Resum

El projecte que es presenta permet analitzar els avantatges i inconvenients d'una programació orientada a *hardware* i d'una programació orientada a *software* a partir del desenvolupament de dos dissenys, un cronòmetre i un freqüencímetre en cadascun dels modes de programació. Donat que en les dues aplicacions es requereix alta precisió de temps (µs) i flexibilitat en el control, la solució final que es proposa és un disseny "mixt" amb dos mòduls *hardware* específics (cronòmetre i freqüencímetre) integrats en un NIOS/CPU sobre una FPGA. Els dos mòduls es controlen per *software* sobre un sistema Linux empotrat (µCLinux).

Resumen

El proyecto que se presenta permite analizar las ventajas e inconvenientes de una programación orientada a *hardware* y de una programación orientada a *software* a partir del desarrollo de dos diseños, un cronómetro y un frecuencímetro en cada uno de los modos de programación. Dado que en las dos aplicaciones se requiere alta precisión de tiempo (µs) y flexibilidad en el control, la solución final que se propone es un diseño "mixto" con dos módulos *hardware* específicos (cronómetro y frecuencímetro) integrados en un NIOS/CPU sobre una FPGA. Los dos módulos se controlan por *software* sobre un sistema Linux embebido (µCLinux).

Overview

The project presented analyzes the advantages and disadvantages of a hardware-oriented programming and a software-oriented programming from the development of two designs, a chronometer and a frequencymeter in each of the programming modes. Because the two applications require high-precision time (µs) and control flexibility, the proposed final solution is a design "mixed" with two specific hardware modules (chronometer and frequencymeter) integrated in a NIOS/CPU on a FPGA. The two modules are controlled by software on a Linux embedded system (µClinux).

UAB

Projecte Fi de Carrera

Enginyeria Electrònica

Disseny d'un cronòmetre i d'un freqüencímetre integrats en un NIOS controlats sobre µClinux

Àngel Gutiérrez i Bravo

Directors: Jordi Sacristán i Riquelme
Núria Barniol i Beumala
Departament d'Enginyeria Electrònica

Escola d'Enginyeria

Universitat Autònoma de Barcelona (UAB)

Universitat Autònoma de Barcelona
Escola Tècnica Superior d'Enginyeria

Els Drs. Jordi Sacristán i Riquelme i Núria Barniol i Beumala, professor Associat i Catedràtica, respectivament, del Departament d'Enginyeria Electrònica de la Universitat Autònoma de Barcelona

CERTIFIQUEN

que la memòria *Disseny d'un cronòmetre i d'un freqüencímetre integrats en un NIOS controlats sobre µClinux* que presenta Àngel Gutiérrez i Bravo com a projecte final de carrera d'Enginyeria Electrònica, s'ha realitzat sota la seva direcció.

Bellaterra (Cerdanyola del Vallès), 14 de Setembre de 2011.

Dr. Jordi Sacristán i Riquelme
Dra. Núria Barniol i Beumala

Agraïments

A l'hora de realitzar el projecte he tingut el suport de les següents persones:

Miriam Sánchez (la meva xiqueta): amb el seu amor inesgotable, els moments més difícils i crítics del projecte s'han alleugerit enormement. Li agraeixo enormement l'esforç diari que ha realitzat per alliberar-me de moltes tasques que m'ha permès tenir el temps necessari per desenvolupar aquest treball. Ha sabut valorar perfectament la gran importància que tenia per mi la realització d'aquest projecte.

Ángel Gutiérrez, Juani Bravo, Esther Gutiérrez, Pau Farell, Marie Bravo (pares, germana, cunyat i tieta): la meva família sempre ha confiat en mi i en les meves possibilitats per tirar aquest repte endavant. La confiança que els meus pares han dipositat en mi durant els anys d'universitat espero que es vegi recompensada amb la realització d'aquest projecte final de carrera.

Jordi Sacristán i Núria Barniol (directors del projecte): en Jordi ha estat el gran guia i referent que necessitava per la realització d'aquest projecte amb les seves pautes tècniques imprescindibles. Li agraeixo moltíssim la seva dedicació i la seva total disponibilitat per adaptar-se als meus horaris força complicats. M'ha animat molt i m'ha donat tota la confiança necessària en els moments més crítics. En definitiva, ha estat un plaer treballar amb ell. La Núria m'ha donat uns bons consells en la realització d'aquesta memòria tècnica que m'han servit de gran ajuda.

Ros (el meu gosset): m'ha sabut distreure en molts moments de col·lapse i m'ha donat moltíssima companyia durant els dies intensos dedicats al projecte.

A Tots ells:

Us dedico aquest projecte i, sobretot, MOLTÍSSIMES GRÀCIES,

Ángel.

1. Introducció

L’evolució de la electrònica i, en concret, de la microelectrònica ha permès el desenvolupament de les FPGAs (*“Field Programmable Gate Array”*) com a resultat de la convergència entre dues tecnologies diferents, els PLDs (*“Programmable Logic Devices”*) i els Circuits Integrats d’Aplicació eSpecífica (ASIC).

La història dels PLDs va començar a començament de la dècada dels 70 amb els primers dispositius PROM *(“Programmable Read-Only Memory”)*. A mitjans dels 70 se’ls hi va afegir més versatilitat amb els PLA *(“Programmable Array Logic”)* que permetien més entrades i registres.

D’altra banda, els ASICs sempre han estat dispositius molt optimitzats però el seu desenvolupament ha requerit tradicionalment una gran inversió temporal i econòmica.

El pas final era combinar les dues estratègies amb un mecanisme d’interconnexió que pogués programar-se utilitzant fusibles o cel·les RAM i seguint aquesta premissa, a mitjans dels anys 80, Xilinx va desenvolupar la primera FPGA.

Actualment, una FPGA és un dispositiu molt versàtil capaç d’implementar sistemes d’aplicació específica que incloguin microprocessadors, filtres, mòduls de comunicacions, memòries, etc.

1.1 Marc del projecte

Des del departament d'Enginyeria Electrònica de la Universitat Autònoma de Barcelona (UAB) que se n'ocupa de la realització de pràctiques de programació de VHDL a la carrera d'Enginyeria de Telecomunicacions de la UAB, va sorgir la necessitat d'ampliar les actuals sessions de pràctiques orientades principalment a la programació *hardware* amb llenguatges descriptius.

Es pretén l'ampliació de les pràctiques per tal que els alumnes comprovin de primera mà els avantatges i inconvenients dels dos àmbits de programació a partir del desenvolupament d'un mateix projecte en cadascun dels modes de programació: la programació *hardware* amb un llenguatge de descripció *hardware* –per exemple, VHDL- o la programació *software* amb un llenguatge d'alt nivell com, per exemple, C/C++.

El projecte desenvolupat en cadascun dels àmbits de programació ha estat un cronòmetre que contempla les opcions bàsiques de funcionament. A partir de la realització dels dissenys *hardware* i *software* del cronòmetre, s'ha desenvolupat un disseny "mixt" del que combina una programació *software* i *hardware* per tal d'aprofitar els avantatges de cadascun dels àmbits de programació per aconseguir un disseny que assoleixi els dos grans objectius del projecte: facilitat i flexibilitat en el control i alta precisió en la mesura de temps (µs).

També s'ha realitzat el disseny "mixt" d'un freqüencímetre que ha permès validar la precisió assolida (µs) per comparació de la mesura d'aquest freqüencímetre amb la mesura de freqüència realitzada per un oscil·loscopi.

1.2 Objectius del projecte

Els objectius concrets del projecte són els següents:

- Disseny d'un cronòmetre i d’un freqüencímetre a partir de la programació en un llenguatge de descripció *hardware* de dos mòduls *hardware* específics.

- Integració dels mòduls *hardware* dissenyats en una CPU *embedded*.

- Disseny d'un cronòmetre *software* a partir d'una aplicació implementada en llenguatge d’alt nivell.

- Integració de l'aplicació del cronòmetre *software* en un sistema operatiu Linux *embedded*.

- Anàlisi comparatiu entre els avantatges i inconvenients d’una programació *hardware* i d'una programació *software*.

A partir dels dissenys *hardware* i *software* del cronòmetre l’objectiu és realitzar un disseny “mixt” que aprofiti els avantatges de cadascun dels modes de programació:

- Disseny “mixt” d'un cronòmetre *hardware* integrat en una CPU *embedded* controlat per *software* sobre un sistema Linux *embedded*.

Amb l’objectiu de validar la precisió assolida en el projecte desenvolupat (µs) es pretén realitzar el disseny “mixt” d’un freqüencímetre:

- Freqüencímetre *hardware* integrat en una CPU *embedded* controlat per *software* sobre un sistema Linux *embedded*.

A partir de l’assoliment dels anteriors objectius, es pretén proposar la creació d’un

laboratori de pràctiques pel desenvolupament d’unes sessions pràctiques en les quals els alumnes hauran de dissenyar el cronòmetre en els dos àmbits de programació i amb un disseny “mixt” *hardware*/*software*.

1.3 Estructura de la memòria

La present memòria tècnica del projecte es divideix, principalment, en quatre blocs.

En el primer dels blocs es realitza una introducció al disseny amb FPGA identificant les tècniques i eines típiques de disseny, introduint els sistemes digitals *embedded* i descrivint l’entorn de desenvolupament del projecte.

En el segon bloc es detalla el disseny del cronòmetre realitzat en tres fases seqüencials: un primer disseny *hardware*, un segon disseny *software* i un tercer disseny “mixt” *hardware/software*. La descripció es realitza detalladament en mode tutorial per petició expressa del departament d’Enginyeria Electrònica amb l’objectiu que el projecte desenvolupat pugui ser fàcilment adaptable a un guió de pràctiques de laboratori.

En el tercer bloc es descriu el disseny “mixt” *hardware/software* del freqüencímetre seguint els mateixos passos que en el cas anterior.

En el quart bloc es proposa la creació d’un laboratori de pràctiques de manera que els alumnes puguin desenvolupar un sistema en els dos àmbits de programació per acabar desenvolupant un sistema “mixt” *hardware/software*.

2. Introducció al disseny amb FPGA

Una **FPGA** és un dispositiu semiconductor que conté blocs lògics que poden ser configurats per reproduir des de funcions senzilles com poden ser portes lògiques o sistemes combinacionals (per exemple, descodificadors o funcions matemàtiques) fins a sistemes més complexos com poden ser memòries o altres sistemes seqüencials basats en biestables. A la Figura 1 s'observa l'arquitectura interna d'una FPGA:

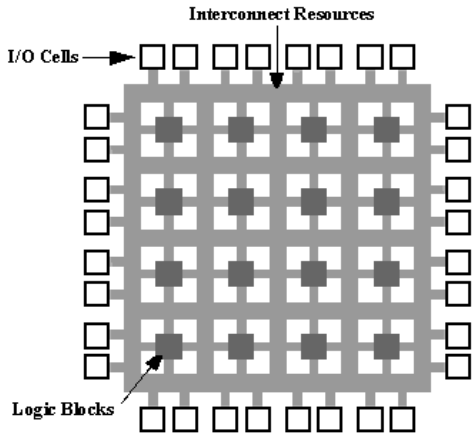


Figura 1 Arquitectura interna d'una FPGA

L'arquitectura es basa en cel·les d'entrada/sortida interconnectades amb blocs lògics programables (ALM , *"Adaptive Logic Module"*) formats per lògiques combinacionals adaptables (LUTs, "*Look-up Tables*), sumadors i registres tal com s'observa a la Figura 2 ([1] 2011):

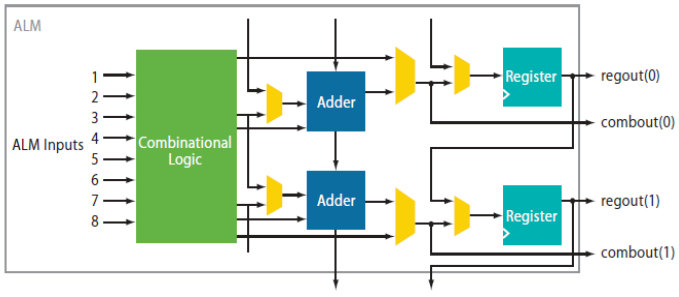


Figura 2 Arquitectura del bloc lògic programable d'una FPGA (ALM)

En comparació amb els ASICs, les principals avantatges que presenten les FPGAs són, d’una banda, la possibilitat de reprogramació que implica una gran flexibilitat en el flux de disseny i una reducció del temps de desenvolupament i, d’altra banda, el baix cost de desenvolupament per produccions de baixes quantitats.

El principal inconvenient que tenen les FPGAs respecte els ASICs és que són de propòsit general i no estan optimitzades exclusivament per una sola aplicació. D’aquest aspecte es deriven els següents inconvenients en els dissenys realitzats sobre FPGAs: augment del temps de resposta, augment del consum de potència i gran dificultat pel disseny de sistemes complexos.

2.1 Tècniques i eines de disseny

Tècniques de disseny:

En el disseny de sistemes *software* i electrònics programables, bàsicament, existeixen dues tècniques emprades: la ***top-down*** i la ***bottom-up***.

En el model *top-down* es realitza un resum del sistema, sense especificar detalls. Cada part del sistema es refina dissenyant-la més detalladament fins que el sistema complert és suficientment detallat per validar-ho. Un dels avantatges del model *top-down* és que normalment es dissenya amb l’ajuda de “caixes negres” que fan més fàcil el compliment dels requeriments tot i que no es defineixin amb detall els components individuals que formen cada “caixa negra”.

En aquest model és molt important la planificació i el coneixement complert del sistema. En sistemes programables, la programació no pot començar fins que no s’ha assolit un nivell de detall suficient, almenys en alguna part del sistema. Aquest aspecte deriva un inconvenient d’aquest model provocant un retard de les proves de les unitats funcionals del sistema fins que gran part del disseny s’ha completat.

En el disseny *bottom-up* les parts individuals es dissenyen molt detalladament i, posteriorment, s'uneixen per formar components més grans, que a la seva vegada, s'uneixen fins que es forma el sistema complert.

En aquest model és molt important la programació i les proves provisionals que es poden encetar tan bon punt s'hagi desenvolupat el primer mòdul. D'aquest aspecte es deriva un inconvenient d'aquest model que té el risc de programar cadascun dels mòduls sense considerar la connexió amb la resta del sistema i aquesta connexió pot resultar molt més complicada del previst inicialment. Un dels avantatges d'aquest model és la reutilització del codi font en diversos mòduls.

En el disseny de sistemes programables, normalment, es combinen tècniques *top-down* i *bottom-up*. Inicialment es recomanable partir d'un coneixement complert del sistema per abordar el disseny correctament (*top-down*). No obstant, en la majoria de dissenys s'intenta reaprofitar codi font existent pel desenvolupament de cadascun dels mòduls (*bottom-up*).

Eines de disseny:

En el disseny de sistemes digitals, existeixen una sèrie d'eines *software* bàsiques incloses en el programari de disseny habitual que permeten desenvolupar un circuit digital i simular-lo sense necessitat de tenir físicament el circuit electrònic: esquemàtic, llenguatges de descripció hardware i simulacions digitals.

- Esquemàtic:

És un diagrama pictòric que s'utilitza en el disseny de circuits digitals. Mostra els diferents components del circuit i es poden observar les connexions d'alimentació i de senyal entre cadascun d'ells.

- Llenguatges de descripció hardware:

Un Llenguatge de Descripció Hardware (HDL, *Hardware Description Language*) és un llenguatge de programació que permet descriure les interconnexions i el comportament d'un circuit electrònic, sense la necessitat d'utilitzar els diagrames esquemàtics.

Existeixen diferents HDL que es poden ordenar pel seu grau d'abstracció de més baix nivell a més alt nivell, per exemple: Verilog, VHDL (acrònim que resulta de VHSIC *"Very High Speed Integrated Circuit"*) i HDL *"Hardware Description Language"*) i SystemC.

- Simulació digital:

Els programes de disseny de sistemes digitals acostumen a incloure eines de simulació que després de desenvolupar el sistema, a partir dels models lògics dels components, permeten realitzar simulacions funcionals i/o temporals de tot el circuit digital complert. Les simulacions es realitzen observant el comportament de les formes d'ona del/s senyal/s de sortida a partir de la generació d'estímuls o senyals d'entrada.

2.2 Sistemes *embedded*

Els sistemes empotrats (*embedded*) són sistemes de computació dissenyats i programats per realitzar una o poques funcions dedicades freqüentment en un sistema de computació en temps real ([2] 2003). Els primers equips *embedded* els va desenvolupar IBM en els anys 80.

En un sistema empotrat la majoria dels components es troben inclosos en la mateixa placa base. Solen utilitzar un microprocessador i una memòria de poca capacitat i pocs perifèrics d'E/S per reduir costos. També consta d'un programa dedicat a una aplicació concreta emmagatzemat permanentment en memòria.

Tot el *hardware* i *software* del que consta el sistema, està dissenyat i optimitzat buscant

la mínima circuiteria i les menors dimensions per una aplicació particular. Normalment es dedica a controlar algun procés mitjançant una sèrie de sensors.

El consum d’energia pot ser determinant en el desenvolupament d’alguns sistemes empotrats que necessàriament s’alimenten amb bateries, el que provoca que el temps d’utilització del sistema vingui marcat per la duració de les bateries.

Aquests sistemes es poden localitzar en electrodomèstics com, per exemple: en televisors, reproductors de DVD, rentadores, alarmes, telèfons inalàmbrics, etc.

També es localitzen gran varietat de sistemes empotrats en automòbils com, per exemple: en el sistema de control de tracció, en el sistema de direcció assistida, en el sistema de frens antiblocatge (ABS), etc.

Els principals fabricants d’aquests sistemes són STMicroelectronics (família STPC), AMD (família Geode), Motorola (família ColdFire) i Intel.

En quant als sistemes operatius específics utilitzats pels sistemes *embedded* es troben, per exemple: Windows CE, QNX, VxWorks, µCLinux, etc.

2.3 Entorn de desenvolupament del projecte

Es desenvolupa el projecte sobre una electrònica *hardware* i un programari *software* de disseny específics.

2.3.1 Hardware

Pel disseny del projecte s'ha disposat d'una placa de desenvolupament d'Altera ideal per àmbit educatiu anomenada comercialment “DE2 Development and Education Board” ([3] 2011) disponible al laboratori de pràctiques. A la Figura 3 s’observa aquesta placa:

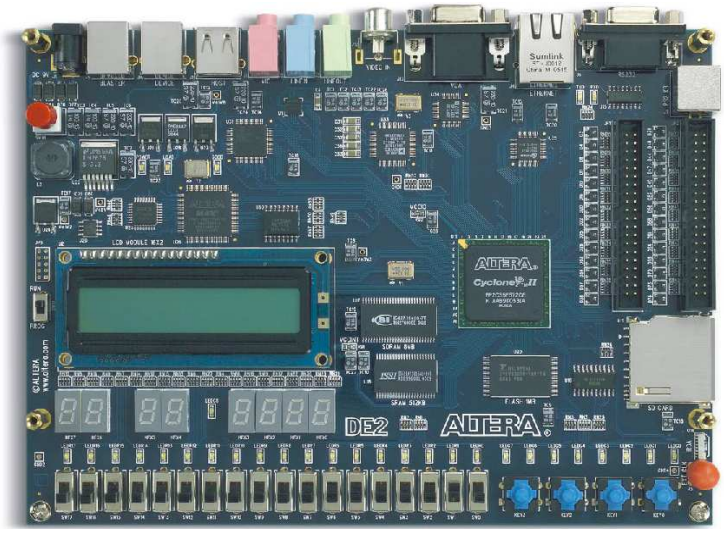


Figura 3 DE2 ***Development and Education Board***

Aquesta placa és ideal per l'àmbit educatiu ja que disposa de múltiples ports de comunicació (Ethernet, RS-232, USB, PS/2, entrada/sortida de vídeo, entrada/sortida d' àudio, infraroigs, etc), dispositius perifèrics (polsadors, Leds, etc.) memòries (SDRAM, SRAM, Flash, etc.) i rellotges (27 MHz, 50 MHz, entrada externa) oferint gran flexibilitat pel desenvolupament d'aplicacions diverses. Pel present projecte s'han utilitzat els següents dispositius i ports de comunicació:

- FPGA Cyclone II EP2C35F672C6.
- USB Blaster per la configuració de la FPGA (inclosa la descàrrega de µCLinux).
- 3 polsadors pel control del cronòmetre.
- 8 "displays" de 7-segments per la indicació del temps transcorregut del cronòmetre.
- Memòria SDRAM de 8 Mbytes per descarregar el µCLinux.
- Rellotges interns de 50 MHz i 100 MHz pels dissenys del cronòmetre, del freqüencímetre i del NIOS/CPU.
- NIOS/CPU per la integració del cronòmetre i del freqüencímetre.
- Ethernet per establir comunicació Telnet i NFS (*Network File System*) entre el µCLinux i un ordinador portàtil extern.
- 1 entrada del bus d'expansió GPIO pel senyal extern del freqüencímetre.
- 1 LED per monitoritzar el senyal a mesurar del freqüencímetre.

2.3.2 Software

Els sistemes operatius que s'han utilitzat pel disseny del projecte són els següents:

- Windows i Linux:

Sistemes operatius utilitzats per executar el programari de disseny: Quartus II, Nios II Command Shell, Emacs i el Notepad++. Si el Nios II Command Shell s’executa sota Windows cal fer-ho sobre un emulador de Linux (cygwin). El Linux és necessari per compilar el sistema µCLinux.

- µCLinux:

Sistema operatiu *embedded* basat en la versió 2.6 del kernel de Linux utilitzat per executar sobre una CPU *embedded* les aplicacions desenvolupades.

S’utilitza habitualment en microcontroladors que no disposen d’una unitat de gestió de memòria (MMU). Se suporta sobre diferents arquitectures, com per exemple: Altera NIOS, ADI Blackfin, ARM ARM7TDMI, Axis ETRAX, Freescale m68k, aka Motorola 68000 family, Fujitsu FR-V, Hitachi H8, Hyperstone E1/E2 (anomenat hyLinux), Intel i960, MIPS, Motorola ColdFire, NEC V850E, Xilinx MicroBlaze i Lattice Mico32. I sobre diferents productes, com ara: routers, càmeres de seguretat, DVD’s o reproductors MP3, telèfons VOIP, escàners, lectors de targetes, etc.

El programari que s'ha utilitzat pel disseny del projecte és el següent:

- Emacs i Notepad ++ (editors de text):

L’Emacs és un programa lliure que inclou un entorn de programació apte per programar en l’estàndard VHDL definit per l’IEEE i que ofereix macros predefinides que faciliten i fan molt intuïtiva la programació. En el projecte s’ha utilitzat aquest entorn Emacs per realitzar els dissenys del cronòmetre i del freqüencímetre en VHDL en comptes

del propi Quartus justament perquè aquestes macros faciliten enormement la tasca de programació.

El Notepad++ és un editor de text lliure que permet la programació en diferents llenguatges de programació d'alt nivell com, per exemple, Java i C/C++. En el projecte s'utilitza aquest *software* per programar en C aplicacions de control que interactuïn sobre la CPU dissenyada, i en concret, sobre els dos mòduls *hardware* específics dissenyats.

- Quartus II ([4] 2011):

És un *software* d'Altera que permet crear dissenys *hardware* a partir de llenguatges de descripció *hardware* com VHDL i Verilog. També permet realitzar simulacions del disseny creat així com assignar pins a una FPGA i crear una netlist que s'utilitza per configurar la FPGA. La versió "Web Edition" és gratuïta i està disponible a la web d'Altera. En el projecte s'ha utilitzat aquest *software* per importar i compilar els dissenys del cronòmetre i del freqüencímetre realitzats en VHDL sobre l'entorn d'Emacs i poder descarregar-los sobre la FPGA per provar tota la seva funcionalitat sobre la placa de desenvolupament.

- NIOS II ([5] 2011):

És el nucli (*core*) *software* d'un processador "*embedded*" fabricat per Altera que permet ser integrat dins del disseny global d'una CPU. Existeixen 3 tipus de NIOS que es poden triar en funció dels requisits de cada projecte tal com s'observa a la Figura 4: el NIOS II/e (versió "econòmica"), el NIOS II/s (versió "estàndard") i el NIOS II/f (versió "ràpida"). En el cas del present projecte i, en tractar-se, d'una aplicació estàndard d'àmbit educatiu s'ha triat el NIOS II/s:

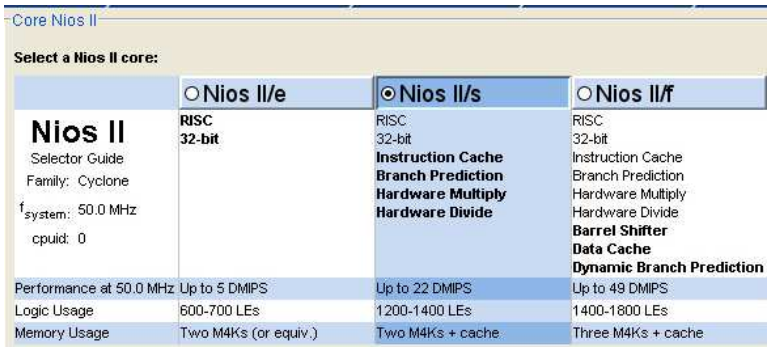


Figura 4 Diferents nuclis (*core*) Nios II d'Altera

La CPU basada en NIOS controla els diferents perifèrics de la placa de desenvolupament mitjançant el bus intern de comunicació Avalon desenvolupat per Altera. Aquest mateix bus s'utilitza per comunicar amb el mòdul *hardware* a desenvolupar.

- SOPC Builder *("System On a Programmable Chip Builder")*:

És una eina *software* inclosa en el Quartus II d'Altera que permet la personalització d'una CPU "*embedded*" a partir de la integració d'un *NIOS* amb els drivers i bussos necessaris per la comunicació amb els diferents dispositius de la placa de desenvolupament.

- Nios II Command Shell:

És una consola de comandes (terminal) del programa NIOS II EDS que, entre d'altres funcionalitats, permet la configuració d'una FPGA mitjançant un port JTAG, la descàrrega d'una "imatge" d'un sistema µCLinux compilat sobre la FPGA, obrir una consola (terminal) del sistema µCLinux descarregat, etc.

2.3.3 Desenvolupament hardware en VHDL

Qualsevol sistema digital es basa en blocs combinacionals i/o blocs seqüencials.

Els blocs combinacionals són aquells en els que les seves sortides només depenen de l'estat de les seves entrades en un moment donat. Aquests blocs estan formats per portes lògiques.

Els blocs seqüencials són aquells en que les sortides a més de dependre de l'estat de les seves entrades en un moment donat també depenen d'estats previs. Aquests blocs estan formats per elements de memòria que emmagatzemen informació d'estats anteriors (registres).

En els dissenys *hardware* desenvolupats en el projecte, es parteix d'un diagrama de blocs dels diferents components que formen cada sistema i es programen directament en VHDL. A la Figura 5 s'observa la programació d'un bloc combinacional format per un restador i un multiplexor:

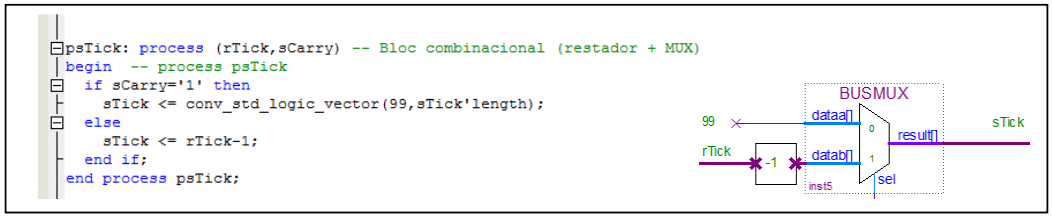


Figura 5 Programació en VHDL d'un bloc combinacional (restador i multiplexor)

A la Figura 6 s'observa la programació d'un bloc seqüencial format per un registre:

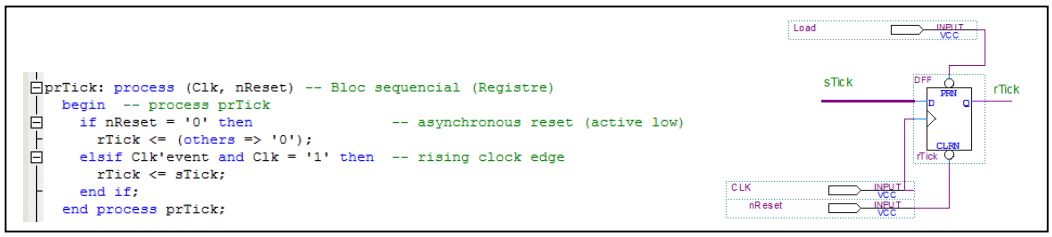


Figura 6 Programació en VHDL d'un bloc seqüencial (registre)

3. Desenvolupament del cronòmetre

El projecte del cronòmetre s'ha subdividit en 2 fases inicials per demostrar comparativament, a partir del disseny del cronòmetre en els dos àmbits de programació, *software* i *hardware*, els avantatges i inconvenients de cada àmbit. Finalment, s'ha realitzat una tercera fase "mixta" en que l'objectiu és l'aprofitament dels avantatges dels dos àmbits de programació.

A la primera fase s'ha realitzat tot el disseny *bottom-up* del cronòmetre a baix nivell amb un llenguatge de descripció *hardware* (VHDL) tant del propi cronòmetre com del seu sistema de control amb l'assignació dels controls a polsadors de la placa DE2.

Durant la segona fase, s'ha realitzat el disseny del cronòmetre *software* a partir d'una aplicació realitzada en llenguatge d'alt nivell (C) que contempla tant la programació del propi cronòmetre a partir de la llibreria "*time.h*" com una interfície d'usuari que implementa els controls bàsics del cronòmetre. L'aplicació s'integra sota un sistema operatiu µCLinux.

Partint dels sistemes desenvolupats a la primera i segona fase, durant la tercera fase s'ha realitzat un disseny "mixt" *bottom-up* integrant el cronòmetre com un mòdul *hardware* addicional dins d'un NIOS/CPU estàndard i s'ha realitzat una nova aplicació de control en C per controlar aquest mòdul mitjançant el bus Avalon del NIOS. Ha estat necessari crear una interfície software/hardware en VHDL per tal que aquesta aplicació *software* de control interaccioni directament sobre el mòdul *hardware* que es dedica només al compte de temps.

3.1 Disseny hardware del cronòmetre amb control per hardware (Fase I)

El disseny del cronòmetre *hardware* controlat per *hardware* s'ha realitzat d'acord als requisits funcionals següents:

- Utilització mínima de portes lògiques.
- Única freqüència d'operació de 100 MHz (rellotge intern de la placa DE2).
- El cronòmetre ha de disposar dels controls típics següents: engegar/aturar, inicialitzar a zero, cronòmetre ascendent/descendent.
- Visualització del temps transcorregut en el 8 displays de 7-segments.

S'ha dissenyat el cronòmetre *hardware* tenint en compte que el temps es pogués visualitzar pels 8 displays de 7-segments que incorpora la placa DE2. Considerant que l'objectiu de la Fase I és demostrar una alta precisió de temps (µs) que es pot assolir amb un disseny totalment *hardware* dedicat exclusivament al compte de temps, s'ha considerat adient l'assignació de precisió en els 8 displays que s'indica a la Figura 7:

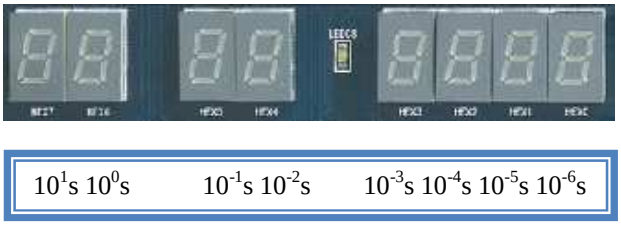


Figura 7 Assignació de la precisió del cronòmetre en els 8 displays

El diagrama de flux del disseny *hardware* del cronòmetre amb control per *hardware* s'observa a la Figura 8:

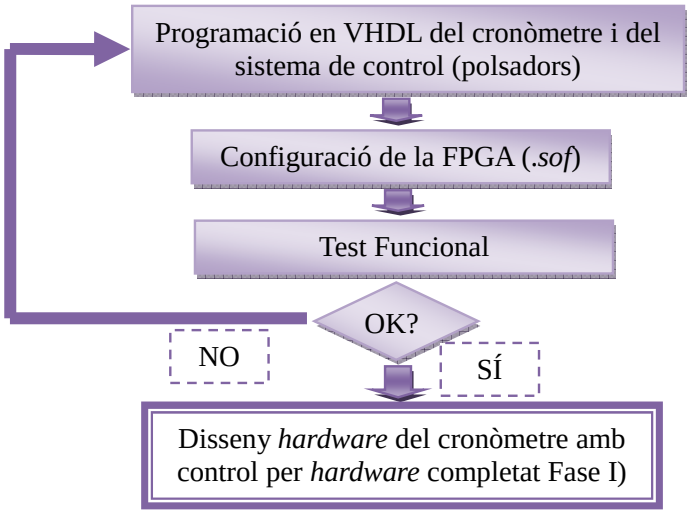


Figura 8 Diagrama de flux del disseny *hardware* del cronòmetre amb control per *hardware* (Fase I)

Considerant els requisits funcionals descrits anteriorment , s'ha realitzat el disseny a partir d'una base de compte d'un microsegon aconseguida gràcies a un comptador que amb una freqüència de 100 MHz compta des de 99 fins a 0 (Equacions 1 i 2) proporcionant a la sortida un senyal de rellotge amb període d'un microsegon que serveix com a freqüència d'operació pel primer comptador de 0 a 9 que té com a sortida l'últim display amb la precisió de microsegons:

$$\frac{1}{100MHz} = 0{,}01\mu s \qquad \textbf{Eq. 1}$$

$$\frac{1\mu s}{0{,}01\mu s} = 100 vegades \qquad \textbf{Eq. 2}$$

Aquest primer comptador de 0 a 9 realitza el compte fins a 9 o fins a 0 depenent del tipus de compte (ascendent o descendent) i proporciona a la sortida el senyal de rellotge del segon comptador de 0 a 9 (el display correspon a la precisió de 10^{-5}s segons) amb un període 10 vegades més gran. Així, successivament, es va connectant la sortida d'un comptador de 0 a 9 a l'entrada de rellotge del següent comptador de 0 a 9 de manera que cadascun dels 8 displays mostren una base de temps independent tal com s'indica a la Figura 7.

L'esquema de blocs que formen el cronòmetre i el connexionat entre blocs es descriu a

l'arxiu *"aCrono.vhd"* ([6] 2011) resultant el disseny que s'observa a la Figura 9:

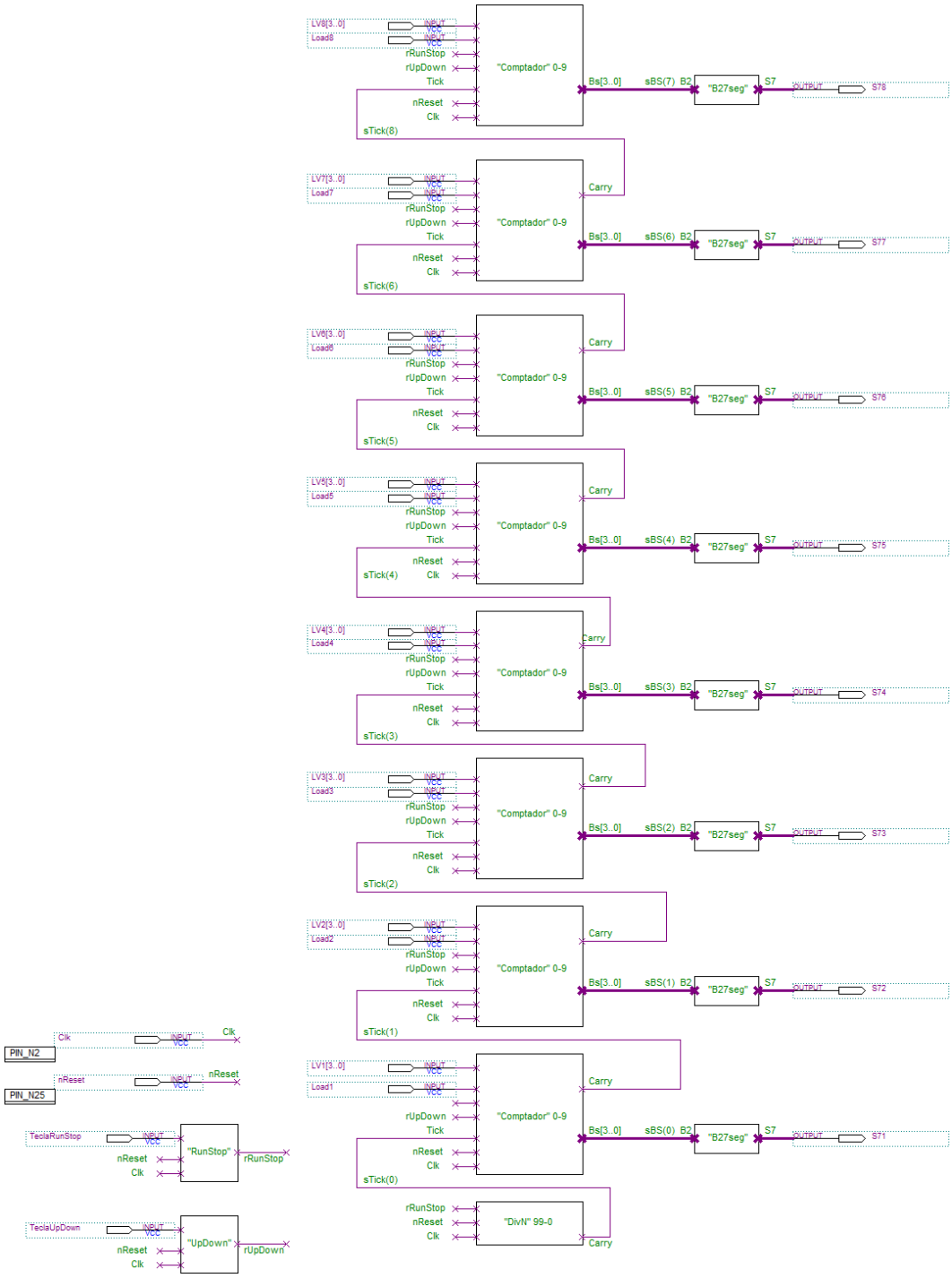


Figura 9 Esquemàtic del cronòmetre (Fase I)

El cronòmetre té com a entrades un rellotge global ("Clk") que provoca un sol event de rellotge en tot el circuit i un reset global per definir un valor inicial ("nReset") que eviten que el sistema dissenyat pugui trobar-se en un metaestat o un estat desconegut. La resta d'entrades són el control que permet canviar el tipus de compte ascendent/descendent ("TeclaUpDown") i el control que permet engegar o aturar el cronòmetre ("TeclaRunStop").

Les 8 sortides que contempla el cronòmetre són els 8 displays de 7-segments que monitoritzen el compte del cronòmetre ("S71" a "S78").

Es mostren en detall les arquitectures internes de cadascun dels blocs representats a la Figura 9. A la Figura 10 s'observa el primer bloc anomenat "divN" que descriu l'arquitectura del comptador de 99 a 0:

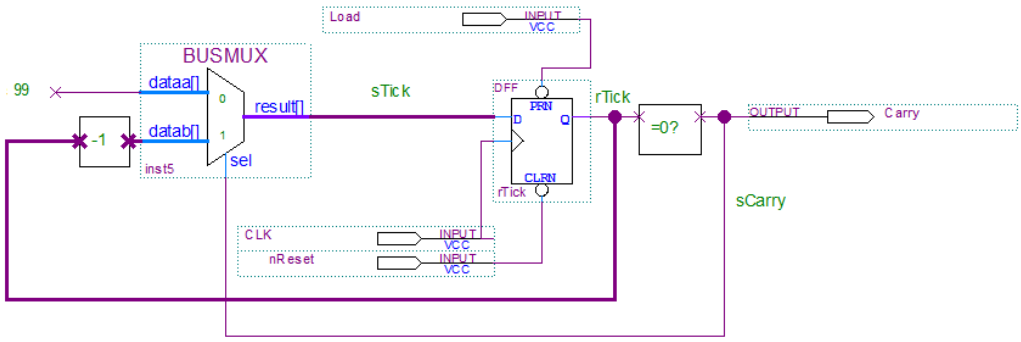


Figura 10 Bloc "divN"

L'esquema es basa en un registre amb una freqüència d'operació de 100 MHz (freqüència per defecte del rellotge de la placa), un multiplexor, un decrementador i un comparador que activen la sortida "Carry" només quan el valor decrementat des de 99 arriba a 0 (a cada iteració es comprova si el valor arriba a 0). D'aquesta manera s'aconsegueix el comptador de 100 desitjat utilitzant el mínim nombre de portes lògiques. Es tria que el registre sigui de 7 bits ja que contindrà el compte instantani amb valors de 99 a 0 (2^7=128).

A la Figura 11 s'observa el segon bloc anomenat "Comptador" que descriu l'arquitectura del comptador de 1 a 9:

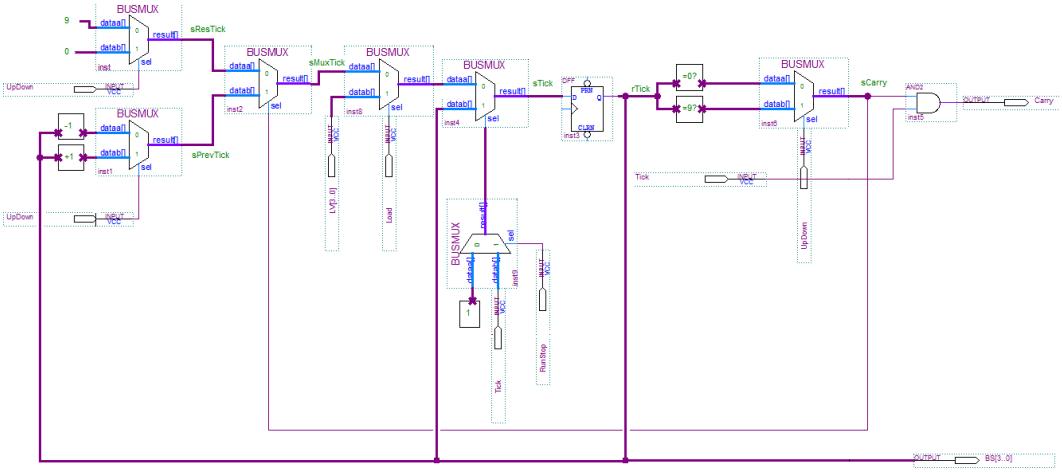


Figura 11 Bloc "Comptador"

L'esquema es basa en un registre, 7 multiplexors, un incrementador, un decrementador, dos comparadors (format per portes lògiques) i una porta lògica AND que activen la sortida "Carry" només quan el valor decrementat –si el compte és descendent- arriba a 0, o bé, quan el valor incrementat –si el compte és ascendent -arriba a 9.

El control "UpDown" s'encarrega de seleccionar el tipus de compte (ascendent o descendent) i el control "RunStop" s'encarrega d'aturar/engegar el cronòmetre. També es permet la càrrega d'un valor en el comptador per inicialitzar el cronòmetre i començar a comptar des d'aquest valor concret (controls "Load" i "LV"). No obstant, per la poca practicitat que resulta de realitzar la càrrega amb els possibles controls (polsadors) que ofereix la placa DE2 no s'ha considerat la càrrega durant aquesta Fase I.

Per controlar la freqüència de compte s'utilitza el control "Tic" que en el comptador amb precisió de 10^{-6} segons anirà connectat a l'anterior sortida "Carry" del bloc "DivN". En el control "Tic" del comptador amb precisió de 10^{-5} segons es connectarà la sortida "Carry" del comptador amb precisió de 10^{-6} segons i així successivament aconseguint un reescalatge de 10^{-1} segons en la freqüència de compte dels 8 comptadors de 0 a 9 o de 9 a

0. Per mantenir la sincronització, es controla que l'activació de la sortida "Carry" de cada comptador només es produeixi sempre amb l'arribada d'un event de rellotge ("Tic") gràcies a l'ús d'una porta lògica AND.

D'altra banda, es tria que el registre sigui de 4 bits ja que contindrà el compte instantani amb valors de 0 a 9 (2^4=16). La sortida del registre "Bs[3..0]" es connectarà a un descodificador ("B27seg") perquè descodifiqui de 4 bits a 7 bits i es pugui visualitzar el número en un display de 7-segments de la placa. A la Figura 12 s'observa el tercer bloc anomenat "B27seg" que descodifica els 4 bits a 7 bits per tal que es puguin encendre cadascun dels 7 leds que formen un display de 7-segments (no es té en compte el led del punt):

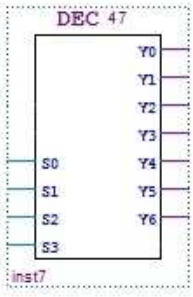


Figura 12 Bloc "B27seg"

A la Figura 13 s'observa la disposició dels diferents leds en un 7-segments que s'ha considerat ([7] 2011):

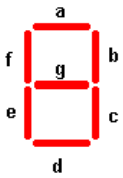


Figura 13 Disposició dels leds en un 7-segments

A partir d'aquesta disposició i utilitzant una lògica negativa (els zeros provoquen l'encesa d'un led), s'utilitza la Taula 1 de descodificació per "traduir" els 4 bits a 7 bits:

Número	Bs_3	Bs_2	Bs_1	Bs_0	a	b	c	d	e	f	g
					$S7_6$	$S7_5$	$S7_4$	$S7_3$	$S7_2$	$S7_1$	$S7_0$
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	1	1	0	0

Taula 1 Descodificació del Bloc "B27seg"

A la Figura 14 s'observa el quart bloc anomenat "RunStop" que descriu l'arquitectura del mòdul de control que s'encarrega d'engegar i aturar el cronòmetre:

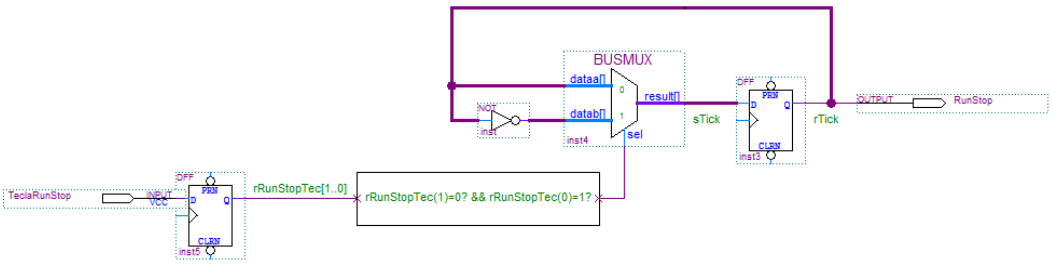


Figura 14 Bloc "RunStop"

L'esquema es basa en dos registres, 1 multiplexor, un comparador (format per portes lògiques) i una porta lògica NOT que controlen l'estat de la tecla de control d'engegada/aturada del cronòmetre ("TeclaRunStop").

El biestable de l'esquerra registra cada tic de rellotge l'estat de la tecla de control "TeclaRunStop" i a la sortida d'aquest es connecta un comparador que "detecta" si s'ha

produït un canvi (flanc) respecte l'anterior tic de rellotge. Depenent de si s'ha produït o no aquest canvi d'estat, el flip-flop de la dreta, manté sincronitzadament la sortida de control "RunStop" amb el mateix estat o, en el cas d'haver-se produït un canvi, actualitza el seu valor. Es defineixen dos estats de canvi de la següent manera:

0 → 1 (Canvi: cronòmetre aturat → cronòmetre engegat)
1 → 0 (Canvi: cronòmetre engegat → cronòmetre aturat)

A la Figura 15 s'observa el cinquè bloc anomenat "UpDown" que descriu l'arquitectura del mòdul de control que s'encarrega de configurar el cronòmetre amb compte ascendent o descendent:

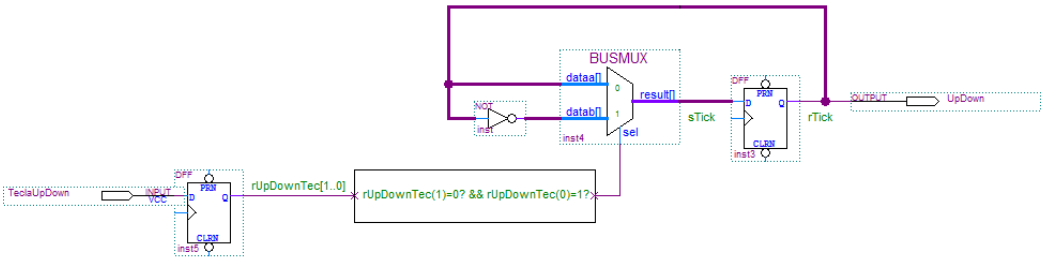


Figura 15 Bloc "UpDown"

L'esquema es basa en dos registres, 1 multiplexor, un comparador (format per portes lògiques) i una porta lògica NOT que controlen l'estat de la tecla de control que configura el cronòmetre ascendent o descendent ("TeclaUpDown").

El biestable de l'esquerra registra cada tic de rellotge l'estat de la tecla de control "TeclaUpDown" i a la sortida d'aquest es connecta un comparador que "detecta" si s'ha produït un canvi (flanc) respecte l'anterior tic de rellotge. Depenent de si s'ha produït o no aquest canvi d'estat, el flip-flop de la dreta, manté sincronitzadament la sortida de control "UpDown" amb el mateix estat o, en el cas d'haver-se produït un canvi, actualitza el seu valor. Es defineixen dos estats de canvi de la següent manera :

0 → 1 (Canvi: cronòmetre descendent → cronòmetre ascendent)
1 → 0 (Canvi: cronòmetre ascendent → cronòmetre descendent)

A partir del disseny exposat anteriorment, s'ha realitzat el projecte en Quartus II, programant amb el *software* Emacs en VHDL ([8] 1998) cadascun dels blocs ([6] 2011).

A l'arxiu "*cronoprueba.qsf*" ([9] 2011) es mostra l'assignació complerta dels pins utilitzats en el disseny del cronòmetre. Els controls d'entrada (TeclaRunStop, TeclaUpDown i nReset) s'han assignat a polsadors de la placa, les 8 sortides (S71[6..0], S72[6..0], S73[6..0], S74[6..0], S75[6..0], S76[6..0], S77[6..0], S78[6..0]) s'han assignat als 8 displays de 7 segments i el rellotge (Clk) s'ha assignat al clock intern de la placa ([7] 2011). La Figura 16 mostra l'assignació de pins del reset, d'un display de sortida i del rellotge:

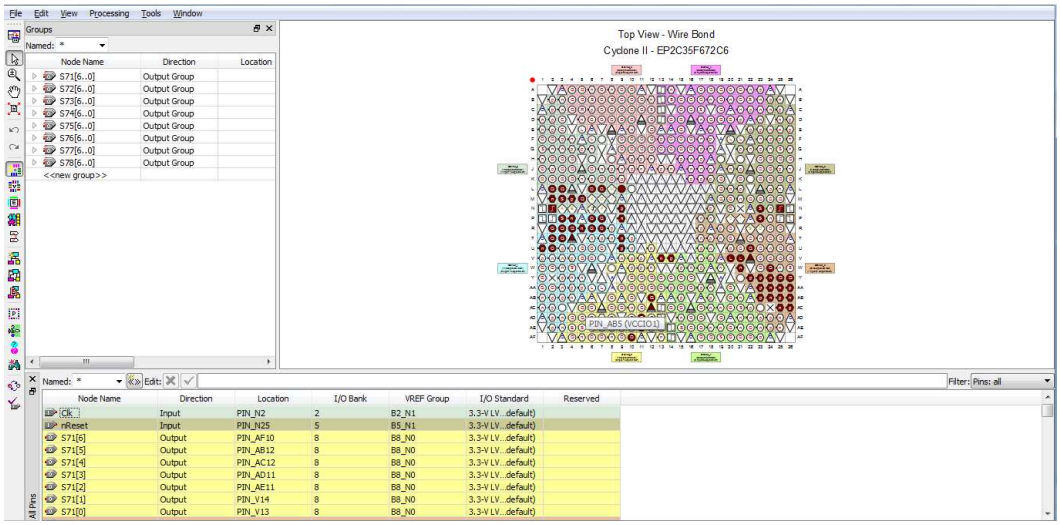


Figura 16 Assignació de pins del cronòmetre (Fase I)

Tal com s'observa a la Figura 17 es configura la FPGA amb el cronòmetre dissenyat descarregant l'arxiu "*cronoprueba.sof*" ([9] 2011):

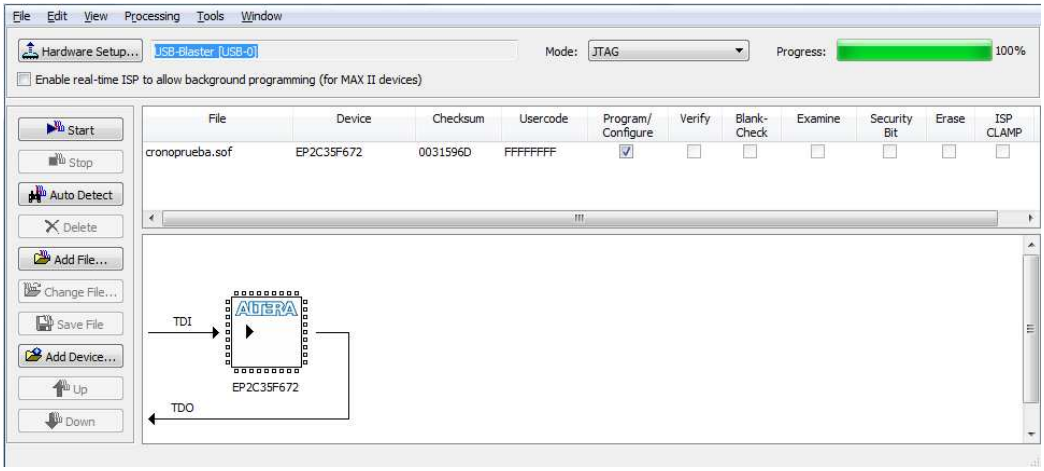


Figura 17 Configuració de la FPGA

3.2 Resultats del cronòmetre hardware amb control per hardware (Fase I)

Per testejar el disseny realitzat es realitzen les proves funcionals interactuant directament sobre els controls definits de la placa DE2 d'Altera. Tenint en compte l'assignació de pins realitzada, els controls del cronòmetre que s'han definit actuen de la següent manera:

- Per engegar o aturar el compte s'actua sobre el KEY1 (control "TeclaRunStop").
- Per canviar el tipus de compte, de compte ascendent a compte descendent o a l'inrevés, s'actua sobre el KEY2 (control "TeclaUpDown").
- Per resetejar el compte s'actua sobre el SW0 (control "nReset").

La Figura 18 mostra un exemple del cronòmetre comptant i s'indiquen els controls que interaccionen satisfactòriament sobre el cronòmetre dissenyat:

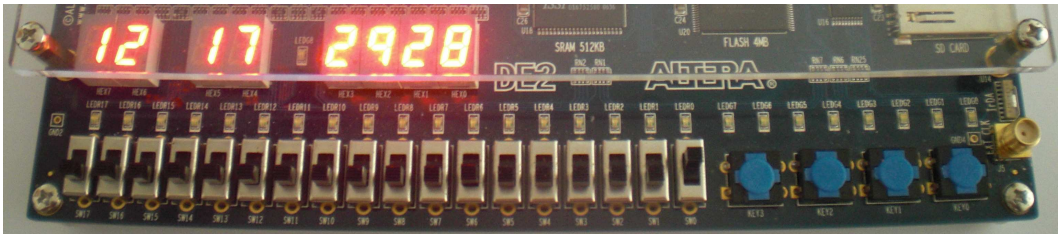


Figura 18 Exemple de funcionament del cronòmetre

La Figura 19 mostra el resultat de resetejar el cronòmetre (SW0 a "Off"):

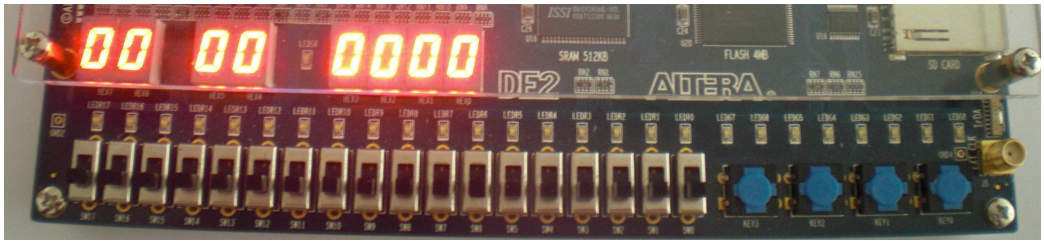


Figura 19 Reset del cronòmetre

La càrrega d'un valor d'inicialització s'ha descartat en aquesta Fase I ja que no és possible amb un control directe que un usuari pugui carregar directament el valor d'inicialització desitjat en el cronòmetre. Aquest aspecte és un inconvenient que té el disseny *hardware* que degut a que no disposa d'una interfície de control ofereix poca flexibilitat en el control del sistema.

3.3 Disseny software del cronòmetre amb control per software (Fase II)

El disseny del cronòmetre *software* controlat per *software* ha de complir els següents requisits funcionals:

- S'ha de mostrar el compte en temps real per un terminal *(Shell*).
- L'usuari ha de poder disposar de les funcionalitat típiques del cronòmetre controlades per les tecles d'un teclat: engegar/aturar, triar el tipus de compte (ascendent o descendent) i inicialitzar a zero.
- Pel cas del compte enrere, s'ha de poder inicialitzar el cronòmetre amb el valor que indiqui l'usuari i s'ha de permetre aturar el cronòmetre automàticament quan el compte arriba a zero o quan l'aturi manualment l'usuari.

A l'hora de dissenyar l'aplicació del cronòmetre i la interfície d'usuari de control, s'han de tenir en consideració els següents requisits de programació:

- La programació de l'aplicació es realitzarà en C per integrar-la com una aplicació dins de µCLinux i descarregar-la sobre la FPGA.
- Qualsevol llibreria utilitzada en C (*.h*) ha de ser compatible amb µCLinux.
- L'aplicació ha d'ocupar el mínim espai de memòria possible ja que la memòria SDRAM de 8 Mbytes sobre la que es descarrega el µCLinux és bastant justa.
- Per la compilació del µCLinux que inclou l'aplicació del cronòmetre, és necessari l'ús d'una CPU que inclogui com a mínim els següents components: kernel NIOS, el driver SDRAM per descarregar el µCLinux, el driver JTAG per comunicar amb la placa DE2, el driver DM9000 per establir comunicació Ethernet i definir rellotges de 50 MHz i 100 MHz tal com s'observa a l'esquema de la Figura 20:

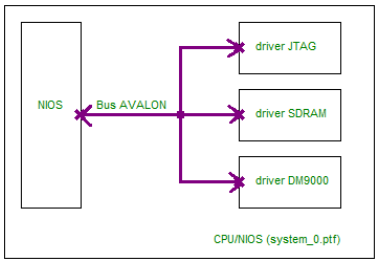


Figura 20 Esquema del NIOS/CPU utilitzat pel cronòmetre

El diagrama de flux del disseny *software* del cronòmetre amb control per *software* s'observa a la Figura 21:

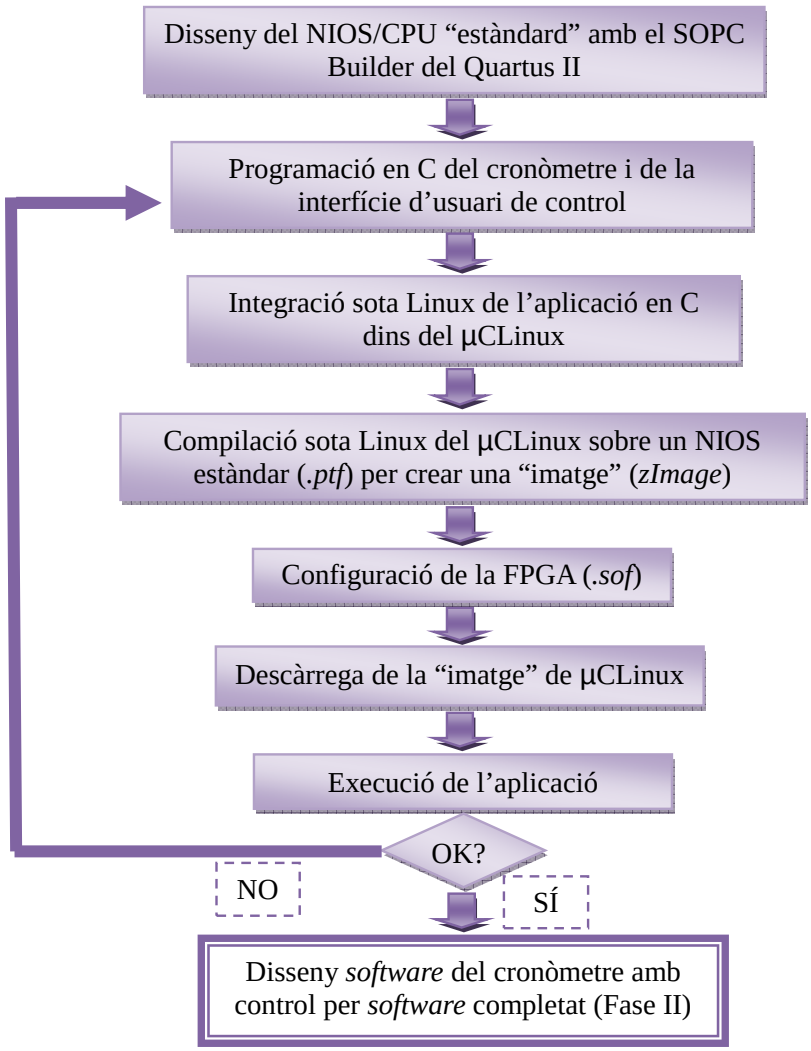


Figura 21 Diagrama de flux del disseny *software* del cronòmetre amb control per *software* (Fase II)

A partir dels requisits funcionals descrits anteriorment, s'ha realitzat el disseny d'un cronòmetre programant una aplicació en C que permet a l'usuari observar el compte en temps real pel terminal (*Shell*), controlar l'engegada/aturada del cronòmetre en qualsevol moment, triar el tipus de compte (ascendent o de compte enrere), inicialitzar a zero el cronòmetre i, pel cas del compte enrere, inicialitzar el compte enrere a un valor desitjat i aturar el cronòmetre en arribar el compte a zero o bé abans d'arribar a zero quan l'usuari ho desitgi.

Per portar el control del temps, en un primer disseny de l'aplicació, s'ha triat l'ús de la macro CLOCKS_PER_SEC de la llibreria estàndard *time.h*. Aquesta macro conté un valor aproximat del número de *tics* que els processadors actuals són capaços de fer per segon. Multiplicant el valor d'aquesta macro pel número de segons s'obté una base temporal amb precisió de segons.

El codi font de l'aplicació en C del cronòmetre amb l'ús de la macro CLOCKS_PER_SEC s'adjunta a l'arxiu *"cronometre_fase2a.c"* ([10] 2011).

Prèviament a la integració d'aquesta aplicació en µCLinux, és necessari partir d'un µCLinux instal·lat que serveix com a plataforma estàndard de desenvolupament per qualsevol aplicació que es vulgui desenvolupar en la placa DE2. Per obtenir aquesta plataforma estàndard cal seguir els passos descrits a l'Annex A sobre una distribució de Linux. Un cop s'obté el µCLinux instal·lat, per afegir i compilar una nova aplicació en µCLinux cal seguir els passos descrits també a l'Annex A.

És necessari seleccionar un NIOS/CPU (arxiu.ptf) compatible amb els drivers preinstal·lats en µCLinux (Annex A) i la memòria sobre la que es descarregarà el µCLinux sobre la FPGA. Se selecciona l'arxiu *"system_0.ptf"* que conté el NIOS/CPU que gestionarà el sistema operatiu tal com s'indica a la Figura 22 ([9] 2011). Es parteix d'un NIOS/CPU que inclou, com a mínim, els següents components: kernel NIOS, el driver SDRAM per descarregar el µCLinux, el driver JTAG per comunicar amb la placa DE2 i la definició de rellotges de 50 MHz i 100 MHz ([11] 2011). També se selecciona la memòria SDRAM sobre la que es descarregarà el sistema tal com es mostra a la Figura 23:

```
angelillo@angelillo:~/nios2-linux/uClinux-dist$ make vendor_hwselect SYSPTF=/media/disk/proyecto_fpga/DE2_demonstrations_copia/cronometre_hardware_software/system_0.ptf
make ARCH=nios2 -C vendors vendor_hwselect
make[1]: se ingresa al directorio `/home/angelillo/nios2-linux/uClinux-dist/vendors'
make -C /home/angelillo/nios2-linux/uClinux-dist/vendors/Altera/nios2/. dir_v=/home/angelillo/nios2-linux/uClinux-dist/vendors/Altera/nios2/. -f /home/angelillo/nios2-l
inux/uClinux-dist/vendors/vendors-common.mak vendor_hwselect
make[2]: se ingresa al directorio `/home/angelillo/nios2-linux/uClinux-dist/vendors/Altera/nios2'
[ -d /home/angelillo/nios2-linux/uClinux-dist/romfs/$i ] || mkdir -p /home/angelillo/nios2-linux/uClinux-dist/romfs
make ARCH=nios2 CROSS_COMPILE=nios2-linux-uclibc- -C /home/angelillo/nios2-linux/uClinux-dist/../linux-2.6 O=/home/angelillo/nios2-linux/uClinux-dist/linux-2.6.x hwsele
ct
make[3]: se ingresa al directorio `/home/angelillo/nios2-linux/linux-2.6'
  no emulation specific options.
  RUNNING hwselect

--- Please select which CPU you wish to build the kernel against:

(1) cpu_0 - Class: altera_nios2 Type: f Version: 7.08100

Selection: 1
```

Figura 22 Selecció del NIOS/CPU

```
--- Please select a device to execute kernel from:

(1) sram_0
        Class: sram_16bit_512k
        Size: 524288 bytes

(2) sdram_0
        Class: altera_avalon_new_sdram_controller
        Size: 8388608 bytes

(3) epcs_controller
        Class: altera_avalon_epcs_flash_controller
        Size: 2048 bytes

(4) cfi_flash_0
        Class: altera_avalon_cfi_flash
        Size: 4194304 bytes

Selection: 2
```

Figura 23 Selecció de la memòria SDRAM

Així es crea la imatge "*zImage*" que conté el sistema µCLinux amb l'aplicació del cronòmetre que es descarregarà amb el NIOS II Command Shell.

Es crea un directori en el path on s'hagi instal·lat el NIOS II. Dins d'aquest directori es copia l'arxiu de configuració de la FPGA *DE2_NIOS_HOST_MOUSE_VGA.sof* ([10] 2011) i el *zImage* que s'ha generat de la compilació del µCLinux.

S'executa el NIOS II Command Shell, es descarrega l'arxiu de configuració de la FPGA tal com s'indica a la Figura 24 i, tot seguit, es descarrega el *zImage* tal com s'observa a la Figura 25:

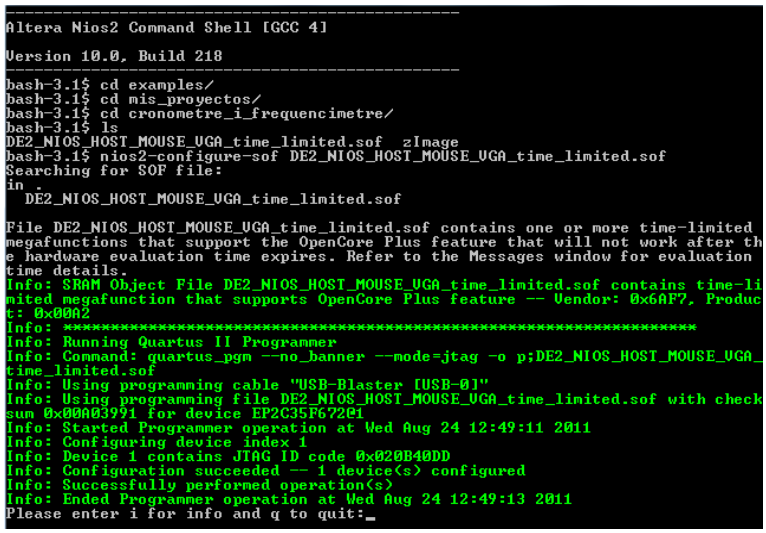


Figura 24 Configuració de la FPGA

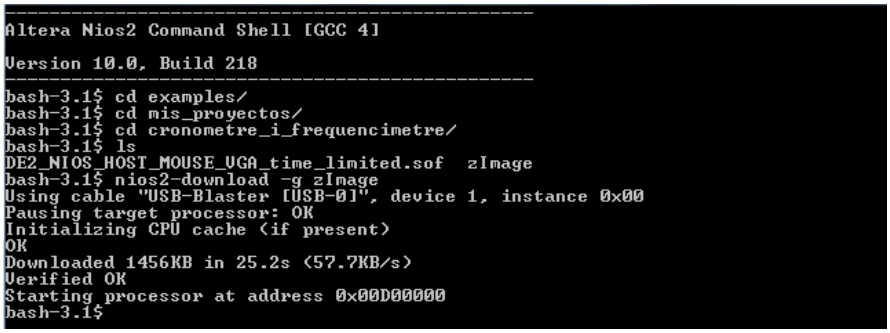


Figura 25 Descàrrega de la *zImage* de µCLinux

S'executa la comanda "nios2-terminal" per obrir un terminal (Shell) en el sistema µCLinux i el resultat s'observa a la Figura 26:

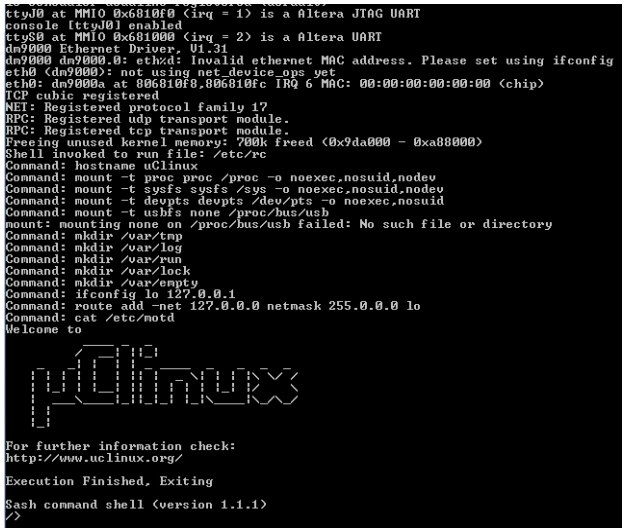


Figura 26 Terminal (Shell) de µCLinux

I s'executa l'aplicació "cronòmetre_fase2a" tal com es mostra a la Figura 27:

Figura 27 Aplicació de control del cronòmetre (Fase II)

3.4 Resultats del cronòmetre software amb control per software (Fase II)

Es fan diverses proves funcionals de l'aplicació dissenyada i els resultats obtinguts no són òptims ja que el temps mesurat pel cronòmetre és molt més gran que el temps programat per l'usuari.

Bàsicament, el motiu que afecta directament al retard de mesura que té el cronòmetre dissenyat és que l'ús de la macro CLOCKS_PER_SEC de la llibreria *time.h* utilitza el rellotge del sistema que es veu afectat negativament i de manera proporcional al nombre de tasques que el processador *embedded* estigui executant en el mateix moment en que s'executa l'aplicació del cronòmetre.

Un dels requisits que ha de complir el disseny del cronòmetre és que l'aplicació imprimeixi el temps transcorregut pel terminal (*Shell*) a cada iteració del bucle *while* que controla si l'usuari atura manualment el cronòmetre o el cronòmetre de compte enrere arriba a zero. Aquest procés d'escriptura contínua en el terminal (*printf* en C) carrega molt la CPU *embedded* i, de forma directa, al rellotge del sistema.

Per demostrar-ho, donat que el NIOS/CPU (*system_0.ptf*) incorpora el driver DM9000A per la comunicació Ethernet amb la FPGA s'estableix una comunicació per Telnet amb el µCLinux, s'executa una aplicació "*en background"* i es comprova que si, a part de l'aplicació cronòmetre, s'executa una nova aplicació el processador es "carrega" i el cronòmetre es rellenteix perquè el rellotge del sistema es veu afectat.

Per solucionar aquest aspecte crític i que l'aplicació del cronòmetre funcioni perfectament independentment de les aplicacions que s'estiguin executant en aquell moment i, en definitiva, no dependre del rellotge del sistema, la solució és utilitzar directament el rellotge *hardware* del processador (*Real Time Clock*, RTC).

Per treballar directament amb el rellotge *hardware* i mantenir la precisió de µs

requerida en el projecte, és necessari que l'aplicació del cronòmetre agafi la base temporal de la funció "*gettimeofday()*" de la llibreria "*time.h*".

El codi font de la nova aplicació en C del cronòmetre per utilitzar la funció "*gettimeofday()*"s'adjunta a l'arxiu *"cronòmetre_fase_2b.c"* ([10] 2011).

La nova aplicació s'integra en el µCLinux seguint els mateixos passos que amb l'anterior aplicació i el µCLinux es torna a compilar per incloure aquesta nova aplicació del cronòmetre.

Es fan diverses proves funcionals de la nova aplicació modificada del cronòmetre "cronòmetre_fase_2b" i els resultats obtinguts ara sí que són els esperats ja que el temps mesurat pel cronòmetre és exactament el temps programat per l'usuari i no es produeix cap retard en la mesura.

La precisió obtinguda de microsegons per aquesta nova aplicació programada amb la funció "*gettimeofday()*" és la mateixa que s'ha aconseguit utilitzant el cronòmetre *hardware* dissenyat en la fase I, perquè en ambdós casos s'utilitza directament un rellotge *hardware* com a base temporal.

L'objectiu amb el desenvolupament de la Fase III és integrar el cronòmetre *hardware* dissenyat en la fase I com un mòdul *hardware* específic dins d'un NIOS/CPU, dissenyar una interfície *hardware-software* i una aplicació de control per tal d'obtenir uns resultats equivalents als obtinguts amb l'aplicació del cronòmetre que utilitza la funció "*gettimeofday()*". És a dir, desenvolupar un RTC (*Real Time Clock*) amb les prestacions de control que es marquin en els requisits funcionals.

3.5 Disseny hardware del cronòmetre amb control per software (Fase III)

El disseny del cronòmetre *hardware* controlat per *software* s'ha realitzat d'acord als requisits funcionals següents:

- L'usuari ha de poder disposar de les funcionalitats típiques del cronòmetre controlades per les tecles d'un teclat: engegar/aturar, triar el tipus de compte (ascendent o descendent) i inicialitzar a zero. Pel cas del compte enrere, s'ha de poder inicialitzar el cronòmetre amb el valor que indiqui l'usuari i s'ha de permetre aturar el cronòmetre automàticament quan el compte arriba a zero o quan l'aturi manualment l'usuari.
- Visualització del temps transcorregut en el 8 displays de 7-segments.

Per satisfer aquests requisits funcionals, la opció més idònia és integrar el cronòmetre *hardware* de la Fase I com un mòdul *hardware* específic dins d'un NIOS/CPU "estàndard" i controlar aquest mòdul directament amb una aplicació de control *software*.

El diagrama de flux del disseny *hardware* del cronòmetre amb control per *software* s'observa a la Figura 28:

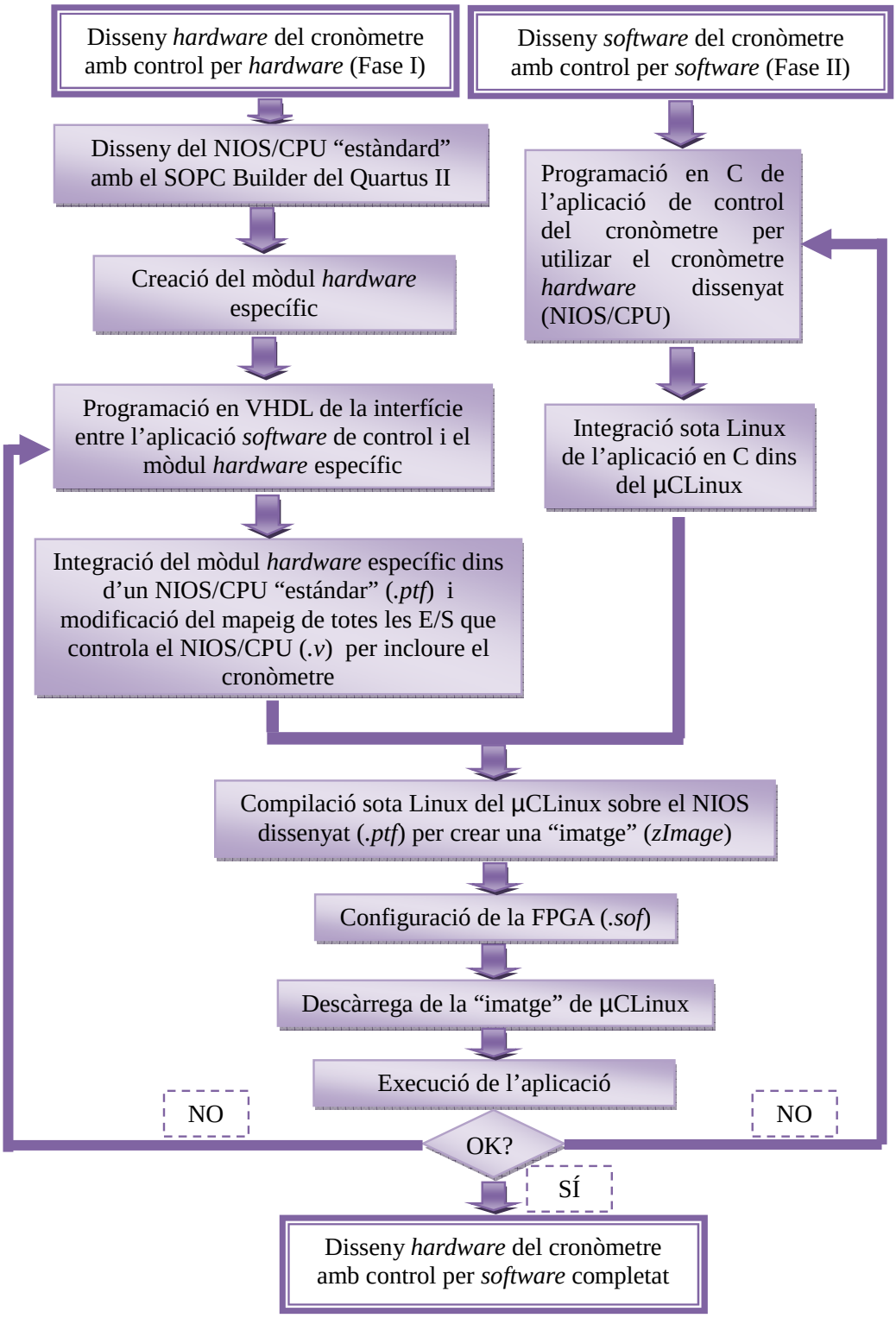


Figura 28 Diagrama de flux del disseny *hardware* del cronòmetre amb control per *software* (Fase III)

Per complir amb els requisits funcionals de la Fase III, es parteix del mateix NIOS/CPU utilitzat en la Fase II ([11] 2011). A més dels "components" que inclou aquest NIOS/CPU, la idea és afegir un nou "component" (mòdul hardware) anomenat *"niosCronometre"* que inclogui tot el *hardware* que descriu la funcionalitat del cronòmetre.

Aquest nou component *"niosCronometre"* tindrà entrades per ser controlat amb l'aplicació de control en C i com a sortida els 8 displays de 7-segments que indicaran el temps transcorregut del cronòmetre.

Abans de crear el nou component, per satisfer els requisits funcionals d'aquesta Fase III es parteix del cronòmetre *hardware* de la Fase I per crear un nou projecte de la fase III en Quartus II ([12] 2011) que amplia la seva funcionalitat permetent, pel cas del compte enrere, la càrrega del valor d'inicialització que indiqui l'usuari per *software* i permetent l'aturada automàtica del cronòmetre quan el compte arriba a 0.

Per interactuar per *software* mitjançant una aplicació de control amb qualsevol component del NIOS/CPU –en aquest cas, el mòdul *hardware"niosCronometre"*-, és necessari considerar en el disseny de qualsevol mòdul una interfície *software*/*hardware* programada en VHDL que, a partir de la direcció *"address[3..0]"*, utilitzi les entrades "*write_n*" (*enable*) i "*writedata[7..0]*" (*dades*) per llegir una dada del bus Avalon procedent de l'aplicació de control i actualitzi l'estat de les entrades del cronòmetre (controls d'engegar/aturar, de compte ascendent/descendent, de càrrega d'un valor d'inicialització). També cal la entrada "*read_n*" (*enable*) i la sortida "*readdata[7..0]*" (*dades*) per escriure una dada en el bus Avalon i poder llegir les sortides del cronòmetre mitjançant l'aplicació de control (compte del temps transcorregut que indiquen els 8 displays de 7-sements). A la Figura 29 es mostra un esquema de la interfície *software*/*hardware* mitjançant el bus Avalon:

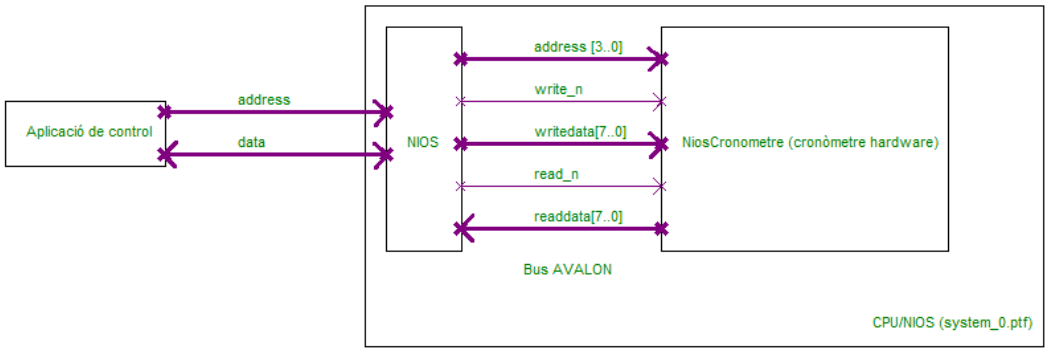


Figura 29 Esquema de la interfície *software*/*hardware* mitjançant el bus AVALON

S'ha inclòs aquesta interfície *software*/*hardware* en el arxiu *"niosCronometre.vhd"* del projecte de la fase III ([13] 2011). La nova arquitectura del cronòmetre de la fase III que inclou aquesta interfície *software*/*hardware* s'observa a la Figura 30:

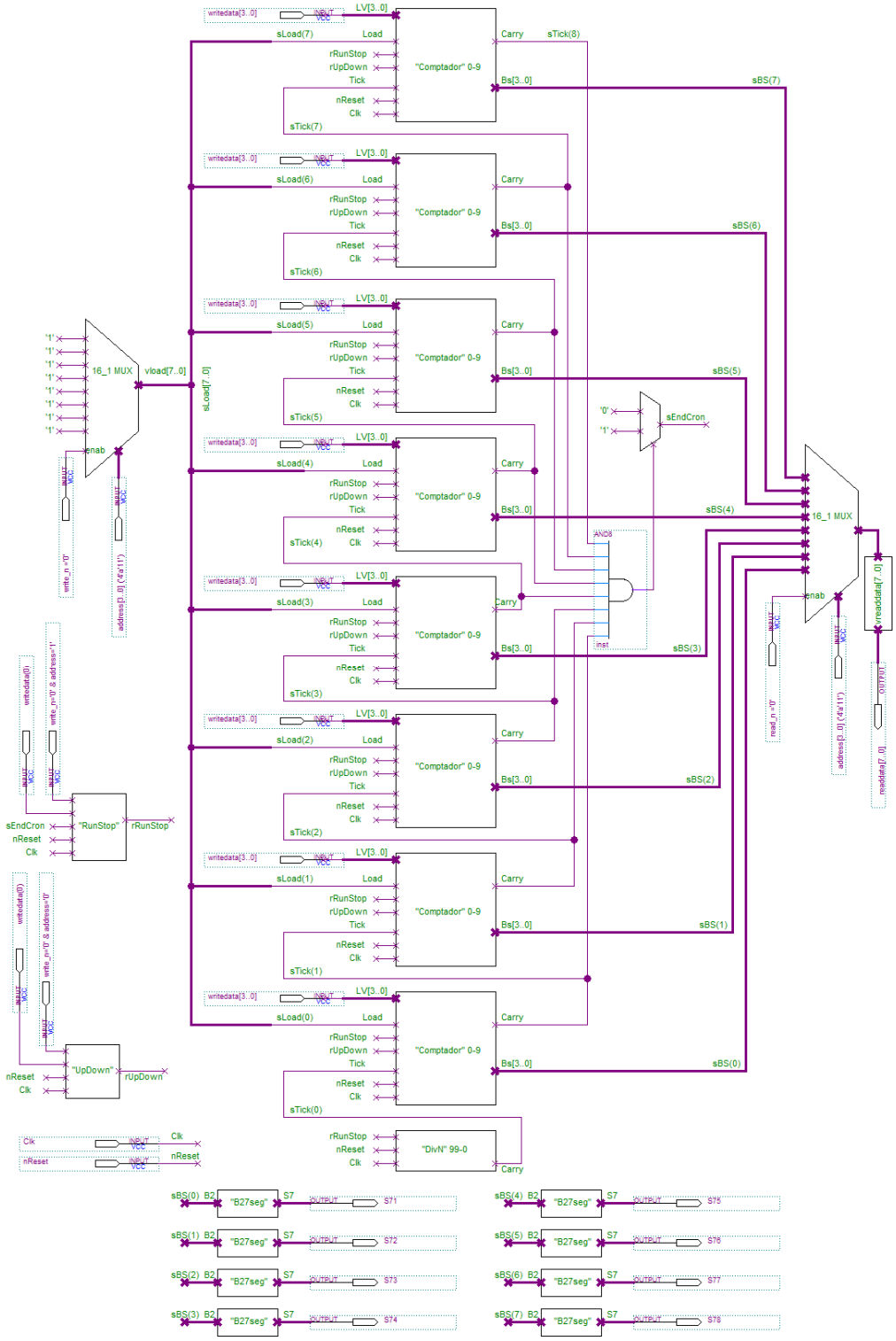


Figura 30 Esquemàtic del cronòmetre (Fase III)

L'esquema del cronòmetre es basa en l'esquema de la fase I, tot i que en aquest esquema de la fase III s'ha insertat la interfície *software/hardware* i les entrades de control no es connecten a polsadors i interruptors de la placa, sinó que s'assignen als senyals d'escriptura del bus Avalon (*"write_n"* i *"writedata[7..0]"*). A partir de l'assignació d'una adreça diferent (*"address[3..0]"*) per cadascuna de les entrades, es permet controlar el cronòmetre directament des de l'aplicació de control.

Les sortides del cronòmetre es continuen assignant als 8 displays de 7-segments però, a més, amb la incursió de la interfície *software/hardware* s'assignen les 8 sortides als senyals de lectura del bus Avalon (*"read_n"* i *"readdata[7..0]"*). A partir de l'assignació d'una adreça diferent (*"address[3..0]"*) per cadascuna de les sortides, es permet llegir el compte transcorregut del cronòmetre des de l'aplicació de control.

Per integrar tot el *hardware* descrit del cronòmetre de la fase III en un sol component *"niosCronometre"* que pugui ser afegit al NIOS/CPU cal seguir els passos descrits a l'apartat B.1 de l'Annex B.

A la Figura 31 es mostra el NIOS/CPU amb el nou component *"niosCronometre"* afegit:

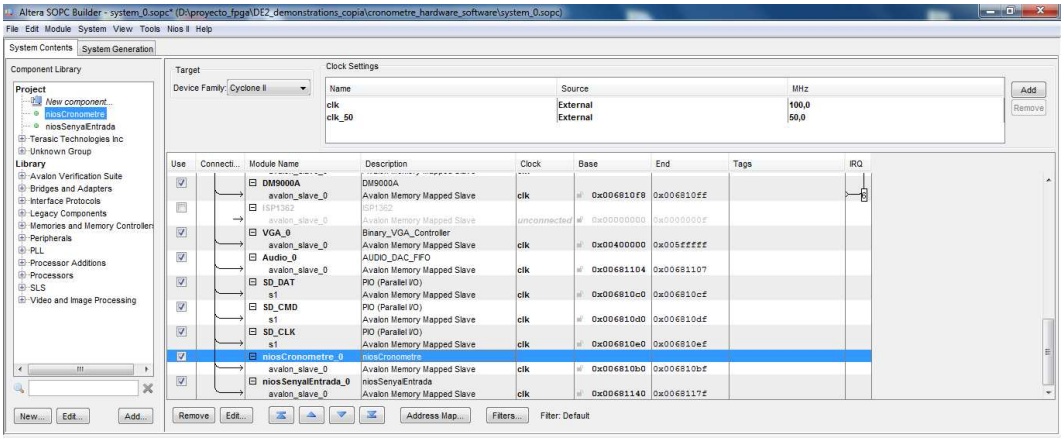


Figura 31 Component *"niosCronometre"*afegit al NIOS/CPU

El SOPC Builder assigna automàticament un rang de memòria pel nou component afegit (0x006810b0 → 0x006810bf). Per accedir directament –obviant la memòria *cache* del NIOS/CPU- des de l'aplicació de control a les posicions de memòria que ocupa el mòdul del cronòmetre dins del NIOS/CPU cal considerar el rang de memòria assignat, però, a més, cal considerar el primer bit a '1' de cada posició de memòria. A la Taula 2 s'especifica el direccionament de memòria per cada control del cronòmetre que s'ha considerat en el disseny, tant de la interfície *software/hardware* com de l'aplicació de control:

Adreça	R/W	Adreça (accés directe)	Control
0x006810b0	W	0x806810b0 (address='0')	Ascendent/descendent (Up/Down)
0x006810b1	W	0x806810b1 (address='1')	Engegar/Aturar (Run/Stop)
0x006810b4	W	0x806810b4 (address='4')	Carregar en dígit 0 (sLoad[0])
0x006810b5	W	0x806810b5 (address='5')	Carregar en dígit 1 (sLoad[1])
0x006810b6	W	0x806810b6 (address='6')	Carregar en dígit 2 (sLoad[2])
0x006810b7	W	0x806810b7 (address='7')	Carregar en dígit 3 (sLoad[3])
0x006810b8	W	0x806810b8 (address='8')	Carregar en dígit 4 (sLoad[4])
0x006810b9	W	0x806810b9 (address='9')	Carregar en dígit 5 (sLoad[5])
0x006810ba	W	0x806810ba (address='10')	Carregar en dígit 6 (sLoad[6])
0x006810bb	W	0x806810bb (address='11')	Carregar en dígit 7 (sLoad[7])
0x006810b2	R	0x806810b2 (address='2')	Consultar aturada a 0 (compte descendent)
0x006810b4	R	0x806810b4 (address='4')	Llegir dígit 0 (sBS[0])
0x006810b5	R	0x806810b5 (address='5')	Llegir dígit 1 (sBS[1])
0x006810b6	R	0x806810b6 (address='6')	Llegir dígit 2 (sBS[2])
0x006810b7	R	0x806810b7 (address='7')	Llegir dígit 3 (sBS[3])
0x006810b8	R	0x806810b8 (address='8')	Llegir dígit 4 (sBS[4])
0x006810b9	R	0x806810b9 (address='9')	Llegir dígit 5 (sBS[5])
0x006810ba	R	0x806810ba (address='10')	Llegir dígit 6 (sBS[6])
0x006810bb	R	0x806810bb (address='11')	Llegir dígit 7 (sBS[7])

Taula 2 Assignació d'adreces de memòria del component *"niosCronometre"*

Per tal que el NIOS/CPU consideri les 8 sortides dels 8 displays de 7-segments com a sortides del cronòmetre cal indicar-ho a l'arxiu descrit en Verilog *"DE2_NIOS_HOST_MOUSE_VGA.v"* on s'especifica el mapeig de totes les entrades i sortides que controla el NIOS/CPU dissenyat ([12] 2011).

A l'arxiu DE2_NIOS_HOST_MOUSE.qsf ([12] 2011) es mostra l'assignació complerta dels pins utilitzats en el disseny del cronòmetre de la fase III d'acord als components que controla el NIOS/CPU utilitzat.

En compilar el projecte *"DE2_NIOS_HOST_MOUSE.qsf"* s'actualitza l'arxiu *"DE2_NIOS_HOST_MOUSE_time_limited.sof"* ([12] 2011) i, tot i que és d'ús restringit per part d'Altera, conté el nou disseny del NIOS/CPU que inclou el mòdul del cronòmetre dissenyat i permet configurar posteriorment la FPGA amb aquest disseny.

Es crea una nova aplicació de control que s'encarrega únicament de controlar el mòdul *hardware* (cronòmetre) integrat en el NIOS/CPU, deixant la tasca de compte temporal com a responsabilitat del propi mòdul *hardware*.

Es programa l'aplicació mitjançant l'ús d'apuntadors que actuen sobre el rang de memòria indicat a la Taula 2, canviant l'estat dels controls del cronòmetre, o bé, llegint directament el temps transcorregut de cadascun dels 8 displays de 7-segments del cronòmetre dissenyat.

El codi font de l'aplicació *software* de control que controla el cronòmetre *hardware* s'adjunta a l'arxiu *"cronòmetre_fase_3.c"* ([12] 2011).

De la mateixa manera que s'ha realitzat amb les dues aplicacions desenvolupades durant la Fase II, s'integra aquesta aplicació dins del sistema µCLinux i es crea la "imatge" que s'ha de descarregar sobre la FPGA (*zImage*).

Amb el NIOS II Command Shell es descarrega l'arxiu de configuració de la FPGA (*DE2_NIOS_HOST_MOUSE_time_limited.sof*), la "imatge" de µCLinux (*zImage*), s'obre

un terminal en el µCLinux i s'executa l'aplicació de control del cronòmetre "cronometre_fase3" que interactua sobre el cronòmetre *hardware* tal com s'observa a la Figura 32:

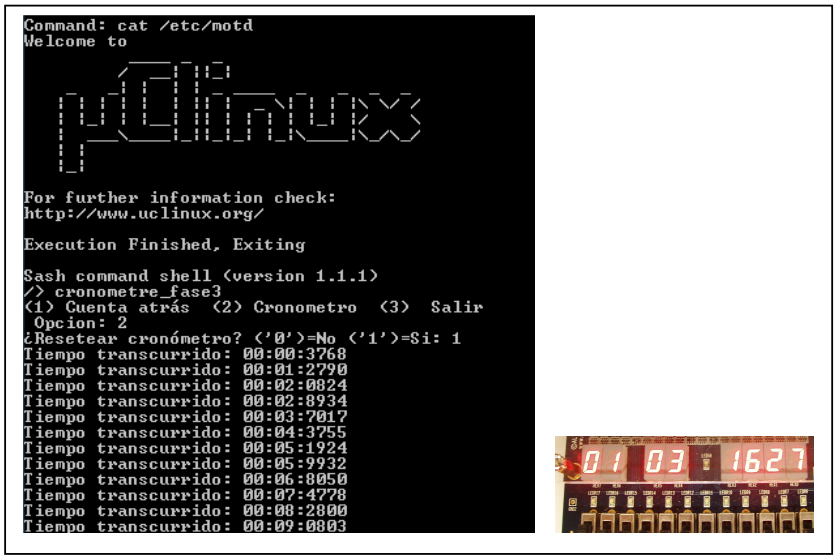


Figura 32 Aplicació de control del cronòmetre (Fase III)

3.6 Resultats del cronòmetre hardware amb control per software (Fase III)

Es realitzen diverses proves funcionals amb l'aplicació de control desenvolupada i s'observa que des de l'aplicació de control s'aconsegueix tenir el control del cronòmetre *hardware* (engegada/aturada, compte ascendent/descendent, càrrega d'un valor d'inicialització de compte enrere, etc.) i es pot visualitzar al mateix temps el compte del cronòmetre tant pels 8 displays de 7-segments com per l'aplicació de control.

Gràcies a que el mòdul *hardware* (cronòmetre) s'encarrega exclusivament del compte temporal, s'aconsegueix la precisió requerida de µs i l'exactitud d'aturada en arribar el compte descendent a 0. Aquesta precisió de µs és la mateixa que s'ha aconseguit durant la Fase II amb l'ús de la funció *gettimeofday()* de la llibreria *time.h* ja que en ambdós casos es treballa amb una base temporal *hardware*.

Amb aquest disseny “mixt” *software*/*hardware* s’aconsegueix gran flexibilitat en la programació de l’aplicació de control per afegir noves funcionalitats al cronòmetre, com per exemple, la càrrega d’un valor d’inicialització pel cas de compte enrere.

Un altre avantatge derivat d’aquest disseny mixt és l’estalvi de components *hardware* destinats inicialment al control del cronòmetre ja que no són necessaris en fer-se tot el control del cronòmetre mitjançant una aplicació *software*.

4. Desenvolupament del freqüencímetre

El projecte del freqüencímetre es desenvolupa, bàsicament, amb tres objectius:

- Demostrar que a partir de l'experiència adquirida en el disseny anterior del cronòmetre *hardware* integrat en un NIOS/CPU i partint d'un µCLinux instal·lat que serveix com a plataforma estàndard de desenvolupament, es pot afegir en el mateix NIOS/CPU un nou component (freqüencímetre) i una nova aplicació de control per aquest nou component de manera relativament ràpida i intuïtiva.

- Validar la base temporal de microsegons assolida amb el disseny del cronòmetre a partir d'un nou disseny *hardware* d'un freqüencímetre controlat per *software*. La validació es realitza a partir de la comparació entre la lectura de freqüència (rang de MHz) realitzada pel freqüencímetre dissenyat i la lectura de freqüència realitzada per un oscil·loscopi extern.

- Identificar el límit freqüencial que serà capaç de mesurar el freqüencímetre que es dissenyi considerant que el mòdul utilitzarà la base de rellotge per defecte de 100 MHz del NIOS/CPU.

4.1 Disseny hardware del freqüencímetre amb control per software

El disseny del freqüencímetre *hardware* controlat per *software* s'ha realitzat d'acord als requisits funcionals següents:

- A més del principal bloc mesurador de freqüència, el *hardware* ha d'incloure un bloc generador de senyal per tenir la possibilitat de generar internament per *hardware* el senyal a mesurar a partir de la freqüència especificada des de

l’aplicació de control. El bloc generador ha de permetre generar, com a mínim, un senyal amb un rang freqüencial de 0,1 Hz a 1MHz. La freqüència d’operació del rellotge de tots els blocs és de 100 MHz.

- Ha de permetre mesurar directament la freqüència d’un senyal extern mitjançant una entrada de tensió del bus d’expansió de la FPGA (GPIO) en un rang freqüencial, com a mínim, de 0,1 Hz a 1 MHz (període d’1 µs en el domini temporal).
- L’aplicació de control i el *hardware* específic han de permetre triar la mesura del senyal generat internament o la mesura d’un senyal extern.
- L’aplicació de control i el *hardware* específic han de permetre una mesura “*hardware*” de la freqüència del senyal mitjançant el freqüencímetre dissenyat i una mesura “*software*” de la freqüència del senyal directament des de l’aplicació de control per realitzar la comparativa de resultats entre una mesura *hardware* i una mesura *software*.

Per satisfer aquests requisits funcionals, s’ha realitzat la integració d’un freqüencímetre com un nou mòdul *hardware* específic dins del mateix NIOS/CPU utilitzat pel projecte del cronòmetre i es controla aquest mòdul directament amb una nova aplicació de control.

El diagrama de flux del disseny *hardware* del freqüencímetre amb control per *software* contempla les etapes que s’observen a la Figura 33:

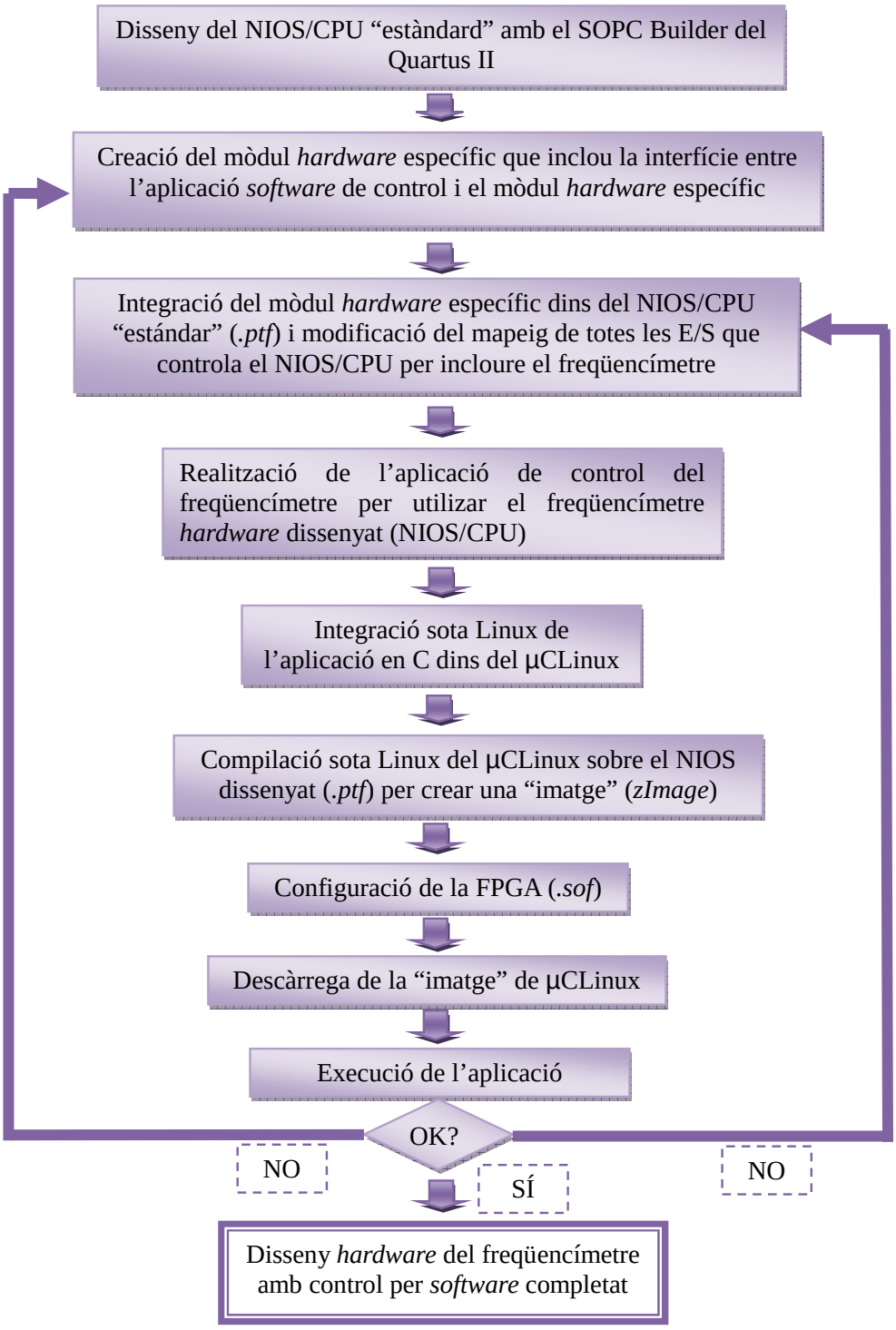


Figura 33 Diagrama de flux del disseny *hardware* del freqüencímetre amb control per *software*

A partir dels requisits funcionals del freqüencímetre, s'ha realitzat el disseny *hardware* implementant tres blocs *hardware* consecutius: un bloc "Generador" del senyal intern, un bloc "Selector" del senyal (intern/extern) i un bloc "Mesurador" de freqüència.

Es mostren en detall les arquitectures internes de cadascun dels 3 blocs que composen el *hardware* del freqüencímetre.

A la Figura 34 s'observa el primer bloc "Generador" que descriu l'arquitectura del bloc que genera un senyal intern a partir de la freqüència especificada en l'aplicació de control:

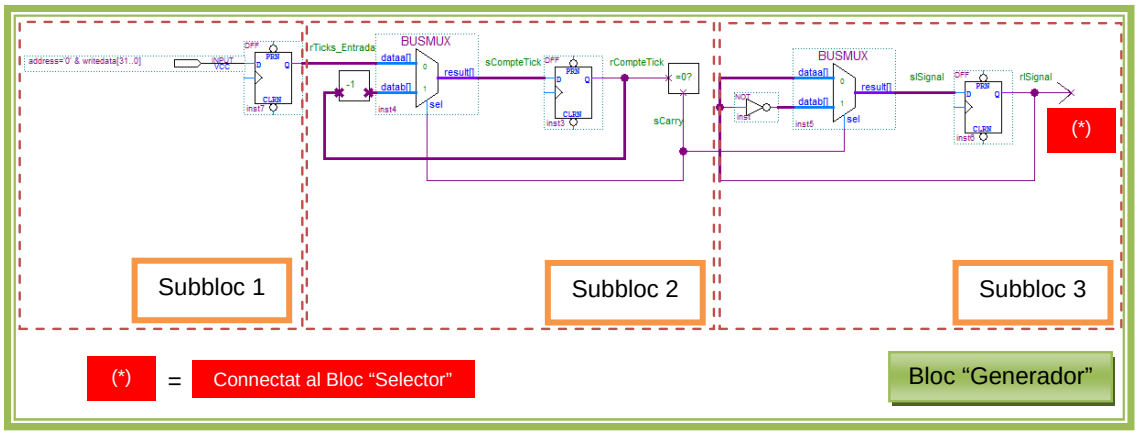


Figura 34 Bloc "Generador"

A partir de la freqüència del senyal seleccionada per l'usuari mitjançant l'aplicació de control del freqüencímetre, aquesta aplicació calcula el número de tics necessaris per generar mig període del senyal (*"numticksgenerar/2"*) tenint en compte la freqüència d'operació del rellotge del NIOS/CPU (freqüència de 100 MHz) tal com es mostra a la Equació 3:

$$\frac{\text{numticksgenerar}}{2} = \frac{1}{2}\left(\frac{Període}{0{,}00000001}\right) = \frac{1}{2}\left(\frac{100000000}{Freqüència}\right) \qquad \textbf{Eq. 3}$$

Aquest *"numticksgenerar/2"* es transmet des de l'aplicació de control al bloc *hardware* generador del senyal mitjançant el bus Avalon del NIOS/CPU. A la Figura 34 s'observa

l'entrada *"writedata[31..0]"* que correspon al *"numticksgenerar/2"*.

La sortida del bloc serà un senyal quadrat amb el període/freqüència especificat per l'usuari (senyal "rISignal").

L'esquema es basa en un comptador que compta el número de tics per generar mig període del senyal seleccionat amb l'aplicació de control. Per realitzar el compte es decrementa per cada tic de rellotge el "numticksgenerar/2" fins arribar el compte a 0. Mentre el compte no arriba a 0, el senyal de sortida –generat- es troba en estat '0'. En el moment en que el compte arriba a 0, s'activa el senyal "sCarry" i s'inverteix el senyal generat de '0' a '1'. Es torna a realitzar el compte des de "numticksgenerar/2" fins a 0 i quan el compte torna a arribar a 0, "sCarry" es torna a activar i s'inverteix el senyal generat novament de '1' a '0'. D'aquesta manera s'aconsegueix generar a la sortida d'aquest bloc un senyal quadrat amb la freqüència indicada per l'usuari a l'aplicació de control.

El *hardware* que forma el bloc "Generador" està format en un primer subbloc per un registre que emmagatzema l'últim valor de *"numticksgenerar/2"* (meitat del període) monitoritzant contínuament l'últim valor de freqüència seleccionat per l'usuari. El segon subbloc està format pels mateixos components *hardware* que formen els comptadors del cronòmetre, és a dir, dos multiplexors, un decrementador, un registre i un comparador. I el tercer subbloc de components *hardware* permet invertir el senyal i mantenir el seu estat mentre no s'activi el senyal "sCarry"i està format per un inversor, un MUX i un registre.

Els registres dels dos primers subblocs són de 32 bits perquè, segons els requisits funcionals, han d'emmagatzemar el número de tics mínim corresponent a mig període *("numticksgenerar/2"*) d'un senyal seleccionat amb una freqüència de 0,1 Hz (període=10s). A la Equació 4 es calcula el número mínim de tics que ha de registrar el bloc "Generador" per un senyal amb freqüència de 0,1 Hz prenent com a referència la Equació 3 que considera una freqüència d'operació del NIOS/CPU de 100 MHz:

$$Freq = 0{,}1Hz \rightarrow \frac{\text{numticksgenerar}}{2} = \frac{1}{2}\left(\frac{100000000}{0{,}1}\right) = 500000000ticks \qquad \textbf{Eq. 4}$$

Considerant que la màxima capacitat del senyal de dades del bus Avalon és de 32 bits, els registres es trien amb aquesta capacitat i a la Equació 5 es calcula el màxim número de tics que es poden registrar:

$$2^{32} = 4294967296 ticks \qquad \textbf{Eq. 5}$$

A partir d'aquest número de tics màxim que es pot registrar, a la Equació 6 es calcula la freqüència mínima d'un senyal que es pot generar mitjançant el bloc "Generador" dissenyat:

$$Freq_{mín} = \frac{1}{2}\left(\frac{100000000}{2^{32}}\right) = 0{,}0116 Hz \qquad \textbf{Eq. 6}$$

El registre del tercer subbloc que s'encarrega d'invertir el senyal generat, es tria d'1 bit perquè només ha de registrar l'estat del senyal quadrat que es genera, és a dir un '1' o un '0'.

A la Figura 35 s'observa el segon bloc "Selector" que descriu l'arquitectura del bloc que permet seleccionar mitjançant l'aplicació de control l'origen de la mesura del senyal, és a dir, un senyal generat internament o la mesura d'un senyal extern introduït directament pel bus d'expansió GPIO:

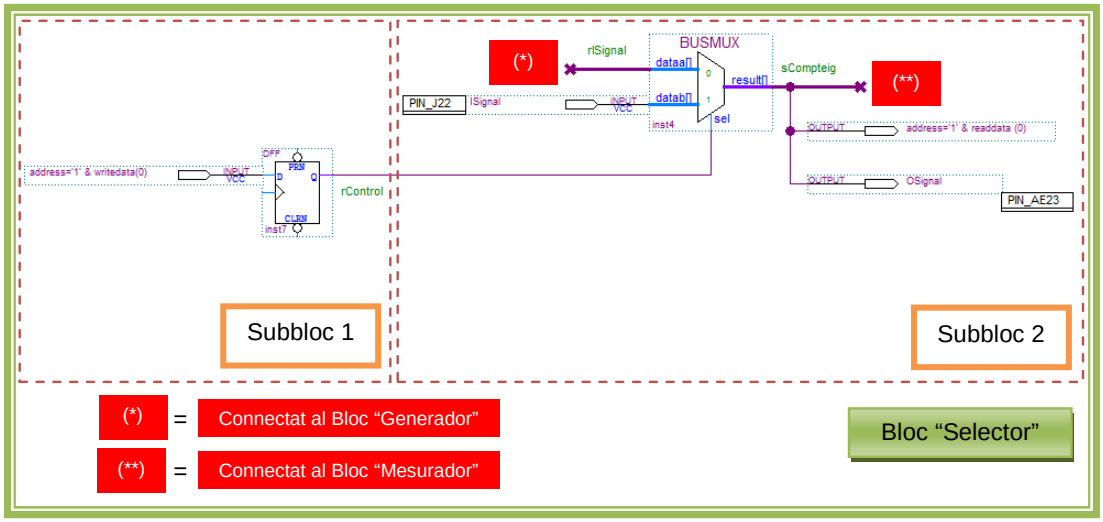


Figura 35 Bloc "Selector"

Aquest bloc té una entrada d'un bit procedent de l'aplicació de control que permet seleccionar el senyal generat internament pel bloc "Generador", o bé, seleccionar el senyal extern procedent directament del bus d'expansió GPIO.

Les dues entrades restants són el senyal generat internament ("rISignal") i el senyal extern procedent directament del bus d'expansió GPIO ("ISignal").

La sortida del bloc ("sCompteig") correspondrà a un dels dos senyals possibles, és a dir, a l'intern que genera el bloc "Generador" o a l'extern que prové del GPIO. Aquesta sortida es connecta amb el bloc "Mesurador" per tal de mesurar la freqüència del senyal i a la sortida d'un led vermell ("OSignal") per monitoritzar el senyal que es pretén mesurar. I també es connecta a una sortida d'un bit per tal de transmetre aquest senyal a l'aplicació de control i poder realitzar una mesura de freqüència per *software* que pugui ser comparada amb la mesura de freqüència *hardware* que realitzi el bloc "Mesurador".

El *hardware* que forma el bloc "Selector" està format en un primer subbloc per un registre que permet guardar contínuament la última selecció triada per l'usuari en quant a l'origen del senyal a mesurar , és a dir, '0' si desitja mesurar un senyal generat internament o '1' si desitja mesurar un senyal extern que provingui del bus GPIO. Per aquest motiu, el

registre és d'un bit.

El segon subbloc està format per un multiplexor que permet la selecció del senyal intern o extern a mesurar.

A la Figura 36 s'observa el tercer bloc "Mesurador" que descriu l'arquitectura del bloc que permet mesurar el número de tics de 0,01µs ("*numticksmesurar*" per una freqüència de 100 MHz) que es realitzen durant un període del senyal intern o extern i que permetrà a l'aplicació de control obtenir la freqüència del senyal.

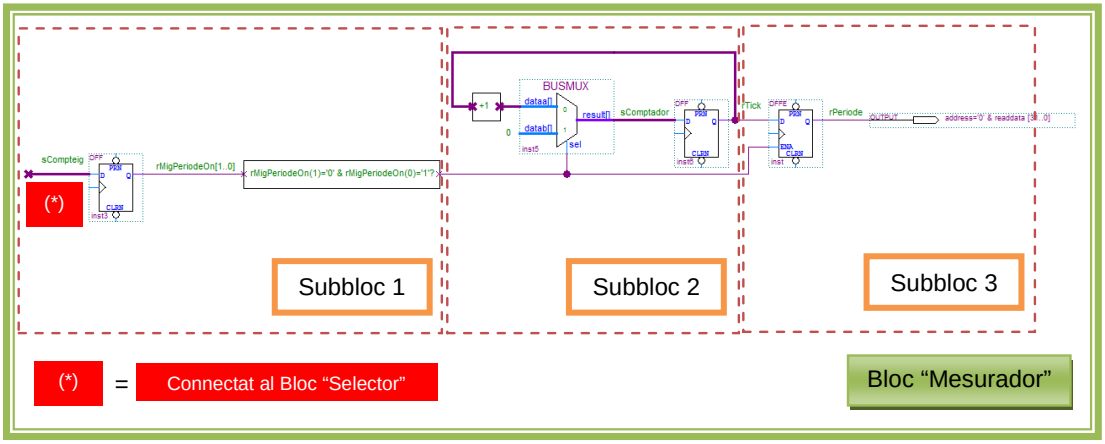


Figura 36 Bloc "Mesurador"

Aquest bloc té una entrada procedent del bloc "Selector" que contindrà el senyal a mesurar seleccionat per l'usuari.

El bloc té una sortida de 32 bits (igual a la capacitat de "*numticksgenerar*") que contindrà el número de tics mesurats ("*numticksmesurar*"). A partir de la mesura del número de tics, l'aplicació de control del freqüencímetre calcula la freqüència del senyal mesurat amb la Equació 7:

$$Freqüència(Hz) = \frac{100000000 ticks/s}{\text{numticksmesurar}} \qquad \textbf{Eq. 7}$$

Es calcula la precisió teòrica que tindrà el freqüencímetre en cada rang de mesura

considerant que el rellotge que utilitza el sistema de mesura té una freqüència de 100 MHz i calculant el marge d'error en *Hertz* que s'obtindria en mesurar un tic més o menys en cada mesura amb la Equació 8:

$$\text{error\_un\_tic(s)} = \left[\left(\frac{periode\_a\_mesurar}{periode\_"Mesurador"}\right) \pm 1\right] \times periode\_"Mesurador" = \left[\frac{freq\_"Mesurador"}{freq\_a\_mesurar} \pm 1\right] \times \frac{1}{freq\_"Mesurador"} \qquad \textbf{Eq. 8}$$

Aquest error temporal equival al error en freqüència que s'indica a la Equació 9:

$$\text{error\_un\_tic(Hz)} = \frac{1}{\text{error\_un\_tic(s)}} = \frac{1}{\left[\frac{freq\_"Mesurador"}{freq\_a\_mesurar} \pm 1\right] \times \frac{1}{freq\_"Mesurador"}} \qquad \textbf{Eq. 9}$$

A partir de la Equació 9, a la Equació 10 es calcula la precisió teòrica en la mesura d'1 MHz:

$$\text{error\_un\_tic(Hz)} \xrightarrow{\text{freq=1 MHz}} = \frac{1}{\left[\left(\frac{100E(6)}{1E(6)}\right) \pm 1\right] \times \frac{1}{100E(6)}} = \begin{cases} 1010101{,}01 \text{ Hz} = 1{,}01 \text{ MHz} \\ 990099{,}0099 \text{ Hz} = 990{,}099 \text{ KHz} \end{cases} \qquad \textbf{Eq. 10}$$

La precisió teòrica de la mesura d'1 MHz és de ± 10,10101 KHz.

A la Equació 11 es calcula la precisió teòrica en la mesura de 0,1 Hz:

$$\text{error\_un\_tic(Hz)} \xrightarrow{\text{freq=100 mHz}} = \frac{1}{\left[\left(\frac{100E(6)}{0{,}1}\right) \pm 1\right] \times \frac{1}{100E(6)}} = \begin{cases} 0{,}1000000001 \text{ Hz} \\ 0{,}0999999999 \text{ Hz} \end{cases} \qquad \textbf{Eq. 11}$$

La precisió teòrica de la mesura de 0,1 Hz és de ± 0,0000000001 Hz.

El *hardware* que forma el bloc "Mesurador" està format en un primer subbloc per un biestable i un comparador que permeten monitoritzar contínuament el moment en que es produeix un flanc positiu en el senyal a mesurar (inici d'un nou període) per tal de

començar a comptar els tics del període i realitzar una mesura fiable des de l'inici fins al final d'un període, evitant la mesura del número de tics abans de la finalització del cicle la qual cosa provocaria una mesura incorrecta de la freqüència per part de l'aplicació de control.

El segon subbloc consta d'un comptador (registre + incrementador) que permet comptabilitzar el número de tics durant el cicle del senyal. Per resetejar el comptador, es fa servir un multiplexor que carrega un "0" en el moment en que es produeix un flanc positiu en el senyal a mesurar (inici d'un nou període).

En el tercer bloc, s'utilitza un nou registre que conté el valor de tics mesurats totals ("*numticksmesurats"*) en la mesura del cicle del senyal. Aquest valor s'actualitza només quan es produeix una nova mesura del cicle del senyal, és a dir, quan es produeix un flanc positiu del senyal la qual cosa indica la finalització del cicle que s'estava mesurant i l'inici d'un nou cicle. Per tant, es tracta d'un registre amb senyal d'activació (*enable*) tal com es pot observar a la Figura 36.

A partir del disseny exposat anteriorment, s'ha realitzat el projecte en Quartus II, programant amb el *software* Emacs en VHDL cadascun dels tres blocs i interconnectant-los tal com s'observa a les Figures 34, 35 i 36 ([14] 2011).

Per integrar tot el *hardware* descrit del freqüencímetre en un sol component *"niosSenyalEntrada"* que pugui ser afegit al NIOS/CPU cal seguir els passos descrits a l'apartat B.2 de l'Annex B.

A la Figura 37 s'observa el nou component del freqüencímetre ("*niosSenyalEntrada"*) insertat en el mateix NIOS/CPU que pel cas del cronòmetre:

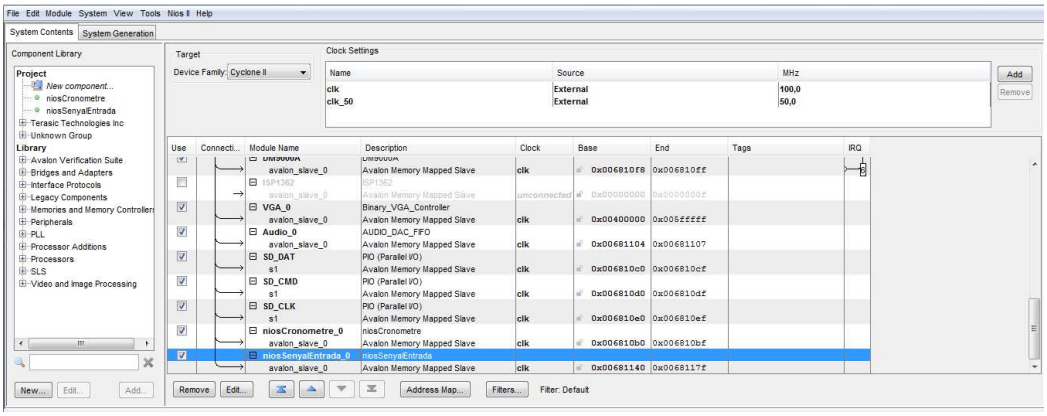


Figura 37 Component *"niosSenyalEntrada"*afegit al NIOS/CPU

El SOPC Builder assigna automàticament un rang de memòria pel nou component afegit (0x00681140 → 0x0068117f). Per accedir directament –obviant la memòria *cache* del NIOS/CPU- des de l'aplicació de control a les posicions de memòria que ocupa el mòdul del cronòmetre dins del NIOS/CPU cal considerar el rang de memòria assignat, però, a més, cal considerar el primer bit a '1' de cada posició de memòria. A la Taula 3 s'especifica el direccionament de memòria per cada control del freqüencímetre que s'ha considerat en el disseny tant de la interfície *software/hardware* com de l'aplicació de control:

Adreça	R/W	Adreça (accés directe)	Control
0x00681140	W	0x80681140 (address='0')	*numticksgenerar/2*
0x00681144	W	0x80681144 (address='4')	Selecció de senyal intern o extern
0x00681144	R	0x80681144 (address='4')	Senyal seleccionat
0x00681140	R	0x80681140 (address='0')	*Numticksmesurar*

Taula 3 Assignació d'adreces de memòria del component *"niosSenyalEntrada"*

Per tal que el NIOS/CPU consideri la entrada i sortida hardware del freqüencímetre, *"ISignal"* (senyal extern) i *"OSignal"* (senyal seleccionat per mesurar) respectivament, cal indicar-ho a l'arxiu descrit en Verilog *"DE2_NIOS_HOST_MOUSE_VGA.v"* on s'especifica el mapeig de totes les entrades i sortides que controla el NIOS/CPU dissenyat ([15] 2011).

A l'arxiu DE2_NIOS_HOST_MOUSE.qsf ([15] 2011) es mostra l'assignació complerta dels pins utilitzats en el disseny del freqüencímetre d'acord als components que controla el NIOS/CPU utilitzat ([7] 2011). La entrada *"ISignal"* s'ha assignat a una entrada del bus d'expansió GPIO i la sortida *"OSignal"* a un led vermell tal com s'observa a les Figures 38 i 39:

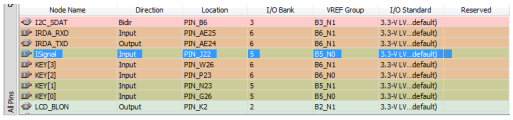


Figura 38 Assignació de pins del freqüencímetre (I)

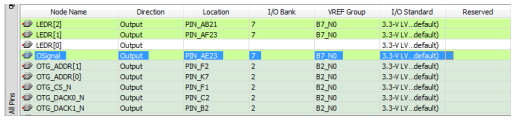


Figura 39 Assignació de pins del freqüencímetre (II)

En compilar el projecte *"DE2_NIOS_HOST_MOUSE.qsf"* s'actualitza l'arxiu *"DE2_NIOS_HOST_MOUSE_time_limited.sof"* ([15] 2011) i, tot i que és d'ús restringit per part d'Altera, conté el nou disseny del NIOS/CPU que inclou el mòdul del freqüencímetre dissenyat i permet configurar posteriorment la FPGA amb aquest disseny.

Es crea una nova aplicació de control que s'encarrega de controlar el mòdul *hardware* (freqüencímetre) integrat en el NIOS/CPU. Es programa l'aplicació mitjançant l'ús d'apuntadors que actuen sobre el rang de memòria indicat a la Taula 3, permetent a l'usuari:

- Triar el senyal a mesurar: un senyal generat internament o un senyal extern.
- Seleccionar la freqüència del senyal generat internament.
- Seleccionar el tipus de mesura de freqüència a realitzar: *software* (directament des de l'aplicació de control) o *hardware* (des del bloc "Mesurador").
- Visualitzar la freqüència mesurada pel terminal (*Shell*).

El codi font de l'aplicació *software* de control que controla el freqüencímetre *hardware* s'adjunta a l'arxiu *"frequencimetre.c"* ([15] 2011).

De la mateixa manera que s'ha realitzat amb les aplicacions desenvolupades del cronòmetre, s'integra aquesta aplicació dins del sistema µCLinux i es crea la "imatge" que s'ha de descarregar sobre la FPGA (*zImage*).

Amb el NIOS II Command Shell es descarrega l'arxiu de configuració de la FPGA (*DE2_NIOS_HOST_MOUSE_time_limited.sof*), la "imatge" de µCLinux (*zImage*), s'obre un terminal en el µCLinux i s'executa l'aplicació de control del freqüencímetre "frequencimetre" tal com s'observa a la Figura 40:

Figura 40 Aplicació de control del freqüencímetre

4.2 Resultats i validació del freqüencímetre hardware amb control per software

Amb l'objectiu de validar el disseny del freqüencímetre es realitza la mesura *software* i *hardware* de freqüència d'un senyal extern generat per un generador de funcions connectat a una de les entrades del bus d'expansió GPIO i es compara amb la mesura de freqüència del mateix senyal realitzada per un oscil·loscopi connectat directament a la sortida del generador de funcions. A la Figura 41 es mostra el muntatge realitzat on s'observen dos connectors "cocodril" vermell i negre que provenen del generador de funcions i una PCB amb una cinta plana que "transporta" el senyal del generador de funcions cap a una de les entrades del bus d'expansió GPIO:

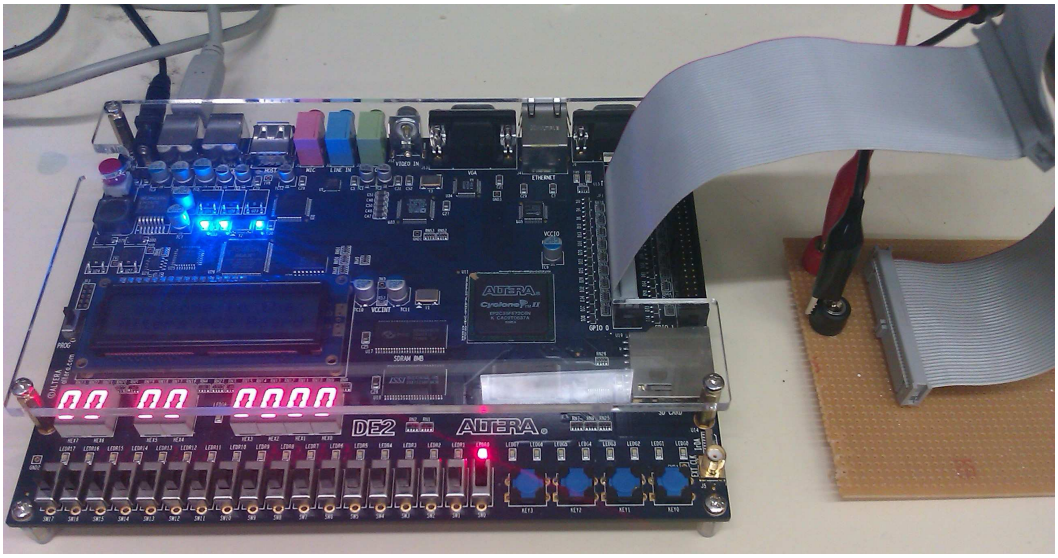


Figura 41 Muntatge per la validació del freqüencímetre

Es realitzen diverses mesures de freqüència comparatives en un rang freqüencial de 0,1 Hz fins a 1 MHz. Els resultats obtinguts s'indiquen a la Taula 4:

Oscil·loscopi	Freqüencímetre *hardware*	Freqüencímetre *software*	Precisió teòrica *hardware* (Hz)	Precisió mesurada *hardware* (Hz)
0,1 Hz	0,100 Hz	Mesura errònia	0,0000000001	0,000000000
0,5 Hz	0,5 Hz	Mesura errònia	0,0000000025	0,0000000025
1 Hz	1 Hz	Mesura errònia	0,00000001	0,00000001
100 Hz	100 Hz	Mesura errònia	0,0001	0,0001
1 KHz	1 KHz	Mesura errònia	0,01	0,01
100 KHz	100 KHz	Mesura errònia	100,1	100,1
500 KHz	500 KHz	Mesura errònia	2512,56	2512,56
1 MHz	1 MHz	Mesura errònia	10101,01	10101,01

Taula 4 Mesures de freqüència comparatives entre l'oscil·loscopi i el freqüencímetre dissenyat

Aquestes mesures conclouen que les mesures *software* de freqüència realitzades pel freqüencímetre no són fiables i, en canvi, les mesures *hardware* sí que ho són fins a 1 MHz amb la precisió indicada. Així es demostra que el freqüencímetre assoleix la precisió requerida d'1 µs en el domini temporal de la mateixa manera que s'aconsegueix durant la Fase III del cronòmetre.

Les mesures "*software*" de freqüència no són correctes, bàsicament per dos motius. El primer motiu és que l'aplicació de control necessita comptar contínuament el número de tics de la sortida "OSignal" del bloc "Selector" per calcular la freqüència i per "*software*" no és possible assegurar que el número de tics mesurats correspongui al 50% del cicle del senyal i es pugui realitzar la mesura correcta de la freqüència. El que acostuma a ocórrer es que el número de tics pot ser el resultat d'haver comptat un % del cicle menor del 50% cosa que provoca una mesura incorrecta de la freqüència per part de l'aplicació de control. El segon motiu és que el sistema operatiu en ser multitasca ha de gestionar a l'hora diferents aplicacions i en els instants en que no li dóna prioritat a l'aplicació de control produeix un retard en la lectura del número de tics que provoca una mesura incorrecta de la freqüència del senyal.

5. Creació d'un laboratori de pràctiques

A partir del projecte realitzat, es proposa la creació d'un laboratori de pràctiques amb l'objectiu que es puguin realitzar unes sessions pràctiques en les quals els alumnes puguin desenvolupar el projecte del cronòmetre en els dos àmbits de programació *hardware* i *software*, comprovin de primera mà els avantatges de cadascun dels dos àmbits de programació i, finalment, puguin desenvolupar un disseny "mixt" que aprofiti precisament els avantatges de cada àmbit.

A la Figura 42 es mostra el muntatge del laboratori de pràctiques:

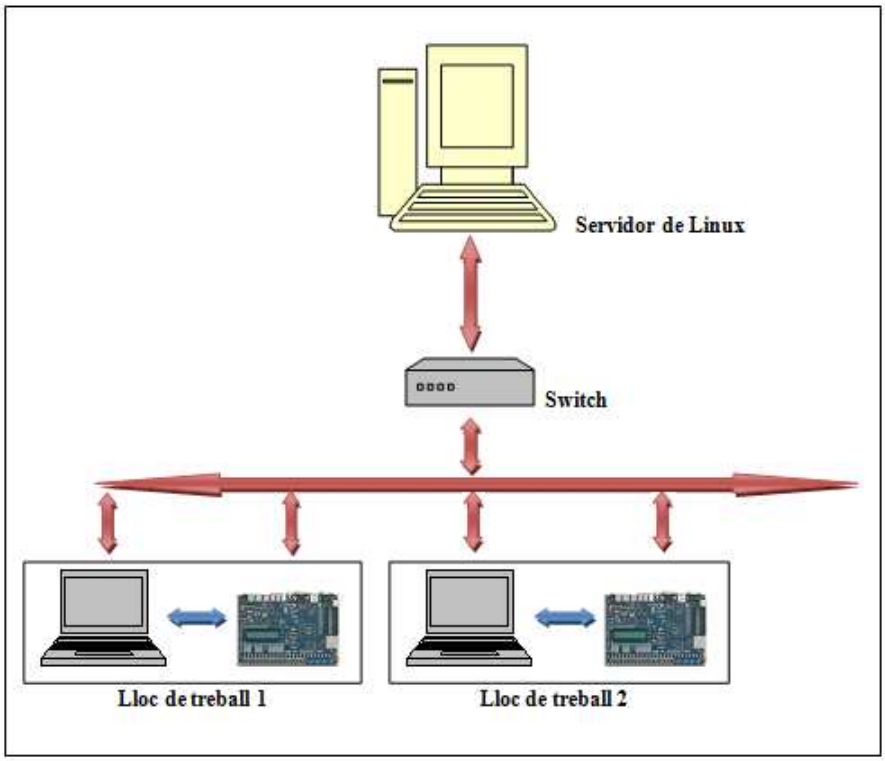


Figura 42 Muntatge del laboratori de pràctiques

A l'esquema de la Figura 42 només s'han representat dos llocs de treball, però realment es podrien muntar fins a 255 llocs de treball, cadascun d'ells amb un PC amb Windows o Linux que inclogués el programari especificat al capítol 2.3.2. El NIOS II Command Shell

pot instal·lar-se en Windows sobre un emulador de Linux (*cygwin*). Cada PC del lloc de treball es connectarà al *switch* amb un cable RJ45 i a la placa DE2 d'Altera per USB. El port Ethernet de cada placa també es connectarà al *switch* amb un cable RJ45.

Es disposarà d'un servidor de Linux amb un compte personal per cada lloc de treball, amb *NFS* instal·lat i un compilador *gcc* que permetrà als alumnes compilar i executar les aplicacions que desenvolupin, indiferentment si el S.O. de desenvolupament és Linux o Windows. Aquest servidor també es connectarà al *switch*.

Les sessions pràctiques d'implementació del cronòmetre es divideixen en tres parts: disseny *hardware*, disseny *software* i disseny "mixt" *software/hardware*.

En la primera part, els alumnes desenvoluparan el disseny *hardware* del cronòmetre programant el disseny en VHDL tal com s'ha realitzat a la Fase I descrita al capítol 3. La descàrrega del disseny sobre la FPGA la faran amb Quartus II. Realitzaran proves funcionals del disseny realitzat interactuant sobre els controls del cronòmetre assignats a la placa DE2 d'Altera.

En la segona part de les pràctiques, els alumnes desenvoluparan un disseny del cronòmetre purament *software* tal com s'ha realitzat a la Fase II descrita al capítol 3. L'aplicació *software es* programarà en C sobre un editor de text.

Per compilar i executar l'aplicació desenvolupada els alumnes es connectaran amb el seu PC al compte personal del servidor de Linux i "pujaran" l'aplicació desenvolupada a una carpeta compartida. Descarregaran sobre la FPGA el µCLinux amb el NIOS II Command Shell seguint els passos indicats al capítol 3.3 i obriran un terminal per seguir els passos indicats a l'Annex C. Hauran de configurar la xarxa Ethernet des de µCLinux per comunicar amb el servidor de Linux mitjançant una comunicació *Telnet* i hauran d'establir una sessió *NFS* (*Network File System*) com a *NFS client* que els permeti executar la seva aplicació dissenyada des del servidor (*NFS server*).

A la FPGA, al PC de cada lloc de treball i al servidor de Linux s'assignen IPs i adreces

MAC diferents dins de la mateixa subxarxa (192.168.1.0). A la Taula 5 s'indiquen les IPs que es configurarien per un muntatge amb cinc llocs de treball:

Dispositiu	IP	Adreça MAC
Servidor de Linux	192.168.1.1	Predefinida
PC 1	192.168.1.2	Predefinida
FPGA 1	192.168.1.3	00:07:ed:0a:03:00
PC 2	192.168.1.4	Predefinida
FPGA 2	192.168.1.5	00:07:ed:0a:03:01
PC 3	192.168.1.6	Predefinida
FPGA 3	192.168.1.7	00:07:ed:0a:03:02
PC 4	192.168.1.8	Predefinida
FPGA 4	192.168.1.9	00:07:ed:0a:03:03
PC 5	192.168.1.10	Predefinida
FPGA 5	192.168.1.11	00:07:ed:0a:03:04

Taula 5 Assignació d'IPs i adreces MAC als dispositius del laboratori de pràctiques

Per la tercera part de les pràctiques, a cadascun dels llocs de treball es facilitaran els arxius del disseny del cronòmetre realitzat a la Fase III del capítol 3 ([12] 2011) que inclou la interfície *software/hardware* de control i l'arxiu *"DE2_NIOS_HOST_MOUSE_time_limited.sof"* que utilitzaran per configurar la FPGA. I es facilitarà la Taula 2 del capítol 3 per tal que els alumnes tinguin present com s'ha dissenyat la interfície software/hardware i puguin desenvolupar una aplicació de control en C del cronòmetre *hardware*. També se subministrarà una "imatge" (*zImage*) amb un µCLinux compilat que els alumnes hauran de descarregar sobre la FPGA. Aquesta "imatge" s'haurà creat amb les opcions que es detallen a l'Annex A (*TCP/IP, Telnet, NFS)* d'acord al NIOS/CPU dissenyat.

Per compilar i executar l'aplicació de control desenvolupada els alumnes descarregaran el µCLinux amb el NIOS II Command Shell seguint els passos indicats al capítol 3.2 i obriran un terminal per seguir els passos indicats a l'Annex C.

6. Pressupost del projecte

Pel desenvolupament del present projecte cal disposar del *kit* de disseny DE2 *Development and Education Board* d'Altera. Depenent del tipus de destinatari del material, Altera ofereix dos preus diferents en la seva pàgina web ([3] 2011):

Ús acadèmic: 269$ (186,86€).
Ús comercial: 495$ (343,83€).

Per la creació del laboratori de pràctiques és necessari l'equipament i el material descrit al capítol 5. A la Taula 6 es mostra un llistat de preus aproximat de l'equipament i el material que costaria el muntatge de les pràctiques amb 12 llocs de treball:

Equipament/material	**Preu/unitat (€)**	**Quantitat**	**Subtotal (€)**
Placa DE2	186,86	12	2242,32
Servidor de Linux	800	1	800
PC personal	600	12	7200
Switch 32 ports	600	1	600
Cable ethernet	9	25	225
		TOTAL (aprox.)	**11067,32**

Taula 6 Llistat de preus de l'equipament o material per la creació del laboratori de pràctiques

7. Pla temporal del projecte

Les tasques realitzades en el projecte, la temporització de cadascuna d'elles i el percentatge del recurs utilitzat s'indiquen a la Taula 7. Es considera que una utilització del 100% del recurs correspondria a una dedicació de 8 hores de treball en el projecte:

Número tasca	Tasca	Inici	Final	Recurs	% recurs utilitzat
1	Desenvolupament del cronòmetre (Fase I)	06/09/2010	27/10/2010	Àngel G.	30%
2	Creació plataforma de desenvolupament en µCLinux	28/10/2010	16/12/2010	Àngel G.	30%
3	Desenvolupament del cronòmetre (Fase II)	17/12/2010	03/02/2011	Àngel G.	30%
4	Desenvolupament del cronòmetre (Fase III)	04/02/2011	06/04/2011	Àngel G.	30%
5	Desenvolupament del freqüencímetre	07/04/2011	09/06/2011	Àngel G.	30%
6	Proposta de creació d'un laboratori de pràctiques	10/06/2011	24/06/2011	Àngel G.	30%
7	El·laboració de la memòria tècnica	25/06/2011	31/08/2011	Àngel G.	80%

Taula 7 Pla temporal del projecte

A la Figura 43 s'observa el Diagrama de Gantt del projecte:

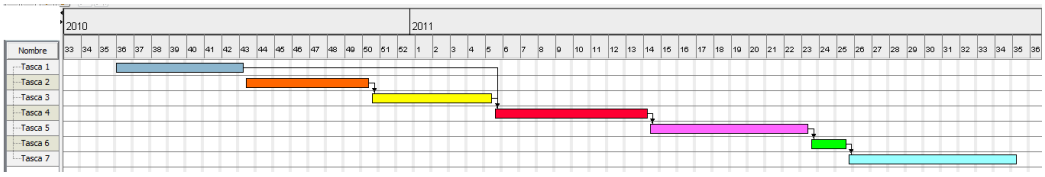


Figura 43 Diagrama de Gantt del projecte

8. Conclusions

A partir d'un model de disseny *bottom-up*, els dissenys purament *hardware* del cronòmetre i del freqüencímetre han permès obtenir la precisió de microsegons requerida. L'inconvenient d'un disseny totalment *hardware* és que degut a que no disposa d'una interfície de control ofereix poca flexibilitat a l'hora d'afegir de manera directa controls addicionals al sistema.

En el disseny purament *software* del cronòmetre integrat sobre un sistema µCLinux, configurat en una FPGA i gestionat per una CPU *embedded*, s'ha observat que l'avantatge que s'aconsegueix respecte el disseny *hardware* és una alta flexibilitat a l'hora d'afegir controlabilitat al sistema. Com a contrapartida, si l'aplicació dissenyada treballa amb una funció que pren com a base temporal el rellotge del sistema, el processador *embedded* se satura, el cronòmetre es rellenteix i no s'assoleix la precisió requerida.

En canvi, si l'aplicació *software* treballa amb una funció que pren com a base temporal el rellotge *hardware* del processador (*Real Time Clock*, RTC), el cronòmetre compta en temps real sense cap retard i amb la mateixa precisió que en el cas del disseny *hardware*.

Amb el desenvolupament dels dissenys "mixts" *hardware/software*, tant del cronòmetre com del freqüencímetre, s'han obtingut dos dissenys que aprofiten els avantatges d'ambdós àmbits de programació: alta precisió en la mesura de temps (µs) i una gran flexibilitat en la programació del sistema de control. Addicionalment, donat que el sistema de control es realitza per *software*, es produeix un estalvi de components *hardware* destinats inicialment al control del cronòmetre.

A partir dels resultats obtinguts en la mesura de freqüència *software* directament des de l'aplicació de control del freqüencímetre o *hardware* a partir del bloc "*Mesurador*" del freqüencímetre, es confirma que les mesures *software* no són fiables. El motiu que provoca que les mesures *software* no siguin fiables és que si la mesura es realitza directament des de l'aplicació de control, per "*software*" no és possible assegurar la medició d'un cicle

complert del senyal mesurat. En canvi, les mesures *hardware* són fiables fins a 1 MHz amb una precisió de ±10 KHz. Considerant-ho en el domini temporal, les mesures són fiables fins a 1 µs amb una precisió de ±10ηs. Per obtenir una major precisió tant en el cronòmetre com en el freqüencímetre seria necessari l’ús d’un rellotge amb una freqüència superior als 100 MHz del rellotge utilitzat.

Amb el desenvolupament dels dos projectes s’ha comprovat que segons els requeriments funcionals que s’hagin de satisfer, és millor optar per un o altre tipus de programació – *hardware* o *software*-, o bé, triar un disseny “mixt” que aprofiti els avantatges de cadascun dels dos modes de programació. Això és el que comprovaran els alumnes de primera mà amb la creació del laboratori de pràctiques.

9. Possibles línies de futur del projecte

Les aplicacions de control del cronòmetre i del freqüencímetre s'han desenvolupat per executar-les en línea de comandes d'un terminal de µCLinux.

Una millora que es podria realitzar a les aplicacions seria desenvolupar-les amb un entorn gràfic de control aprofitant la sortida VGA de la placa DE2 d'Altera. La programació es podria realitzar, per exemple, amb MicroWindows. D'aquesta manera s'oferiria a l'usuari un sistema de control més intuïtiu i pràctic.

D'altra banda, amb l'obtenció del µCLinux instal·lat s'ha creat una plataforma estàndard de desenvolupament que permet afegir ràpidament qualsevol aplicació programada en C i executar-la en µCLinux sobre la FPGA de la placa DE2 d'Altera. En el projecte s'ha aprofitat aquesta plataforma creada per afegir les aplicacions de control del cronòmetre i del freqüencímetre, però es poden afegir infinitat d'aplicacions que interactuïn directament sobre el NIOS/CPU i, a la vegada, aprofitin tots els perifèrics i ports de comunicació que ofereix la placa DE2 d'Altera.

10. Bibliografia

[1]. *Altera Corporation, FPGA Architecture.* 12 / Agost / 2011. http://www.altera.com/literature/wp/wp-01003.pdf (últim accés: 11 / 5 / 2011).

[2]. A *Embedded Systems Design*, de Steve Heath. Newnes, 2003.

[3]. *Altera Corporation, DE2 Development and Education Board.* 12 / Agost / 2011. http://www.altera.com (últim accés: Març / 7 / 2011).

[4]. *Altera Corporation, Quartus II.* 12 / Agost / 2011. http://www.altera.com (últim accés: 3 / Gener / 2011).

[5]. *Altera Corporation, Nios II.* 12 / Agost / 2011. http://www.altera.com (últim accés: 11 / Febrer / 2011).

[6]. *Gutiérrez Bravo, Ángel. CD adjunt del present PFC; carpeta "cronometre_fase_1"; subcarpeta "vhdl files"*. Barcelona, 3 / Juny / 2011.

[7]. *Altera Corporation, DE2 User Manual.* 12 / Agost / 2011. ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf (últim accés: 11 / Gener / 2011).

[8]. En *VHDL Lenguaje Estándar de Diseño Electrónico*, de Lluís Terés, Yago Torroja, Serafín Olcoz y Eugenio Villar. McGraw-Hill, 1998.

[9]. *Gutiérrez Bravo, Ángel. CD adjunt del present PFC; carpeta "cronometre_fase_1"*. Barcelona, 3 / Juny / 2011.

[10]. *Gutiérrez Bravo, Ángel. CD adjunt del present PFC; carpeta "cronometre_fase_2"*. Barcelona, 3 / Juny / 2011.

[11]. *Altera Corporation, DE2 demonstrations.* 12 / Agost / 2011. ftp://ftp.altera.com/up/pub/de2_demonstrations.zip; carpeta DE2_NIOS_HOST_MOUSE_VGA (últim accés: 20 / Gener / 2011).

[12]. *Gutiérrez Bravo, Ángel. CD adjunt del present PFC; carpeta "cronometre_fase_3"*. Barcelona, 21 / Juny / 2011.

[13]. *Gutiérrez Bravo, Ángel. CD adjunt del present PFC; carpeta "cronometre_fase_3"; subcarpeta "vhdl files"*. Barcelona, 21 de Juny de 2011.

[14]. *Gutiérrez Bravo, Ángel. CD adjunt del present PFC; carpeta "frequencimetre"; subcarpeta "vhdl files"*. Barcelona, 27 de Juny de 2011.

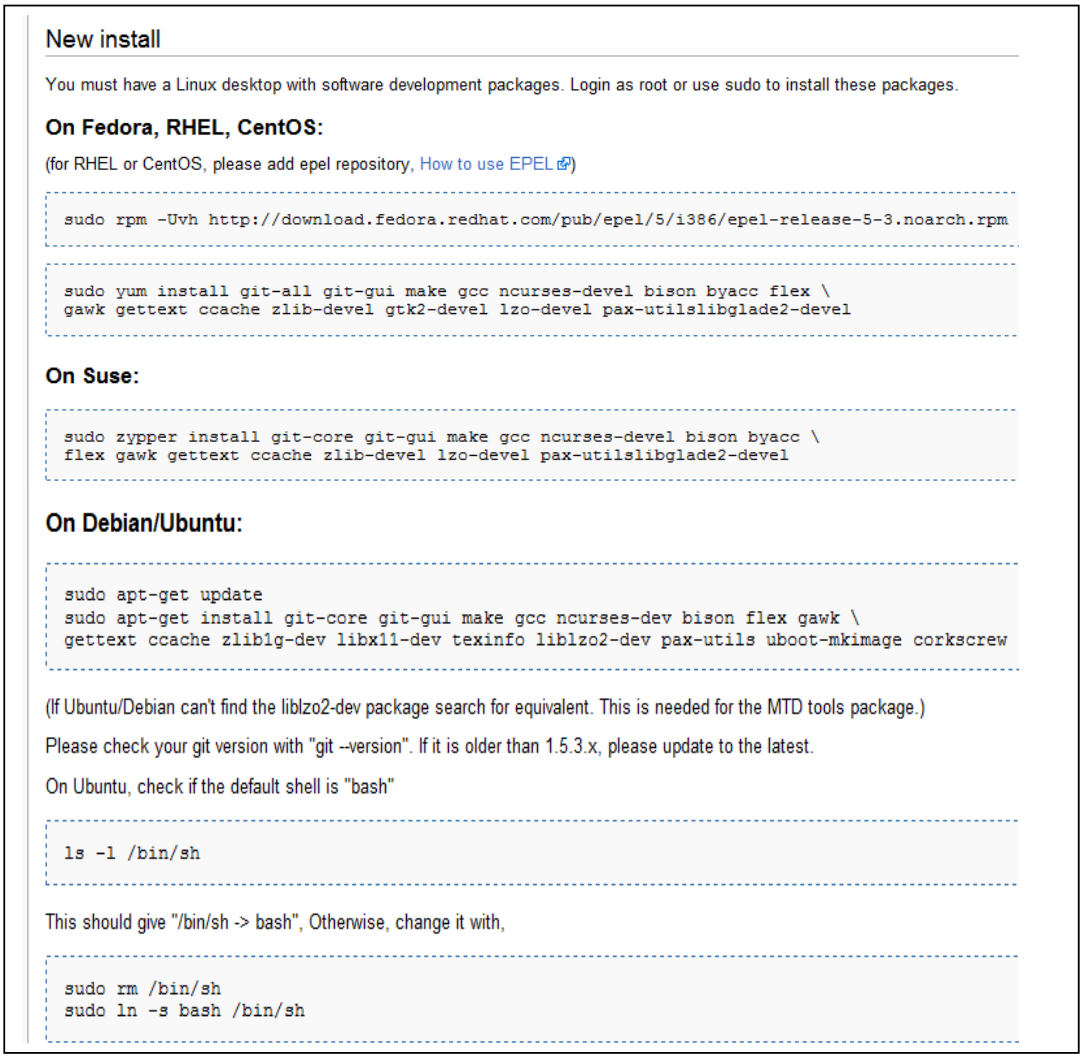
[15]. *Gutiérrez Bravo, Ángel. CD adjunt del present PFC; carpeta "frequencimetre"*. Barcelona, 27 de Juny de 2011.

[16]. *Altera Corporation, NIOS FTP.* 12 / Agost / 2011. http://www.niosftp.com (últim accés: 20 / Setembre / 2010).

[17]. *Altera Corporation, NIOS Wiki.* 12 / Agost / 2011. http://www.alterawiki.com (últim accés: 17 / Març / 2011).

Annex A: Compilació de µCLinux

El sistema µCLinux es troba disponible en un repositori d'Internet ([16] 2011). Per aconseguir instal·lar-lo obtenint així una plataforma estàndard de desenvolupament preparada per afegir noves aplicacions, com per exemple, les aplicacions de control del cronòmetre i del freqüencímetre, cal seguir una sèrie de passos sota un sistema Linux ([17] 2011). A les Figures 44, 45, 46 es presenta el manual d'instal·lació del µCLinux:



New install

You must have a Linux desktop with software development packages. Login as root or use sudo to install these packages.

On Fedora, RHEL, CentOS:

(for RHEL or CentOS, please add epel repository, How to use EPEL)

```
sudo rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm
```

```
sudo yum install git-all git-gui make gcc ncurses-devel bison byacc flex \
gawk gettext ccache zlib-devel gtk2-devel lzo-devel pax-utilslibglade2-devel
```

On Suse:

```
sudo zypper install git-core git-gui make gcc ncurses-devel bison byacc \
flex gawk gettext ccache zlib-devel lzo-devel pax-utilslibglade2-devel
```

On Debian/Ubuntu:

```
sudo apt-get update
sudo apt-get install git-core git-gui make gcc ncurses-dev bison flex gawk \
gettext ccache zlib1g-dev libx11-dev texinfo liblzo2-dev pax-utils uboot-mkimage corkscrew
```

(If Ubuntu/Debian can't find the liblzo2-dev package search for equivalent. This is needed for the MTD tools package.)

Please check your git version with "git --version". If it is older than 1.5.3.x, please update to the latest.

On Ubuntu, check if the default shell is "bash"

```
ls -l /bin/sh
```

This should give "/bin/sh -> bash", Otherwise, change it with,

```
sudo rm /bin/sh
sudo ln -s bash /bin/sh
```

Figura 44 Instal·lació de µCLinux (I)

followed by a logout and log back in again.

As root, check if you have "cc" which is a symlink to "gcc".

```
which gcc

gcc -v
which cc

cc -v
```

if not,

```
cd /usr/bin

ln -s gcc cc
```

GCC4.2

Before compiling the 20090703 toolchain on a linux ubuntu/kubuntu 9.04 you need to make sure to install gcc version 4.2 (older) and gcc's file system link is properly pointing to it:

```
sudo apt-get install gcc-4.2
sudo rm /usr/bin/gcc
sudo ln -s gcc-4.2 /usr/bin/gcc
```

If this isn't done tool chain compilation will buffer overflow on nios2-linux-uclibc-ar

Nios2 Linux Tarball

Make sure you have 6GB or more free disk space. Please use wget to download the tar file nios2-linux-20100621.tar , 1.6GB. Best thanks to Altera kindly hosting these files. Please verify the sha1sum after download. (but don't use Windows ftp, it corrupts the files)

```
wget http://www.niosftp.com/pub/linux/nios2-linux-20100621.tar
```

```
sha1sum nios2-linux-20100621.tar
```

We use "git" to keep the source. You make check out GitServer later. The git database inside the tar file was compressed, so we don't do compression on the tar file.

You can build in any working directory as a user account, eg. in your home (say /home/hippo). The tar file contains a nios2-linux dir.

Figura 45 Instal·lació de µCLinux (II)

```
tar xf <path_to>nios2-linux-20100621.tar  # untar the package


cd nios2-linux
ls   # see what's in


binutils  insight          u-boot              use_http_for_update
checkout  linux-2.6        uClibc              use_ssh443_for_update
elf2flt   README           uClinux-dist
gcc3      sshkey           update
glibc     toolchain-build  use_git_for_update

# check out the source
```

```
./checkout
```

Now the source files for the Nios2 uClinux and gnutools are ready.

- **linux-2.6**: the Linux kernel source, which is stock Linux kernel plus Nios2 specific patches. The current version is v2.6.30.
- **uClinux-dist**: the uClinux userspace libraries and applications. We will build uClinux here.
- **binutils, gcc3, elf2flt, insight**: the gnu tools source.
- **uClibc**: the main userspace libraries, which is smaller that glibc. please note, newlib doesn't support Nios2 uClinux.
- **u-boot**: a powerful boot loader and monitor, DasUBoot.

Toolchain

We will proceed to build the toolchain. If you don't want or fail to build it yourself, you may use the prebuild BinaryToolchain.

If you work on x86_64, 64 bits platform, you will need to change the arch to 32 bits , with "setarch i386" , to build gcc.

```
cd toolchain-build

gcc --version
git clean -f -x -d   # clean on restart
make gcc elf2flt gdb-host   # setarch i386 make gcc elf2flt gdb-host,  if you work on x86_64 platform



                          # make -j6 gcc elf2flt gdb-host #if you have a multi-core machine, -j(number of cores +2)


cd ..
```

It might take hours to build.

The default installation path is toolchain-build/build/nios2. Then setup the PATH for the tools, you can add a line at the end of file ~/.bash_profile (or ~/.bashrc on Debian/Ubuntu) (the file is hidden, you have to use "ls -a" to find it . For "gedit" use open Location, and enter the file name), like this

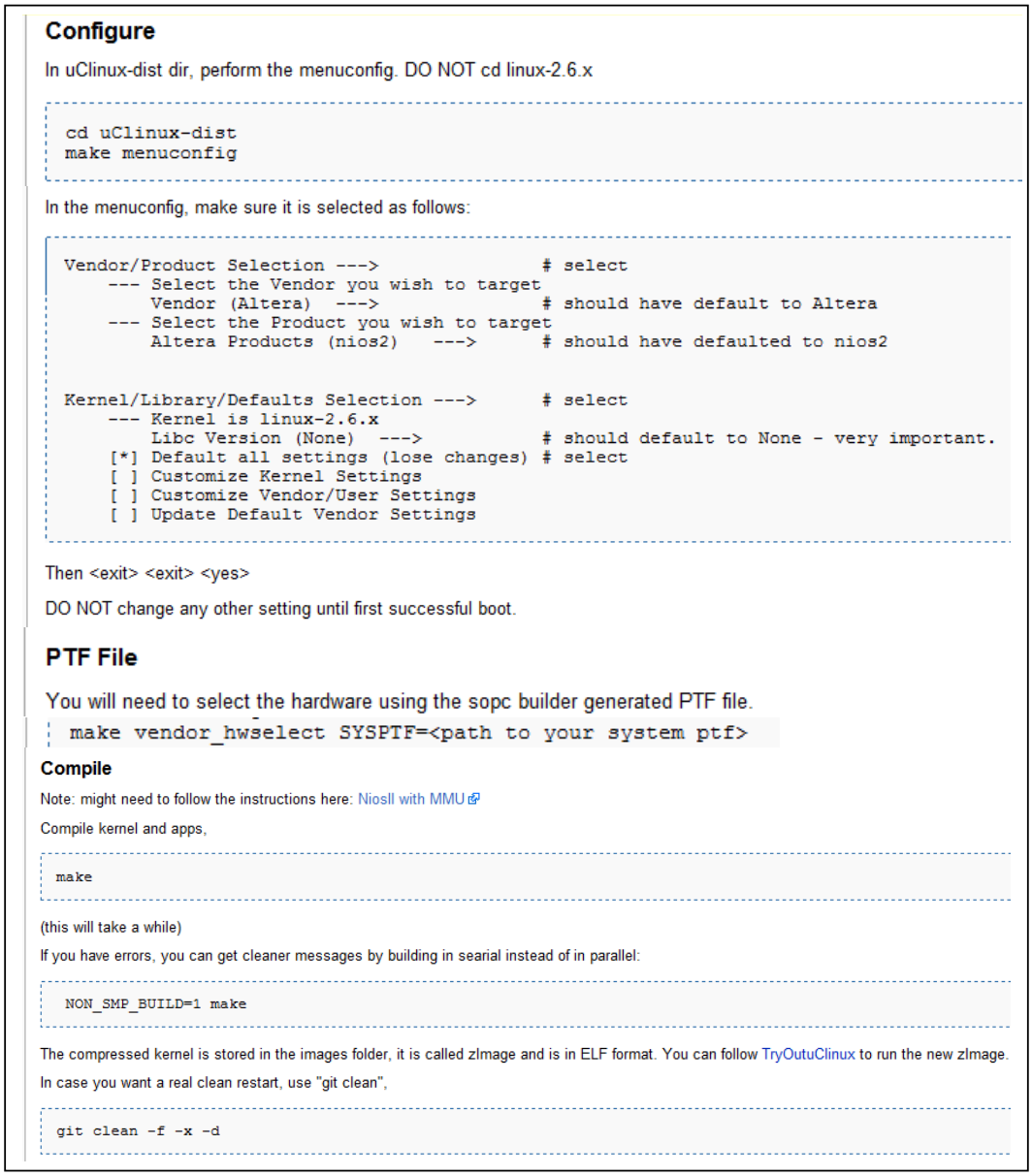
```
PATH=$PATH:/home/hippo/nios2-linux/toolchain-build/build/nios2/bin
```

Logout and login again. You can use the tools now. Run this to verify that you have it in your command search path,

```
nios2-linux-uclibc-gcc -v
```

Figura 46 Instal·lació de µCLinux (III)

Després d'instal·lar el µCLinux es comprova que es permet compilar amb les opcions per defecte i generar una "imatge" (*zImage*) seguint els passos indicats a la Figura 47:



Configure

In uClinux-dist dir, perform the menuconfig. DO NOT cd linux-2.6.x

```
cd uClinux-dist
make menuconfig
```

In the menuconfig, make sure it is selected as follows:

```
Vendor/Product Selection --->               # select
    --- Select the Vendor you wish to target
        Vendor (Altera)  --->               # should have default to Altera
    --- Select the Product you wish to target
        Altera Products (nios2)   --->      # should have defaulted to nios2


Kernel/Library/Defaults Selection --->      # select
    --- Kernel is linux-2.6.x
        Libc Version (None)  --->           # should default to None - very important.
    [*] Default all settings (lose changes) # select
    [ ] Customize Kernel Settings
    [ ] Customize Vendor/User Settings
    [ ] Update Default Vendor Settings
```

Then <exit> <exit> <yes>

DO NOT change any other setting until first successful boot.

PTF File

You will need to select the hardware using the sopc builder generated PTF file.

```
make vendor_hwselect SYSPTF=<path to your system ptf>
```

Compile

Note: might need to follow the instructions here: NiosII with MMU

Compile kernel and apps,

```
make
```

(this will take a while)

If you have errors, you can get cleaner messages by building in searial instead of in parallel:

```
NON_SMP_BUILD=1 make
```

The compressed kernel is stored in the images folder, it is called zImage and is in ELF format. You can follow TryOutuClinux to run the new zImage.

In case you want a real clean restart, use "git clean",

```
git clean -f -x -d
```

Figura 47 Configuració i compilació de µCLinux amb les opcions per defecte

Un cop es genera la “imatge” satisfactòriament, el següent pas és afegir noves aplicacions al sistema µCLinux, en aquest cas, les aplicacions de control del cronòmetre i del freqüencímetre. Es presenten dos mètodes per compilar i executar una nova aplicació en µCLinux:

- Mètode 1: Integrar-la directament en µCLinux i compilar-la conjuntament amb el sistema.
- Mètode 2: Amb el µCLinux compilat per defecte, obrir un terminal, comunicar per *Telnet* amb un PC amb Linux, obrir una sessió *NFS* per muntar en µCLinux un directori compartit i compilar i executar l’aplicació des del PC amb Linux. Aquest mètode és el que s’ha proposat en la creació del laboratori de pràctiques del capítol 5 i es descriu a l’Annex C.

A la Figura 48 es presenten els passos a seguir necessaris per afegir una nova aplicació (en aquest exemple, “*hello.c”*) segons el primer mètode:

Add hello apps to uClinux-dist (recommended)

Please follow the doc to add your user apps,

nios2-linux/uClinux-dist/Documentation/Adding-User-Apps-HOWTO

But change foo to hello.

```
cd nios2-linux/uClinux-dist
<p>mkdir user/hello
</p>
```

Edit user/Makefile,

add a line in the sorting order.

```
    dir_$(CONFIG_USER_HELLO_HELLO)             += hello
```

Edit user/Kconfig

after menu "Miscellaneous Applications".

```
config USER_HELLO_HELLO
   bool "hello"
   help
     This program prints hello world.
```

Create user/hello/Makefile

EXEC = hello

```
OBJS = hello.o

all: $(EXEC)

$(EXEC): $(OBJS)

    $(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)

romfs:

    $(ROMFSINST)     /bin/$(EXEC)
clean:
    -rm -f $(EXEC) *.elf *.gdb *.o
```

Create user/hello/hello.c.

```
#include <stdio.h>

int main(void)
{
    printf("hello world\n");
}
```

Then make menuconfig and select [*] hello. Now you can make. Boot nios2 and run "hello".

Figura 48 Passos per integrar directament una aplicació en µCLinux

Prèviament a la compilació definitiva del µCLinux, cal accedir novament al menú de configuració de µCLinux per afegir els drivers "a la carta" d'acord al NIOS/CPU dissenyat i seleccionant les aplicacions de control del cronòmetre i del freqüencímetre afegides tal com es mostra a les Figures 49, 50, 51, 52. Per obtenir ajuda en la selecció dels drivers i aplicacions es recomana consultar la wiki de NIOS ([17] 2011).

Next Steps

Networking

Now, you can config uClinux-dist for costmize kernel and select your apps.

Still in uClinux-dist dir, DO NOT cd linux-2.6.x

```
make menuconfig
Kernel/Library/Defaults Selection --->
(linux-2.6.x) Kernel Version
(None) Libc Version
[ ] Default all settings (lose changes)
[*] Customize Kernel Settings    <== to change kernel config
[*] Customize Vendor/User Settings  <== to change user apps config
[ ] Update Default Vendor Settings
```

Then <exit> <exit> <yes> . It will enter kernel config first, then it will enter user apps config, you can select more apps. After you change config,

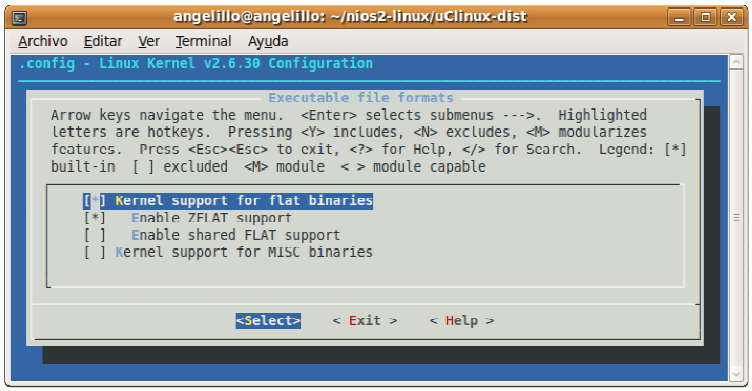


Figura 49 Configuració de µCLinux d'acord al cronòmetre i freqüencímetre dissenyats (I)

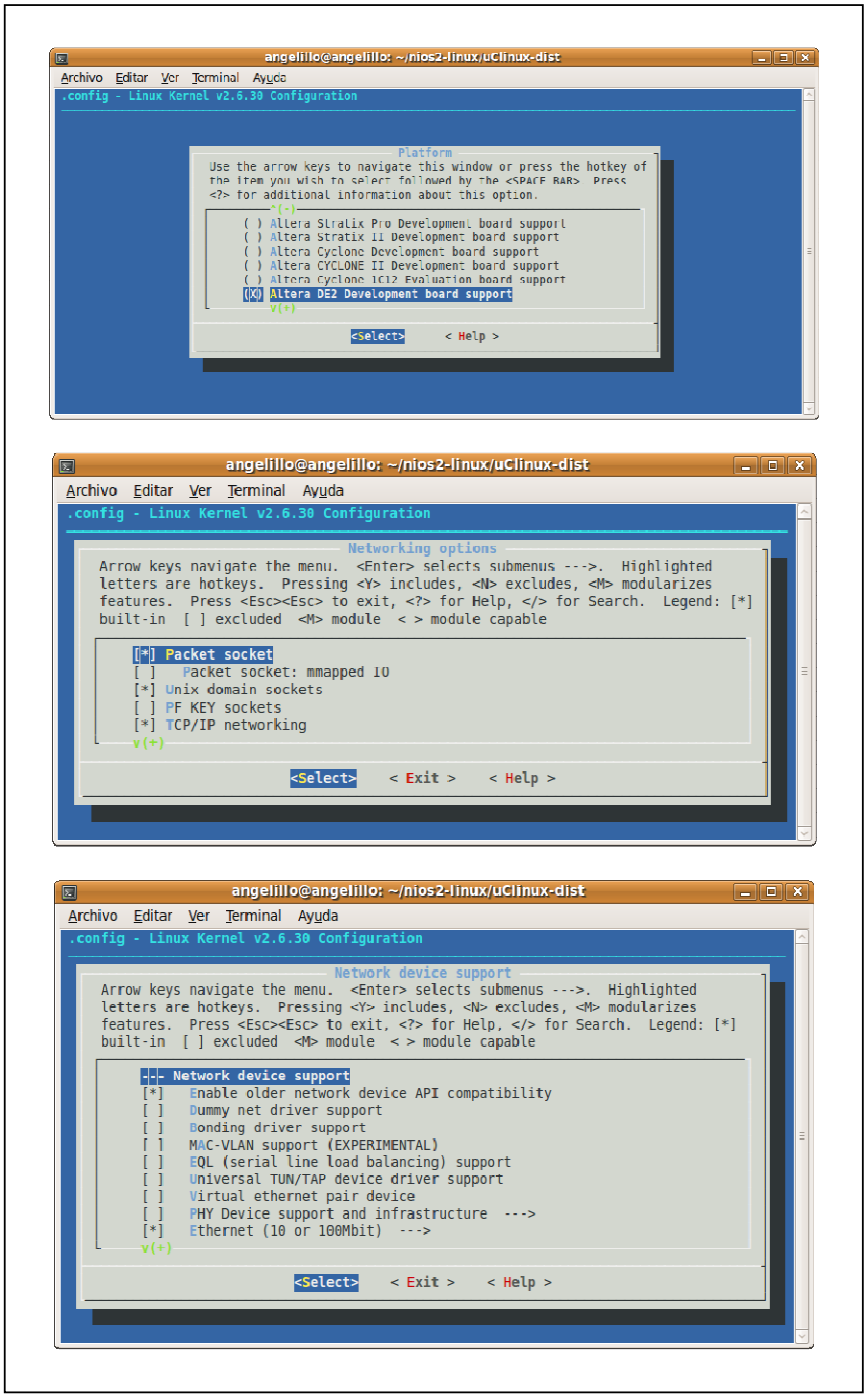


Figura 50 Configuració de µCLinux d'acord al cronòmetre i freqüencímetre dissenyats (II)

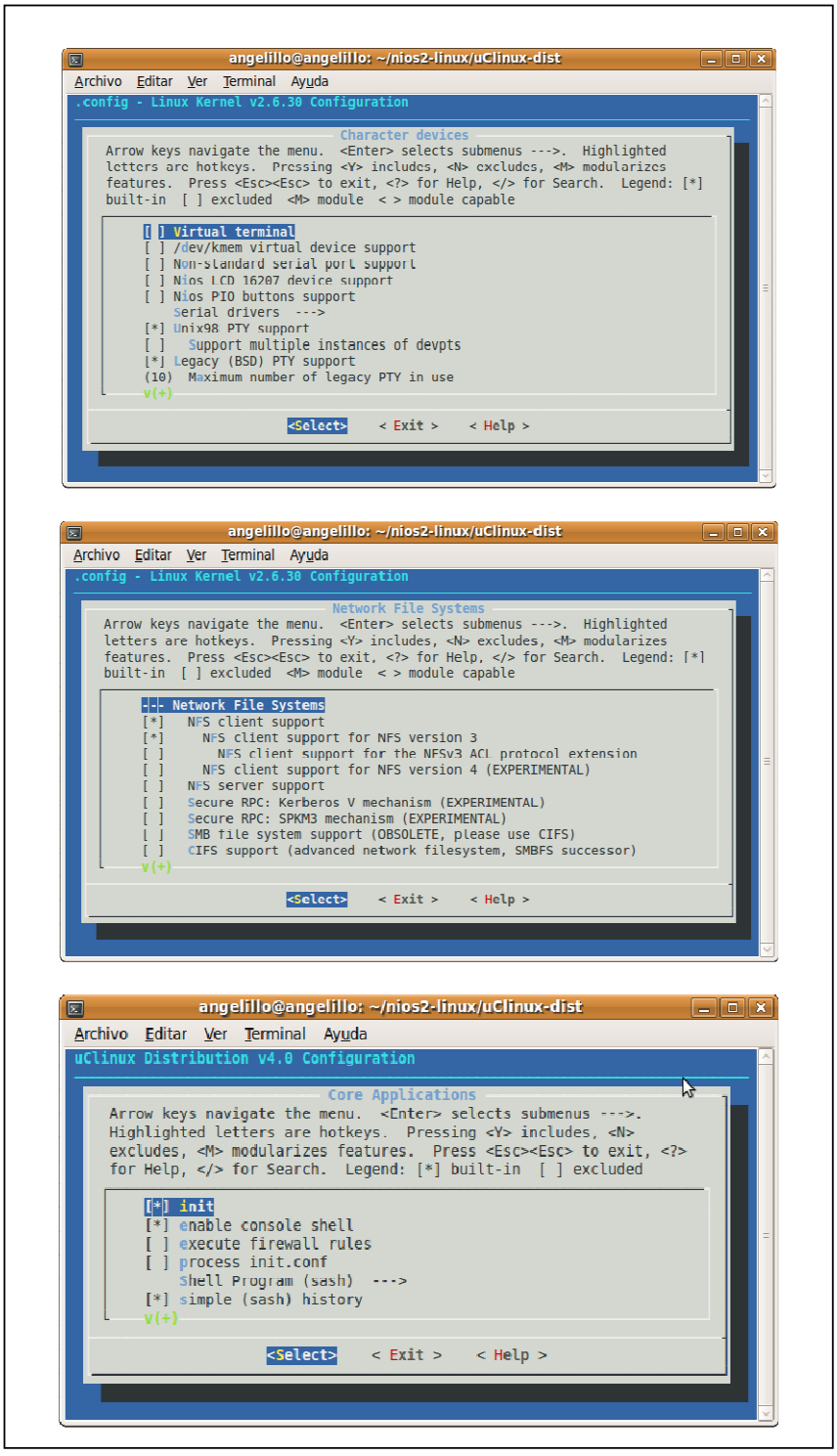


Figura 51 Configuració de µCLinux d'acord al cronòmetre i freqüencímetre dissenyats (III)

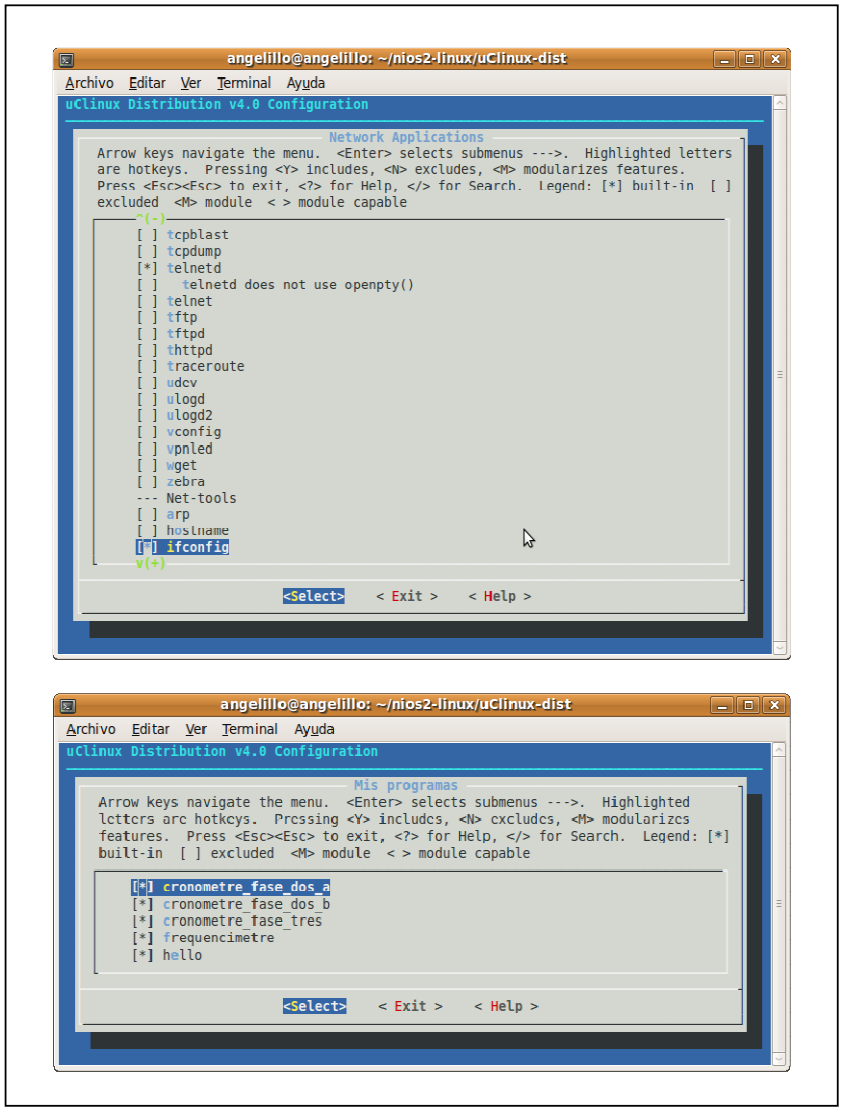


Figura 52 Configuració de µCLinux d'acord al cronòmetre i freqüencímetre dissenyats (IV)

I, per últim, amb la comanda "*make*" es compila el sistema µCLinux generant la "imatge" (*zImage*) que es descarregarà sobre la FPGA i que conté les aplicacions de control del cronòmetre i del freqüencímetre.

Annex B: Integració dels mòduls *hardware* en un NIOS/CPU

B.1 Integració del component "niosCronometre"

Per integrar tot el *hardware* descrit del cronòmetre dins del NIOS/CPU cal seleccionar el menú *Tools→SOPC Builder* del Quartus II. S'obre la eina SOPC Builder que permet dissenyar el NIOS/CPU i, com es parteix del projecte *DE2_NIOS_HOST_MOUSE_VGA* ([11] 2011), a la Figura 53 s'observa que automàticament apareix el NIOS/CPU "estàndard" que s'havia dissenyat per aquest projecte. Pel projecte del cronòmetre, d'aquests components s'utilitzen els següents: "CPU" (*kernel*, NIOS), la memòria "SDRAM" per descarregar el µCLinux i el "jtag_uart" per configurar la FPGA des d'un PC extern :

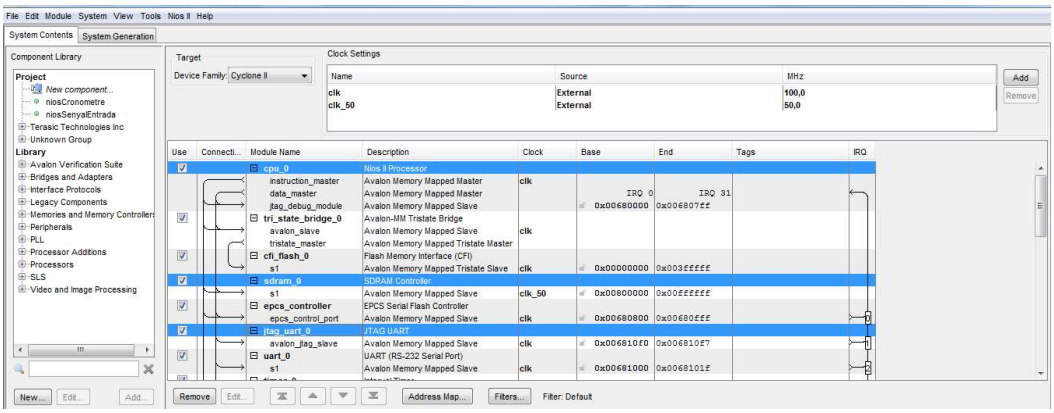


Figura 53 NIOS/CPU utilitzat pel cronòmetre

S'han mantingut tots el components que apareixen excepte el *"ISP1362"* que s'utilitza per la comunicació amb el port USB ja que pels requisits del projecte no s'utilitza i tampoc s'instal·la el driver corresponent en el µCLinux.

El següent pas és la creació del mòdul *hardware* específic anomenat *"niosCronometre"* per afegir-lo com un mòdul addicional *("component")* al NIOS/CPU anterior. Per fer-ho se selecciona *"New Component..."* i com s'observa a la Figura 54 a l'etiqueta *"HDL Files"* s'afegeix l'arxiu definit com a *"Top"* en el disseny *hardware* (*"niosCronometre.vhd"*) i l'arxiu *"pcrono.vhd"* que és el *"package"* dels components descrits a l'arquitectura del

cronòmetre:

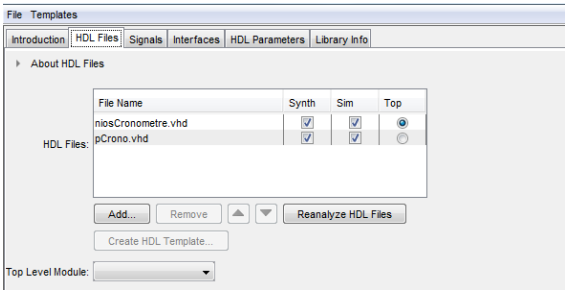


Figura 54 Arxius HDL per la creació del component *"niosCronometre"*

A la següent etiqueta anomenada *"Signals"* es realitza l'assignació dels *"signals"* a cadascuna de les interfícies, bàsicament al bus Avalon i a les entrades i sortides del propi *hardware* específic. Per realitzar l'assignació s'han de configurar correctament els camps *"Signal Type"* i *"Direction"* tal com s'observa a la Figura 55:

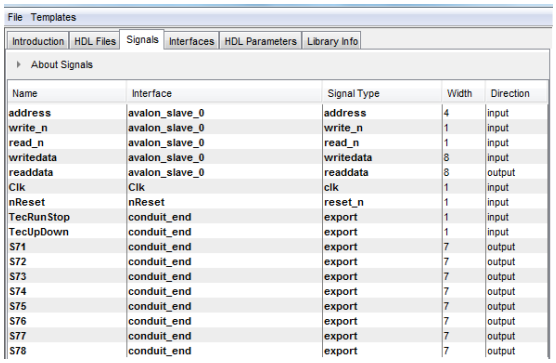


Figura 55 Senyals per la creació del component *"niosCronometre"*

La següent etiqueta *"Interfaces"* permet configurar cada interfície de manera específica. A les Figures 56, 57, 58, 59 s'observa la configuració realitzada per cadascuna de les interfícies:

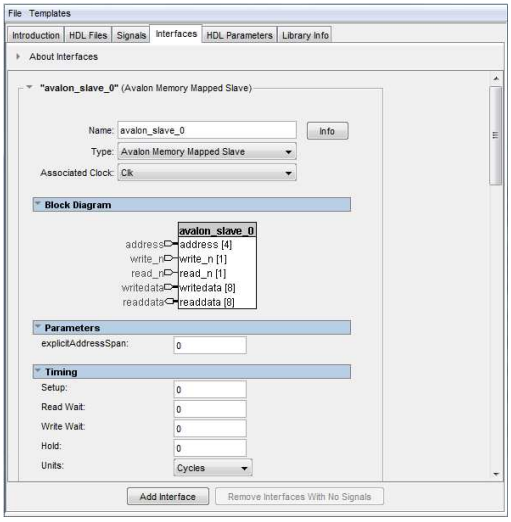


Figura 56 Interfícies per la creació del "niosCronometre" (I)

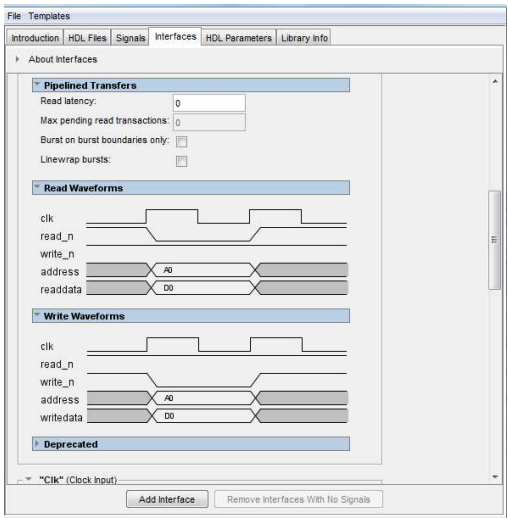


Figura 57 Interfícies per la creació del "niosCronometre" (II)

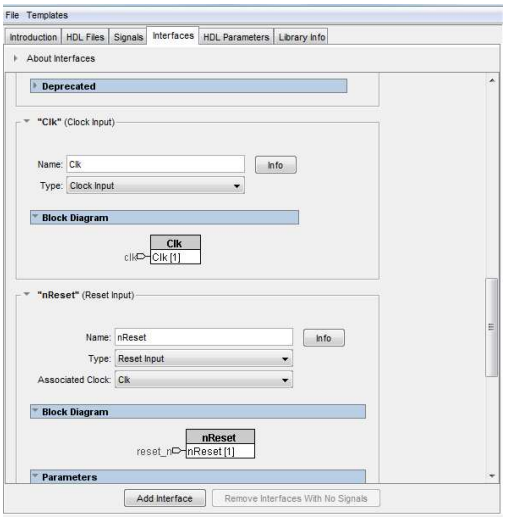


Figura 58 Interfícies per la creació del "niosCronometre" (III)

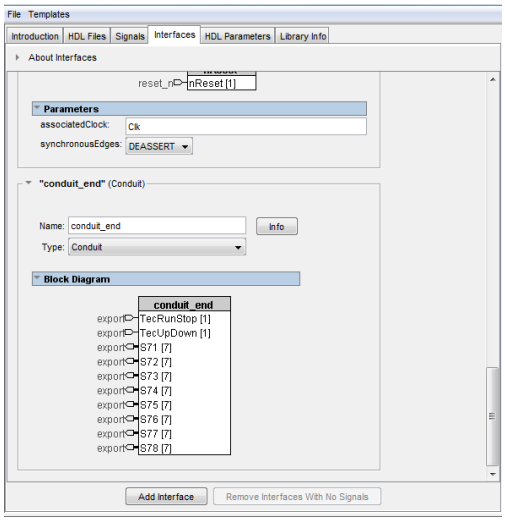


Figura 59 Interfícies per la creació del "niosCronometre" (IV)

Per últim, cal seleccionar *"Finish"* per completar la creació del component *"niosCronometre"*.

Per finalitzar la integració del nou component al NIOS/CPU "estàndard", només cal seleccionar *"Generate"* per actualitzar el NIOS/CPU amb el nou mòdul *hardware* específic afegit (cronòmetre).

B.2 Integració del component "niosSenyalEntrada"

Els passos a seguir per a la integració són exactament els mateixos que s'han explicat pel cas del cronòmetre. A la Figura 60 s'observa l'arxiu HDL que és necessari definir com a *"Top"* per a la creació del nou component:

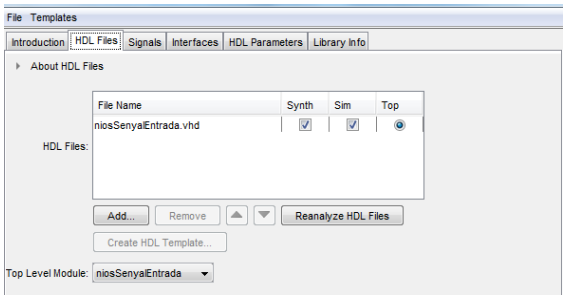


Figura 60 Arxiu HDL per la creació del component *"niosSenyalEntrada"*

El nou component tindrà les mateixes senyals (*"Signals"*) que en el cas del cronòmetre per accedir al component mitjançant el bus Avalon des de l'aplicació de control. És a dir, *"address[3..0]", "write_n", "read_n", "writedata[31..0]" i "readdata[31..0]"*. A més, com en el cas del cronòmetre, tindrà dues entrades pel rellotge i pel reset dels registres (*Clk i nReset*). Addicionalment, tindrà una entrada *hardware* per connectar a una entrada del bus d'expansió GPIO que permeti connectar un senyal extern generat a partir d'un generador de funcions i una sortida *hardware* per connectar a un led vermell que permeti monitoritzar el senyal a mesurar. A la Figura 61 s'observen tots els *"Signals"* del nou component:

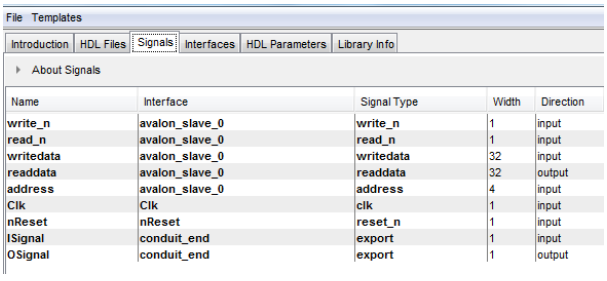


Figura 61 Senyals per la creació del component *"niosSenyalEntrada"*

Un cop es completa la creació del nou component ja apareix en el menú "*Component Library*" per ser seleccionat.

Annex C: Tutorial bàsic de µCLinux aplicat a la sessió pràctica

Aquest tutorial bàsic de µCLinux ([17] 2011) pretén servir de guia pel muntatge de la sessió pràctica proposada en el capítol 5.

Per establir la comunicació Ethernet des de µCLinux, primer es defineix l'adreça MAC hardware de la placa. El seu valor ha d'estar dins del rang següent: 00:07:ed:0a:03:<valor aleatori 00-ff>. Per fer-ho, dins de µCLinux es configura amb la comanda de la Figura 62:

```
/> ifconfig eth0 hw ether 00:07:ed:0a:03:29
```

Figura 62 Comanda per definir l'adreça MAC en Linux

Es configuren les adreces IP de la placa (192.168.1.3) i del servidor de Linux (192.168.1.1) amb la comanda de la Figura 63 amb direccions que es troben dins de la mateixa subxarxa (192.168.1.0):

```
/> ifconfig eth0 192.168.1.3
```

Figura 63 Comanda per definir l'adreça IP en Linux

S' "aixeca" la xarxa des del terminal de µCLinux utilitzant la comanda que s'indica a la Figura 64:

```
/> ifconfig eth0 up
```

Figura 64 Comanda per "aixecar" una xarxa en Linux

Es comprova que hi ha enllaç físic entre la placa DE2 i el servidor de Linux executant la comanda que s'indica a la Figura 65:

```
/> ping 192.168.1.1
```

Figura 65 Comanda per comprovar l'enllaç físic d'una xarxa Ethernet en Linux

Un cop existeix enllaç físic, s'estableix una comunicació *Telnet* des del µCLinux i el servidor de Linux amb la comanda indicada a la Figura 66:

```
/> telnet 192.168.1.1
```

Figura 66 Comanda per comunicar per *Telnet* en Linux

El següent pas és establir una sessió *NFS* (*NFS client*) per muntar en µCLinux un directori compartit pel servidor de Linux (*NFS server*) que permeti compilar i executar sota Linux qualsevol aplicació amb el compilador *gcc*.

Abans d'establir la sessió NFS és necessari que des del µCLinux es comprovi que el "mapejador" de ports de Linux, *"portmap"*, es troba instal·lat i s'està executant amb la comanda indicada a la Figura 67. Aquest "mapejador" s'encarrega de gestionar l'assignació entre aplicacions i ports:

```
/> portmap &
```

Figura 67 Comanda per comprovar el "mapejador" en Linux

En el µCLinux es crea un directori per muntar el directori compartit del servidor de Linux tal com s'indica a la Figura 68:

```
/> mkdir /mnt/nfs
```

Figura 68 Comanda per crear un directori en Linux

S'estableix la sessió *NFS* entre el µCLinux i el servidor de Linux executant la comanda

indicada a la Figura 69:

```
/> mount 192.168.1.1:/home/nios2 /mnt/nfs
```

Figura 69 Comanda per establir una sessió *NFS* en Linux

Es compila l'aplicació amb la comanda indicada en la Figura 70:

```
/> gcc –o "nomAplicacio" nomAplicacio.c
```

Figura 70 Comanda per compilar una aplicació amb *"gcc"* en Linux

I s'executa l'aplicació de control tal com s'indica a la Figura 71:

```
/> ./nomAplicacio
```

Figura 71 Comanda per executar una aplicació en Linux

Amb les comandes descrites anteriorment, els alumnes podran compilar, executar i comprovar sobre la placa DE2 que l'aplicació de control que dissenyin controla satisfactòriament el cronòmetre *hardware* mitjançant el NIOS/CPU.

El µCLinux és un sistema Linux *embedded* "a la carta" que permet la gestió de gran funcionalitat (ports de comunicació, perifèrics, etc). en funció dels drivers que se seleccionin prèviament a la compilació. A la wiki de NIOS ([17] 2011) s'ofereixen tutorials que ajuden a executar correctament totes les comandes disponibles.

Ángel Gutiérrez i Bravo

Signatura:

Bellaterra, 14 de setembre de 2011