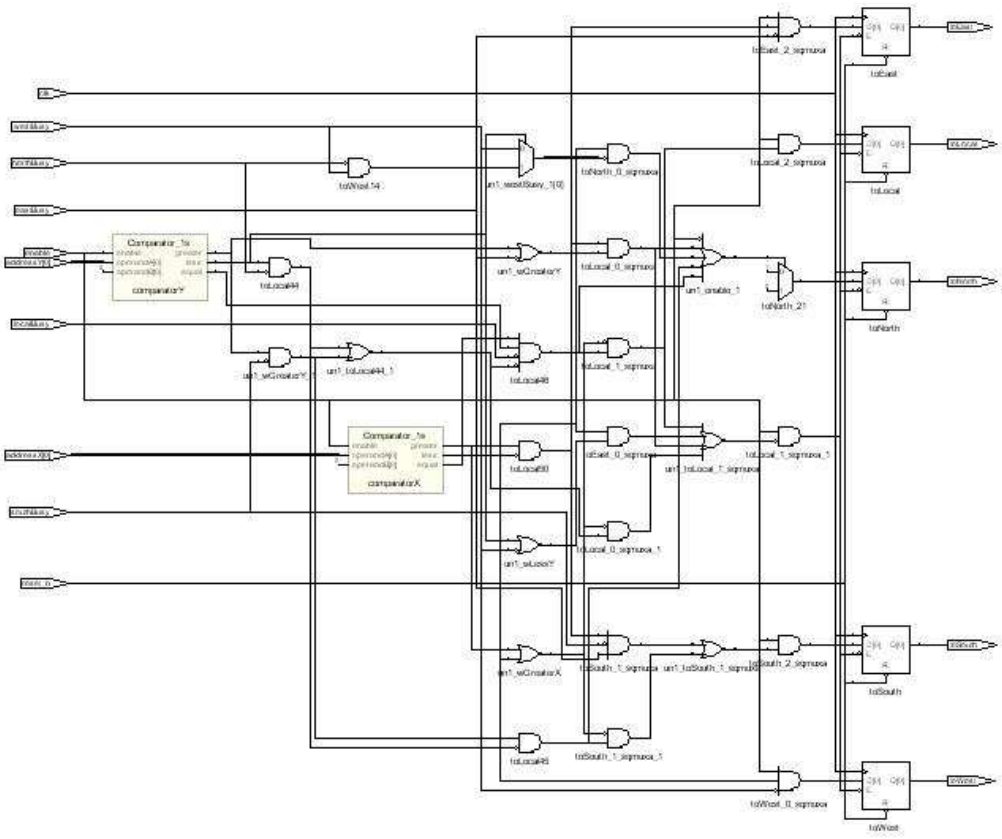


Figura 6.4 Esquema RTL de l'algorisme Negative First

Com ja havíem comentat en el punt anterior, aquest algorismes es basen tots en un senzill esquema, on hi trobem dos comparadors, per donant-se les senyals de greater, equal i less, que ens serviran perquè la lògica combinacional decideixi quina direcció triar en funció de l'algorisme i finalment les sortides registrades.

Com podem veure en les figures anteriors, l'algorisme XY es l'algorisme mes senzill de tots, ja que es un algorisme determinista i molt simple però també es un dels mes eficaços. Comparant els esquemes RTL globalment, veiem que la complexitat de disseny de la lògica combinacional va variant en funció de l'algorisme que dissenyem. Així podem veure com per exemple la figura 6.4 i 6.3 la complexitat de la lògica combinacional augmenta considerablement, fent difícil l'observació de tot el disseny.

6.1 Comparatives entre algorismes

En aquest apartat es comparará amb mes profunditat tots tres algorismes adaptatius, juntament amb l'algorisme determinista XY. Per tal de obtenir dades per poder fer aquesta comparació utilitzarem l'eina Synplicity-Synplify Pro®, la qual després de fer l'anàlisi del diferents tipus de router, un per cadascun dels algorismes esmentats anteriorment, ens proporciona la suficient informació, per poder fer comparatives entre ells.

Per tal de fer l'anàlisi amb aquesta eina, hem hagut d'escollir una FPGA (Field Programable Gate Array) on simular l'estructura dels nostres routers. L'FPGA escollida es una d'Altera, mes concretament l'Altera Stratix EP1S25, aquesta placa la podem veure a la figura 6.5, ens dona un nombre total de 25660 LEs (Logic Elements) on podem sintetitzar el nostre router.

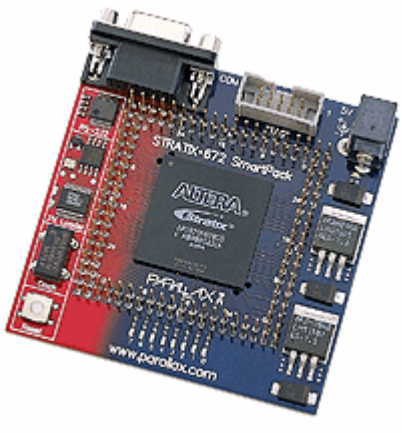


Figura 6.5 Altera Statrix EP1S25

Un cop tenim sintetitzat el nostre router, obtenim una sèrie de fitxers en els quals ens proporcionen un esquema RTL del tot el router amb els mòduls dels que es composa, el qual no il·lustrarem ja que es massa extens i no es distingiria gairebé res. L'altre fitxer important que ens proporciona aquesta eina, es un log, en el qual obtenim per cadascun dels nostres algorismes el numero de LUT's, l'àrea de desenvolupament, etc.

6.1.1 Costos (LEs-LUTs)

El primer paràmetre que observarem en aquest fitxers o logs seran els LEs-LUTs o LUT's, aquests ens indiquen el tamany que ocupa el mòdul, en el nostre cas el router, dintre de la FPGA. A continuació en la següent figura veurem un grafic dels valors en logic elements de cadascun dels routers.

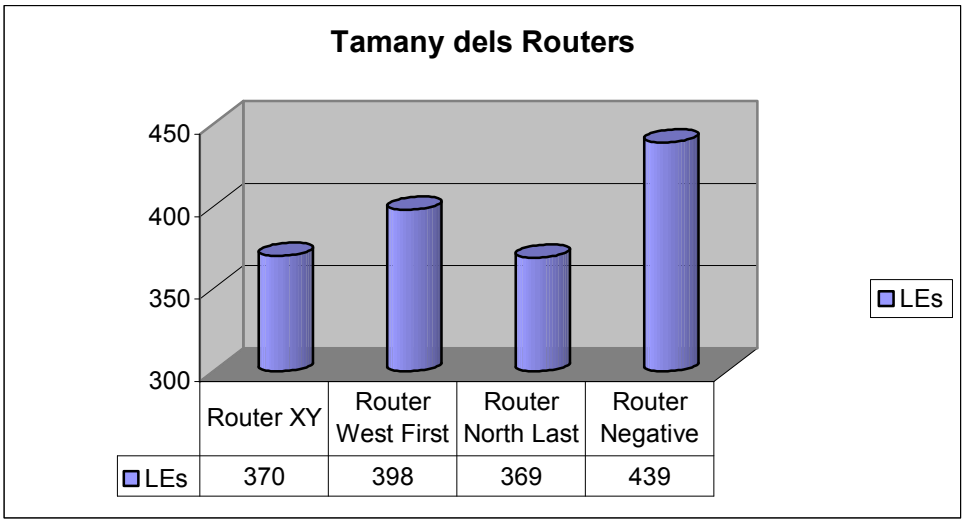


Figura 6.6 Tamany dels diferents routers en LEs

Com podem comprovar el tamany en LEs no varia gaire entre routers, l'únic que es desvia una mica mes d'aquesta afirmació es el router que implementa l'algorisme Negative First, el qual supera en mes de 60 LEs a l'algorisme base XY, respecte els altres dos algorismes, no varia tant aquesta diferencia en el West First son 28 LEs de diferencia i fins i tot a l'algorisme North Last en tenim una de menys respecte l'algorisme XY. Això ens indica una clara similitud entre algorismes ja que es cert que una part de cadascun dels algorismes parcialment adaptatius es fa utilitzant l'algorisme determinista XY, perquè si recordem les explicacions dels algorismes sempre i quan es compleixen les condicions donades, l'algorisme actua d'aquesta manera.

Pel que respecte als fitxers de log en la següent figura observarem un exemple, on es poden veure els LEs utilitzats per combinacional i els utilitzats per seqüencial en cadascun dels algorismes.

```
--------------------------------------------------------------------------------
########   Utilization report for  Top level view:   MeshRouterWestFirst   ########
================================================================================

SEQUENTIAL ATOMS
****************

Name          Total elements     Utilization     Notes
-----------------------------------------------------
REGISTERS     221                100 %
=====================================================
Total SEQUENTIAL ATOMS in the block MeshRouterWestFirst:        221 (27.42 % Utilization)

COMBINATIONAL ATOMS
*******************

Name                Total elements     Utilization     Notes
-----------------------------------------------------------
ATOMS               177                100 %
ARITHMETIC MODE     0                  0.0 %
===========================================================
Total COMBINATIONAL ATOMS in the block MeshRouterWestFirst:     177 (21.96 % Utilization)
```

Figura 6.7 Algorisme West First en LEs

```
--------------------------------------------------------------------------------
########   Utilization report for  Top level view:   MeshRouterNorthLast   ########
================================================================================

SEQUENTIAL ATOMS
****************

Name          Total elements     Utilization     Notes
-----------------------------------------------------
REGISTERS     221                100 %
=====================================================
Total SEQUENTIAL ATOMS in the block MeshRouterNorthLast:        221 (28.44 % Utilization)

COMBINATIONAL ATOMS
*******************

Name                Total elements     Utilization     Notes
-----------------------------------------------------------
ATOMS               148                100 %
ARITHMETIC MODE     0                  0.0 %
===========================================================
Total COMBINATIONAL ATOMS in the block MeshRouterNorthLast:     148 (19.05 % Utilization)
```

Figura 6.8 Algorisme North Last en LEs

```
--------------------------------------------------------------------------------
########   Utilization report for  Top level view:   MeshRouterNegativeFirst   ########
================================================================================

SEQUENTIAL ATOMS
****************

Name          Total elements     Utilization     Notes
-----------------------------------------------------
REGISTERS     221                100 %
=====================================================
Total SEQUENTIAL ATOMS in the block MeshRouterNegativeFirst:    221 (26.25 % Utilization)

COMBINATIONAL ATOMS
*******************

Name                Total elements     Utilization     Notes
-------------------------------------------------------------
ATOMS               218                100 %
ARITHMETIC MODE     0                  0.0 %
=============================================================
Total COMBINATIONAL ATOMS in the block MeshRouterNegativeFirst: 218 (25.89 % Utilization)
```

Figura 6.9 Algorisme Negative First en LEs

6.1.2 Area de desenvolupament

Per a l'àrea de desenvolupament del algorismes, ens centrarem com en el punt anterior en l'anàlisi del fitxer log de cadascun dels algorismes però aquest cop es mesura l'àrea que ocupa cadascun dels algorismes dintre de la FPGA, així doncs el tamany en tan per cent dintre de la NoC ens ajudarà a conèixer la distribució de cadascun del routers i ens facilitarà una mica mes l'anàlisi sobre ells. El nivell d'ocupació de cadascun dels routers és molt petit ja que es calcula observant el numero de LEs que ocupa cada router i dividint-lo pel numero de LEs total del que disposem que com hem dit anteriorment per el tipus de FPGA en la qual estem fent els càlculs es de 25660. Ja que tots els routers tenen un tamany molt similar l'àrea de desenvolupament de cadascun d'ells ronda el 1%, aquest tamany es molt significatiu perquè ens permet obtenir un nivell d'escalabilitat molt alt encara que no sigui real ja que si escaléssim els nostres routers hauríem d'augmentar el numero de bits que hem de fer servir per les direccions, com ja hem vist al punt 4 i per tant tots els mòdul haurien d'augmentar el seu tamany, fent d'aquesta manera que també augmentin els LEs i en conseqüència la nostra area de desenvolupament.

6.1.3 Freqüència de Rellotge Màxima

En aquest punt, analitzarem la freqüencia màxima a la que pot anar cadascun dels routers implementats i les compararem amb el XY, aquesta dada ens

servirà per saber a quina freqüència anirà tot el router i aquesta depèn de cadascun dels seus mòduls. Tal i com estàvem fent amb els anteriors punts, per tal de comprovar la freqüència màxima del rellotge de cadascun dels algorismes hem d'observar el log extret de l'eina Synplicity-Synplify Pro® el qual ens aporta aquesta informació en un dels seus apartats. Aquestes freqüències les podem veure al gràfic de la figura següent:

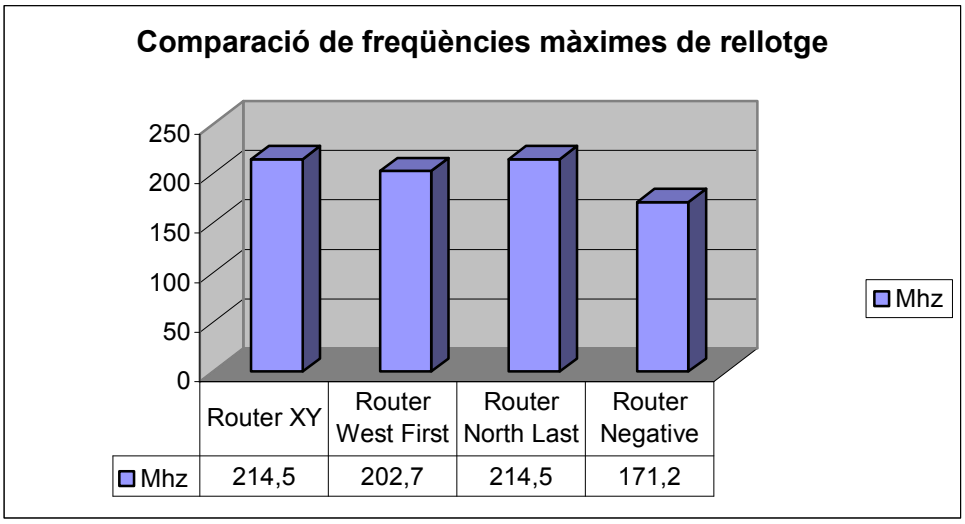


Figura 6.10 Gràfic de freqüències màximes

Com podem observar en la figura, obtenim que un dels algorismes parcialment adaptatius com es el North Last obte una freqüència de rellotge màxima equivalent a la de l'algorisme XY, per contra els altres dos algorismes obtenen una freqüència inferior, això es degut a que com hem vist en els diagrames RTL tenen mes recorregut, ja que aquesta freqüència màxima es la màxima freqüència de rellotge a la que es pot disparar un flip-flop de forma fiable i segura.

6.1.4 Simulació

La simulació dels algorismes es fa, com ja havíem comentat en l'apartat 4.2.2, amb l'eina ModelSim, així doncs aquesta eina executa un fitxer d'estímuls que ens proporciona el comportament del nostre mòdul i ens verifica el seu correcte funcionament.

Aquesta eina és la base per saber si el comportament del nostre mòdul és el correcte, i ens proporciona tota la informació necessària de les senyals d'entrada i sortida del mòdul i com aquestes evolucionen al llarg del temps.

Per tal de facilitar l'enteniment de les gràfiques que aquesta eina ens proporciona dels algorismes adaptatius implementats en aquest projecte, procedirem primer a explicar-ne la de l'algorisme XY, i aixi comprendrem una mica millor com actuen els algorismes adaptatius en front les mateixes condicions.

Com volem que l'algorisme testejat tingui la possibilitat de desplaçar-se a qualsevol direcció, em d'escollir la posició del router idònia perquè no perdi cap direcció. Procedirem doncs a injectar a l'algorisme els vectors de direccions de destinació per tal que ell, s'encarregui de donant-se la direcció on es dirigirà el paquet. Recordant l'explicat al punt 2.3.1, on el router es part d'una mesh 2D, sabem que les posicions on es pot fer les comprovacions mes exhaustives son les del mig. Per testejar els mòduls, passarem vectors d'adreces de 2 bits, el quals servirien per fer una mesh 2D de 3x3, així doncs la posició escollida es la 1,1. En la següent figura podem veure les direccions de destinació segons els vectors d'adreça d'entrada que rep el mòdul i si te el seu respectiu camí bloquejat o no.

Origen	Destí	Busbusy	Direcció
1 , 1	0 , 0	NoActiu	Oest
		Actiu	En espera
1 , 1	0 , 1	NoActiu	Oest
		Actiu	En espera
1 , 1	0 , 2	NoActiu	Oest
		Actiu	En espera
1 , 1	1 , 0	NoActiu	Nord
		Actiu	En espera
1 , 1	1 , 1	NoActiu	Local
		Actiu	En espera
1 , 1	1 , 2	NoActiu	Sud
		Actiu	En espera
1 , 1	2 , 0	NoActiu	Est
		Actiu	En espera
1 , 1	2 , 1	NoActiu	Est
		Actiu	En espera
1 , 1	2 , 2	NoActiu	Est
		Actiu	En espera

Figura 6.10 Taula direccions algorisme XY

Aquestes direccions s'activen sempre i quan el camí escollit no estigui ocupat per algun altre paquet que s'estigui encaminant alhora. Si resulta que la

direcció escollida està ocupada, el paquet haurà d'esperar a que l'altre el desocupi.
Després de saber les reaccions de l'algorisme en front les adreces d'entrada, ja podem entendre el diagrama de simulació de l'algorisme XY que podem veure a la figura següent.

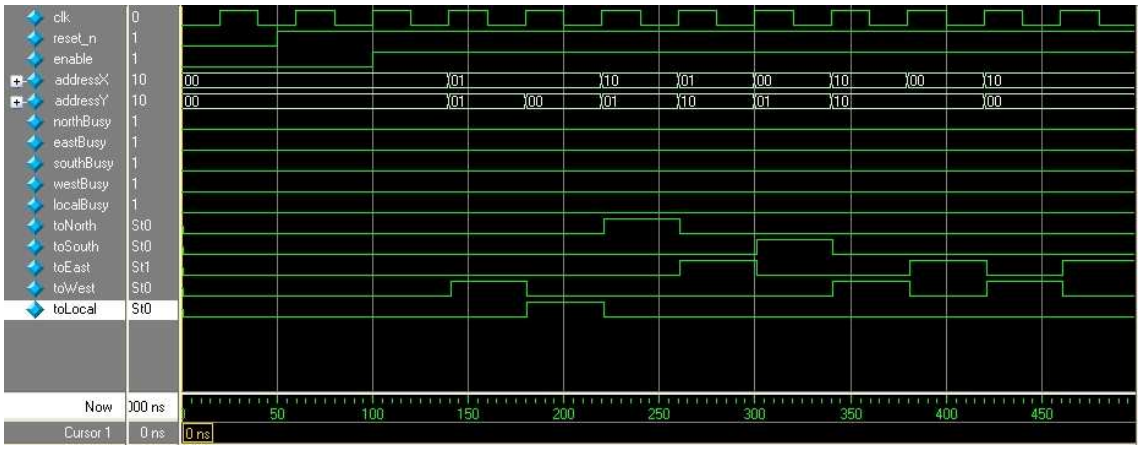


Figura 6.11 Simulació de l'algorisme XY

Com podem veure en el waveform de l'algorisme XY, el mòdul no actua fins que les senyals de reset_n i enable estan actives. Un cop la lògica combinacional de l'algorisme ha resolt en quina direcció ha d'anar, fa que s'activi la senyal de direcció corresponent. Com podem veure en la figura anterior cada cop que canviem d'adreça s'activa la senyal corresponent a aquella direcció, així doncs, per exemple, quan esta activa l'adreça 2, 2, i els senyals de busy estan desactivats, la direcció a la que va es al est, i així passa amb tots els exemples proposats a la figura.

Un cop entenem l'algorisme base del qual em partit, podrem veure amb mes claredat com actuen els algorismes adaptatius en front els mateixos estímuls. Els algorismes adaptatius son molt semblants a l'algorisme XY però en alguns casos, com ja s'ha explicat al punt 3, actuen de manera adaptativa, en els següents wavefoms, nomes explicarem alguns casos on l'algorisme actua de forma adaptativa, ja que explicar tots els casos seria tornar a repetir l'esmentat anteriorment per la majoria dels vectors d'adreces.

Procedirem doncs primer a explicar un cas adaptatiu de l'algorisme West first, on tenim com a adreça d'entrada la direcció 2,0, la qual si recordem hauria

d'anar a l'est, però activarem la senyal d'ocupat de la direcció est que fa que l'algorisme s'adapti i s'encamini cap al nord. Seguidament veurem el waveform de l'algorisme West First on es veurà aquest exemple.

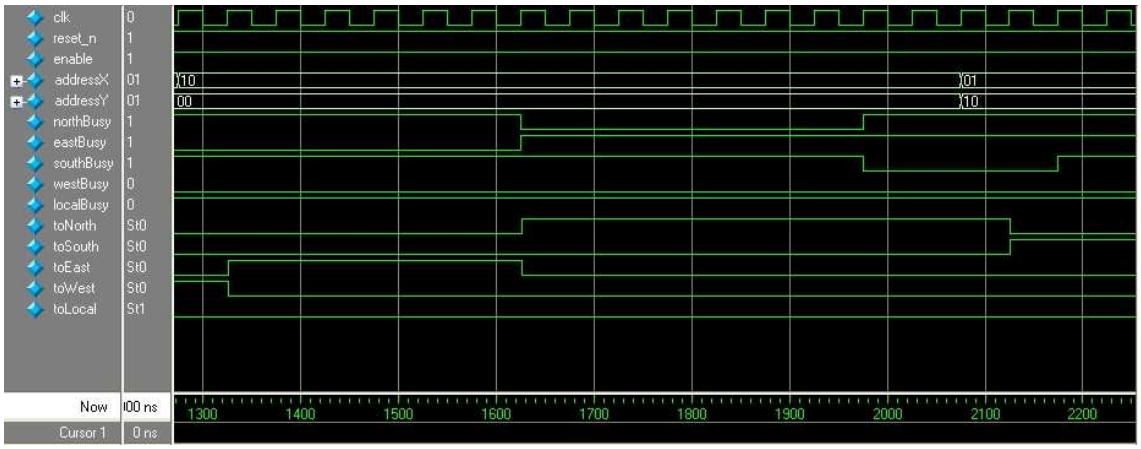


Figura 6.12 Simulació de l'algorisme West First

Com veiem a la figura 6.7, amb l'adreça 2,0 l'algorisme encaminava cap al est (*toEast*), i tot seguit al activar la senyal d'ocupat de l'est (*eastbusy*) el router s'adapta i encamina cap al nord (*toNorth*), si comparem aquest cas amb l'algorisme XY, aquest s'aturaria fins que s'alliberés la senyal *eastbusy* i pogués encaminar cap a l'est.

Per l'algorisme North Last em escollit l'adreça de destinació 2,2, la qual ens demostrarà l'adaptació de l'algorisme a tenir el camí de l'est ocupat.

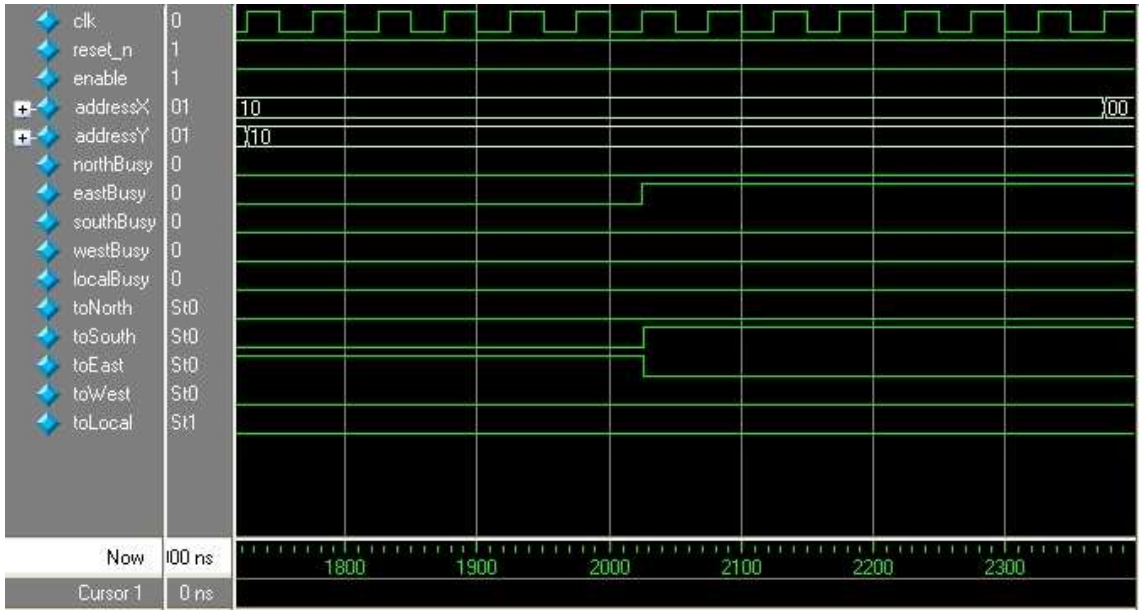


Figura 6.13 Simulació de l'algorisme North Last

Com podem veure a la figura 6.8, amb l'adreça de destinació 2,2 l'algorisme encaminava cap a l'est (toEast), un cop ocupem el camí de l'est (eastbusy) veiem com l'algorisme s'adapta i procedeix a encaminar cap al sud, donant així una alternativa a l'ocupació del camí.

Per l'últim dels algorismes, el Negative First, em escollit l'adreça de destinació 0,0 ja que ens proporciona un gir negatiu, com ja em explicat al punt 4.

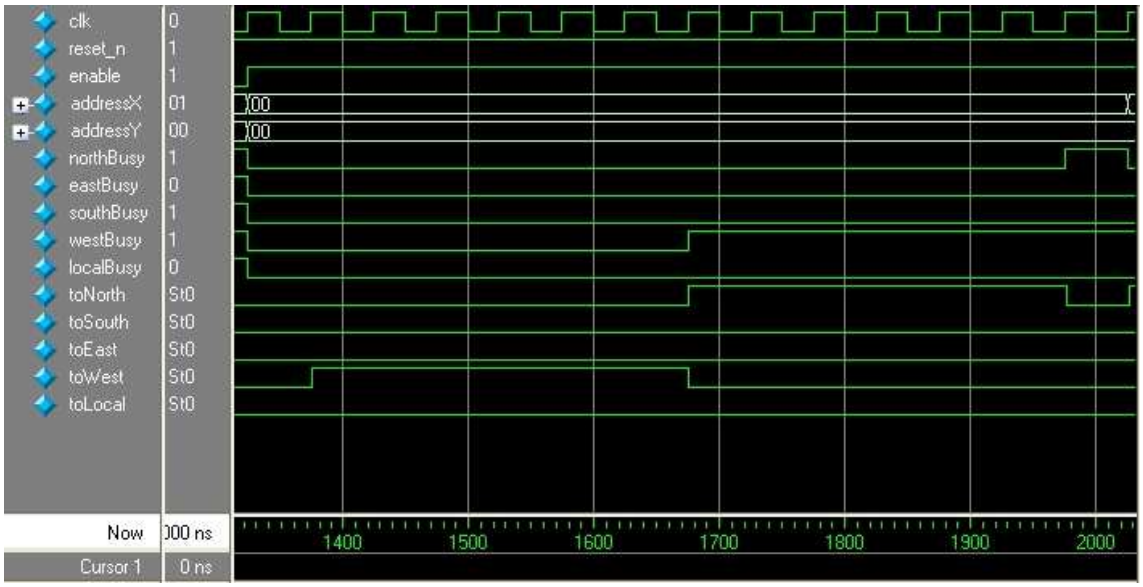


Figura 6.14 Simulació de l'algorisme Negative First

Com podem observar a la simulació de la figura 6.8, l'adreça de destinació introduïda es la 0,0, així doncs l'algorisme encamina cap a l'oest (toWest), però després activem el bloqueig del camí del oest (westBusy) amb la conseqüent adaptació del router a aquest nou estat que fa que passi de tenir activa la senyal del oest (toWest) a activa la direcció del nord (toNorth).

Capítol 7

7 Conclusions i propostes de futur

En aquest últim capítol, anunciem les conclusions del nostre projecte vists els resultats obtinguts a l'anterior capítol i les línies de futur que es poden seguir.

7.1 Conclusions

En aquest punt doncs, veurem les conclusions a les quals hem pogut arribar durant i després de la realització del projecte. Partim doncs de la base, que ja coneixem els nostres tres algorismes adaptatius situats al nostre entorn, dintre d'una NoC, i que vist els resultats obtinguts la seva implementació no es gaire mes costosa que la de l'algorisme al qual sempre hi fem referència i el qual es la nostra base de comparació, el XY. Per tant si ens preguntem quin d'aquests algorismes es el millor a utilitzar la resposta no es tan clara, ja que cap d'aquest suposa una gran diferencia dels altres pel que respecte a àrea de desenvolupament o costos en LEs. Així doncs les avantatges en l'aspecte de l'encaminament presentades pels algorismes parcialment adaptatius realitzats en aquest projecte semblen suficients per tal de treure conclusions positives pel que fa a les situacions que s'eviten d'espera en els diferents camins on s'activa l'encaminament adaptatiu.

Ara bé, el criteri mes important que ens queda per acabant-se de decidir a proposar un algorisme o un altre es fer un estudi exhaustiu del sistema que vulguem incorporar dintre de la NoC, la seva carrega de trànsit o quines transicions són les mes utilitzades, ja que són aquest detalls els que faran definitivament decantant-se cap a una banda o cap a l'altre. Al ser aspectes d'un estudi molt mes exhaustiu tant dels IP cores que conformen la NoC com les carregues en les transicions, quines son les mes utilitzades i quines no s'utilitzen mai. Només amb aquest aspectes podrem saber certament l'algorisme a escollir ja que com hem dit anteriorment no presenten diferencies significants o si mes no molt significants un dels altres en tamany i/o consum.

Durant l'evolució d'aquest projecte s'ha aconseguit especificar, generar i validar el codi HDL necessari per la implementació del router especificat per cadascun dels algorismes parcialment adaptatius que composaven la realització d'aquest projecte. Per assolir aquest objectius s'ha estudiat el funcionament de la NoC i dels seus components principals.

Pel que respecte a la part personal, les conclusions que es poden extreure de la realització d'aquest treball són molt positives, ja que ens ha permès estudiar

i analitzar una de les parts de hardware mes avançades del nostre temps com son les NoCs, i aprofundir en el seu funcionament, tanmateix hem creat tota una part molt important de la mateixa com és un router i s'han pogut contemplar les diferencies entre algorismes. Tot això s'ha aconseguit fer amb un llenguatge al qual no estàvem habituats a treballar i que es la base al món laboral, el verilog, tot i que es cert que el llenguatge no difereix en molts aspectes del que ja coneixíem, el VHDL, si que ens ha fet entendre millor les diferencies que hi ha entre ells. A part d'això em aprés a fer servir les eines Modelsim i Synplicity-Synplify Pro® les quals són les utilitzades en gran mesura pel mon laboral i que ens serviran per entrar en aquest mon amb una millor preparació.

7.2 Propostes de futur

De cara a futures aplicacions els algorismes d'encaminament implementats en aquest projecte no son ni els únics, ni segurament els mes òptims que es poden representar, tot i així donen una idea de l'optimització que representa utilitzar aquest algorismes respecte l'algorisme determinista del qual partíem. Així doncs la complementació d'aquest projecte es pot fer de moltes maneres ja que al ser una estructura jeràrquica de hardware en la qual el components estan formats per mòduls i alhora aquest mòduls estan formats d'altres mòduls, fa que la implementació d'aquest algorismes es pugui traslladar a altres topologies i d'altres tipus de connexions.
Així doncs una possible aplicació de futur la podríem proporcionar unint els nostres algorismes parcialment adaptatius amb unes diferents tècniques de comunicació, com es el cas del wormhole, ja que la principal avantatge d'aquesta tècnica en front del ephimeral circuit switching és que la retransmissió dels missatges es realitza punt a punt i no end to end com ho fa aquesta, a part de tenir una cua que permet transmetre paquets i emmagatzemar-los per després enviar-los i així no haver d'aturar la transmissió de paquets.
Així doncs es podria millorar amb la composició dels algorismes parcialment adaptatius que farien d'aquest sistema molt mes efectiu ja que no s'hauria

d'aturar quan el camí estigues bloquejat sinó que trobaria d'altres camins alternatius per arribar al seu destinació.

Una altra proposta de futur que es podria realitzar seria l'anàlisi del tràfic que travessa la NoC, l'anàlisi de tràfic es la extracció i la inferència d'informació a la xarxa incloent els temps i volums dels paquets de xarxa així com les adreces visibles de la xarxa des de el seu origen fins al seu destinació. Amb aquest anàlisis podríem veure el tràfic que recorre la nostra NoC, quins nodes tenen mes tràfic i quins camins son els mes recorreguts.

Acabarem amb una altra proposta de futur, que seria la de crear el router proposat, amb un encaminament dinàmic per tal de prevenir les fallades que es poden ocasionar en una NoC, aquest router miraria l'estat dels seus veïns i dinàmicament crea la seva taula d'encaminaments possibles per tal de que si un node fa un fallida aquest s'obvia de la taula i ja no es te en compte.

Bibliografia

En aquest punt, exposarem totes les fonts utilitzades ja siguin físiques o virtuals.

Referències a llibres

- [Castells06] David Castells-Rufas, Jaume Joven, Jordi Carrabina, "A validation and performance evaluation tool for ProtoNoC", International Symposium on System-on-Chip'06, 2006.

- [Gaghan93] Patrick T. Gaughan, Sudhakar Yalamanchili, "Adaptive routing protocols for hypercube interconnection networks", Computer, vol. 26, no. 5, pp. 12-23, May, 1993.

- [Hu04a] Jingcao Hu, Radu Marculescu, "DyAD - Smart Routing for Networks-on-Chip", 41st Conference on Design Automation Conference (DAC'04), 2004.

- [Jantsch03] Axel Jantsch i Hannu Tenhunen. "*Networks on chip*", Kluwer Academic Publishers (2003).

- [LluísTeres06] L.Terés. "Visión global preliminar de la microelectronica".cnm-imb (CSIC) 2006.

- [LluísTeres98] Ó E. Lecha, M. Moré, F. Rincón, L. Terés y J. Vidal, "el lenguaje vhdl", cnm-uab, Seminario de VHDL, 1998-00.

- [MELLOt] Aline Mello, Luciano C. Ost, Fernando G. Moraes, Ney L. Calazans. "Evaluation of Routing Algorithms on Mesh Based NoCs", Internal Technical Report temporally available in http://www.inf.pucrs.br/tr/tr040.pdf, 2004.

- [Ogras05] U. Y. Ogras, J. Hu, R. Marculescu, "Key Research Problems in NoC Design: A Holistic Perspective", in Proc. of the International Conf. on HWSW Codesign and System Synthesis, Sep. 2005.

- [Schafer05] M. K. F. Schafer, T. Hollstein, H. Zimmer, M. Glesner, "Deadlockfree routing and component placement for irregular mesh-based networks-on-chip", ICCAD '05: Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design, ISBN: 0-7803-9254-X, 2005.

Referències electròniques.

http://www.altera.com/

http://www.model.com/

http://www.synplicity.com/

Resum

Aquest projecte presenta una avaluació de les diferents alternatives d'encaminament per a una NoC amb una topologia mesh 2D. Per tal d'exposar aquestes alternatives s'ha estudiat la composició d'un router implementat amb l'algorisme determinista XY i s'ha adaptat per tal que aquest suportés els algorismes parcialment adaptatius West First, North Last i Negative First. Un cop tenim els routers implementats es disposa un estudi dels diferents algorismes i com cadascun d'aquests actuen en front uns mateixos estímuls per tal de crear una comparativa entre ells que ens faciliti una elecció a priori.

Resumen

Este proyecto presenta una evaluación de las diferentes alternativas de encaminamiento para una NoC con topología mesh 2D. Con tal de exponer estas alternativas se ha estudiado la composición de un router implementado con el algoritmo determinista XY y se ha adaptado para que este soporte los algoritmos parcialmente adaptativos West First, North Last y Negative First. Una vez tenemos los routers implementados se dispone un estudio de los diferentes algoritmos y como cada uno de ellos actúan delante de los mismos estímulos para crear una comparativa entre ellos que nos facilite una elección a priori.

Summary

This project presents an evaluation of the different alternatives of routing for a NoC with mesh 2D topology. To exposing these alternatives we have studied the composition of the router implemented with the deterministic algorithm XY and it has been adapted to support the partially adaptative algorithm West First, North Last and Negative First. Once we have the routers implemented, a study of the different algorithm and how each one act in front of the same stimulations has been studied to create a comparative between them that provide a priory choice.