Universitat
Autònoma
de Barcelona

Sistema Gràfic de depuració per xarxes (NoC)

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Miquel Àngel Gallego Parejo
i dirigit per
Antoni Portero Trujillo
amb el suport de
Aitor Rodríguez Alsina
Bellaterra, 1 de Juny de 2007

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, ...,
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 Miquel Àngel Gallego Parejo

I, perquè així consti, signa aquest certificat.

Signat: .....

Bellaterra (Cerdanyola del Vallès), 15 de Juny de 2007

Prefaci

La memòria d'aquest projecte de final de carrera ha estat redactada seguint l'estàndard del IEEE que busca la capacitat de síntesi al centrar el text en un article no massa extens, que condensa la part teòrica principal de l'aportació del projecte. Les parts més tècniques poden ser consultades als apèndixs.

Conté, a més, en forma d'apèndix, el marc teòric i la introducció dels complements per a entendre tota la terminologia, els tecnicismes i la tecnologia emprada en el desenvolupament de l'eina en la que es basa.

Per tal de profunditzar més en la matèria s'inclou, a més, una guia d'execució i una breu explicació de com utilitzar aquesta eina així com, les consideracions a tenir en compte per tal d'obtenir resultats destacables en la tasca de depuració de dissenys de NoC.

RESUM

Aquest projecte descriu el disseny i desenvolupament d'una eina gràfica per a la depuració de projectes desenvolupats amb un llenguatge de descripció de sistemes com és el SystemC. Amb aquest llenguatge s'ha desenvolupat una NoC (Network on Chip). L'eina desenvolupada mostra de forma visual l'arquitectura de la xarxa NoC, els valors dels senyals que es transmeten a través de la xarxa i estadístiques sobre aquests per tal de poder fer un seguiment exhaustiu i agilitzar la recerca d'errors com interbloquejos, pèrdua de dades i d'altres. Al concentrar en un únic entorn la descripció de la NoC i les dades relatives a les senyals en temps de simulació, proporciona un valor afegit a altres eines disponibles per a realitzar aquesta tasca.

RESUMEN

Este proyecto describe el diseño y desarrollo de una herramienta gráfica para la depuración de proyectos desarrollados con un lenguaje de descripción de sistemas como es el SystemC. Con este lenguaje se ha desarrollado una NoC (Network on Chip). La herramienta desarrollada muestra de forma visual la arquitectura de la red NoC, los valores de las señales que se transmiten a través de la red y estadísticas sobre estas para poder hacer un seguimiento exhaustivo y agilizar la búsqueda de errores como interbloqueos, pérdida de datos u otros. Al concentrar en un único entorno la descripción de la NoC y los datos relativos a las señales en tiempo de simulación, proporciona un valor añadido a otras herramientas disponibles para realizar esta tarea.

ABSTRACT

This project describes the design and development of a graphical tool to debug projects that are developed with a system description language like is the SystemC. A NoC(Network on Chip) was developed with this language. The developed tool shows visually the NoC network architecture, the signals values that are transmitted through the network and statistics about this signals to get an exhaustive record and improve the errors search such as: deadlocks, data loss and others. Including in just one environment the NoC description and the information relative to the signals in simulation time providing an added value to other available tools to realize this task.

Taula de continguts

Article.
Graphical Environment to Network on Chip communication Debug

At the beginning of this project, the idea was to develop a GUI to show visually a NoC schema for different embedded IPs connected to it and after, with this tool try to help to debug multi-threading code that has bugs in the data and control flow. We try to stand out the different communications signals that are part of the GALS (Globally Asynchronous Locally Synchronous) communication inside the NoC. The main proposal is locate incoherencies in inter tile communication to detect possible bugs as deadlocks, data-lost, stack information etc.

Sistema gràfic de depuració de xarxes NoC

MiSE, ETSE
08193 Bellaterra (Juny 2007)
Miquel A. Gallego Parejo

***Resum*— En aquest treball es presenta el desenvolupament d'una eina per visualitzar les dades que passen a través de xarxes (NoC) en sistemes encastastats. Em de pensar que el número de processadors que pot tenir un SoC cada vegada creix més, per exemple el Cell Processor. Ja que s'ha vist que no es milloren prestacions només amb un processador. El processador d'Intel P4, per exemple, no té una freqüència de més de 3.8GHz en canvi utilitza un *quad core* (varis processadors fent tasques diferenciades). Una possibilitat que està agafant força dins del món acadèmic de com connectar aquests processadors és utilitzant una xarxa que comunica els processadors, ja que un bus segmentat pot no donar les prestacions adequades si el número de processadors que pengem creix suficient i no és parametritzable. En aquest treball, s'ha desenvolupat una eina que permet veure la informació que circula per la xarxa. Degut a que són varis processos que de forma concurrent accedeixen a la xarxa, la detecció d'errors amb les eines actuals de depuració pot ser molt tediosa.**
Aqueta eina pot ser de valuosa utilitat ja que pot detectar errors de comunicació dins de la xarxa que amb les eines de depuració es trigaria molt més temps.

I. Introducció

En el disseny de sistemes encastats es necessita cada vegada més reduir el temps de disseny i producció, maximitzar la reutilització dels dissenys, millorar la flexibilitat i dels complexes mòduls i simplificar els procesos de verificació. El número de processadors que pot tenir un SoC cada vegada creix més, per exemple el *Cell Processor* de IBM[1]. El ràpid desenvolupament dels microprocessadors fa que segons la llei de Moore cada 18 mesos es dobli la capacitat en nombre de transistors dins d'un xip (SoC). Aquest poder d'integració no va acompanyat per les eines de desenvolupament de maquinari (*hardware*), ja que cada vegada es necessiten més enginyers/hora per fer un xip. En aquest treball es presenta una eina per mostrar la representació gràfica de xarxes que comuniquen diferents processadors dins d'un xip.

Degut a les exigències i els límits esmentats, constantment sorgeixen noves idees, noves metodologies, nous processos, noves eines i nous paradigmes.

Un exemple de nou paradigma, centrant-nos en el disseny microelectrònic, és el paradigma de les *Network-on-Chip* (NoC).

Els SoC són sistemes encastats en un xip, que proporcionen una completa funcionalitat, gràcies a integrar en ell mòduls complexes, blocs IP (*Intellectual Property Blocks*) perfectament provats, etc. però que tenen limitacions en les comunicacions, ja que tots els mòduls comparteixen un bus, o busos optimitzats, però sempre connexions point-to-point

En una NoC, en canvi, mòduls com nuclis de processador, memòries, blocs IP i clústers de funcionalitat intercanvien dades codificades en els paquets de bits, fent servir una xarxa com a subsistema de transport per a dirigir el tràfic de la informació. Aquest nou paradigma complica en certa forma el disseny dels nous sistemes i la seva verificació, fet que implica una creixent necessitat d'eines per a dur a terme aquestes tasques d'una forma més sencilla

Una d'aquestes eines és el llenguatge SystemC (mirar Apèndix 6,A.6) que és una llibreria per a descripció de hardware (HDL) amb una sintaxi molt coneguda, com és la sintaxi C/C++. Aquesta nova llibreria ha possibilitat implementar d'una forma més intuïtiva la funcionalitat dels mòduls hardware ja que possibilita fer codisseny *Hardware-Software* a diferents nivells d'abstracció i ens permet treballar a nivells TLM *(Transactional Level Modeling)*. Si comparem SystemC amb altres llenguatges HDL com Verilog o VHDL, que també permeten dissenyar sistemes amb nivells d'abstracció superiors i que treballen a nivells RTL, ens trobem amb la facilitat d'implementació en SystemC per als programadors degut a la familiaritat de la sintaxi. Podem també dissenyar sistemes abstraient els detalls d'implementació, és a dir, seguint una metodologia *top-down*.

Amb la metodologia top-down es poden modelar sistemes complexes, com un sistema basat en una NoC (com en el que es basa aquest projecte), amb independència del hardware o software que implementi cada una de les parts integrants.

II. Anàlisi i Disseny

Aquest treball està explicat en tres punts diferenciats. En l'apartat primer s'ha fet una introducció a la necessitat de les NoC i les eines per tal de treballar amb elles. A l'apartat segon es fa un anàlisi i disseny del sistema i dels mòduls ens els que es divideix. Finalment, s'expliquen les conclusions i treballs futurs relacionats amb l'eina desenvolupada.

I. Introducció a la Problemàtica de les NoCs

A. Problemàtica

Per tal d'avaluar els possibles beneficis de les NoC a diferents desenvolupaments s'estan implementant solucions fent servir llenguatges de codisseny com el SystemC, que encara que agilitzen moltíssim el temps de disseny, no inclouen eines de depuració prou avançades per visualitzar la concurrència, necessària quan parlem de maquinari *(hardware).* Aquesta raó provoca una complicada depuració pels dissenyadors, degut a que no es controlen tots els valors de les variables observades en el mateix instant de temps, o se'ls fa difícil i tediós el posterior seguiment de traces generades pel seu propi disseny, degut al paral·lelisme i la concurrència propia dels sistemes basats en maquinari. Un exemple el tindríem en una solució proposada per Ramon Pla Casablancas a la seva tesina. Proposa una NoC per un sistema MPEG (A.7) encastat.

Aquest sistema està realitzat per a avaluar l'aportació de les NoCs en un projecte desenvolupat mitjançant la llibreria SystemC.

Per tal de fer un seguiment de l'actuació de les senyals que intervenen en aquest sistema es fa un seguit de generacions de fitxers de tracejat (logs) d'extenció Visual Core Dump (vcd) (A.6) que és un estàndard de generar formes d'ona, que serviran per a que les carregin a programes, com el *gtkwave [13]* fet per la visualització de formes d'ona de les senyals.

El problema ve produït per l'enorme quantitat de senyals que intervenen en un sistema com aquest i quan es vol fer un seguiment pot ser costós i pesat, a més de veure el progrés de les senyals en el sistema donat les seves característiques de xarxa.

Per pal·liar el problema es proposa un sistema gràfic de seguiment d'aquest tipus de traces per tal de mostrar, d'una forma clara i ràpida, el progrés d'aquests senyals i així també veure les comunicacions entre els diferents mòduls que composen el sistema.

Per tal de no limitar el sistema gràfic a un sol disseny, es proposa també la generació d'un model de descripció del sistema per tal de tenir un pas previ i necessari de representació de la informació. Ens servirà per tenir les dades importants en un únic arxiu, poder revisar la correspondència d'aquest amb el sistema en SystemC.

B. Consideracions

En el primer moment donat que anava a desenvolupar tot el sistema a distància, vaig prendre la decisió d'optar per que funcionaria sobre un sistema web.

Més endavant, un cop començat el projecte es va poder comprovar que el sistema web, encara que per la primera part fos més simple, no proporcionava totes les avantatges a l'hora de generar un sistema amb una interfície gràfica(GUI) fàcilment modificable com amb l'opció adoptada.

C. Requisits

Ens fixem com a objectius mostrar visualment l'esquema d'una xarxa NoC per a un sistema que conecta un compressor de video MPEG encastat, en una xarxa de topologia 2-d mesh de NxN, i ressaltar de forma visual les senyals que s'estan trasmeten a la xarxa en aquesta topologia.

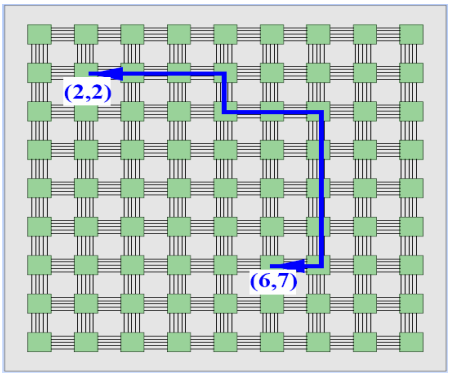


Fig. 1. Exemple de xarxa 2-d mesh de 9x9..

Es planteja com a requisit obtenir a partir del sistema original una descripció dels objectes en un document del tipus XML(A.1) que és un estàndard de marcació d'elements de tot tipus, i sobretot de descripció d'objectes. Aquest document ha de ser clar, simple i sobretot estar pensat per facilitar una futura utilització per altres eines.

A partir d'aquest XML es vol fer una representació gràfica de la NoC amb tots els seus routers, mòduls i wrappers.

Per últim, a partir de la representació gràfica de la NoC, i d'un arxiu *vcd* prèviament generat, diferenciar les senyals que estan actives de les que no ho estan en un instant de temps.

Aquests requisits ens separen el sistema complet en tres parts ben diferenciades, per funcionalitat, entrades i sortides.

D. Requisits funcionals

La primera part d'aquest sistema, la podem considerar com un mòdul de funcionalitat ben delimitada. S'encarregarà de d'analitzar sintàcticament el codi font del projecte programat amb la llibreria SystemC sota C++.

Per a analitzar sintàcticament de forma correcta, s'ha de tenir en compte que cada mòdul ve definit en un arxiu *header*, i s'ha de fer una primera passada per tal d'extraure la definició dels mòduls; i una segona per a exportar les instàncies d'aquests objectes en el programa principal o *main* del codi font. Aquestes dues passades han de generar dos arxius XML.

Per a dur a terme aquest propòsit hem plantejat l'anàlisi sintàctic en el llenguatge *Java*[5] mitjançant la classe Regex(A.1). Aquesta classe Java permet analitzar sintàcticament cadenes de text fent servir expressions regulars, i per la generació dels arxius XML s'ha optat per la llibreria JDOM(*Java Document Object Model*) [7] de Java.

La segona part serà un analitzador sintàctic de XML[6],. Utilitzem la interfície de programació (API) SAX[8] de Java (es basa en els events per tal de separar etiquetes XML).

Aquesta part ha de traduir les descripcions generades en la primera part i crear les estructures de dades adients per a dibuixar l'esquema buscat. Posteriorment, amb la llibreria *graphics* representarem l'estructura construïda recentment.

Per últim, la tercera part analitzar sintàcticament els arxius de traces (log) del tipus vcd, fent canviar el dibuix en funció dels arxius esmentats.

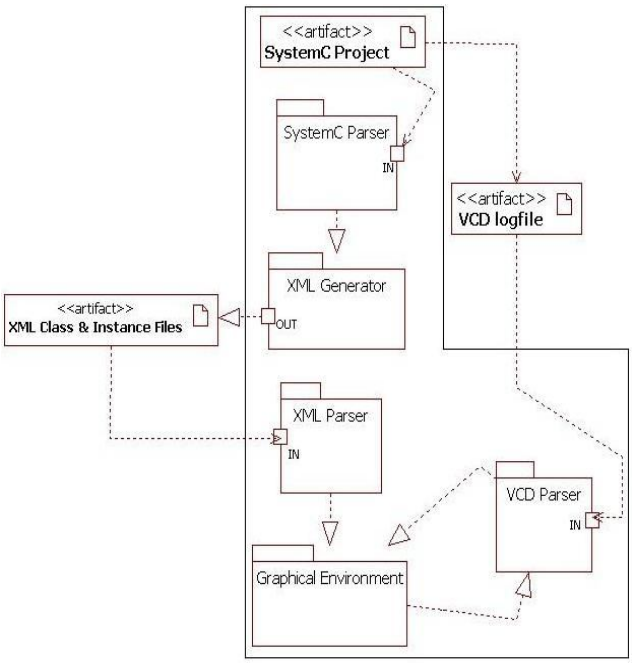


Fig. 2. Diagrama de components del sistema proposat.

II. MÒDULS I SUBPROGRAMES

Com hem comentat abans, dividim el sistema proposat en quatre mòduls o subprogrames, de funcionalitat específica.

Així doncs, concretant la funcionalitat de cada un, concloem en els algorismes a seguir per cada mòdul.

Analitzador sintàctic SystemC:
El primer mòdul segueix els següents passos:

1. Llegir el programa principal (*main* del projecte *SystemC*) buscant els arxius de capçalera (*headers*) que és a on es declaren els objectes SystemC.
2. Obtenir els noms de les classes i els seus *SC_PORTS* (ports de connexió que serviran per enviar/rebre les dades.
3. Construir l'XML amb les descripcions d'aquestes classes mitjançant JDOM(A.1) que és una llibreria de codi font per a analitzar sintàcticament, manipular i crear documents XML

```
<NOC>
  <SC_MODULE num="n">
    <NAME>ControlMPEG</NAME>
    <CONNECTIONS>
      <CLK_CONNECTIONS>
        <CLK_CONNECTION num ="n">
          …
        </CLK_CONNECTION>
      </CLK_CONNECTIONS>
      <IN_CONNECTIONS>
        <IN_CONNECTION num="n">
          …
        </IN_CONNECTION>
      </IN_CONNECTIONS>
      <OUT_CONNECTIONS>
        <OUT_CONNECTION num="n">
          …
        </OUT_CONNECTION>
      </OUT_CONNECTIONS>
    </ CONNECTIONS>
  </SC_MODULE>
  …
</NOC>
```

4. Obtenir les instàncies de les classes al *main*, juntament amb l'assignació de cada port dels objectes instanciats amb la senyal corresponent.
5. Construir l'XML amb aquestes instàncies.

Per a aquest últim pas hem considerat que els objectes podran ser de tres tipus segons la seva funció dins de la NoC (que no dins del sistema). Per tant tindrem en compte els **Mòduls** (tenen una funcionalitat lògica dins del sistema, es a dir, implementen la funcionalitat del sistema), **Routers** (s'encarreguen de dirigir el tràfic de les senyals cap a un mòdul o cap a un altre) i els **Wrappers**, que s'encarreguen d'establir la comunicació entre els routers i els mòduls per tal d'alliberar a aquests últims de la feina "bruta".

Així doncs, obtenim un XML com aquest:

```
<NOC>
  <SC_MODULES>
    <SC_MODULE num="0">
      <SC_MODULE_TYPE></SC_MODULE_TYPE>
      <SC_MODULE_NAME> </SC_MODULE_NAME>
      <SC_MODULE_CONNECTIONS>
        <SC_MODULE_CONNECTION num="n">
          <SC_MODULE_PORT> </SC_MODULE_PORT>
          <SC_MODULE_SIGNAL></SC_MODULE_SIGNAL>
        </SC_MODULE_CONNECTION>
      </SC_MODULE_CONNECTIONS>
    </SC_MODULE>
      …
  </SC_MODULES>

  <SC_ROUTERS>
      …
  </SC_ROUTERS>

  <SC_WRAPPERS>
      …
  </SC_WRAPPERS>
</NOC>
```

Ens hem de fixar que en la part de les connexions també especificarem a la etiqueta si ens referim a la part dels *routers* o dels *wrappers*.

Analitzador sintàctic XML
Aquest mòdul rep com entrada els dos arxius XML, el de la descripció de les classes del programa i el de la descripció de

les instàncies dels *SC_Modules*.

1. Llegim el fitxer XML de les classes gràcies a un objecte analitzador sintàctic generat pel mètode estàtic de SAX i a un manejador (*handler*) que reescriu els mètodes de la súper classe *DefaultHandler*
2. Creem una estructura de dades amb les classes del programa principal. Aquesta serà una llista d'objectes que representen aquestes classes.
3. Llegim el fitxer XML dels objectes seguint els mateixos passos que al pas 1, però amb mètodes i manejadors propis per cada tipus d'objecte per tal de construir llistes separades d'objectes segons el seu tipus (*SCModule*, *SCRouter* i *SCWrapper*).
4. Als objectes creats se'ls assignen les característiques pròpies de cada classe amb la que coincideixen del programa original SystemC, establint la quantitat de ports d'entrada, de sortida, les senyals que entren, les senyals de surten, la seva situació a la NoC, etc.
5. Un cop tenim les llistes d'objectes, passem a generar un objecte anomenat *NoCConf* que serà l'estructura que contindrà tota la informació de la nostra NoC, amb tres vectors bidimensionals, per situar els mòduls, els enrutadors i els empaquetadors respectivament a la posició de la xarxa NoC que correspongui.

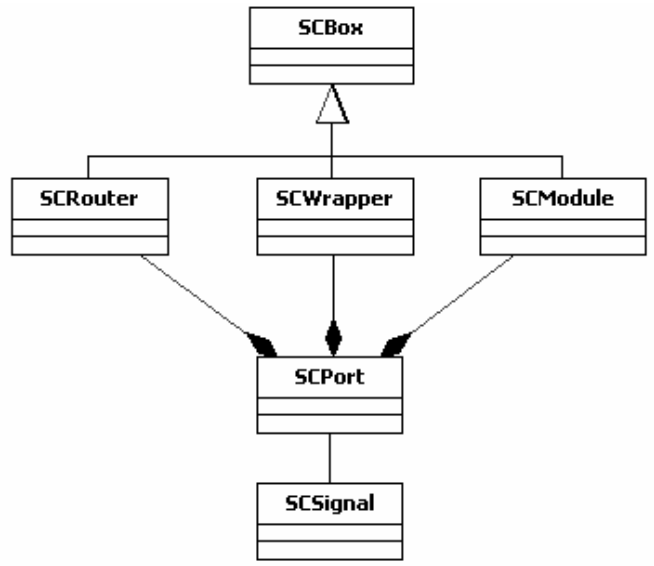


Fig 3: Diagrama de classes resultant del Parser XML

El Sistema Gràfic

El sistema gràfic s'encarrega de mostrar d'una forma visual l'estructura de dades acabada de crear, la *NoCConf*.

Per a dur a terme aquest procés, creem un objecte anomenat *DrawPanel* que reescriu la classe *JPanel* de l'API Swing per a generar entorns gràfics. Aquest panell, gràcies a la classe *graphics* de *Java* (A1), ens permet mostrar visualment la NoC per pantalla.

Seguim aquest algorisme:

1. Dibuixem els *SCRouters* segons la seva posició en el vector bidimensional de l'objecte *NoCConf*. Pintem també els seus ports.
2. Dibuixem a continuació els *SCModuls*, seguint la mateixa proporció que abans i seguint també el criteri de situació dins de la xarxa NoC que ve donat per la situació de l'objecte en el vector bidimensional d'*SCModuls* dins de l'objecte *NoCConf*.
3. Seguim el mateix patró per dibuixar els *SCWrappers*.
4. Passem a dibuixar les connexions entre els enrutadors, primer tots els ports i després, per tal de no repetir fils només mostrarem les connexions de la part est i de la part sud dels enrutadors. També tindrem en compte els enrutadors que son de frontera per tal de no dibuixar connexions que pengen
5. Mostrem per últim les connexions entre els *SCModuls* i els *SCWrappers* i les connexions entre els *SCWrappers* i els *SCRouters*.

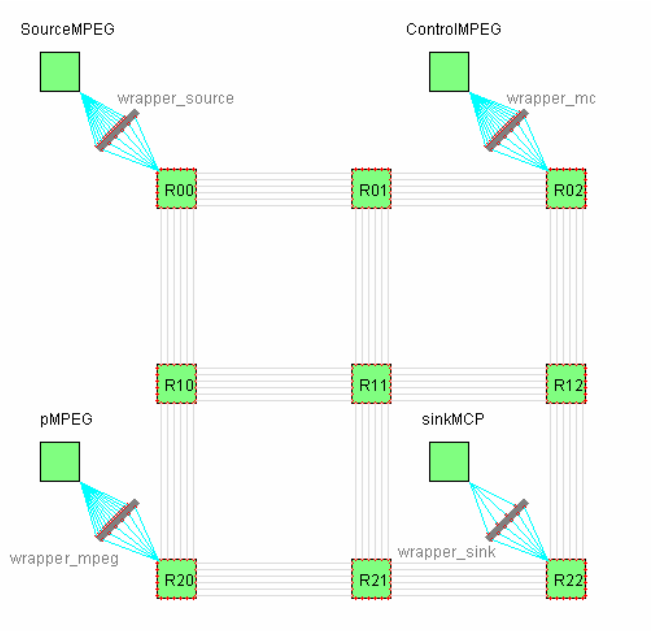


Fig. 4. Dibuix d'una NoC 3x3.

Un cop tenim el panell acabat, i per facilitar la manipulació del programa s'opta per construir una interfície gràfica, basada també en *Swing*(A.1) que és una interfície de programació d'aplicacions (API) de Java que proporciona uns mètodes i plantilles per crear un entorn de finestres amb plantilles de panells, botons, menús i events associats a aquests objectes.

Degut a la modularitat buscada des de el principi, i a les facilitats que dóna *Java* per integrar classes en un mateix paquet, es crea un únic programa que des del seu entorn podrem obrir els arxius XML i generar el dibuix de la NoC sense fer servir un navegador. Aquesta interfície gràfica també ens servirà per obrir fàcilment els arxius de traces VCD, seguir els estats guardats a aquest arxiu, així com veure els valors que tenien les senyals en temps d'execució (sempre segons l'arxiu VCD).

Analitzador sintàctic VCD

L'analitzador sintàctic vcd segueix un algorisme en el que intervé activament l'usuari.

El parser segueix el següents passos després de que l'usuari hagi obert el fitxer de traces VCD:

1. Llegir el fitxer VCD establint la concordança entre senyals, és a dir, el sistema vcd assigna àlies a les senyals tracejades, i nosaltres hem d'establir aquesta correspondència.
2. Anem a l'estat inicial, i activem o no les connexions de cada objecte.
3. Guardem l'estat de tots els objectes a l'arxiu de projecte (Base de dades encastada basada en el motor DB4O – *DataBase 4(for) Objects)* (A.1)
4. Passem l'estructura de dades resultant al sistema gràfic per tal de que en temps d'execució d'aquest triï les senyals que han de ser pintades en un altre color segons si estan actives o no.
5. Imprimeix a la taula els valors de l'objecte actiu.
6. Un cop s'ha mostrat el resultat gràfic, la GUI permet prémer un botó que ens durà al següent pas, ja a partir de l'estat inicial, l'analitzador sintàctic del *vcd* va agafant cada nou instant de temps mostrat a la traça i repeteix els passos d'activar o no les senyals segons ens digui la traça, per posteriorment tornar a pintat el resultat obtingut.

III. Resultats

Després de l'execució del programa creat, obtenim una representació visual clara i efectiva, el programa és ràpid i ens proporciona la representació gràfica fidel al projecte SystemC analitzat.

Es proposa, a més, generar uns valor estadístics que s'imprimeixen en una taula a una finestra independent a petició de l'usuari si en un estat qualsevol prem el botó d'estadístiques. Aquestes mostren valors útils com el temps en que s'ha activat un senyal per primera vegada, el temps de l'ultima activació, el % d'ús de la connexió (si es un bus de dades), el temps que porta actiu un senyal etc. A l'apèndix 2 hi ha un exemple(s), de com funciona l'eina.

Prenent com exemple el sistema explicat a l'apèndix 7, un MPEG-4 SP[10] que codifica macro-blocks del tipus I, P i B. L'experiment està basat en una compressió de 4 imatges de tamany 320x288 CIF. Les imatges es llegeixen desde fitxer amb pel mòdul amb nom *SourceMPEG* (0,0) posteriorment el mòdul *pMPEG*(2,0) les comprimeix i les envia al *sinkMCP(2,2)* i es mostra amb un decodificador de MPEG. El mòdul *ControlMPEG*(0,2) controla el tipus d'imatge que s'esta comprimint. L'exemple està basat en diferents mòduls conectats a la NoC com es pot veure a la Fig.5. Un d'ells es la font (és una memòria L2) a on estan les 4 imatges que es codificaran a un GOP (*Group of Pictures*). El mòdul de control(*ControlMPEG*) té que controlar el tipus de GOP que podem codificar (IIII, IPPPP, IBBP etc.) El control pot decidir si el següent macro-block es I, P o B. És una tasca prou dinámica. El mòdul amb nom *sink* es una memòria a on es guardaran les imatges comprimides. S'executa el codi i es genera l'arxiu VCD.

Hi ha 4 fils d'execució concurrents a la NoC i estan sincronitzats amb el protocol *handshake.*

Si simulem el sistema es molt difícil trobar a on es trenquen les connexions.

Aquesta eina permet anar a l'intant desitjat i visualitzar que està passant en aquest moment i trobar inconsistencies.

IV. Tests

En successives proves s'intenta complicar el correcte funcionament, traçant 14 senyals per cada SCRouter i el sistema respon de forma correcta.

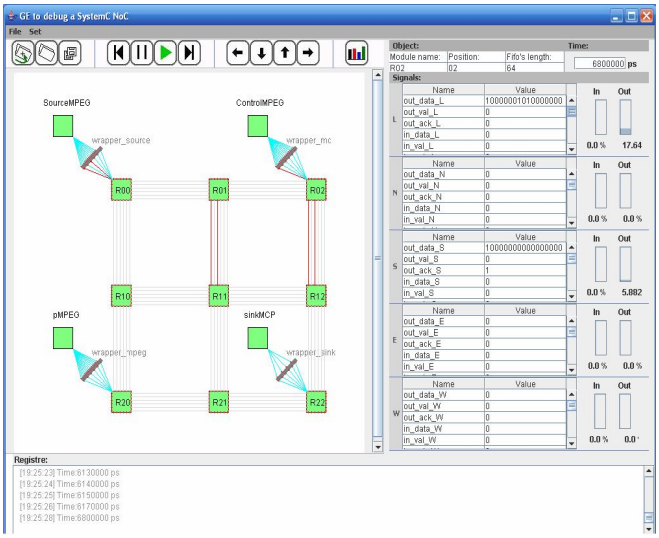


Fig. 5. Dibuix d'una NoC 3x3 seguint traces vcd.

A continuació s'enumeren algunes situacions a on la eina seria molt útil per tal de trobar errors:

Situació 1- Pèrdua de dades

Aquesta eina permet obtenir estadístiques sobre l'us de les interconnexions entre els diferents nodes. Un error molt comú és el que es produeix al enviar un paquet de n bits d'un mòdul a un altre i en canvi es reben n-1 degut a una deficient sincronització (un wait incorrecte al codi SystemC) entre l'emissor i el receptor. Per contra, a la traça generada es detecta però es fa realment dificil de visualitzar quin fil *(thread)* és el responsable de la perdua. L'eina permet controlar les dades que s'han enviat i visualitzar en quin fil d'intercanvi de dades amb *handshake* s'ha perdut la informació.

Situació 2.- Interbloqueig (Dead-lock)

Aquesta situació és comú en desenvolupaments de MPSoC. Quan es modela el disseny mitjançant C++ amb *eclipse*, es veu a la simulació que falla pero no es detecta fàcilment el fil d'excecució está produint el bloqueig. Una tècnica consisteix

en observar les senyals de control que estan entrant just abans de que es bloquegi la simulació. Mirar totes les senyals ocupades en els diversos fils d'execució, en un esquema gràfic de dos dimensions, ajuda a detectar el procés que està produïnt el bloqueig.

Situació 3.-Races

En aquesta situació hi han dos un processos *pMPEG*. El segon *pMPEG* està situat a la posició (2,2). Els dos tenen característiques semblants (p.ex tempsde procés, temps de codificació, etc). Enviem el primer *macro block* en uns quants paquets desde el *SourceMPEG* I un segon *macro block* al segon codificador. Si la qualitat (Q) està definida el segon codificador pot acabar abans que el primer produïnt un *data race*. Això és perquè el temps per codificar un macro block no es dependència directa dels píxels d'entrada. Un *block* amb més soroll que un altre pot costar més temps a ser codificat. En aquest cas podem posar diferents etiquetes segons les posicions dels mòduls per tal de veure si l'ordre en el temps es correcte.

V. Possibles millores futures

Com a possibles millores podríem enumerar varies:

- Millorar l'analitzador sintàctic de *SystemC* ja que l' estar basat en un projecte tancat no s'han establert unes regles prou generals per tal de poder llegir qualsevol projecte en *SystemC* sense unes adaptacions prèvies, així com millorar l'anàlisi sintàctic del programa principal de *SystemC* en la part referent als enrutadors ja que la creació d'aquests i l'assignació de senyals als ports es calcula en temps d'execució, i en el nostre cas s'ha extret aquest codi i s'ha simulat durant l'anàlisi sintàctic degut a l'extensió del programa, però no de forma genèrica, per a qualsevol programa *SystemC*.
- La possibilitat de crear un fil d'execució que calculi els estats futurs dels senyals mentre l'usuari va explorant els valors dels senyals. Aquest càlcul paral.lel a l'entorn gràfic donaria peu a un ampli ventall d'opcions per tal de trobar errors ja que es podria crear un formulari de búsqueda per a estats futurs filtrant per alguna condició com el temps d'activació d'un senyal o una condició de percentatge d'us d'alguna memòria intermèdia, així com fer una animació de la simulació i anar mostrant les activacions gràficament pausant aquesta quan fos necessària.

VI. Conclusions

He desenvolupat la eina proposant un sistema de descripció del codi SystemC en XML. Aquest serveix per donar una primera validació al disseny inicial ja que mostra tots els mòduls, enrutadors i empaquetadors de la NoC analitzada i es pot validar la seva correcció.

Encara que aquest pas es podria haver saltat passant directament del codi SystemC a l'objecte *NoCConf* vaig considerar important el pas de generar les descripcions XML ja que s'obté una informació de la xarxa sense que sigui codi font d'un programa, a més de poder-se mantenir sense tornar a analitzar el codi SystemC si no s'ha canviat el sistema.

En el desenvolupament dels analitzadors de SystemC i dels arxius VCD se m'han plantejat molts problemes i errors amb els patrons de text a tractar i he hagut d'establir un ordre d'anàlisi molt estricte i moltes regles no contemplades a l'inici de plantejament dels mòduls analitzadors.

La part de la creació i el posterior anàlisi dels arxius XML ha sigut la part més sencilla, ja que *Java* está molt lligat a la tecnologia de l'XML. No aixií la part de la interfície d'usuari ja que les diferents possibilitats d'execució, la forma de multi-fil de l'entorn gràfic i el dibuix

La fàcil divisió del sistema dissenyat per al desenvolupament de l'eina en mòduls més petits, m'ha facilitat el procés de creació del sistema, a més de poder considerar cada mòdul per separat i profunditzar de forma molt més eficient en cadascú d'aquests.

En qualsevol cas, ha estat un projecte motivador ja que el progrés ha estat escalonat, les eines utilitzades són agradables pel programador i la tecnologia tractada és molt interessant

Referències

[1] Dac C. Pan et Al, "Overview of the Architectura, Circuit Design, and Physical implementation of a Frist- Generation Cell Processor", IEEE Journal of Solid-State Circuit, Vol41, No.1, January 2006

[2] Benini L.; al. "Networks on chips: a new SoC paradigm". IEEE Computer, 35(1), Jan. 2002, pp. 70-78.

[3] CaEin Ciordus, Twan Basten, Andrei Rddulescu, Kees Goossens ,"AN EVENT-BASED NETWORK-ON-CHIP MONITORING SERVICE", High-Level Design Validation and Test Workshop, 2004. Ninth IEEE International.

[4] Santiago Gonzalez Pestana, Edwin Rijpkema, Andrei R˜adulescu, Kees Goossens and Om Prakash Gangwal, "Cost-Performance Trade-offs in Networks on Chip: A Simulation-Based Approach", Proceedings of the Design, Automation and Test in Europe Conference 1and Exhibition (DATE'04).

[5] D. Castells-Rufas, J. Joven, J. Carrabina, "A Validation and Performance Evaluation Tool for ProtoNoC". International Symposium on System-on-Chip 2006 (SOC2006). Tampere, Finland, November 13-16, 2006.

[6] Bruce Eckel, "Thinking in Java", 3rd edition BETA, Jan 2003 .

[7] Elliotte Rusty Harold," Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and TrAX",Barnes & Noble, 2003

[8] A. Portero, G. Talavera, M. Montón, B. Martinez, M. Moreno, F. Cathoor, J. Carrabina. "Energy-aware MPEG-4 Single Profile in HW-SW Multi-platform implementation" . SOCC: IEEE International SOC Conference. September 24-27, 2006. Austin, USA

[9] A. Portero, R. Pla, A. Rodriguez, J. Carrabina. "NoC Design of a Video Encoder in a Multiprocessor System on Chip Solution". The 17th IEEE International Conference on Microelectronics (ICM2005).Islamabad, Pakistan, December, 13-15, 2005.

[10] Ahmed Jerraya and Wayne Wolf (editors), "Multiprocessor System-on-Chip", Elsevier Morgan Kaufmann, San Francisco, California, 2005.

[11] Grant Martin, "Overview of the MPSoC Design Challenge", DAC July 24-28, 2006 Sant Francisco.

[12] Eclipse - an open development platform Documentation v.3.2 http://www.eclipse.org/

[13] http://home.nc.rr.com/gtkwave/ (gtkwave website)

[14] http://www.systemc.org (SystemC Organization)

Apèndix 1. Entorn de desenvolupament del projecte

En aquest apèndix s'expliquen les tecnologies utilitzades durant desenvolupament de l'eïna en la que es basa aquesta memòria.

S'ha utilitzat XML per a guardar les descripcions de les classes i els objectes del sistema *SystemC* i com a llenguatge per a desenvolupar el programa en si s'ha optat per *Java* i tot un conjunt d'APIs (*Application Programming Interface*) i llibreries de classes que s'han triat per facilitar la programació dels diferents mòduls del que consta el projecte.

XML

XML (eXtensible Markup Language) és un conjunt de regles (es poden considerar com directrius o convencions) per a dissenyar documents de text, les quals permeten l'estructuració de les dades. XML és un meta-llenguatge extensible d'etiquetes, desenvolupat pel *World Wide Web*(W3C), i permet definir la gramàtica de llenguatges específics.

Juga un paper fonamental en l'intercanvi d'una gran varietat de dades. No cal ser programador per usar-ho o per aprendre'l. A més facilita que l'ordinador generi dades, en llegeixi, o s'asseguri que l'estructura de les dades no és ambigua i evita problemes típics del disseny de llenguatges: és extensible, independent de la plataforma, i suporta la internacionalització i la traducció.

```
<persona>
    <nom>Miquel Ángel</nom>
    <cognom>Gallego</cognom>
</persona>
```

Figura 1.1. Exemple de document XML

XML permet definir un nou format de document tot combinant i reutilitzant d'altres formats. Com que dos format desenvolupats independentment poden contenir elements o atributs amb el mateix nom, s'ha de tenir cura quan es combinen. Per eliminar la confusió de noms quan es combinen formats, XML ofereix un sistema d'espais de noms. XSL i RDF són bons exemples de formats basats en XML que usen espais de noms. Els esquemes XML estan dissenyats per a emular aquest suport de la modularitat, ja que simplifiquen la combinació de gramàtiques per a produir una gramàtica derivada que cobreix una estructura fusionada de documents.

Escollir XML com a base per a un projecte, és accedir a una comunitat amb moltes eines i enginyers amb experiència en la tecnologia. llavors, com que XML no necessita llicència, es pot muntar sobre XML qualsevol desenvolupament. El fet que el suport sigui gran i creixent comporta no estar lligats a un únic proveïdor. [11]

El llenguatge Java

És un llenguatge de programació orientat a objectes que permet l'execució d'un mateix programa en múltiples sistemes operatius gràcies a que el programa s'executa sobre una màquina virtual que és la que varia segons el sistema operatiu i que s'encarrega de traduir el programa a codi màquina.

Les seves característiques es basen en que és un llenguatge simple, orientat a objectes, distribuït, interpretat, robust, segur, amb una arquitectura neutra, portable, amb altes prestacions, multi-fil (*multithread*)

Inclou treball en xarxa per defecte, es pot executar codi generat per ell en sistemes remots de forma segura i és fàcil agafar el millor d'altres llenguatges orientats a objectes (com el C++). Comparteix la sintaxis amb aquest llenguatge però elimina l'ús de punters i té un model d'objectes més simple[12].

Expressions regulars amb Regex

Aquesta classe de Java ens permet fer un anàlisi sintàctic del codi *SystemC* ja que es basa en els patrons (*pattern*), o expressions regulars, per treballar.

Els objectes de la classe *pattern* de Java tenen mètodes associats per tal de detectar-los en un text d'entrada, i posteriorment substituir-lo, recuperar el text que compleix l'expressió regular, etc.

Definint patrons com la paraula *include* podem detectar tots els arxius capçalera (*header*) referenciats en l'arxiu principal (*main)* del *SystemC*, amb el patró SC_MODULE, podem detectar les definicions dels mòduls en aquests arxius capçalera i continuant definint patrons determinats per les paraules reservades pròpies del *SystemC* podem anar construint una descripció de les classes que modelen la NoC i posteriorment els objectes que instancien aquestes classes, junt amb l'assignació dels ports a aquestes senyals.

Així doncs, aquesta classe ens facilita la trobada dins el codi font *SystemC* (i posteriorment dins de les traces VCD) de patrons per a extreure la informació necessària per a construir la nostra estructura d'objectes que serviran per mostrar una representació gràfica de la NoC estudiada.
A continuació es veu un exemple de la potència dels patrons a la classe *Regex*

```
/**
 * Funcio que parseja el arxiu main en busca dels arxius de headers
 */
public void buscaHeaders()
{
  String sFileLine = new String();
  Pattern pattern1 = Pattern.compile("^#include ");
  Pattern pattern2 = Pattern.compile("[a-z|A-Z|0-9]+.h");
  Pattern pattern3 = Pattern.compile("\"");
  Pattern pattern4 = Pattern.compile("[^systemc.h]");

  try
  {
    BufferedReader in = new BufferedReader(new FileReader(this.systemCMainFile));

    String nom;
    Matcher m;

    while ((sFileLine = in.readLine()) != null)
    {
      m = pattern1.matcher(sFileLine);

      if(m.find())
      {
        nom = m.replaceAll("");
        m = pattern2.matcher(nom);
        if(m.find())
        {
```

Figura 1.2. Exemple de codi que busca mitjançant patrons els noms d'arxius capçalera

SAX (*Simple API for XML*)

SAX son les sigles de *Simple API for XML*. Originàriament era una API per a l'anàlisi sintàctic de XML per a Java però posteriorment va passar a ser un estàndard de facto. Inclús s'ha propagat el seu ús a altres llenguatges de programació.

SAX es basa en fer un recorregut seqüencial pels elements del document XML. Això es fa detectant events d'inici de document (`<?xml version="1.0" encoding="UTF-8"?>`), inici d'etiqueta d'un element, caràcters interns entre dues etiquetes , final d'etiqueta i final de document.

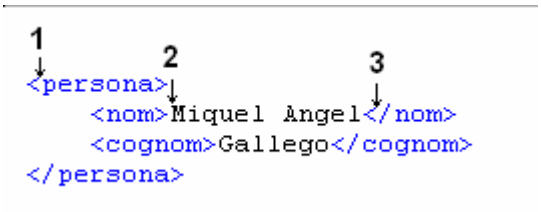


Figura 1.3. Exemple d'inici d'etiqueta (1), caràcters entre etiquetes (2) i final d'etiqueta

Aquests estan definits en un *handler* (o manipulador) que s'ha de reescriure per tal d'aplicar la funcionalitat particular del programa. Així doncs, es reescriuen els mètodes d'inici de document i d'etiqueta per tal d'inicialitzar els atributs del nostre programa, o reescriure el mètode de llegir caràcters entre dues etiquetes perquè actuï segons necessitem (per afegir-lo a les llistes d'elements de la nostra NoC, o a la llista de connexions de dintre el nostre *Router*, etc)

Les avantatges d'aquesta API respecte a altres alternatives, es que no consumeix gaire memòria ja que no es té que generar l'arbre del document en memòria com fa una altra alternativa anomenada DOM (*Document Object Model*)

Les característiques d'aquesta API fa que sigui la candidata perfecta pel nostre propòsit de llegir arxius XML, ja que no consumirà gaire memòria i al mateix temps que es va recorrent l'arxiu XML, es va generant l'estructura que simularà la NoC.

JDOM (*Java Document Object Model*)

És una llibreria de codi font per a analitzar sintàcticament, manipular i crear documents XML. Encara que s'assembla a DOM, que pertany al consorci *World Wide Web*(W3C), és una alternativa no inclosa en DOM. Està optimitzada per ser més fàcil d'utilitzar pels programadors de Java, i és ràpida, consumeix pocs recursos i fiable.

Aquesta facilitat per a manipular i crear documents XML partint de Java la fa molt interessant ja que seguint ordenadament una estructura jeràrquica d'objectes es pot construir el corresponent XML que el descriu. Un exemple es pot veure a continuació:

```
Element root = new Element ("NOC");

Element item1 = new Element ("SC_ROUTER");
item1.setAttribute ("num", "1");

...

root.addContent (item1);          // Afegim l'itmem al root de l'element

XMLOutputter outputter = new XMLOutputter ("",true);
try{
   outputter.output (new Document(root),new FileOutputStream ("classes.xml"));
} catch (Exception e){
   e.getMessage();
}
```

Figura 1.4. Exemple d'una construcció simple d'un document XML amb JDOM

D'aquesta manera serveix per a crear els documents XML descriptius de les classes i els objectes de la NoC implementada en SystemC, un cop parsejat aquest codi font.

DB4O (*Data Base 4 (for) Objects*)

En un projecte de programari (Software), en la fase del disseny, es té que arribar a la conclusió de si farà falta una base de dades o no. Inicialment pot ser únicament per guardar l'estat de l'aplicació, i la majoria de vegades amb l'objectiu de mantenir el valor de les dades independentment de si el programa està executant-se o no.

Si es decideix que aquesta es necessària, i el llenguatge en el que el projecte de programari serà realitzat és un llenguatge orientat a objectes com Java, entra en joc en el disseny del programa el disseny pel model relacional per tal d'emmagatzemar les dades, a part del disseny pel model d'objectes en el que es basarà l'aplicació.

Aquests dos models, s'han de fusionar d'alguna manera per tal de mantenir una coherència entre els objectes i les dades que s'emmagatzemaran, per tant, s'ha de dissenyar una capa que controli el mapeig dels objectes cap al model relacional, que és el que entén una base de dades tradicional. Aquesta capa també controla el camí invers, és a dir, el mapeig de les dades des del model relacional cap al model d'objectes.

Per a solucionar aquest increment en el temps de disseny, i facilitar la manipulació i la persistència dels objectes, neixen les bases de dades orientades a objectes.

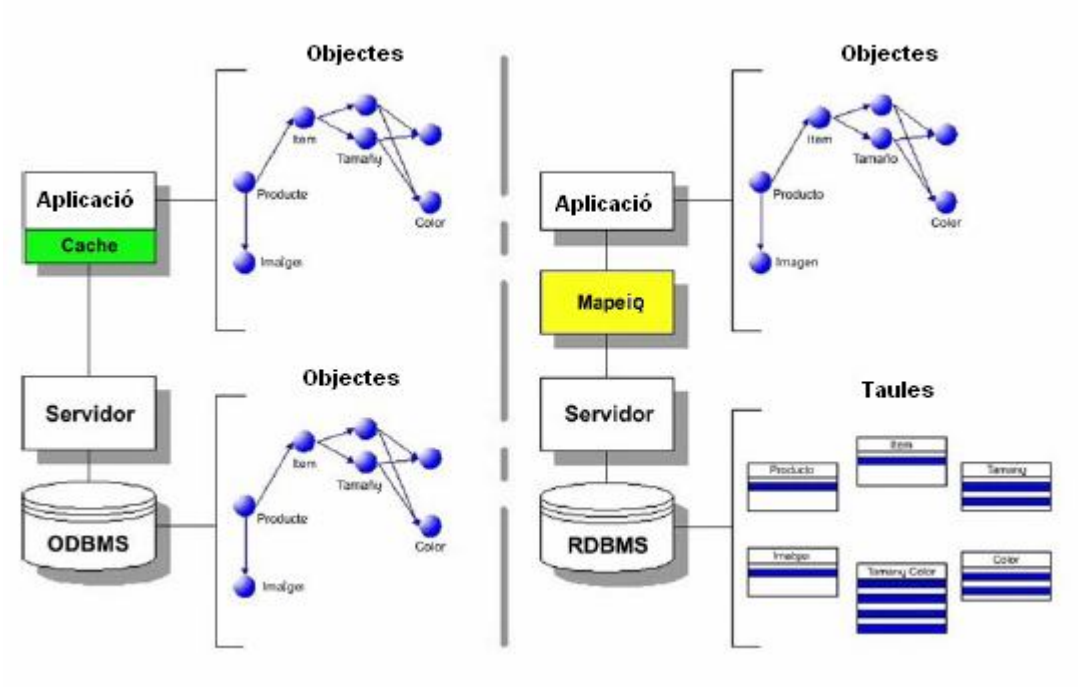


Figura 1.5. Emmagatzematge directe vs. Mapeig d'objectes

DB4O (*Data Base 4(for) Objects*) és un modern motor de base de dades orientat a objectes (o base d'objectes) que funciona com una llibreria per a Java i per a .NET per tal d'emmagatzemar objectes, en comptes de les bases de dades relacionals tradicionals. Està desenvolupat per una empresa amb el mateix nom *db4objects, Inc* i ja tenen disponible la versió 6.d'aquest motor, utilitzat per grans projectes com el control del trens AVE (Indra.)

La seva llicència es mixta (GPL/Comercial), si va destinada a un programa de codi obert no es té que pagar una llicència però si va destinat a un programari privatiu la llicència s'ha de pagar,

DB4O és un motor molt potent i ràpid, sobretot en la seva versió encastada (*embebed)* que no necessita un llenguatge de consultes com SQL per recuperar els objectes de la base d'objectes, sinó que es fa aplicant restriccions als atributs dels objectes a recuperar i definint la classe del objecte a recuperar.

En aquest projecte es fa ús d'aquest motor, en la forma encastada (*embebed),* per tal de guardar les rutes dels arxius XML de classes i d'objectes que descriuen la NoC i no tenir que indicar altra vegada quins arxius són, a més de guardar la ruta i l'estat del parseig de l'arxiu

VCD, de forma que quan volem tornar enrere en el temps de les traces VCD només hem de recuperar els objectes corresponents a l'estat anterior de la base d'objectes i no tornar a calcular els atributs d'aquests. Un exemple de la recuperació de l'estat de la NoC de la Base d'objectes:

```
NoCState nocState = new NoCState();
nocState.setNTimePosition(nInTimePosition);
nocState.setNocConf(inNoC);

if(nInTimePosition < this.numStates)
{
  Query q = this.dataBase.query();
  q.constrain(PortTimeState.class);
  q.descend("nTime").constrain(nInTimePosition);
  ObjectSet result = q.execute();

  while(result.hasNext())
  {
    PortTimeState portTimeState = new PortTimeState();
    portTimeState = (PortTimeState)result.next();

    nocState.setPortState(portTimeState);
    nocState.setSTime(portTimeState.getSTime());
  }
  return nocState;
}
```

Figura 1.6. Exemple de recuperació d'objecte de la Base d'Objectes

Interfícies gràfiques amb Swing

Swing és una biblioteca gràfica de Java que forma part de les JFC (*Java Foundation Classes*) des de la versió 2. Conté objectes predefinits que es poden invocar i configurar per a incloure en una *GUI*, com poden ser els camps de text i botons (molt útils per a formularis), taules, panells, llistes desplegables, taules de valors.

Aquestes característiques junt amb l'existència de *plugins* per a la creació gràfica per les eines de desenvolupament de Sw (IDE) per Java com *Eclipse* i *NetBeans*, fa que sigui molt utilitzada per a la creació d'interfícies gràfiques d'usuari (GUIs).

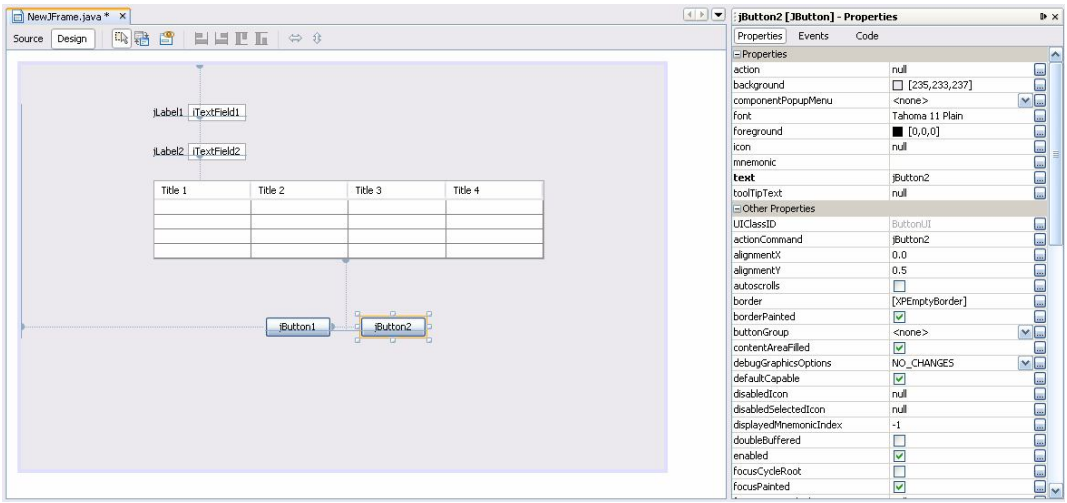


Figura 1.7. IDE Netbeans amb plugin per a Swing

Classe *graphics*

Aquesta classe és el sistema bàsic de totes les operacions gràfiques en Java. Pertany al paquet *java.awt*.

Possibilita al programador dibuixar o col·locar imatges dins d'un component contenidor, com un JPanel de Swing, ja que suporta un gran ventall de mètodes.

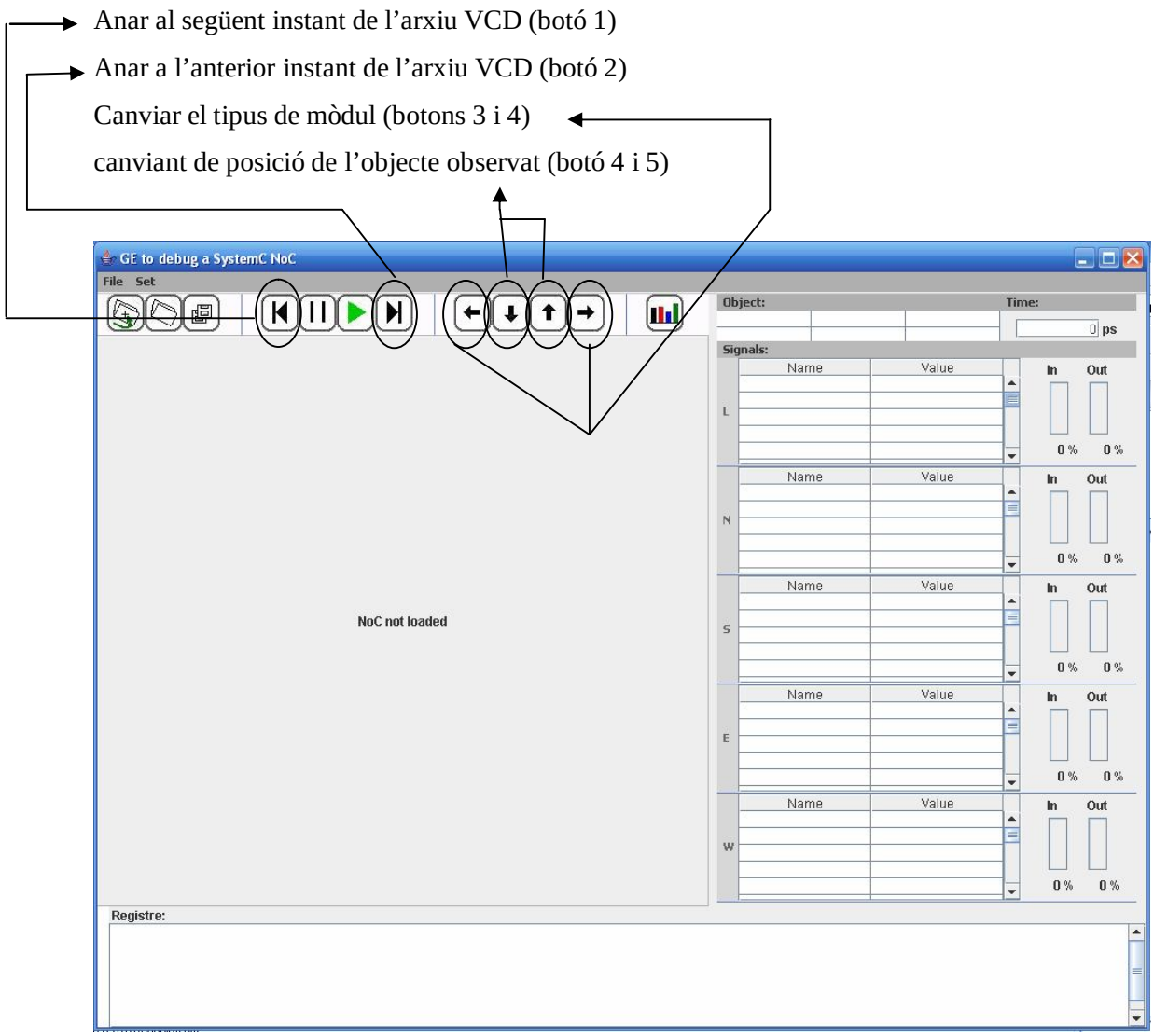
El seu funcionament consisteix en encapsular tota la informació que afecti a les operacions de dibuix en un context gràfic, de manera que la sortida estàndard, que és una finestra o panell, té que passar primer per aquest context.

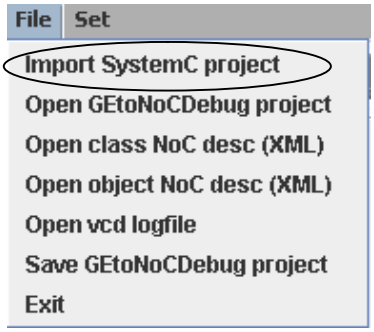
És una classe abstracta, per tant no es pot crear un objecte directament, sinó que és té que passar el context al programa a través dels seus mètodes *paint() repaint() i update()*.

Aquesta classe, la fem servir per a dibuixar els diferents mòduls de la NoC, els ports d'aquests, així com, les connexions entre els diferents mòduls.

Apèndix 2. Guia del programa

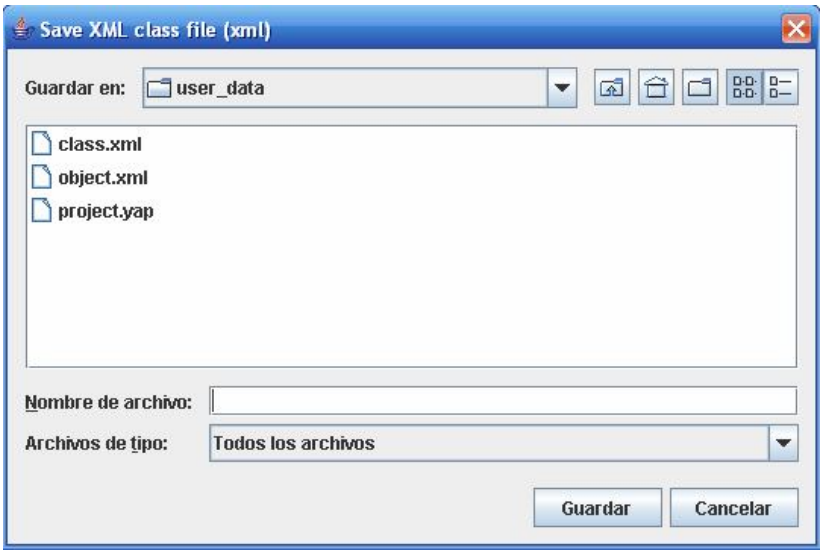
En aquest apèndix es dóna una guia per a l'execució de l'eina amb imatges d'exemple. Per a l'execució del programa ens hem basat en el codi font d'una NoC on hi ha una tasca que és un compressor MPEG.



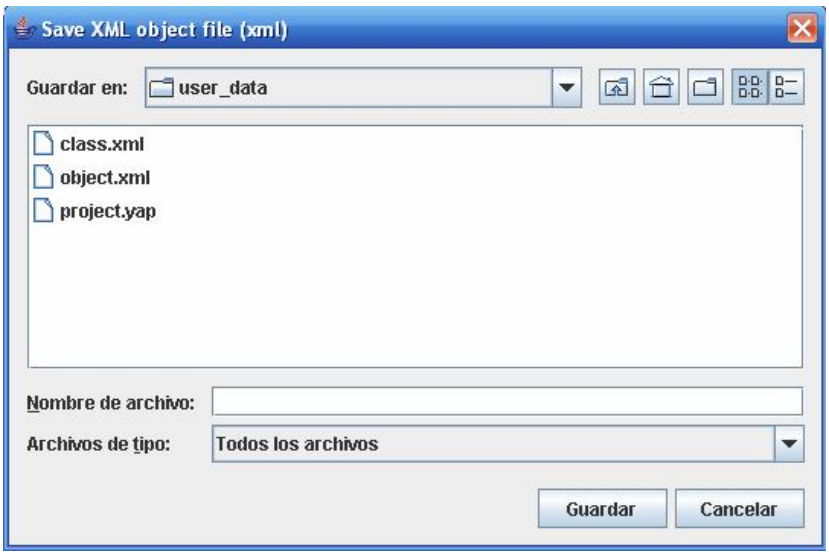


En una execució primer obrirem el projecte *SystemC* fent una importació de l'arxiu principal (*main)* del projecte per tal de poder analitzar el codi font i extreure la configuració de la NoC i dels mòduls, *router*s i *wrappers* que la contenen

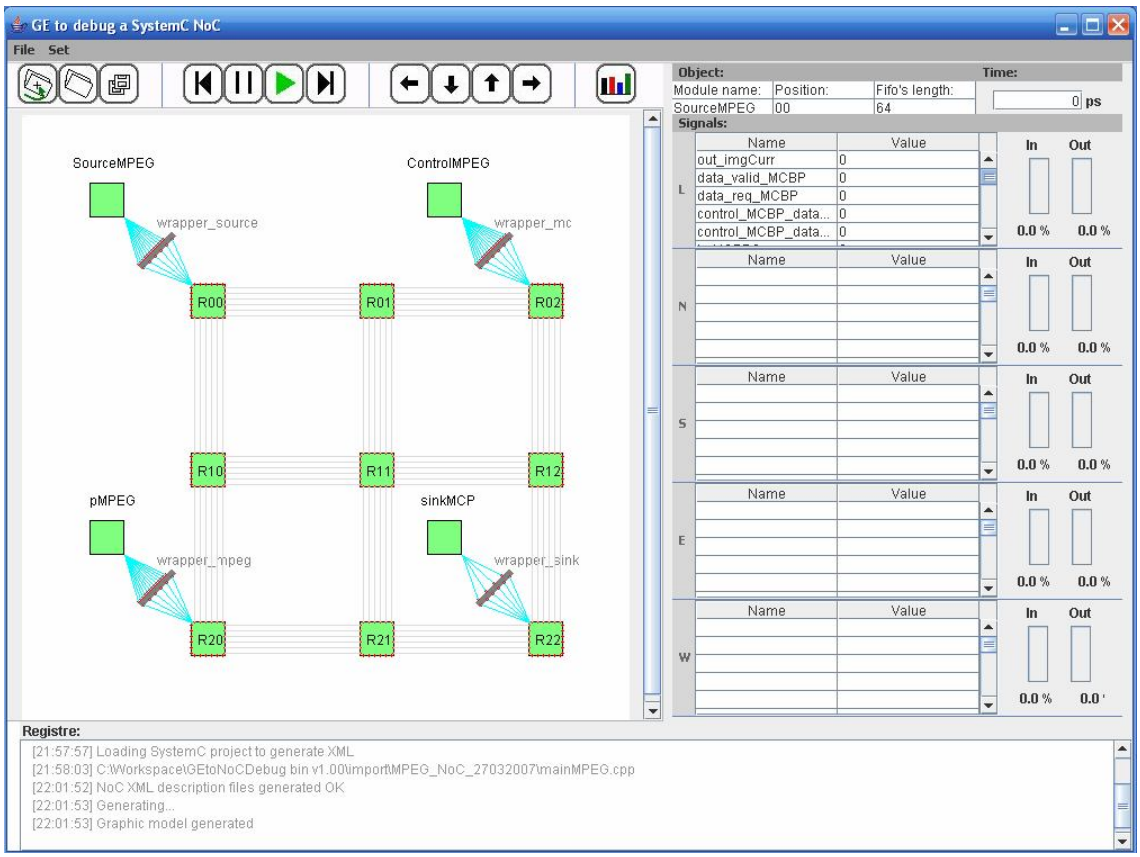
Un cop importat el projecte ens demanarà el nom dels arxius XML per a guardar una descripció de les classes:



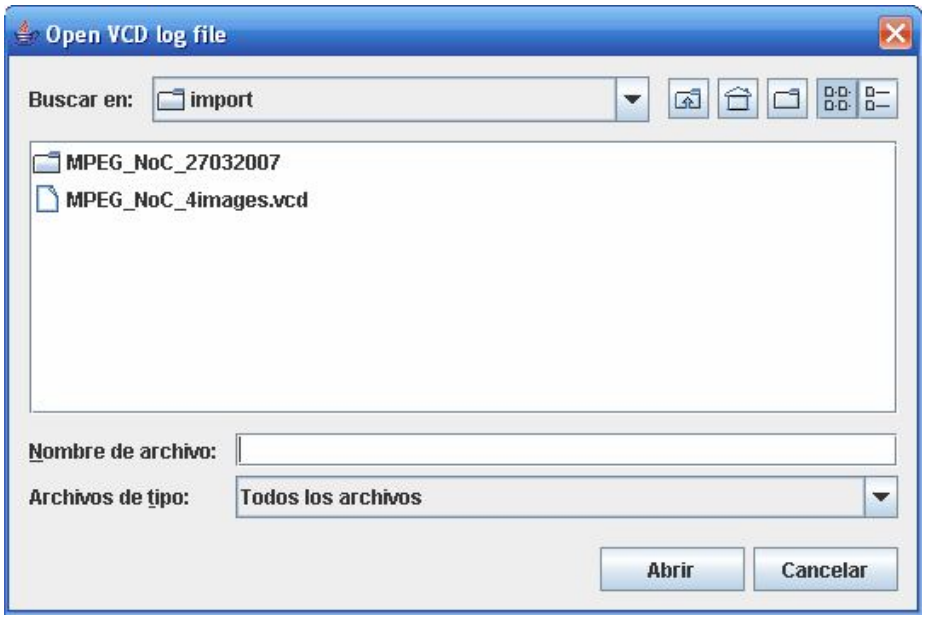
I per a guardar una descripció dels objectes:



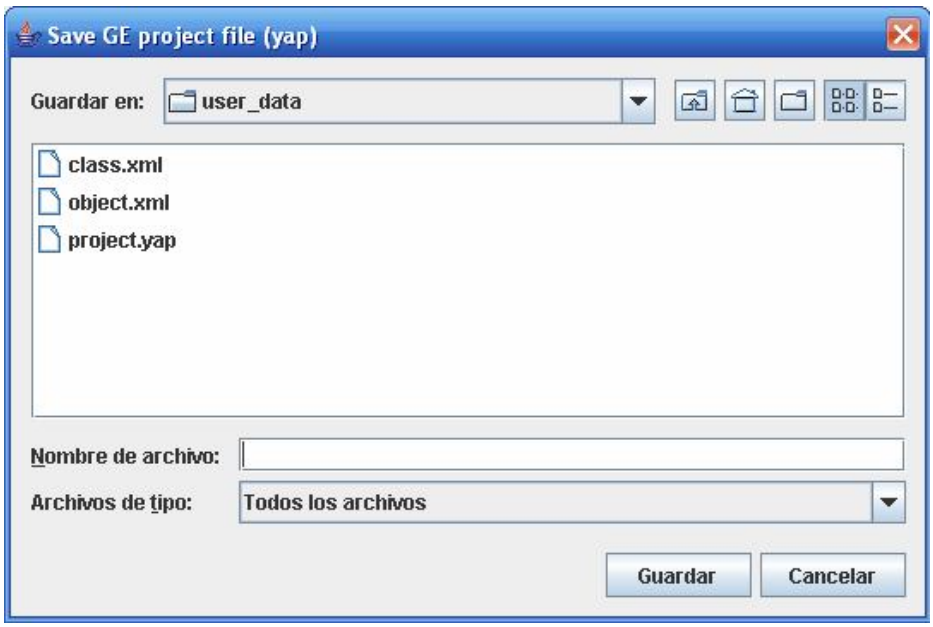
Tot seguit, el programa dibuixa la representació gràfica de la NoC descrita al projecte *SystemC*:



Una vegada el programa ha dibuixat la NoC obrirem l'arxiu VCD per tal de començar el seguiment de l'execució:

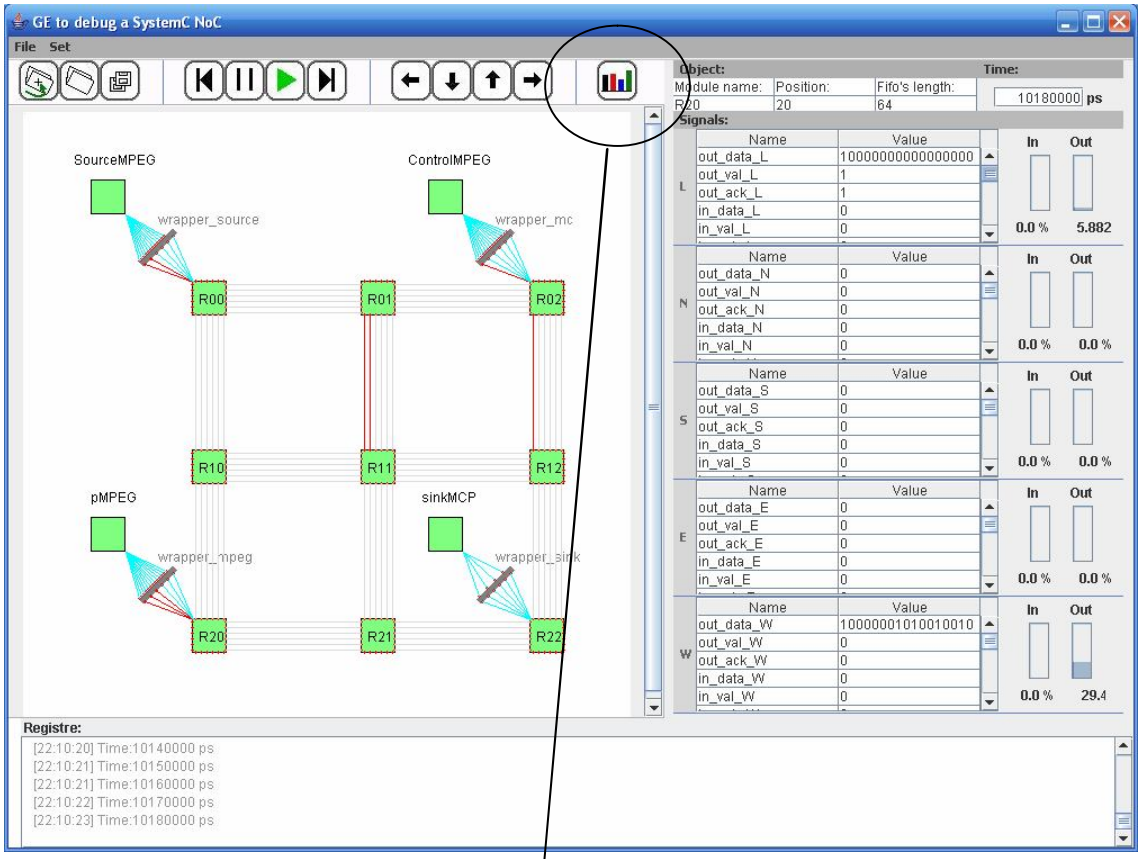


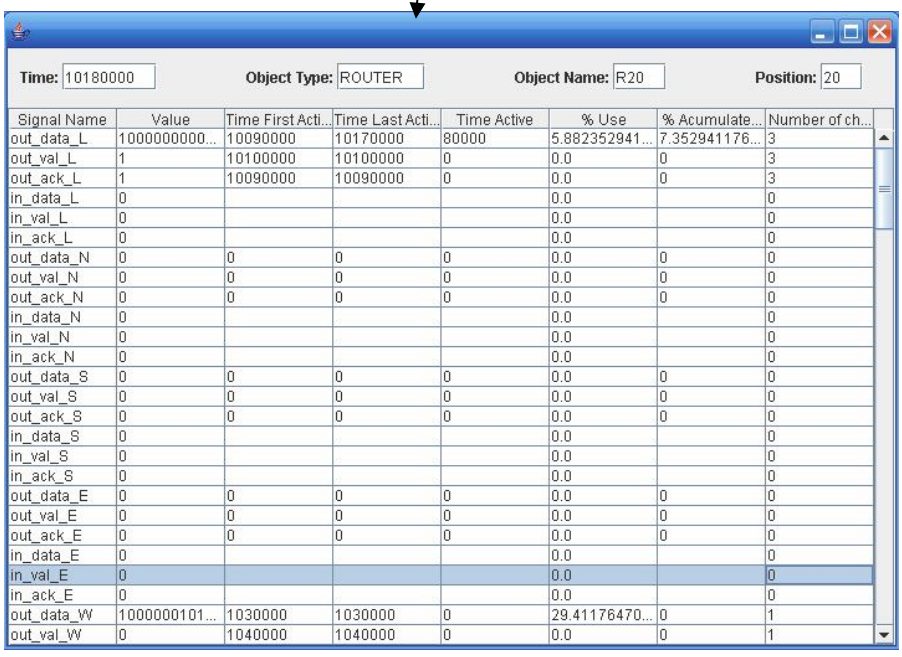
En aquests moments el programa ens demana el nom d'un arxiu per tal de guardar el nostre projecte. Aquest arxiu té que tenir l'extensió *.yap* ja que és l'extensió suportada pel motor DB4O:



Un cop a arribat aquest punt, el programa inicialitza tots els objectes que serviran per fer el seguiment de l'execució que tenim descrita a l'arxiu VCD i es comença a analitzar sintàcticament l'arxiu. El programa es queda en l'instant 0 esperant a que nosaltres fem clic en els botons d'anar al següent instant de l'arxiu VCD (el botó 1) o enrere (el botó 2) o moure'ns pels diferents mòduls de la NoC canviant els tipus de mòduls per tal de veure els valors de les seves connexions en l'instant en el que estem (botons 3 i 4) o canviant de posició de l'objecte observat (botons (4 i 5)).

També es pot veure una taula amb un seguit de valor estadístics prou interessants com el temps en que s'ha activat per primera vegada una senyal, o el temps de la última activació, o el % d'us de la memòria intermèdia de sortida o d'entrada de qualsevol mòdul, etc.





Time: 10180000 Object Type: ROUTER Object Name: R20 Position: 20

Signal Name	Value	Time First Acti...	Time Last Acti...	Time Active	% Use	% Acumulate...	Number of ch...
out_data_L	1000000000...	10090000	10170000	80000	5.882352941...	7.352941176...	3
out_val_L	1	10100000	10100000	0	0.0	0	3
out_ack_L	1	10090000	10090000	0	0.0	0	3
in_data_L	0				0.0		0
in_val_L	0				0.0		0
in_ack_L	0				0.0		0
out_data_N	0	0	0	0	0.0	0	0
out_val_N	0	0	0	0	0.0	0	0
out_ack_N	0	0	0	0	0.0	0	0
in_data_N	0				0.0		0
in_val_N	0				0.0		0
in_ack_N	0				0.0		0
out_data_S	0	0	0	0	0.0	0	0
out_val_S	0	0	0	0	0.0	0	0
out_ack_S	0	0	0	0	0.0	0	0
in_data_S	0				0.0		0
in_val_S	0				0.0		0
in_ack_S	0				0.0		0
out_data_E	0	0	0	0	0.0	0	0
out_val_E	0	0	0	0	0.0	0	0
out_ack_E	0	0	0	0	0.0	0	0
in_data_E	0				0.0		0
in_val_E	0				0.0		0
in_ack_E	0				0.0		0
out_data_W	1000000101...	1030000	1030000	0	29.41176470...	0	1
out_val_W	0	1040000	1040000	0	0.0	0	1

Apèndix 3. A Planificació del desenvolupament del projecte i Presupost.

En aquest apèndix s'inclou la planificació seguida per a desenvolupar el sistema dissenyat. Que va des de l'especificació del projecte, fins al desenvolupament de la memoria, en total 33 setmanes, on li he dedicat una mitja de 15 hores per setmana. En total li he dedicat unes 495hores que està bastant més enllà dels 15 credits ECTS (15*25 hores alumne = 375 hores) d'un Projecte Final de Carrera a Enginyeria Informàtica. Aquest projecte s'ha allargat a més de la complexitat que ha tingut que no ha estat menyspreable al fet que treballo a jornada complerta.

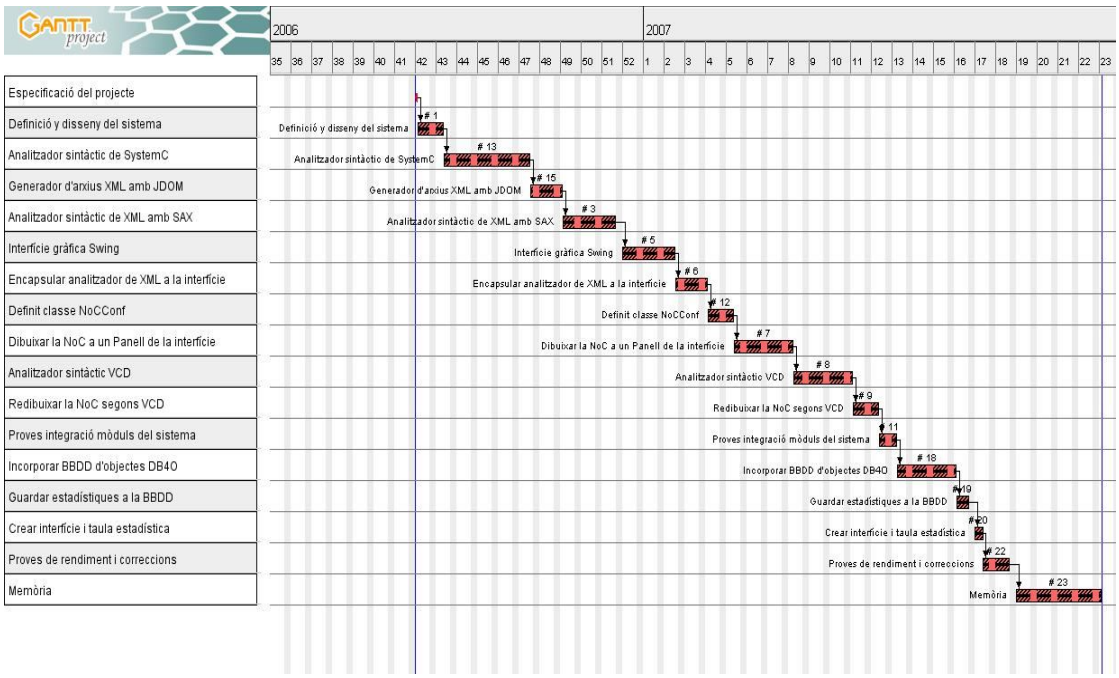


Figura 3.1. Diagrama de gantt amb la planificació del projecte

Presupost

A partir de les hores dedicades en aquest projecte s'ha fet una estimació de presupost. El presupost està estimat a partir del que un Enginyer Informàtic guanya. El que pot guanyar un Enginyer Informàtic pot variar bastant depenent de la seva experiencia. Hem fet dos pressupostos un de mínims i un altre de màxims.

Presupost orientatiu mínim (30 euros/hora)
-Hores dedicades per setmana 495 hores * 30 euros/hora
Total……. 14850
+16%IVA ……2376
Total a pagar…… 17226

Presupost orientatiu màxim (60 euros/hora)
-Hores dedicades per setmana 495 hores * 60 euros/hora
Total……. 29700
+16%IVA ……4752
Total a pagar…… 34452

Apèndix 4. Explicació dels SoC i una forma d'interconnectar els IPs a través d'una NoC

System on Chip (SoC)

Amb la proliferació de mòduls ja provats i verificats que modelen components especialitzats, que es coneixen tècnicament com *Intellectual Property blocks* (IP), cada vegada més petits, gràcies a la creixent densitat d'integració dels transistors, es dona la possibilitat de implementar els *System on Chip* (SoC). Aquests són més estructurats, i el seu desenvolupament no comporta un cost excessiu, així com tampoc una complicació del disseny

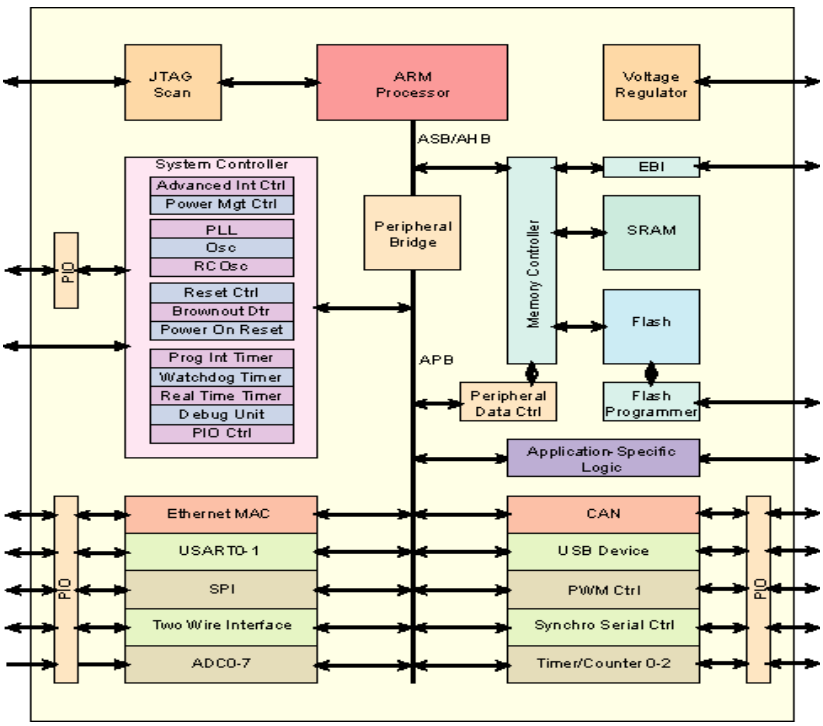


Figura 4.1. Microcontrolador basat en una SoC

La principal manera de connectar components són les connexions punt a punt i els busos. Encara que aquests últims ens permetin una compartició del recurs i unes velocitats altes no tenen un cost baix ja que la longitud i la complicació afegida de prestar atenció al dibuix dels fils que el formen compliquen el disseny. Solen tenir també un excés de càrrega que afecta al consum i a la velocitat.

S’intenten millorar aquestes interconnexions adoptant solucions com el bus segmentat amb ponts entre components contigus (situats així per la freqüència de comunicació entre ells) però no acaba d’eliminar el problema.

Aquest gran problema junt amb la dificultat creixent de la validació dels sistemes dissenyats i les necessitats latents en termes de comunicacions i de reutilització (cada sistema té un esquema diferent de comunicació) fa que comencin a proliferar nous paradigmes com les *Network on Chip* (NoC).

Network on Chip (NoC)

Últimament havien sorgit varis paradigmes partint de la necessitat d’una nova forma d’interconnexió en xarxa. L’escollida per damunt de totes és la *Network on Chip* (NoC).

En una NoC, mòduls com nuclis de processador, memòries, blocs IP (Intellectual Property blocks) i clusters de funcionalitat intercanvien dades codificades en els paquets de bits, fent servir una xarxa com el subsistema de "transport públic" pel tràfic de la informació.

Les solucions NoC aporten mètodes de xarxes de commutació per a la comunicació interna d’un xip.

Una NoC es construïda a partir de múltiples enllaços de transmissió point-to-point interconnectats per interruptors (*switch*/*routers*), tal que els missatges poden ser retransmesos des de qualsevol mòdul font a qualsevol mòdul de destí a través de varis enllaços, fent les decisions d’encaminament als enrutadors. Un NoC es similar a una xarxa de telecomunicacions moderna, fent servir la commutació de paquets de bits sobre enllaços multiplexats.

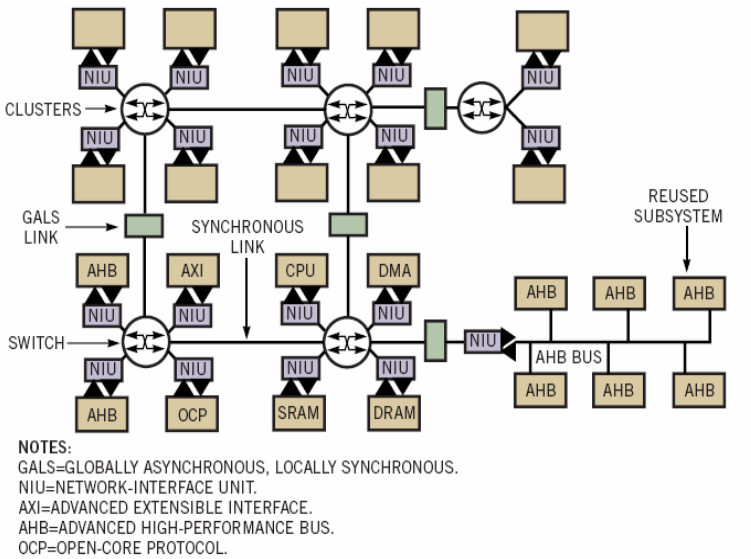


Figura 4.2. Exemple de *Networking* aplicat a les comunicacions on-chip per l'empresa Arteris

El paral·lelisme i l'adaptabilitat

Les connexions en els enllaços de la NoC són compartits per moltes senyals. S'aconsegueix un alt nivell de paral·lelisme, perquè tots els enllaços en la NoC poden funcionar simultàniament sobre paquets de dades diferents. Per tant, com la complexitat de sistemes encastats segueix creixent, una NoC proporciona el funcionament millorat i l'adaptabilitat en comparació amb arquitectures anteriors.

Varies forces condueixen a l'adopció de l'arquitectura NoC: des d'un punt de vista de disseny físic, en la nanotecnologia CMOS, les interconnexions dominen tant el funcionament i la dissipació dinàmica d'energia, com la propagació del senyal a través del xip que requereix múltiples cicles de rellotge.

Els enllaços de la NoC poden reduir la complexitat de dissenyar connexions al poder preveure una velocitat, energia, soroll, fiabilitat, etc, gracies a la seva estructura regular i controlada.

Encara que a nivell de NoC les comunicacions són asíncrones, a nivells interns es pot mantenir una sincronització que permet la comunicació eficaç entre dos mòduls. Permeten també una modularitat en el disseny, reutilització parcial (mòduls i IP) i, d'aquesta forma es nota en el cost de temps de producció, per tant s'augmenta la productivitat.

Encara que la NoC pot agafar conceptes i tècniques del domini ben establert de xarxes de computadors, és poc pràctic reutilitzar a cegues les característiques de xarxes de computadors "clàssiques" i multiprocessadors simètrics. En particular, els interruptors(routers) NoC deuen de ser petits, eficients amb l'energia, i ràpids. Els algoritmes d'encaminament deurien de ser posats en pràctica per la lògica simple, i el número de buffers de dades deu de ser mínim. La topologia i les propietats de la xarxa poden ser específiques d'una aplicació concreta.

I no es trivial l'elecció d'una topologia de xarxa ja que el que, a simple vista, pot semblar una topologia idònia, però si s'analitza en profunditat s'arriba a la conclusió que la topologia triada aporta uns beneficis que no concorden amb el cost de producció, i potser ni aporten les avantatges de temps ni de latència que a priori l'herència de la teoria de xarxes de computadors ens invita a deduir. És a dir, que s'ha de tenir una especial cura amb les característiques implícites del xips en el moment de triar l'arquitectura de la xarxa,

No obstant, molts problemes de investigació estan pendents de ser solucionats a tots els nivells, des de la capa d'enllaç físic fins la capa d'enllaç, arribant fins l'arquitectura del sistema i el software d'aplicació.

Apèndix 5. MPEG

El vídeo és un conjunt de imatges o *frames*, que mostrades de forma continua i regular en el temps donen la seqüència de vídeo. La quantitat de *frames* per segon varia segons el sistema, però el més comú són 30 *frames* per segon (25 en el cas del sistema de TV europeu o PAL).

La compressió engloba la part de compressió i descompressió de manera que la parella codificador/descodificador (coder/decoder – CODEC) serveixen per codificar una imatge font o seqüència de vídeo en un format comprimit, i descodificar-lo per produir una copia o aproximació de la seqüència d'entrada. Un cop s'ha fet la codificació i la descodificació, si la seqüència obtinguda és igual que l'original estarem parlant d'un sistema sense pèrdues (*lossless*) i si es diferent de la seqüència d'entrada, haurà estat amb pèrdues (*lossy*).

Per aconseguir una compressió, s'intenta reduir la redundància temporal que es dóna degut a la similitud entre imatges successives, i la redundància espacial que es dóna degut a la similitud entre píxels propers entre ells. Aquestes redundàncies són explotades normalment mitjançant prediccions de frames futurs per posteriorment compensar el moviment també amb prediccions.

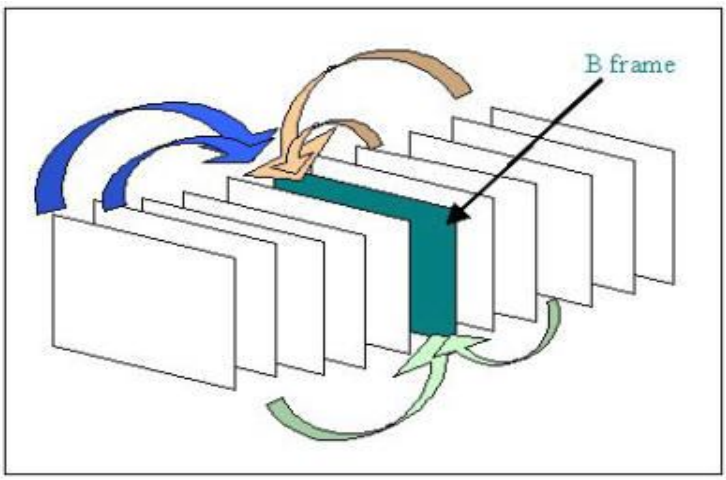


Figura 5.1. Successió d'imatges o frames

MPEG (*Moving Picture Experts Group*), és un grup de treball designat oficialment a través de l'organització d'estàndards ISO/IEC. Aquest grup s'encarrega de desenvolupar estàndards internacionals per a la compressió, descompressió, processat i codificació de vídeo, àudio i altres extensions.

MPEG està basat en MPEG per a imatges de tipus Intra (I), utilitzat en la compressió de les imatges estàtiques i MPEG per a les imatges de tipus P i B, on hi ha també compressió temporal

A partir de la codificació de vídeo, el sistema de compressió MPEG genera imatges codificades a partir de les imatges originals de la seqüència de vídeo. Podem distingir tres tipus d'imatges: imatges I, imatges P i imatges B.

- *I-pictures* o imatges *Intra*, contenen informació de la totalitat de la pantalla, únicament se'ls aplica una compressió espacial, presentant una compressió moderada. Són utilitzades com a referència per a la resta d'imatges.

- *P-pictures* o imatges predites, es codifiquen a partir de l'anterior *frame* I o P més proper, utilitzen algoritmes d'extracció de redundància temporal que permeten una compressió alta.

- *B-pictures* o imatges bidireccionals, es codifiquen a partir d'imatges I o P anteriors i futurs. Aquesta tècnica rep el nom de predicció bidireccional i utilitza els algoritmes d'extracció de redundància temporal presentant una compressió molt elevada. No son referència per a cap altre *frame*.

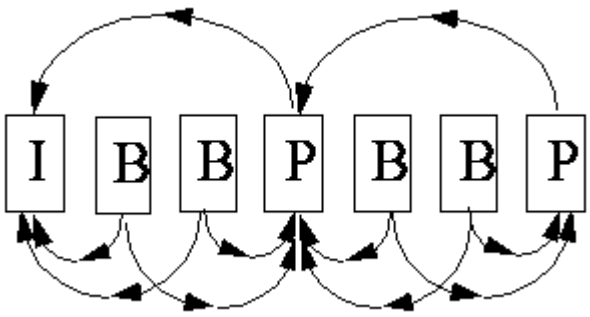


Figura 5.2. Patró d'imatges IBBP

Apèndix 6. SystemC

En el disseny electrònic dels dispositius lògics programables (PLD) s'estan començant a utilitzar eines sorgides per corregir veritables colls d'ampolla en el flux de producció de solucions.

Les noves metodologies de disseny, les exigències del mercat, junt amb la complexitat creixent dels dispositius forcen a trobar solucions alternatives per tal de reduir costos, i mantenir una fiabilitat en els dissenys als que s'arriba després de seguir el procés d'abstracció, jerarquia i estructuració.

Aquestes alternatives intenten millorar les limitacions a les que s'està arribant amb el llenguatge descriptiu del hardware (HDL). Aquest llenguatge es fa servir molt per descriure mòduls hardware que posteriorment seran sintetitzats a la *Field Programable Gate Array* (FPGA).

HDL té una gran importància en el disseny microelectrònic, ja que sobre el mateix han sorgit eines molt eficients, capaces d'optimitzar i sintetitzar un mateix codi HDL en PLDs amb recursos hardware molt diferents.

La naturalesa descriptiva del HDL avarca des de la descripció funcional del sistema fins l'arquitectura d'aquest. Per tant s'arriba a un nivell d'especificació molt alt, però a la vegada molt complexa, i a mesura que els mòduls permeten una major integració de components amb les funcionalitats cada vegada més especialitzades d'aquests, la feina dels dissenyadors es torna molt extensa en el temps.

Aquí és on entren en joc les alternatives que pretenen, i aconsegueixen, reduir aquesta càrrega temporal dels dissenyadors, fent que a partir d'un algorisme (en el cas de la descripció funcional) o una especificació de disseny (en el cas de l'arquitectura) generar codi de descripció hardware.

SystemC permet la generació automàtica de HDL a partir d'un codi C++. Aquesta metodologia és molt potent, ja que qualsevol algoritme expressat en aquest codi podria arribar a ser sintetitzable dins un PLD.

Avantatges temporals i de disseny

L'aparició de *SystemC* es deguda a la necessitat dels productors de dispositius electrònics a reduir les inversions en els temps de disseny, acurtant el cicle d'aquest procés.

I és possible ja que aporta als dissenyadors la avantatge d'estalviar-se la feina extra a l'hora de traduir un primer disseny d'alt nivell, ja que *SystemC* fa servir un llenguatge d'alt nivell com és el C++.

El fet de ser C++ comporta moltes avantatges ja que aquest és obert i flexible, permet definir tipus de dades abstractes fàcilment comprensibles per qualsevol dissenyador de sistemes, és molt usada per fabricants d'FPGAs per modelar i molt conegut a nivell mundial per personal relacionat amb la tecnologia de la programació, tant de la branca de la informàtica com de la electrònica i la física.

Fins l'aparició de *SystemC* l'ús tant comú del C++ en el disseny de sistemes era un handicap important ja que cada empresa dissenyava els seus mòduls adaptats a les seves necessitats i dependents dels seus dissenyadors, el que feia molt difícil el posterior intercanvi i reutilització d'aquests mòduls per altres sistemes al estar obligats a fer una traducció o adaptació d'aquests.

L'objectiu de *SystemC* es situa en la pretensió d'oferir un camí comú en el disseny de sistemes Hw i unificar el desenvolupament del Hw i del Sw.

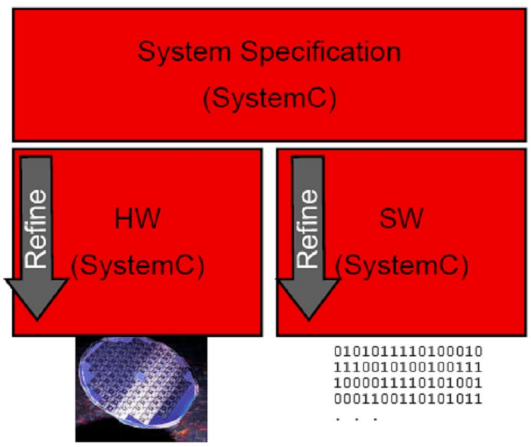


Figura 6.1. Metodologia de disseny Hw/Sw en SystemC

SystemC és un conjunt de llibreries basades en la programació sobre C++, però a diferència d'aquest, permet l'execució de les instruccions de diferents processos en paral·lel, això ens permet simular sistemes *multithread*

Una altra diferència entre C++ i *SystemC* és la corresponent al tipus de dades. C++ té tot un conjunt de tipus de dades, com són el enters, els de punt flotant, etc. i altres que es poden definir més complexes degut a les característiques del llenguatge. *SystemC* treu profit d'aquest avantatge i aconsegueix tipus de dades força útils en la descripció hardware, com són els SC_INT<A>, enters amb A bits per a la seva representació.

Cal destacar també una altra característica que diferència *SystemC* de C++, *SystemC* pot incloure especificacions de temps definides en funció d'un rellotge, és a dir, podem aconseguir simular processos amb determinades especificacions de temps.

Per poder implementar aquestes funcionalitats en C++, SystemC es va crear com una extensió d'aquest llenguatge mitjançant una nova llibreria (**libsystemc.a**).

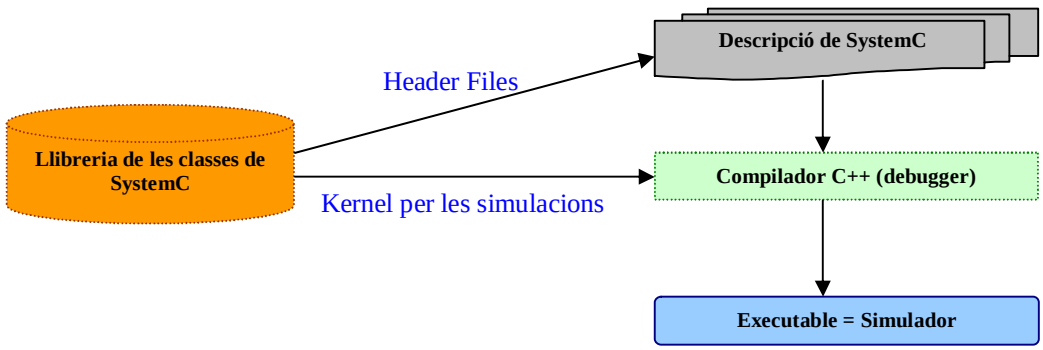


Figura 6.2. Infraestructura de *SystemC*

Modularitat i Comunicacions

La llibreria on s'implementen les funcionalitats addicionals per SystemC, **libsystemc.a**, està construïda a partir de la propietat d'herència que implementa C++ i a partir de la definició de nous tipus de dades que permeten modelar més acuradament el comportament del Hw.

Una vegada s'ha compilat la llibreria de *SystemC*, es pot modelar Hw de la mateixa manera que es programa habitualment en C++ ja que es pot barrejar lliurement les característiques de C++ i de *SystemC*.

L'aplicació C++, únicament haurà de fer un *include* "systemc.h" per poder accedir a les funcionalitats, classes i tipus de dades específiques de *SystemC.*

Arribant a aquest punt, es pot començar a modelar l'especificació del sistema que serà dividida en diferents mòduls per tal de facilitar la implementació de tot el sistema. Aquesta implementació serà desenvolupada sense tenir en compte si es Hw o Sw (co-disseny) i tenint en compte això ens estalviem moltes de les restriccions de recursos en el moment de definir l'arquitectura.

Aquesta especificació està descomposada en blocs funcionals, fet que acurta el desenvolupament al facilitar el disseny, el seguiment de l'evolució de les dades i la correcció d'errors. A més, fa que finalment tinguem un disseny que no és complicat modificar per tal d'adaptar-ho a altres sistemes, el que implica flexibilitat i reutilització.

Els mòduls escrits amb *SystemC* implementen la funcionalitat dels blocs Sw i Hw del sistema. Aquests deriven de la classe de *SystemC* anomenada ***sc_module*** i representen les construccions bàsiques dels blocs de tot el sistema. Cada un dels mòduls implementa la seva funcionalitat gràcies als processos i s'intercomuniquen amb els altres mitjançant les interfícies, els ports i els canals.

Traces VCD

Les traces VCD (*Verilog Change Dump*) són un estàndard (IEEE Standard 1364-1995 in 1995) definit per l'empresa *Verilog* i el seu VHDL. La seva simplicitat i la seva estructura compacta ha permès a altres programes externs seguir les traces aquests per generar formes d'ona que serviran per veure amb detall la resposta en el temps de les senyals implicades en la traça.

Apèndix 7. Sistema exemple

En aquest apèndix es fa una breu introducció al sistema exemple amb el que s'ha treballat per a desenvolupar l'eina de depuració descrita en aquest projecte. Aquest sistema està implementat en *SystemC* i proposa un sistema que conté una xarxa de routers que permet les comunicacions entre els recursos d'un compressor MPEG. Forma part de la tesina de Ramon Pla Casablancas titulada *NoC per un sistema MPEG encastat.*

Router utilitzat

En aquest sistema s'utilitza un Router per una xarxa *mesh* amb transmissió commutada de paquets. Al utilitzar com a tècnica de switching la commutació de paquets en una xarxa indirecta, serà necessària la utilització de wrappers per connectar els recursos als routers de la xarxa. La topologia de la xarxa serà doncs, estrictament ortogonal, degut a ser una topologia *mesh,*que permet un *routing* simple

L'entorn del sistema serà un compressor de vídeo MPEG, i es connectaran als routers els diferents mòduls que composen un compressor MPEG.

Les interconnexions de la xarxa estan formades per dos canals oposats unidireccionals, cadascun amb la seva informació i senyals de control. Cada canal inclou *n* bits (típicament els valors per *n* són 8, 16 o 32 bits) per la informació i un bit per indicar el final d'un paquet de dades: bit *eop (end of packet),* el qual només s'indica en la última paraula del *payload,* que també és el *trailer.*

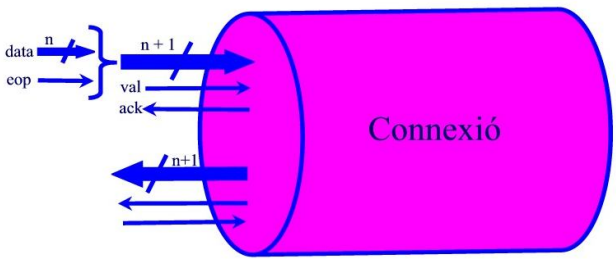


Figura 7.1 Connexió de la NoC

Els bits de control de flux s'utilitzen per validar la informació en el canal (*val*) i per confirmar la informació rebuda (*ack*).

Model de comunicació

El model de comunicació de NoC d'exemple es basa en el pas de missatges. Els nuclis es comuniquen enviant i rebent peticions i respostes de missatges. En aquests models de comunicacions, hi han dues classes de nuclis, els *iniciadors (initiators)* i els *objectius (targets)*. Un iniciador fa una petició, i un objectiu respon. A més, un nucli pot implementar les dues funcionalitats, ser un iniciador i un objectiu per diferents comunicacions (p.e. co-processador).

Algorisme de routing

En la NoC de l'exemple, l'encaminament és determinista i basat en l'origen. Cada remitent d'un paquet ha de determinar el camí que ha d'utilitzar, i incloure la informació de routing en la capçalera del paquet. Com a pla de routing s'utilitza el routing XY, molt utilitzat en topologies ortogonals 2-D.

El routing XY és molt restrictiu i limita l'ús de tota l'amplada de banda disponible de la xarxa. No obstant, és una de les aproximacions més barata per evitar el *deadlock* d'una xarxa.

Arbitratge i flux de control

L'arbitratge de la xarxa és distribuït. A cada router de la xarxa s'aplica una política de prioritats *round-robin* (RR) quan es fa la petició d'un canal de sortida.

Encara que aquest tipus d'arbitratge no és de les arquitectures més barates però és de les més ràpides i assegura l'ús de recursos per igual (cap paquet es queda indefinidament esperant una sortida).

El flux de control de la xarxa es basa en protocol *handshake*. Aquest consisteix en que el remitent activa una senyal de validació (*val*) quan insereix informació en el canal. Si el receptor està preparat per consumir la informació validada, aquest activa la corresponent senyal de

coneixement (*ack*). Aquest *handshake* es realitza a nivell local, és a dir, entre dos elements connectats consecutivament.

Arquitectura del router

El router de la NoC d'exemple està compost per cinc ports: L (Local), N (Nord), S (Sud), E (Est) i O (Oest). Cadascun d'aquests cinc ports està format per dos canals unidireccionals, un d'entrada i un altre de sortida. Cada canal té a més dos senyals de control (*val* i *ack*) que permeten la comunicació basada en *handshake.*

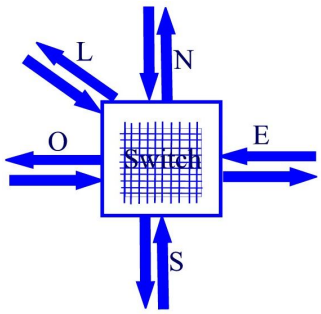


Figura 7.2 Interfície i arquitectura del router

Cada port del router té assignat un procés. Amb cinc processos independents el mateix router pot adreçar simultàniament la informació de diferents ports d'entrada cap a diferents ports de sortida. L'arquitectura del router amb cinc ports independents ens permet implementar fàcilment xarxes amb topologies de malles bidimensionals tipus *mesh* o *torus.*

En els casos dels processos corresponents als ports N, S, E i O, es comprova la informació de la capçalera. Si el router en qüestió té les coordenades del destí, es reenvia al port L que conté el *wrapper*. Si no és el cas, si encara no s'ha arribat al destí, s'actualitza la informació de la capçalera i s'envia al port de sortida corresponent. És a dir, quan entra un paquet per N, S, E o O, es comprova el valor dels camps *Xmod* i *Ymod*. Si aquests són nuls significa que s'ha arribat al destí i que el bloc es tindrà que passar a L. Si pel contrari els valors de *Xmod* i *Ymod* no són nuls, aquests es decrementaran (*Xmod* si aquest no és zero o *Ymod* si *Xmod* és zero) i s'adreçarà el paquet al corresponent port de sortida segons la taula d'encaminament

Wrapper

Al canal L que connecta el router amb el recurs corresponent hi ha un *wrapper* (*W*) o empaquetador. Aquest és l'encarregat de formar el paquet que circularà per la xarxa. A més

afegeix a cada paquet de dades que ha d'entrar per L una capçalera amb la informació de routing i un tràiler per indicar el final del paquet.

La informació de routing d'aquesta capçalera determina el camí a seguir per la xarxa. El wrapper calcula la informació de routing a partir de les coordenades del recurs origen i les coordenades del recurs destí.

El recurs connectat al wrapper determina quin serà el següent recurs en processar la informació. Per tant, cada wrapper genera la seva pròpia taula de routing perquè aquesta depèn del tipus de recurs unit a aquest.

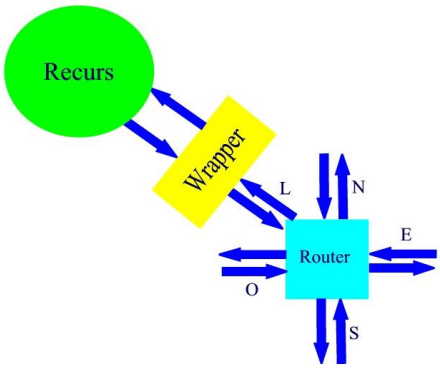


Figura 7.4 Connexió del recurs amb el router a través del wrapper

El wrapper fa d'interfície entre el recurs i la xarxa, conseqüentment, el disseny intern d'aquest està dividit en dues parts: una part connectada a la xarxa que és independent del recurs i una altre part connectada al recurs que és independent de la xarxa.

La part del wrapper connectada a la xarxa es pot reutilitzar per connectar qualsevol altre recurs però la part connectada al recurs dependrà de les senyals disponibles en els pins d'entrada/sortida, com la informació, mida del bus, senyals de control, etc.

BIBLIOGRAFIA

[1] Al Bobik, *"Handbook of Image and Video Processing Compression Standards, Algorithms and Architectures."* Academic Press, 2000.

[2] A. Portero, O. Navas, J. Carrabina, *"Study of High Level design methodologies for a MPEG frames I Compressor directed to FPGA Implementation"*, ICIT 2004 IEEE International Conference on Industrial Technology, Dec 8-10 Hammamet Tunis

[3] A. Portero, O. Navas, J. Carrabina, *"HW-SW design Methodologies used for a MPEG Video Compressor Synthesis"*, ICM 2004, IEEE 16th International Conference on Microelectronics, 06-08 December 2004, Sfax Tunis.

[4] A. Portero, O. Navas, J.V. Moyano, M. Bonamusa, J. Escrig, J. Carrabina, *"A MPEG2 frame I encoder developed with SystemC vs. Matlab DSP Builder"* GSPx, The Internacional Embedded Solution Event, September 27-30, 2004 Santa Clara Convention Center, Santa Clara, CA. EEUU.

[5] C. A. Zeferino and A. A. Susin, *"SoCIN: A Parametric and Scalable Network-on-Chip"*, SBCCI'2003, IEEE CS Press, pp.169-174, Säu Paulo, Brazil, September 08-11, 2003.

[6] C. A. Zeferino, M. E. Kreutz and A. A. Susin, *"RASoC: A Router Soft-Core for Networks-on-Chip"*, DATE'2004 - Designer's Forum, IEEE CS Press, p. 30198, Paris, France, February 16-20, 2004.

[7] ISO/IEC 13818: Generic coding of moving pictures and associated audio (MPEG-2) ITU-T Rec. H.262, ISO/IEC 13818-2,.Standard, Oct.1994

[8] L. Benini and G. De Micheli, *"Networks on Chip: a New SoC Paradigm"*, IEEE Computer, v.35 n.1, pp.70-78, Jan. 2002.

[9] SystemC Organization: www.systemc.org

[10] J. Duato, S. Yalamanchili and L. Ni, *"Interconnection Networks – An Engieneering Approach"*. Morgan Kaufmann Publishers, 2003.

[11] http://www.w3c.es/Divulgacion/Guiasbreves/TecnologiasXML web del W3C

[12] Bruce Eckel, "Thinking in Java", 3rd edition BETA, Jan 2003 .

[13] Elliotte Rusty Harold," Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and TrAX",Barnes & Noble, 2003

[14] http://www.saxproject.org/ web oficial for SAX.

[15] http://java.sun.com/j2se/1.4.2/docs/api/javax/xml/parsers/SAXParser.html SAX API website

[16] http://www.db4o.com/ web oficial for DB4O

Memòria elaborada per Miquel Angel Gallego Parejo,
que la signa per donar-ne fe.