Disseny d'un sistema HW/SW de Control Remot amb dispositius FPGA

Memòria del Projecte Fi de Carrera
d'Enginyeria Electrònica
realitzat per Imar Estopà Mata
i dirigit per Jordi Carrabina Bordoll i Borja Martinez

Bellaterra 16 de Juny del 2008

RESUM

El projecte que es presenta a continuació, té com a objectiu implementar un sistema HW/SW encastat en una FPGA, capaç d'executar funcions de control remot per infraroig en plataformes de televisió flexibles de Sony Corp. El disseny obtingut, s'incorporarà a un sistema més ampli de verificació i test de circuits impresos, dins del marc de producció SMD.

La finalitat d'aquest projecte, és la realització d'un sistema flexible per a la implementació de comandaments de comunicació per infraroig amb circuits impresos.

Prèviament, s'ha estudiat els conceptes bàsics referents a la implementació de sistemes amb FPGAs, la seva metodologia de desenvolupament i les principals característiques de la seva arquitectura. Com a especificacions, s'ha utilitzat l'estàndard de control remot per infraroig de *Sony Corp SIRCS (Sony Infrared remote control system)*

RESUMEN

El proyecto que se presenta a continuación, tiene como objetivo implementar un sistema HW/SW empotrado en una FPGA, que sea capaz de ejecutar funciones de control remoto por infrarrojo en plataformas de televisión flexibles Sony Corp. El diseño obtenido, se incorporara en un sistema más amplio de verificación y testeo de circuitos impresos, dentro del marco de la producción SMD

La finalidad de este proyecto, es la realización de un sistema flexible para la implementación de comandos de comunicación por infrarrojo con circuitos impresos

Previamente, se ha estudiado los conceptos básicos referentes a la implementación de sistemas con FPGAs, su metodología de desarrollo i la principales características de su arquitectura, así como el estándar de control remoto por infrarrojo de *Sony Corp. SIRCS (Sony Infrared remote control system)*

SUMMARY

The project that appears next, it's oriented to the FPGA implementation of embedded HW/SW system, which is capable of executing infrared remote control functions for TV platforms of *Sony Corp.* The design obtained will join to a PCB's verification and testing system.

The objective of this project, is to create flexible system in order to implement an infrared communication command's with PCB's.

Previously, it's been studied the referring basic concepts to the implementation of FPGA's systems, their development methodology, and their main architecture characteristics, as well as the infrared remote control system standard of *Sony Corp*. *SIRCS (Sony Infrared remote control system)*

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, Jordi Carrabina Bordoll

professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFIQUEN:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Imar Estopà Mata

I per tal que consti firma la present.

Signat: .....

Bellaterra, 16 de Juny de 2008.

Índex

Capítol 1.

Objectius i planificació del projecte

1.1. Objectius del projecte

L'objectiu del present projecte és dissenyar un prototip HW/SW encastat en una FPGA, capaç d'executar funcions de control remot per infraroig en plataformes de televisió flexibles de *Sony Corp*. El disseny obtingut, s'incorporarà a un sistema més ampli de verificació i test de circuits impresos, dins del marc de producció SMD.

La finalitat d'aquest projecte, és la realització d'un sistema flexible per a la implementació de comandaments de comunicació per infraroig amb circuits impresos. La figura 1.1 mostra el sistema bàsic utilitzat per a comprovar el correcte funcionament del disseny un cop implementat.

Figura 1.1.- Esquema del sistema de comunicació entre ordinador i plataforma de prototipatge.

Les especificacions en el disseny de la comunicació per infraroig, es regeixen a l'estàndard de comunicació *SIRCS* (*Sony Infrarred Remote Control Systems* - SS-00116) de Sony Corp. on es detallen els paràmetres de les trames i els tipus de transmissió possibles. Des del punt de vista *hardware*, el disseny del el sistema està implementat amb FPGAs d'Altera.

Els dissenys encastats en FPGAs són un camp emergent en el món de la microelectrònica, la capacitat d'implementar tot el sistema en un sòl xip (SoC) i la possibilitat de reconfigurar-ne la implementació segons les necessitats de l'usuari de manera senzilla, fa que la utilització

d'aquests sistemes, ens ofereixi grans avantatges respecte la utilització de microprocessadors convencionals.

La companyia Altera proporciona, de forma gratuïta, un processador virtual (*soft-core*) anomenat Nios II, amb gran flexibilitat de configuració tant de l'arquitectura del processador com del *bus-on-chip* i dels perifèrics. La implementació del sistema de control remot per infraroig es basa en mapar sobre l'arquitectura *system-on-chip* de la FPGA, el nostre disseny (bus, entorn HW/SW, configuració, etc), i executar un *software* de control i test sobre un processador Nios II.

El desenvolupament d'aquest projecte final de carrera s'ha dut a terme conjuntament amb l'empresa "Sony Bcn Tec".

1.2. Contingut de la memòria

L'elaboració de la memòria s'ha estructurat en 6 capítols, on es tracten totes les etapes de desenvolupament d'aquest projecte. El capítol 1 mostra els objectius generals d'aquesta memòria, alguns dels trets distintius i les etapes de desenvolupament d'aquest projecte.

En el segon capítol s'aborda la teoria bàsica i es mostren els conceptes necessaris per a la l'elaboració d'aquest projecte. Entre aquests conceptes hi trobem les especificacions del sistema de control remot per infraroig de *Sony Corp.*, el concepte de component virtual (IP-Cores), els *busos-on-chip*, en especial el bus Wishbone i l'Avalon, utilitzats en aquest projecte, el microprocessador Nios II, les característiques d'aquest, i la descripció de la seva arquitectura. Aquests conceptes formen la base de partida per a l'elaboració del nostre disseny.

Al tercer capítol s'exposa la metodologia de treball seguida en el desenvolupament d'aquest projecte, així com els fluxos de disseny tant *Hardware* com *Software*. També es mostra l'entorn i les eines de disseny utilitzades per a realitzar el disseny d'aquest projecte. Com apunt final del capítol es mostra la l'entorn de prototipat utilitzat, les seves característiques i els elements utilitzats.

En el quart, s'exposa l'arquitectura del sistema, el desenvolupament *hardware* amb els respectius components i mòduls implementats i la interconnexió entre ells. Pel què respecta al desenvolupament del *software*, es dona una visió detalla del programa que ha d'executar el Nios II, així com els paràmetres bàsics a configurar i l'estructura del *software*, i la implementació d'aquest en una llibreria dins del SOPC (*System-on-programable-chip*). Per acabar el capítol, es mostra el disseny de l'entorn gràfic realitzat a nivell d'usuari.

Al capítol cinquè es mostra l'estratègia de test, així com el resultats obtinguts i les simulacions amb oscil·loscopi, simuladors lògics, etc. que verifiquen el correcte funcionament del sistema implementat.

Finalment a l'últim capítol hi trobem les conclusions de la memòria, on es contrasten els resultats obtinguts amb els objectius del projecte i els problemes que han sorgit durant el desenvolupament d'aquest. En aquest últim capítol també s'exposen les impressions obtingudes a nivell personal i professional, així com les perspectives de futur.

1.3. Fases d'anàlisi i desenvolupament del projecte.

El present projecte ha estat estructurat en 8 etapes diferents. Aquestes etapes ens permetran tenir una guia durant el transcurs del projecte i planificar-ne la seva duració.

Les fases de desenvolupament seran les següents:

- 1ª etapa: Estudi de l'estàndard "*SIRCS"* de *Sony Corp*. i documentació de sistemes de comunicació per infraroig
- 2ª etapa: Estudi i disseny de les unitats funcionals *hardware* i de comunicació
- 3ª etapa: Verificació i test del disseny *hardware* de comunicació.
- 4ª etapa: Desenvolupament del disseny del *software*
- 5ª etapa: Validació i comprovació del correcte funcionament
- 6ª etapa: Disseny i validació de l'entorn *software* del sistema complet
- 7ª etapa: Elaboració de la memòria
- 8ª etapa: Entrega i defensa pública

1ª etapa: Estudi de l'estàndard "SIRCS" de Sony Corp. i documentació de sistemes de comunicació per infraroig

En la primera etapa es realitza la documentació referent a sistemes de control remot per infraroig, en especial l'estàndard SIRCS - SS-00116.

2ª etapa: Estudi i disseny de les unitats funcionals *hardware* i de comunicació

El disseny *HW* consta de diversos blocs. En primer lloc es desenvoluparà un Soft-IP amb VHDL basat en les especificacions funcionals de l'estàndard de control remot per infraroig, el segon bloc serà el encarregat de la comunicació entre el primer mòdul i el bus Wishbone. També s'implementarà una interfície entre el bus Avalon i el bus Wishbone per tal de poder accedir al microprocessador Nios II.

3ª etapa: Verificació i test del disseny *hardware* de comunicació.

En aquesta etapa es verificarà els blocs *hardware* a nivell funcional, aquesta etapa ha estat realitzada en paral·lel amb la de desenvolupament *hardware*, i s'ha anat verificant cadascún dels mòduls per separat fins aconseguir les respostes esperades.

4ª etapa: Desenvolupament del disseny del *software*

Per poder usar el sistema Nios II com a processador, és imprescindible que aquest indiqui al nostre sistema que és el què ha de fer. És en aquesta etapa on es desenvolupa el programa que ha d'executar el Nios II. També s'ha implementat una llibreria de funcions a executar pel Nios per tal de simplificar l'aplicació i facilitar-ne la reutilització.

5ª etapa: Validació i comprovació del correcte funcionament

En aquest punt s'executarà el codi *software* sobre el sistema dissenyat. A partir d'aquí s'anirà refinant el sistema fins al seu correcte funcionament, comprovant que es compleixen les característiques tècniques descrites en els objectius d'aquest projecte.

6ª etapa: Disseny i validació de l'entorn *software*

Un cop el sistema estigui validat, es dissenyarà un entorn gràfic còmode per a l'usuari per tal de poder realitzar proves de manera senzilla.

7ª etapa: Elaboració de la memòria

Quan els objectius del projecte estiguin assolits, es recopilarà tota la informació utilitzada per la realització d'aquest projecte i s'elaborarà la memòria per a la posterior defensa. La funció d'aquesta memòria és la de descriure cadascún dels passos seguits en el desenvolupament d'aquest projecte.

8ª etapa: Entrega i defensa

Un cop realitzades les etapes anteriors es procedirà a l'entrega de la memòria i del material de suport necessari per complementar-la i a la defensa pública.

Capítol 2.
Introducció i anàlisi

2.1. Marc del projecte

La producció SMT (*Surface Mount Technology*) és el mètode de construcció de dispositius electrònics més utilitzat actualment. Aquesta tecnologia de fabricació s'empra pel muntatge en PCBs tant de components passius com d'actius, i es basa en el seu posicionat en la superfície del circuit imprès (SMC, *Surface Mount Component*), per a ésser soldat posteriorment.

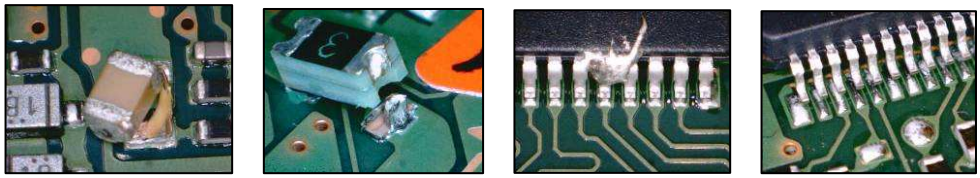
La dimensió dels components SMT és molt reduïda, i els terminals esdevenen molt més prims i curts en el cas de molts circuits integrats, mentre que per a elements passius (com resistències i condensadors) aquests terminals estan fixats directament sobre el cos de l'element. Aquests components estan dissenyats per a ser soldats automàticament (mitjançant robots o maquinària programada), ja que la seva soldadura sòl ser difícil degut a les seves dimensions.

Per a aplicacions en què l'espai, el pes, les interferències electromagnètiques o les prestacions són importants, aquesta tecnologia suposa una millora substancial en tots aquests aspectes, atès que redueixen les pèrdues i interferències que s'originen entre els terminals metàl·lics i la placa de circuit imprès on estan soldats.

A banda dels avantatges que aporta la tecnologia SMT, la reduïda dimensió dels components, i la complexitat en el procés de soldadura i muntatge, dona lloc al principal problema dels sistemes de producció de circuits impresos per inserció automàtica, la fiabilitat. El VITAM és un projecte que es desenvolupa actualment a Sony BCN Tec i té com a objectiu principal la verificació controlada de circuits impresos per a plataformes de televisió.

Són múltiples els factors que donen lloc a un error en un circuit imprès. Degut a la utilització de components SMT, les partícules de pols aporten un probabilitat d'error important, provocant la inclinació de components o connectors amb una mala soldadura (figura 2.1- a), d'altra banda també apareixen problemes originats per d'un excés o falta d'estany en les soldadures dels terminals, provocades per un mala regulació de la temperatura dels forns (figura 2.1- b), la

ruptura de components específics per una manipulació inadequada (figura 2.1 - c), problemes de comunicació, de funcionament, o de muntatge, originats per defectes dels propis dispositius, etc.



a) *b)* *c)* *d)*

Figura 2.1.- Exemples de defectes de producció SMT, a) excés de soldadura als pins d'un CI, b) manipulació indeguda, c) desviació de components degut a partícules de pols, d) IC defectuós

El VITAM es un aparell electromecànic, que executa un test a un circuit imprès per a plataformes de televisió. Aquest test comprova gran part dels problemes originats per el procés de muntatge i soldadura, i verifica la bon funcionament de gran part de les senyals afectades. La part *software* incorpora dispositius reprogramables i processadors soft-core, aquests sistemes HW/SW ens aporten una gran versatilitat en el disseny, proporcionant avantatges com la possibilitat d'actualitzacions amb un menor temps de redisseny, reduir el temps de verificació, flexibilitat, i en definitiva, una implementació del sistema de *software*, reconfigurable a les necessitats de l'usuari.

Figura 2.2.- VITAM

Fins l'actualitat, el test de control remot per infraroig s'ha realitzat manualment, comprovant la comunicació amb la placa, un cop havia estat donada com a bona, provocant així un factor de error. Aquest projecte planteja aquest problema i dona una solució HW/SW integrable al sistema de verificació amb dispositius FPGA ja presents.

2.2. Estàndard *SIRCS* (*Sony Infrarred Remote Control System)*

Els sistemes de control per infraroig cada vegada són més usats degut a l'augment del numero d'equips electrònics que pot adquirir l'usuari. Donades aquestes circumstàncies, i el gran nombre d'aparells que poden conviure en un mateix espai, se n'ha de garantir el correcte funcionament.

L'estàndard *SIRCS* de *Sony* especifica els formats dels codis i les trames per aparells *Sony* i estableix els criteris de fiabilitat per al control de múltiples aparells electrònics, evitant[2]:

- Interferències a equips electrònics en general.
- Interferències a equips electrònics de *Sony.*
- Interferències provocades per equips *Sony*.
- Interferències provocades per equips electrònics en general.

2.2.1. Formats dels codis

Sony utilitza una modulació per amplada de pols. Considerant un temps mínim T de 600μs, cada bit d'informació està codificat mitjançant un nivell baix y un alt: un "0" esta codificat amb un nivell baix y un alt de duració T respectivament. En canvi un "1" esta codificat amb un nivell baix de duració T i un de nivell alt de duració 2T. Cada trama ha d'incorporar inicialment una capçalera de període 3T per indicar l'inici d'una nova trama, seguit de la codificació dels bits començant per l'LSB. La figura 2.3 mostra una trama de 12 bits on s'especifica les amplades dels bits de dades.

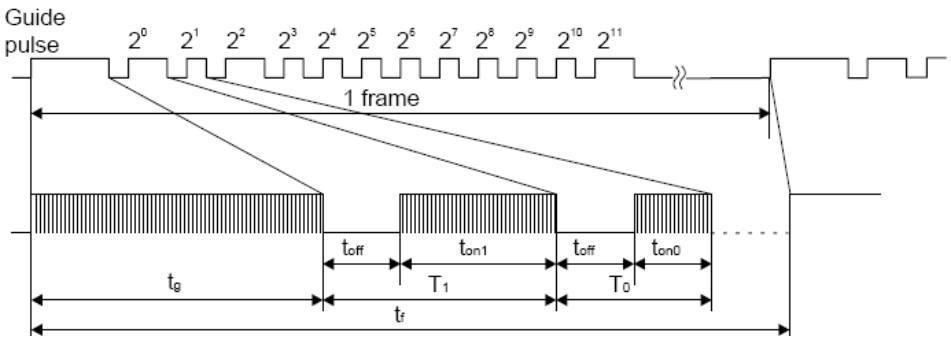


Figura 2.3.- Capçalera i format dels bits de les trames. (*format de 12-bits*).

		Símbol	Temps	Tolerància
Durada de la capçalera		t_g	2.4 ms	±0.015 ms
Bit de dades en nivell baix		t_{off}	0.6 ms	±0.015 ms
Bit de dades en nivell alt	"0"	t_{on1}	1.2 ms	±0.015 ms
	"1"	t_{on2}	0.6 ms	±0.015 ms
Període del bit de dades	"0"	T_1	1.8 ms	±0.03 ms
	"1"	T_0	1.2 ms	±0.03 ms
Període de la Trama		t_f	45 ms	±1.2 ms
Freqüència			40 KHz	

Taula 2.1.- Formats dels bits i períodes.

2.2.2. Formats de les trames

Els formats de les trames emprats per a la comunicació es classifiquen en configuracions de 12, 15 i 20 bits (figura 2.4). Cadascuna d'aquestes configuracions està composada per un codi de dades de 7 bits i un codi anomenat de *Category,* que dona l'adreça del dispositiu amb el qual volem comunicar-nos remotament. Per exemple, la comunicació amb plataformes de TV empra un *Category Code* de 5 bits de valor 0x10 (1000). En canvi la comunicació amb un reproductor de DVD té un *Category Code de* 13 bits de valor 0xB92 (0101110010010).

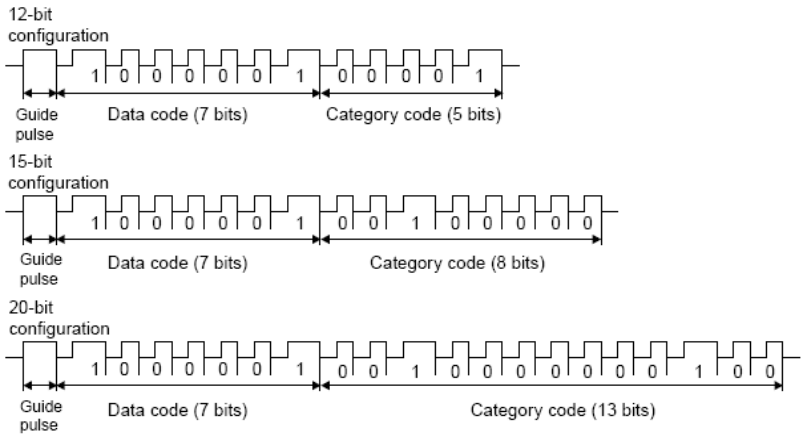


Figura 2.4.- Cada trama consta d'un pols guia per indicar el començament de la trama acompanyat de 12, 15 o 20 bits depenent de la configuració.

2.2.3. Transmissió i recepció del senyal

La transmissió per infraroig es realitza habitualment amb una modulació binaria per desplaçament de fase (*BPSK*). Aquesta tècnica, com la gran part de les modulacions, permet aprofitar el canal de comunicació de manera òptima, brindant la possibilitat de transmetre més informació de forma simultània, protegint-la de possibles interferències o sorolls.

Modulació BPSK: és una forma de modulació angular *PSK* consistent en variar la fase de la portadora, d'acord amb les variacions del senyal modulador, que en aquest cas és un senyal digital. La denominació "*B"* de *BPSK* fa referència a que en aquesta modulació, la portadora pot prendre dos fases de sortida possibles, per una sola freqüència. Una fase de sortida representa un 1 lògic i l'altre un 0 lògic. A la figura 2.5 els mostra el diagrama de blocs estàndard d'un modulador BPSK.[14]

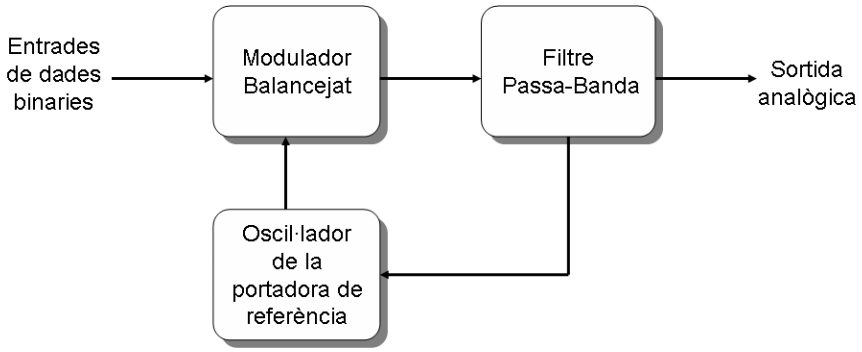


Figura 2.5.- Diagrama de blocs d'un modulador BPSK.

Si l'oscil·lador de referència és de senyal quadrada unipolar, el senyal modulat quedaria d'aquest estil. Aquest tipus de senyal és la que voldrem transmetre a través d'un *LED* infraroig.

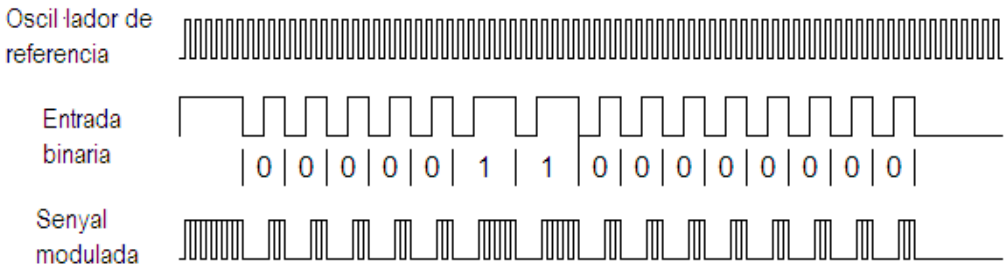


Figura 2.6.- Senyals d'entrada, sortida i oscil·lador BPSK.

L'estàndard SIRCS proposa dos tipus de transmissió per les trames: continua, on la transmissió és constant, mentre les dades són s'hi afegeixen de forma contínua mentre dura la pulsació de la

tecla del comandament a distancia, i per ràfegues on es transmet 3 trames, nombre apropiat per assegurar la recepció sense perill de pertorbacions externes

Per a la detecció de la informació, n'hi hauria prou utilitzant un filtre passa-banda a la sortida del detector. Avui en dia existeixen gran nombre de detectors integrats que incorporen el diode detector de infrarojos, i la circuiteria necessària per obtenir a la seva sortida el senyal desmodulat.

2.3. Components Virtuals *(Intellectual Property Cores)*

Tot i els grans beneficis que ens proporciona el disseny basat en HDL's, encara no pot fer front a la complexitat dels circuit integrats disponibles i la productivitat en el disseny, que augmenta any rere any, segons la llei de Moore. Una de les idees per millorar l'eficiència i la productivitat en el disseny és la reutilització. A l'hora de realitzar un projecte, un enginyer no cal que parteixi de zero i codifiqui totes les parts del sistema, és a dir, pot usar blocs ja creats per altres persones en un altre moment, adaptant el codi a les seves necessitats. Això té una traducció ràpida en qüestió de temps i de costos, ja que existeixen blocs molt complexes i d'ús molt habitual, des dels blocs més bàsics fins els més especialitzats, per tant l'estalvi és considerable.

Les IP (Intellectual Property) Cores. basen el seu flux de disseny en la reutilització i es defineix com el nucli funcional de un xip o sistema que realitza una o varies funcions. Les IP Cores són blocs de codi HDL que descriuen una funció específica, i s'utilitzen per implementar dissenys més grans a nivell de Hw i Soft.

Encara que la portabilitat dels Cores, ens permeti implementar funcions amb un gran estalvi de temps i diners, també existeixen inconvenients. El principal desavantatge de la utilització de IP Cores és que, com que són blocs creats per persones concretes, la gran part són de pagament. Per tal de pal·liar aquest problema la comunitat *OpenCores.org* desenvolupa *IP Cores* d'us i distribució lliure, on podem trobar gran nombre de components de codi obert, disponibles, usables i reutilitzables lliurement, d'aquesta manera s'ajuda a projectes on no es poden adquirir IP comercials per qüestions de finançament tals com, la universitat o empreses petites[17]. Un altre inconvenient és que els dissenys s'han d'elaborar tenint en compte els requeriments i la interfície de cadascuna de les IP Core utilitzades, o readaptar els blocs o mòduls segons les necessitats de cada usuari.

Hard, Firm I Soft IP CORES

Una IP Core pot ser entregada de diferents maneres.

- ***Hard IP Core:*** La opció més rígida. El Core ha estat verificat i implementat en una o més tecnologies específiques. Les especificacions temporals i l'àrea ocupada estan exactament definides per a les tecnologies en les quals s'ha verificat. No es poden fer modificacions ja que no es té accés al codi font
- ***Soft IP Core:*** Es tracta de l'altre extrem. Es pot tenir accés al codi font. Típicament s'han d'especificar-les condicions d'operació com per exemple les freqüències de rellotge, els temps de *set-up* i de *hold*, els requeriments aproximats d'àrea i el tipus de tecnologia on s'ha d'implementar (FPGA o ASIC).
- ***Firm IP Core:*** Es tracta d'un terme entremig. Es disposa d'una *netlist* i es poden fer algunes modificacions en la configuració. Generalment es genera una caixa negra encriptada a partir d'un fitxer de configuració.

2.4. Busos-on-chip

Un bus dona un *link* de comunicació en el que tots els elements de processat poden accedir. Les arquitectures per a comunicacions *on-chip* tenen una gran influència en la velocitat i l'àrea dels dissenys *System-on-Chip* (SoC). Un element clau en els dissenys SoC es la interconnexió entre mòduls.

Els Busos SoC són busos integrats en una ASIC o una FPGA, i estan destinats a la comunicació entre una unitat funcional i una altre (per exemple IP Cores), o bé, interconnectar-la amb alguna part de la lògica que rodeja el SoC, és a dir, defineix les entrades, les sortides i el mètode d'interconnexió cap al mòdul funcional.

Els busos on chip segueixen el mateix concepte de reutilització, efectes com: disposar de memòries, de llibreries de processadors, de controladors DMA, amb busos compatibles permet desenvolupar els nostres disseny amb a un cost molt reduït.

Les característiques principals dels busos *on-Chip* són pràcticament iguals que les de qualsevol bus convencional, excepte pel fet que estan dissenyats i orientats a operar majorment dintre un sol xip, de manera que usar busos amb una amplada superior, és simple, ja que no hi ha limitacions en els pins de distribució.

Les principals característiques dels busos-pn-Chip són:

- Línies d'adreces configurables
- Línies de dades configurables
- Senyals de bidireccionament, encara que el més usual és usar línies de dades diferents per operacions de lectura i d'escriptura
- Línies de control
- Senyal de rellotge
- Un protocol específic

En els apartats següents es dona una visió detallada dels diferents *busos-on-Chip* usats per a la implementació d'aquest projecte.

2.5.1. El bus WISHBONE

L'arquitectura d'interconnexió SoC per IP Cores WISHBONE basa la seva metodologia en un disseny flexible. El seu objectiu és fomentar la reutilització de dissenys disminuint els problemes de la integració SoC. Això ha estat un gran avenç per a la creació de una interfície comú entre els diferents IP-cores. Amb aquesta interfície s'ha contribuït a millorar la portabilitat i la fiabilitat dels sistemes i minimitzat el temps de realització dels projectes.

La interconnexió WISHBONE defineix dos tipus d'interfícies, *Master* i *Slave*. Les *Master* són per a cores capaços de generar i rebre transicions en el bus, en canvi les *Slave* són per cores capaços només de rebre'n, de manera que sempre hi ha un *Master* que escriu (o llegeix) dades a un (o des d'un) *Slave*.

Aquesta interfície és entesa com a propòsit general, definint un estàndard per l'intercanvi de dades entre els diferents IP Cores, encara que això, no limita ni regula les funcions específiques de cada unitat funcional. La implementació del bus es realitza de manera senzilla amb eines CAD, i es pot implementar amb VHDL o Verilog-HDL.

2.5.1.1. Característiques principals

La principal característica que ens aporta el bus WB és que disposa d'una sola arquitectura per a totes les aplicacions, simple i compacta, permetent que el disseny de qualsevol implementació sigui el més senzill possible. Aquest tipus de interconnexió pretén reforçar la compatibilitat entre IP Cores i, a pesar de la senzillesa, permet crear estructures *Master/Slave* amb la presència de múltiples *Masters* i *Slaves* amb l'ajuda d'un sistema eficient d'arbitratge.

Els aspectes més rellevants del bus WISHBONE[13]:

- Espai d'adreces de 64 bit
- Bus de dades extensible (8-64 bits)
- Cicles de lectura i escriptura simple o en ràfega.
- Cicles RMW (*Read Modify Write*)
- Suport de reintent
- Diverses topologies d'interconnexió.
- Protocol *Handshaking*[1] per la transferència de dades.
- Suporta interfície mapada en memòria.

2.5.1.2. Els senyals del WISHBONE

Els senyals que disposa el bus WISHBONE es poden classificar en tres grups, els que són comuns tant en les interfícies *Master* com en les *Slave*, els que són generats pels *Master's* i rebuts pels *Slave's*, i finalment les que són sortides de les interfícies *Slave's* i entrades de les *Masters.* A continuació es mostren les taules amb les senyals de cada grup i les seves descripcions:

Senyal	Descripció
CLK_I	El rellotge d'entrada és el responsable de coordinar totes les activitats de la lògica interna del WISHBONE.
DAT_I ()	Línies de dades d'entrada. Aquest nombre és configurable, amb un màxim de 64 línies.
DAT_O()	Línies de dades de sortida. Aquest nombre és configurable, amb un màxim de 64 línies.
RST_I()	Entrada del *reset*. Reinicia la interfície i força l'estat inicial a totes les FSM.
TGD_I()	El senyal *Data tag type* conté informació associada amb les línies *DAT_I*, i està limitat per el senyal *STB_I.*
TGD_O()	El senyal *Data tag type* conté informació associada amb les línies *DAT_O*, i està limitat per el senyal *STB_O.*

Taula 2.2.- Senyals comuns per a les interfícies *Master* i *Slave*

[1] *Handshaking:* protocol de transferència de dades en el qual, el Master, amb un *strobe,* indica la estabilitat del bus, l'Slave al veure la senyal, activa una senyal ACK amb la que indica que ja ha registrat la dada al bus, i en conseqüència es desactiva l'*strobe* i seguidament l'ACK.

Senyal	Descripció
CYC	Senyal de sortida del cicle de bus generada per el *Master*. S'utilitza per indicar que comença un cicle de lectura o escriptura.
ADR	Conjunt de línies de sortida amb l'adreça en format binari. El *Master* indica l'adreça on vol accedir. La direcció pot ser de 8, 16, 32 i 64 bits.
LOCK	Senyal de sortida de bloqueig. Quan està activada indica que el cicle actual no es pot interrompre.
SEL	El selector indica quines línies de dades d'entrada *DAT_I()* són vàlides durant els cicles de lectura, i quines de *DAT_O()* en els cicles d'escriptura
STB	El senyal de sortida *strobe* indica un cicle vàlid del bus.
WE	Indica si es genera un cicle de lectura o un d'escriptura (quan està actiu).

Taula 2.3.- Senyals de sortida per a les interfícies *Master* (entrades per les *Slave*).

Senyal	Descripció
ACK	Senyal de reconeixement que genera l'*slave* indicant que el cicle del bus ha acabat de manera correcta.
ERR	Senyal d'entrada d'error. Indica una terminació anormal del cicle.
RTY	Indica que la interfície no està llesta en aquest moment per enviar o rebre dades i que s'ha de reintentar el cicle

Taula 2.4.- Senyals d'entrada per a les interfícies *Master* (sortides per les *Slave*).

2.5.2. El bus Avalon©

Avalon és un bus simple dissenyat per Altera per connectar processadors *on-chip* i perifèrics dins un sistema programable on-Chip (SPOC). Aquest bus permet la transferència de dades, entre un dispositiu *Master* i un altre *Slave*, de 8, 16, o 32 bits i la possibilitat d'una arquitectura *multi-master* ,inserint de manera automàtica els mecanismes d'arbitratge corresponents.

Masters i *Slaves* interactuen entre ells usant una tècnica d'arbitratge de bus anomenada *slave-side arbitration*, on es determina quin *Master* té accés a l'*Slave,* en el cas que diversos *masters* vulguin accedir al mateix *Slave* al mateix temps. Aquesta tècnica aporta grans beneficis ja que, poden accedir al bus diversos *Masters* a la vegada com si fos l'únic present, sempre que no intentin accedir al mateix S*lave*.

2.5.2.1. Característiques Principals

L'arquitectura del bus Avalon consisteix en recursos lògics i d'interconnexió del dispositiu PLD. Aquesta arquitectura té una sèrie de propietats i convenis establerts per tal de poder garantir la generació de sistemes, busos i perifèrics.

Els aspectes més rellevants del bus Avalon[1]:

- Permet adreçar fins a 4 GHz de memòria (32 bits).
- Interfície síncrona: tots els senyals de l'Avalon estan sincronitzats amb el senyal de rellotge del bus. Això simplifica molt el bus i facilita la integració de perifèrics d'alta velocitat.
- Línies de dades, control i adreces separades: Els camins de dades i adreces separats proporcionen una interfície molt senzilla per l'usuari, ja que els perifèrics no han de descodificar cicles d'adreçament i de dades.
- *Built-in adress decoding*: El bus Avalon genera automàticament el senyal *chip-select* per a cada dispositiu, simplificant de manera considerable el disseny de perifèrics.
- Arquitectura de múltiple bus *Master*: Diversos dispositius *Master* poden connectar-se al bus simultàniament, generant la lògica necessària per arbitrar-los.
- *Dynamic bus sizing*: El bus Avalon gestiona de forma automàtica les transferències antre perifèrics amb amplades de dades diferents.
- Configuració en mode gràfic: Utilitats gràfiques de ús fàcil a nivell d'usuari.

2.5.2.2. Senyals del bus Avalon

Els senyals que disposa el bus Avalon no es poden classificar com els del bus WISHBONE, ja que hi ha senyals en les interfícies *Master* que no són presents en les interfícies *Slave,* i viceversa, ja que són senyals que proporcionen informació al bus i és aquest l'encarregat de gestionar cada activitat. A continuació es mostren les taules amb les senyals d'entrada i sortida de la interfície Avalon Slave i les seves descripcions, més endavant, es mostraran les corresponents a la interfície M*aster*

Senyal	Descripció
Clk	Senyal global de rellotge amb transaccions síncrones, i asíncrona per transaccions amb perifèrics.
reset	Senyal global de reset. Opcional.
chipselect	Indica quan els senyals del bus van destinats al perifèric, present a tots els *slaves*
address	Línies d'adreces provinents del mòdul del bus Avalon. (1-32 línies).
read	Indica que la petició del *Master* és de lectura. La implementació d'aquesta comporta implementar el senyal *readdata*.
write	Indica que la petició del *Master* és d'escriptura. La implementació d'aquesta comporta implementar el senyal *writedata*.

byteenable	Permet la possibilitat d'activació de part concretes de les línies de dades en transferències majors de 8 bits. (0,2 o 4 línies).
begintransfer	S'activa durant el primer cicle de cada transferència al bus.
writedata	Fins a 32 línies de dades provinents del mòdul del bus per transferència d'escriptura. La implementació d'aquesta comporta implementar el senyal *write*.

Taula 2.5.- Senyals d'entrada de les interfícies Avalon Slave.

Senyal	Descripció
readdata	Fins a 32 línies de dades provinents del mòdul del bus per transferència de lectura. La implementació d'aquesta comporta implementar el senyal *read*.
readdatavalid	Usat només per *slaves* amb latència variable. Indica quan el perifèric pot posar dades valides a les línies *readdata*
waitrequest	S'usa quan les demandes del bus Avalon no poden ser processades immediatament.
Irq	Petició d'interrupció. El dispositiu activa el senyal quan necessita ser servit per un *Master*.
resetrequest	Senyal de reset, permeten que un *Slave* resetegi el sistema sencer.

Taula 2.6.- Principals senyals de sortida de les interfícies Avalon Slave.

Els senyals *readdatavalid, readdata, waitrequest* i *irq* generats per la interfície Avalon Slave són entrades del mòdul Avalon Master, així com les entrades *address, read, write, byteenable, writedata* de l'*Slave* són les sortides de la interfície Avalon Master.

A continuació es dona la descripció de les senyal de la interfície Avalon *Master* que no es troben presents a les interfícies *salve.*

Senyal	Descripció
Clk	Senyal global de rellotge per el mòdul del sistema i el mòdul del bus Avalon. En aquest cas totes les transaccions són síncrones.
reset	Senyal global de reset. Aquest és específic per a cada perifèric
irqnumber	Fins a 6 línies per a establir les prioritats dels *slaves* que estan generant la petició d'interrupció. Aquest és un senyal exclusiu de la interfície *Master*

Taula 2.7.- Senyals específics d'entrada de les interfícies Avalon Master.

Senyal	Descripció
byteenable	La seva activació permet habilitació especifica de pistes de 8 línies. La implementació és especifica del perifèric.
flush	Senyal per a transferències de lectura amb latència. El *Master* pot cancel·lar qualsevol lectura pendent activant el senyal *flush*

Taula 2.8.- Senyals específics de sortida de les interfícies Avalon Slave.

2.5. El NIOS® II

El Nios II és un core d'un processador de propòsit general creat per Altera. Un sistema amb un processador Nios II que inclou una CPU i una combinació de memòries i perifèrics tots comunicats a traves del bus Avalon. L'avantatge dels sistema Nios II respecte les altres estructures, resideix en la facilitat de reconfigurabilitat del sistema, i com a conseqüència, la capacitat d'adaptar el sistema a les especificacions i restriccions de costos del nostre disseny.

2.5.1. Característiques del Nios II

El Nios II és un microprocessador Soft-Core de propòsit general RISC de 32 bits. Les característiques generals són les següents:

- Conjunt d'instruccions, camí de dades i espai d'adreces de 32 bits.
- 32 registres de propòsit general.
- 32 possibles fonts d'interrupció.
- Instruccions simples de multiplicació i divisió de 32 x 32 bits donant un resultat de 32 bits.
- Instruccions dedicades a calcular productes de 64 i 128 bits.
- Instruccions de *barrel shifter*[2]
- Accés a variables de perifèrics on-chip, interfícies de memòria i perifèrics externs.
- Mòdul de depuració assistit per *hardware* que permet parar i arranca, a voluntat a través d'un entorn de desenvolupament integrat (IDE) de control.
- Entorn de desenvolupament *Software* basat en GNU C/C++ i Eclipse (IDE).
- Conjunt d'instruccions compatible amb tots els sistemes basat en processadors Nios II.

El conjunt de propietats descrites formen els paràmetres configurables, és a dir, podem prescindir o utilitzar les característiques del processador més adients, per tal d'adquirir l'entorn més apropiat per els objectius del nostre disseny. El core del Nios II, dona la possibilitat de

[2] *Barrel shifter*: dispositiu capaç de rotar o desplaçar una paraula amb un nombre qualsevol de bits, en una sola

configurar la implementació *hardware* segons diversos objectius, econòmica, estàndard o ràpida, on totes les configuracions són compatibles amb el codi a executar.

Les característiques principals són[2]:

- *Econòmica:* És la configuració més simple, sense *cache* ni pipelines de descodificació d'instruccions. Aquesta versió és la que té una menor ocupació de la FPGA, entre 600-700 elements lògics
- *Estàndard:* Aquesta configuració inclou *cache* de dades, predicció de salts estàtica, 5 nivells de *pipeline* de descodificació d'instruccions i implementació en *hardware* de les operacions de suma i divisió. L'ocupació de la FPGA està entorn de 1200-1400 elements lògics.
- *Ràpida:* És la versió més completa, ja que inclou totes les prestacions de la configuració estàndard, més la implementació *hardware* de *barrel shifter*, i a diferència de l'estàndard inclou un nivell més de *pipeline* de descodificació d'instruccions. La seva ocupació està entorn de 1400 i 1800 elements lògics.

2.6.1. Arquitectura

L'arquitectura del Nios II descriu un conjunt d'instruccions, les quals són implementades per unitats funcionals. El core del Nios II (figura 2.7) no inclou perifèrics ni la lògica de connexió amb l'exterior tan sols defineix les unitats funcionals i la circuiteria necessària per implementar l'arquitectura[3].

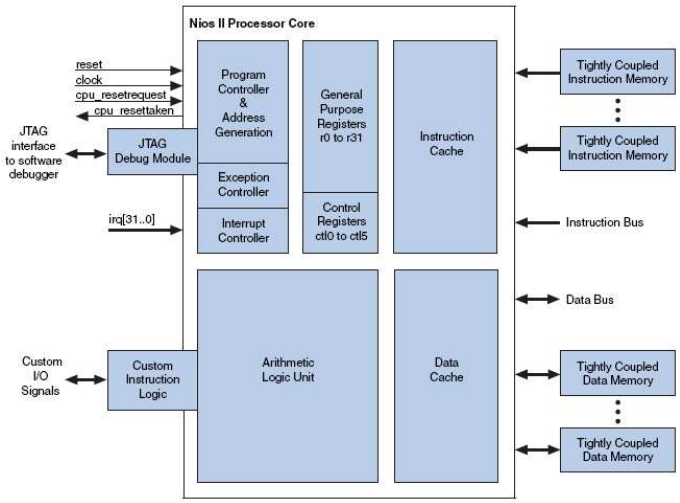


Figura 2.7.- Diagrama de blocs del core microprocessador Nios II *(configuració fast).*

Entre els blocs de la figura anterior podem localitzar la unitat lògica aritmètica, la interfície per a la lògica de les instruccions pròpies, controladors d'excepcions i d'interrupcions, un bus d'instruccions i un de dades, memòries *cache* per a instruccions i per dades, un mòdul de depuració JTAG i el *Register file*[3]. La presencia d'aquests elements en l'arquitectura Nios II bé determinada pel tipus de configuració. La taula 2.9 mostra les característiques de l'arquitectura per cada confinguració.

	Nios II /f Fast	Nios II /s Standard	Nios II /e Economy
Pipeline	6 Stage	5 Stage	None
Multiplier & Barrel Shifter *	1 Cycle	3 Cycle	None
Branch Prediction	Dynamic	Static	None
Instruction Cache	Configurable	Configurable	None
Data Cache	Configurable	None	None
Custom Instructions	Up to 256 Common Interface		

Taula 2.9.- Característiques de les configuracions del processador Nios II.

El sistema Nios II, incorpora una unitat funcional anomenada *custom instructions*. Aquesta unitat està adherida al *hardware* de l'ALU i s'hi accedeix a través d'instruccions màquina especifiques. Les *custom instructions* són mòduls *hardware* que poden efectuar operacions multicicle i són usades en les parts crítiques dels algoritmes *software*, reduint el nombre de cicles utilitzats per executar una part de codi. Podem afegir fins a un total de 256 *custom instructions* o fins a consumir els recursos disponibles a la FPGA. La figura 2.8 representa l'ús i la lògica d'una *custom instruction* extesa a l'ALU d'un Nios II.[4]

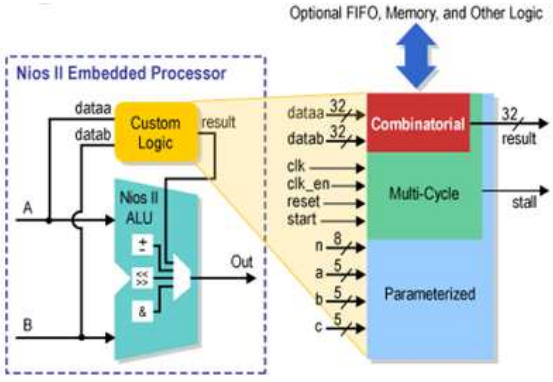


Figura 2.8.- Extensió i lògica d'una *custom instructions* dins de l'ALU.

[3] Register File: És una col·lecció de registres del processador en una entitat central de procés. El conjunt d'instruccions d'una UCP defineix el conjunt de registres usats per guardar dades entre la memòria i les unitats funcionals del processador

Les *custom intruction* consisteix en dos elements essencials: el bloc lògic *custom,* que és el *hardware* que realitza la operació, on l'usuari pot definir-ne fins a 5, i les macros *software*, les quals permeten al dissenyador accedir als blocs lògics a través del codi *software*. Aquests mòduls també és nomenen dispositius.

La utilització d'aquesta unitat difumina la diferència entre RISC i CISC, ja que les custom intructions augmentem la complexitat dels algoritmes interns del microprocessador RISC amb instruccions estàndard de un sol cicle de rellotge millorant el rendiment i la velocitat de la nostra configuració.

En el sistema Nios II podem classificar una sèrie de dispositius perifèrics estàndard, que són presents en el core del processador. Dins l'apartat de memòries podem trobar perifèrics com

- memòries RAM/ROM on-chip, les quals podem establir el tipus i la mida que seran implementades
- controlador per la memòria flash (CFI)
- controlador SDRAM, el qual pot accedir a sistemes SDRAM amb un ample de banda de (8, 16, 32 o 64 bits), amb diferents mides de memòria i múltiples chip-selects.

Els perifèrics de comunicació que podem trobar són

- JTAG UART: Permet establir un sistema de comunicació, usant la circuiteria específica de les FPGA's d'Altera, El host PC pot connectar-se a la FPGA a través dels pins JTAG.
- Dispositius de comunicació sèrie UART: permet la configuració dels paràmetres habituals: bits start i stop, velocitat de transferència, paritat, etc.
- Interfície Ethernet: suporta el protocol ARP, IP, ICMP, UDP i TCP, a través d'una interfície amb el *hardware* i llibreries *software*.
- DMA: Podem escollir la mida de transferència, el tipus de transaccions permeses i activar la transferència per ràfegues.

- SPI: Dispositiu Serial Peripheral Interface que pot comportar-se com a *Master* o *slave*. El bus SPI té tres connexions que permeten una comunicació amb múltiples dispositius. Pot ser gestionat per DMA.

També podem trobar perifèrics tan habituals com Timers i ports paral·lels d'entrada/sortida PIO, aquests últims poden configurar-se en 3 modes: sortida, entrada i bidireccional, amb la opció de definir en nombre de bits per port i la habilitació en cas d'interrupció. .

Capítol 3.

Metodologia de Treball i entorn de desenvolupament

3.1. Metodologia de Treball

En aquest apartat es mostra detalladament el flux de treball seguit per tal d'assolir els objectius marcats per aquest projecte. A grans trets, el disseny d'aquest sistema es basa en la implementació, d'un mòdul de comunicació en VHDL que ens permeti generar les trames necessàries per al control remot del TV, un programa en C a executar per el Nios II, i un entorn gràfic on l'usuari pugui realitzar proves de test de manera senzilla.

3.1.1. Flux de treball HW/SW

El repartiment *Hardware* - *Software* en aquest projecte esta clarament diferenciat, s'implementarà el bloc de comunicació en *Hardware* i es realitzarà la configuració d'aquest per *Software.* Finalment, també s'inclourà una llibreria de funcions a executar pel Nios II i una inter-fase gràfica amb l'usuari.

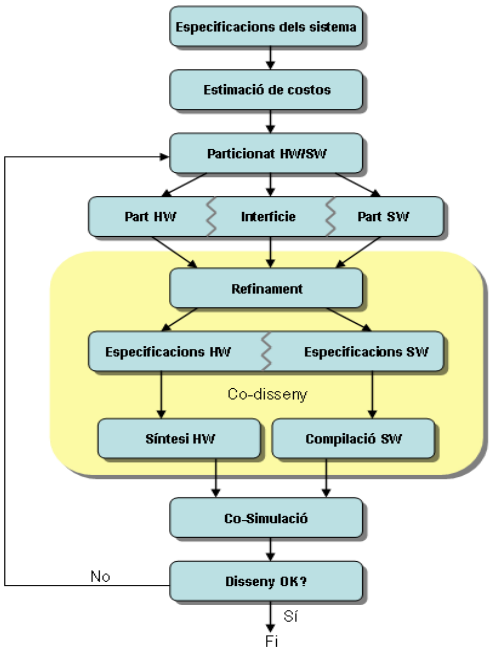


Figura 3.1.- Flux de disseny HW/SW.[9]

Per a comprovar el correcte funcionament del sistema *hardware* i el programa a executar per el Nios II, s'usaven simulacions, i un cop el sistema es va comprovar que funcionava sota simulació funcional, es va baixar el disseny a la FPGA on es va realitzar les captures dels senyals amb l'ajuda del S*ignalTap.* El codisseny *hardware-software* es va realitzar per etapes, a mesura que s'anaven implementant noves funcionalitats al bloc *hardware*, es desenvolupava nou codi a executar per el Nios II, d'aquesta manera es pot comprovar la funcionalitat tant del *Hardware* com de *Software* .

3.1.2. Flux de treball HW

Concretament, l'entorn *Hardware* d'aquest projecte, al formar part d'un sistema de test més ampli, ja tenia unes especificacions establertes que cal respectar. Aquestes especificacions han marcat el flux i el desenvolupament del *Hardware* durant tota la implementació.

Al flux de treball de la figura 3.2 podem veure tres fluxos paral·lels, primerament es va implementar un bloc en VHDL capaç de generar les trames especifiques de l'estàndard "*SIRCS*" i es va verificar nivell funcional. El següent bloc a implementar va ser la interfície entre el primer bloc i el bus WishBone, (la utilització d'aquest ve fixada per les especificacions del sistema), en aquesta etapa també es va realitzar una simulació funcional, per tal de veure que els estímuls d'entrada i sortida eren els correctes i finalment es va crear una interfície per adaptar les senyals del bus Wishbone amb les de l'Avalon, i poder-nos comunicar amb el Nios II.

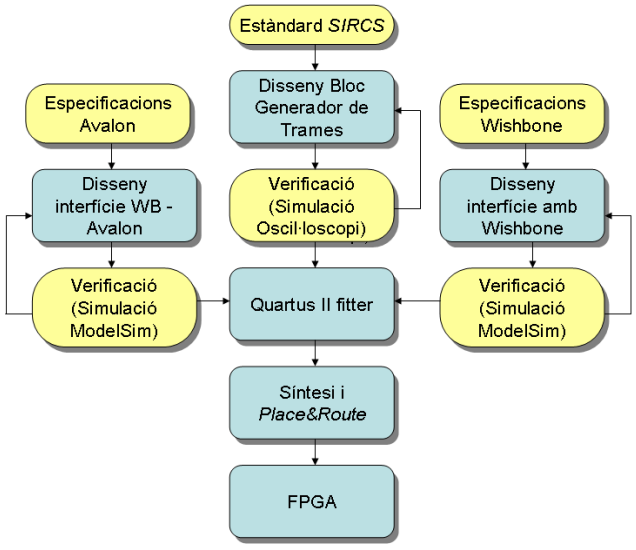


Figura 3.2.- Flux de treball *Hardware.*

Aquest flux de disseny *Hardware* parteix dels fitxers de disseny HDL, es sintetitzen i es fa una simulació funcional, si la simulació funcional és correcte, es realitza el *place and route* amb l'ajuda del Quartus II fitter i si es compleixen els requeriments especificats i el disseny usa un nombre de recursos inferior als de la FPGA, es generen els fitxers de configuració de la FPGA i es realitza una verificació directament en placa. Paral·lelament a aquest disseny hi ha el flux de treball *Software.*

3.1.3. Flux de treball SW

A la figura 3.3 es mostra el flux de desenvolupament software. Aquest flux parteix de les especificacions funcionals referents al tipus de configuració del Nios II i els fitxers HDL sintetitzats. Amb aquests paràmetres i amb l'ajuda de les llibreries de funcions, es genera el codi per a configurar el sistema i el programa d'aplicació especifica que ha d'executar el Nios II amb el compliador *Nios II IDE*. Un cop el sistema està compilat i depurat, es generen els fitxers de configuració i es procedeix a la descarrega del *software* a la FPGA.

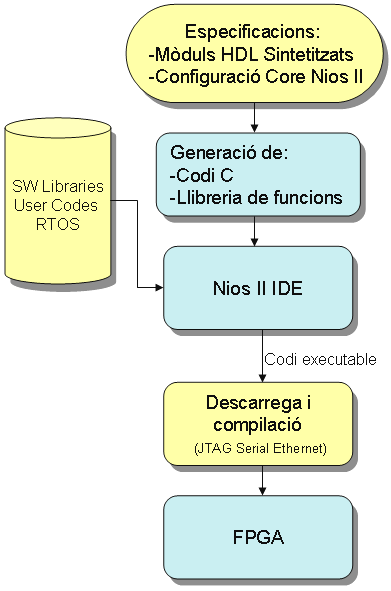


Figura 3.3.- Flux de treball *software*

3.2. Entorns de disseny

En aquest apartat es mostren les eines i els entorns de desenvolupament utilitzats tan per el disseny *hardware* com per el *software.* El disseny *software* també especifica el *Software* d'aplicació per a realitzar l'interfase gràfica amb l'usuari.

3.2.1. Entorn de disseny HW

A l'entorn de disseny *hardware* s'ha utilitzat el CAD de desenvolupament de la companyia Altera, ja que la implementació del disseny HW/SW es realitza sobre la família de FPGAs Cyclone[4], i s'utilitza el Nios II com a processador.

3.2.1.1. Quartus II

El *software* de desenvolupament Quartus II d'Altera permet l'anàlisi i la síntesi de dissenys HDL (*Hardware descripction language*). Quartus II proporciona un entorn familiar i estable per als dissenys SOPC (*System on-a-programable-chip*) permeten desenvolupar la totalitat dels fluxos de disseny *hardware*. Aquest entorn ens dona la opció de crear els components en llenguatge de descripció de *hardware* o través de les llibreries de disseny que porta incorporades. La interconnexió dels sistemes complets es realitza des de un entorn gràfic. Aquest entorn també proporciona les eines necessàries per sintetitzar i fer el *Place&Route* del sistema dissenyat[5].

3.2.1.2. SOPC Builder

El SOPC Builder és un wizard que s'executa sobre Quartus II, i permet crear SoC basats en processadors, perifèrics i memòries[6].

SOPC Builder dona a l'usuari, la possibilitat de configurar el nombre de perifèrics o dispositius que necessiti incorporar al seu sistema , així com el mapeig d'aquestos a memòria i el número de interrupcions que generaran.

3.2.2. Entorn de disseny SW

Com en l'entorn de disseny *hardware*, el CAD de desenvolupament per el *software* a executar pel Nios II, és l'específic de la casa Altera (Nios II IDE). També s'ha creat un entorn gràfic per facilitar les proves de test amb l'usuari.

[4] *Cyclone* : familia de FPGAs de baix cost de la companyia Altera. Existeixen fins a 3 *catergories Cyclone I, II i III.*

3.2.2.1. Nios II IDE

El Nios II IDE (*Integrated Development Enviroment*) és l'entorn de desenvolupament *software* per a plataformes Nios II d'Altera. Nios II IDE està basat en el compilador GNU C/C++ i l'entorn de desenvolupament Eclipse, i inclou l'editor de codi, l'administrador de projectes i eines de compilació i debug.

Aquest entorn treballa conjuntament amb el SPOC *Builder*, en relació amb les simulacions. Quan es compila un *SoC* amb el *SPOC Builder*, existeix una opció per tal que es generi un projecte per al ModelSim per tal de poder simular-lo.

A part de les eines de desenvolupament de *software*, el Nios II IDE ens proporciona el ISS (*Instrucción Set Simulator*), d'aquesta manera podem simular l'execució del *software* sobre un Nios II sense la necessitat de tenir-lo implementat en *hardware*. La figura de continuació mostra l'aspecte de l'entorn de desenvolupament Nios II IDE.

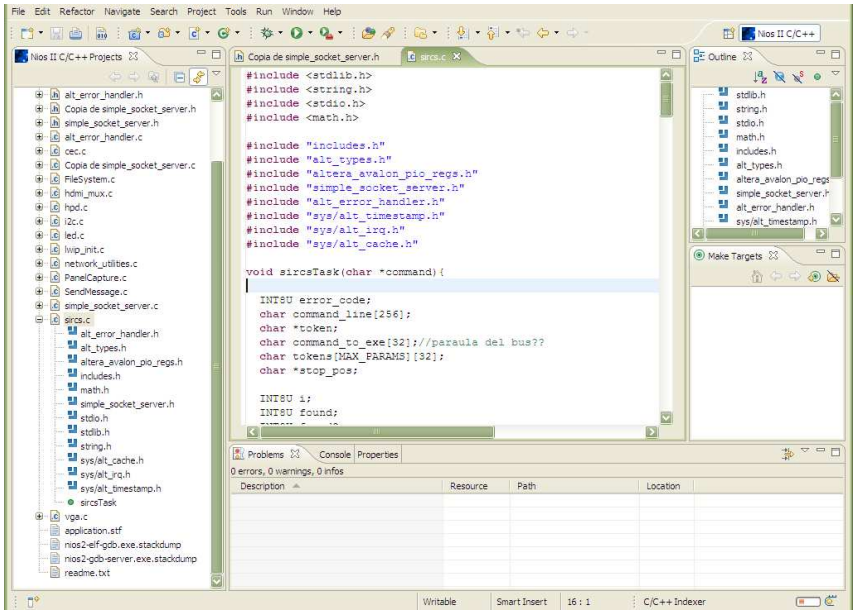


Figura 3.4.- Vista de desenvolupament del *Nios II IDE*

3.2.2.2. LabWindows/CVI

Entorn de desenvolupament proporcionat per National Instruments, amb el qual podem depurar i compilar aplicacions *software* en C/C++. Aquest entorn combina la funcionalitat específica dissenyada pel control d'instrumentació, l'adquisició de dades i l'anàlisi amb una interfície dedicada al control de l'usuari. Un exemple equivalent seria el Visual Studio de Microsoft.

3.3. Entorn de prototipat

L'entorn de prototipat es composa de dues plataformes de desenvolupament, un kit de desenvolupament fabricat per Altera i una plataforma de prototipatge empresarial desenvolupada per Sony Bcn Tec (*ESIICv3*). El desenvolupament del present projecte dins d'aquest entorn, està restringit per les especificacions del sistema actual. De manera que els components *hardware* del nostre disseny, s'instanciarà en una de les FPGA de la *ESIIC*v3, i la tasca a executar per el Nios II s'implementarà a la FPGA del kit de desenvolupament d'Altera. Pel que fa la comunicació, s'utilitzarà el Xip d'Ethernet (TCP/IP) per tal de comunicar-nos amb el PC i poder depurar els dissenys sobre el sistema Nios II. Pel que respecta a les configuracions que volem implementar a les FPGAs, es carregaran a través del port JTAG (*USB-Blaster*) i un cop estigui verificada a nivell funcional, es baixarà a les FPGAs a través de memòries EEPROM.

3.3.1. Plataforma de desenvolupament Nios® II, edició Cyclone® II

La edició *Cyclone II* del kit de desenvolupament Nios II, proveeix de tot el necessari per el desenvolupament SOPC *(System-on-Programable-Chip)*. Està basat en la família de processadors *embedded* Nios II i una FPGA Cyclone II de baix cost. Aquesta plataforma treballa sobre un entorn de desenvolupament Nios II, amb sistema operatiu en temps real i *stack* TCP/IP, també conté llibreria de perifèrics, simulador del set d'instruccions i el *software* de disseny Quartus II que inclou SOPC Builder. Les característiques principals de la placa són les següents:

- FPGA Cyclone II EP2C35 d'Altera
- Memòria SRAM, FLASH, DDR SDRAM
- 8 *LED*'s d'usuari
- 4 interruptors dedicats
- Ports JTAG
- Port sèrie RS232
- Xip d'Ethernet
- Polsador de *reset*
- Dos *displays* de set segments

- 2 x 41 pins d'E/S habilitables.

A la imatge següent es mostra la placa Nios II Kit Development, Cyclone II edition d'Altera

Figura 3.5.- Kit de desenvolupament Nios II, edició *Cyclone* II

3.3.2. Plataforma de prototipat ESIICv3

La plataforma de prototipat ESIICv3 és una placa complementaria al kit de desenvolupament Nios II anterior, i ha estat dissenyada perquè pugui acoblar-s'hi a través dels pins d'E/S. Aquesta plataforma actua coma dispositiu perifèric, i les FPGAs que conté, només contenen elements *hardware*. És una placa d'ús empresarial, orientada a la verificació de circuits impresos de televisió. Les característiques principals són:

- 2 FPGAs, *Cyclone* I EP1C12 d'Altera
- Gran nombre de pins d'E/S
- Múltiples connectors HDMI, VGA, i Audio.
- Potenciòmetres, circuits excitadors de corrent.
- Ports JTAG
- Port sèrie RS-232
- *LED*s d'us general
- Polsador de reset
- Memòries EEPROM
- Connector de Panell

- Connector RCA

A la imatge de la figura 3.6 es mostra la placa de prototipat ESIICv3 desenvolupada per Sony Bcn Tec

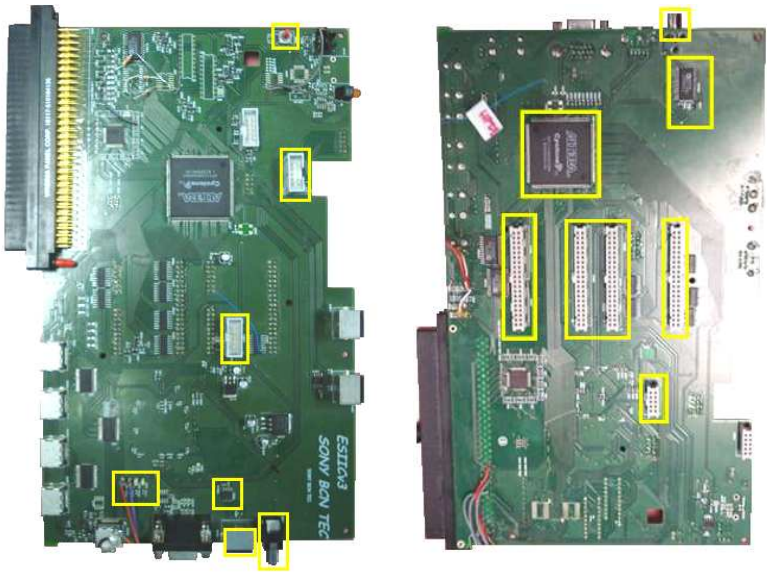


Figura 3.6.- Placa de prototipat ESIICv3.

Per a realitzar la transmissió per infraroig s'ha utilitzat un circuit excitador de corrent com el de l'esquemàtic de la figura 3.7 *a).* Aquest circuit actua com a etapa amplificadora. A l'aplicar el senyal modulat a la base del transistor, aconseguim que es generi el corrent necessari perquè, a la sortida, el *LED* d'infraroig condueixi. El circuit excitador de corrent està implementat físicament amb components *SMD* a la plataforma de prototipat (figura 3.7 *b*), també hi ha implementat un potenciòmetre controlat digitalment per tal de poder simular l'efecte de la distància en les proves de test per control remot. El *LED* infraroig està connectat a la placa mitjançant un connector RCA[5],

a) *b)* *c)*

Figura 3.7.- *a)* Esquema del circuit excitador de corrent, *b)* Circuit excitador de corrent implementat amb components *SMD, c)* vista de la connexió del LED infraroig amb el connector RCA.

[5] Connector RCA: Acrònim de Radio Corporation of America, El connector mascle té un pol al centre (+), rodejat per un anell metàl·lic (-). El connector femella te com a pol central un forat cobert per un anell de metall, més petit que el mascle per a facilitar-ne la subjecció. Ambdós connectors tenen una part intermèdia com a aïllant

Capítol 4.

Disseny i Implementació

4.1. Arquitectura del sistema

Tant el desenvolupament *hardware* com *software* dissenyat, haurà de conviure amb altres aplicacions de test ja implementades a les FPGAs, per tant, s'haurà d'adaptar el nostre sistema a les especificacions i als recursos disponibles. Com s'ha comentat en el capítol anterior, l'arquitectura del sistema consta d'un bloc *hardware* implementat a la *Cyclone I* capaç de generar les trames, una interfície de comunicació entre aquest bloc, el bus Avalon i el bus Wishbone, i tots els recursos necessaris per generar la modulació del senyal. Per tal de controlar aquesta aplicació s'ha dissenyat un programa en *C++* a executar pel Nios II (dins de la *Cyclone II*) i finalment s'ha dissenyat una interfase gràfica per realitzar proves a nivell d'usuari.

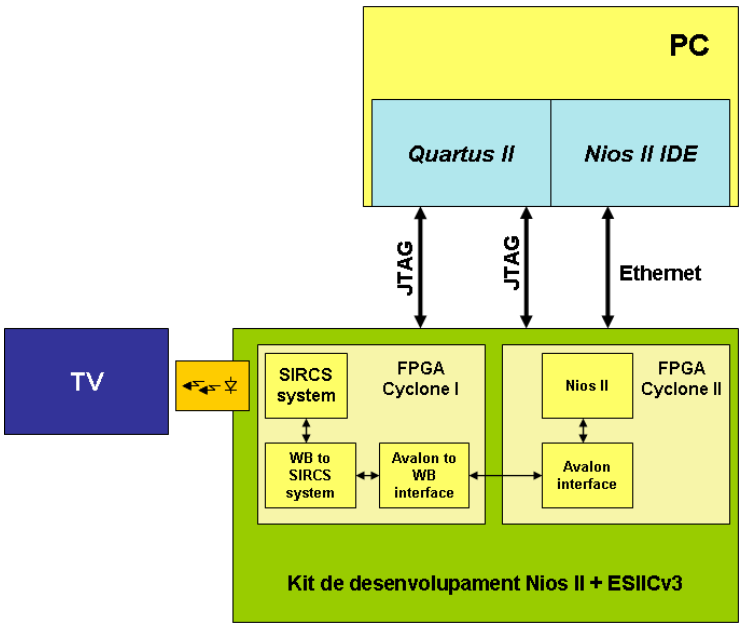


Figura 4.1.- Arquitectura del sistema

S'ha escollit una configuració pel Nios II de tipus *fast,* aquesta versió és la més completa i la seva ocupació dins la FPGA, està entorn de 1400 i 1800 elements lògics.

4.2. Disseny i implementació Hardware

El disseny i la implementació *Hardware* d'aquest projecte, com s'ha esmentat anteriorment, forma part d'un sistema de test molt més ampli que ja té unes especificacions establertes, per tant, s'haurà d'adaptar el disseny a aquestes especificacions. En el cas que ens ocupa, la utilització del bus WishBone seria innecessària, ja que amb les característiques del bus Avalon ja en tindríem prou, però la compatibilitat que aporta aquest bus amb altres IP-Cores, fa que sigui un requeriment del sistema.

4.3. Disseny de components/mòduls

Per poder assolir els objectius d'aquest projecte de final de carrera, es va haver de crear diversos mòduls *hardware;* un *Soft-IP* que generés les trames segons les especificacions funcionals de l'estàndard *SIRCS*, una interfície entre aquest i el bus WishBone i finalment una interfície entre els ports del bus WishBone i els del bus Avalon, per tal d'usar el Nios II com a processador. En els següents apartats es detalla la implementació de cada un d'aquest blocs i s'explica la interconnexió entre ells.

4.3.1. Mòdul *hardware* - Bloc generador de trames

El disseny de la unitat funcional generadora de trames, s'ha basat en una màquina d'estats finits *FSM*. L'objectiu principal d'aquest *soft-IP* és generar les trames corresponents a una entrada binaria, i transmetre de manera continuada o en mode rafega depenen d'una entrada lògica. A continuació es mostra el bloc implementat amb el Quartus II.

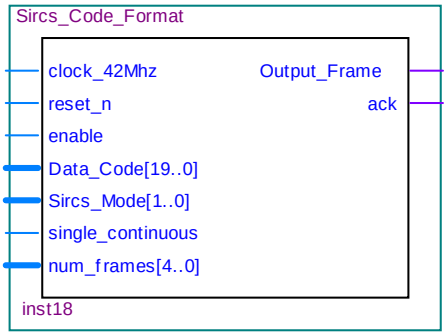


Figura 4.2.- Mòdul VHDL generador de trames

La implementació del mòdul *hardware*, consta de tres senyals d'entrada de propòsit general, una senyal de rellotge de 42MHz, un *reset* i un *enable,* i 28 bits de dades d'entrada, que ens permeten generar les trames escollint-ne la seva configuració i el mode de transmissió.

- Data_Code: Entrada de 20 bits de dades per generar les tres configuracions establertes per l'estàndard *SIRCS*; formades per 7 bits de dades i 5, 8 o 13 bits corresponents a la categoria.

- Sircs_Mode: Entrada de dos bits, que indica si volem generar una trama amb una configuració de 12, 15 o 20 bits, la seva codificació és de "00", "01", i "10" respectivament.

- single_continuous: Quan té un nivell baix la transmissió es realitza de manera continua, amb un nivell alt es realitza per rafegues. L'activació d'aquest senyal comporta la necessitat de establir un nombre determinat de rafegues a transmetre *num_frames*.

- num_frames: Senyal amb el qual especifiquem la quantitat de rafegues que volem transmetre. Fins a 31 trames per rafega (fins 5 bits). Només per transmissions en rafega.

Pel que fa a les sortides del bloc implementat, en tindrem prou en implementar les següents senyals.

- ***Output_frame:*** senyal de sortida de la trama corresponent a l'entrada binaria.

- ***ack:*** Activem la sortida *ack,* per indicar en mode rafega, que s'han transmès el nombre de trames especificat per el senyal d'entrada *num_frames.* L'activació d'aquesta senyal, serà llegida per l'interfase de comunicació i deshabilitarà la senyal d'*enable* parant així la transmissió.

El diagrama d'estats corresponent a aquesta implementació és el que es mostra a la figura 4.3.

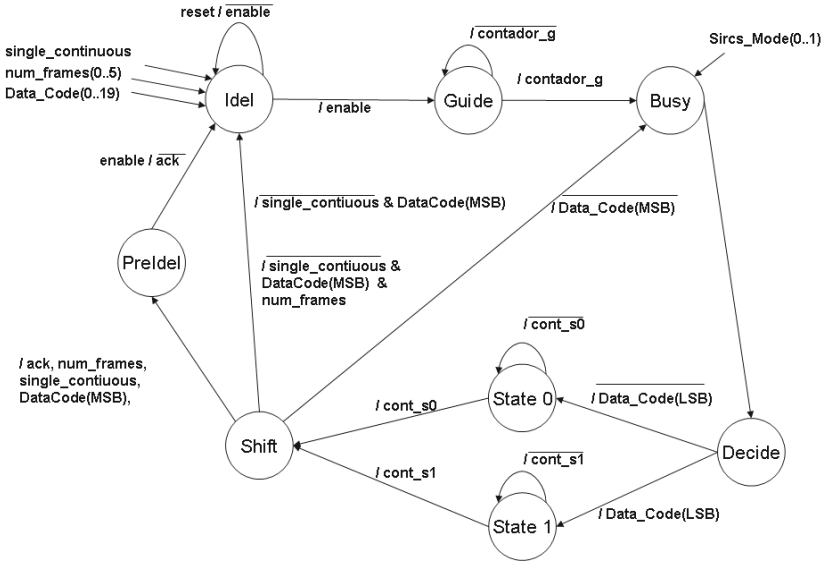


Figura 4.3.- Diagrama d'estats del soft-IP genrador de trames.

Quan s'activa la senyal *enable,* es produeix la transició de l'estat de respòs "*Idel"* a l'estat "*Guide"* on es genera un pols de 1.8ms corresponents a la capçalera inicial. A l'estat "*Busy"* es pren el valor de l'entrada *Sircs_Mode* i es classifica l'entrada *Data_Code* en els primers 7 bits de dades i els restants de categoria començant per l'LSB (primer bit a transmetre). L'estat "*Decide"*, com diu el seu nom, passa a l'estat "*state 0"* o *"state 1"* depenen del bit transmès des de l'estat "*Busy*". Els estats "*state 0"* i "*state 1"* són els que generen els temps per a cada bit de la trama i finalment l'estat "*Shift"* és el que s'encarrega de desplaçar el bits de l'entrada, controlar el període total de la trama i activar o no el senyal ack corresponent al mode rafega. L'estat de "PreIdel" és necessari, només en el mode rafega, perquè el Nios II tingui els cicles necessaris per desactivar la senyal d'*enable,* i parar la transmissió.

4.3.2. Mòdul de comunicació

Per a poder utilitzar el Nios II com a microcontrolador, és necessari adaptar els ports WishBone als port del bus Avalon, i un cop es disposi de senyals WishBone, adaptar aquests al mòdul *hardware* explicat en l'apartat anterior. Per fer-ho, és necessari que totes les entrades Wishbone es puguin generar a partir de senyals Avalon i que totes les sortides que necessita el mòdul del bus Avalon per a funcionar, es generin a partir de senyals WishBone o d'altres senyals disponibles a l'interior del sistema. A continuació s'explica detalladament aquestes interfícies, i la seva implementació dins de l'FPGA.

4.3.2.1. Interfície entre Wishbone – Mòdul HW

El bloc de la figura 4.4, mostra la implementació de la interfície dissenyada per adaptar els senyals de un Wishbone Slave als del *soft-IP* que genera les trames. Com s'observa a la figura s'han afegit les entrades del bloc generador de trames com a sortides de la interfície. Les especificacions del bus són, un conjunt de 6 línies d'adreça i 32 bits d'amplada de dades. A la secció 2.5.1.2 es detallen les descripcions i funcions del senyals referents al WishBone Slave

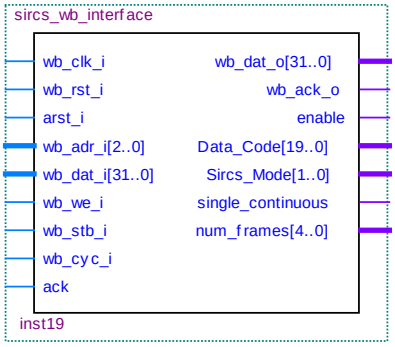
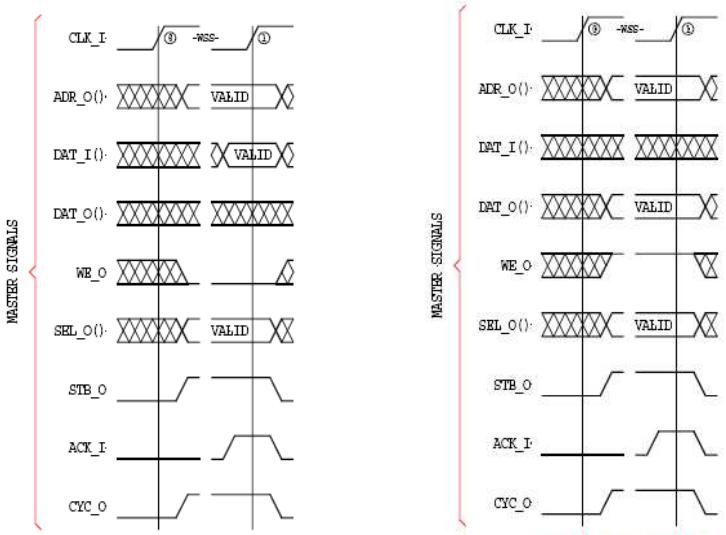


Figura. 4.4.- Interfície entre el bus Wishbone i el Soft_IP Core generador de trames.

Cal recordar que el bus WishBone usa una seqüència *handshanking*, és a dir, que el *Master*, amb un senyal d'*strobe*, indica la estabilitat del bus. L'*Slave* al veure el senyal, activa un senyal *ACK* amb el que indica que ha rebut les dades (en el cas d'escriptura) o que estan disponibles al bus (en un cas de lectura), i finalment el dispositiu *Master,* respon a l'activació del senyal *ACK* per part de l'*Slave,* desactivant l'*strobe*, i finalitzat el cicle al bus. Les figures següents mostren els cicles necessaris per a realitzar lectures i escriptures d'un bit a través del bus WishBone.



a) *b)*

Figura 4.5.- cicles de lectura *a)* i escriptura *b)* d'un bit amb el bus WishBone

La generació dels senyals necessaris per tal d'adaptar dels ports WishBone als port del bus Avalon s'expliquen a l'apartat 4.3.2.2. A continuació s'explica breument, les característiques més rellevants de la implementació dels ports d'entrada i sortida i els processos necessaris per efectuar les operacions de lectura i escriptura.

Per facilitar la implementació del bloc, s'han creat diversos senyals interns:

• Un senyal d'accés a escriptura o lectura, per a facilitar la implementació el protocol *handshaking*; *wb_acc*

• Un senyal per adaptar els ports *wb_rst_i* i *arst_i* permeten realitzar *resets* síncrons i asíncrons respectivament. (*rst_i)*

• Un senyal per a la generació del registre d'escriptura/lectura. (*cr)*

• Un senyal per a poder generar el final d'un cicle. (*iak_o*)

Finalment s'ha implementat diversos processos per a poder generar els cicles d'escriptura i lectura de manera adient. Un dels processos genera el senyal intern amb el qual indiquem el final del cicle, i l'envia al port *wb_ack_o*. El procés de generació del registre d'escriptura, simplement, realitza la transferència de les dades provinents del mòdul del bus Avalon (port *wb_dat_i*) al registre (*cr*), si el senyal *wb_acc* té un nivell alt (*wb_acc=1*). El procés de lectura (*wb_acc=0*), s'utilitza per a poder desactivar la transmissió; un cop s'ha verificat que el senyal "*ack"* provinent del Core generador de trames té un nivell alt (fi de la trama), deshabilita el bit corresponent a l'*enable* del registre (*cr*). Al finalitzar les transferències, es divideix el registre (*cr*) en cadascuna de les diverses sortides corresponents al bloc generador de trames. El format del registre *cr* esdevé com el de la figura 4.6:

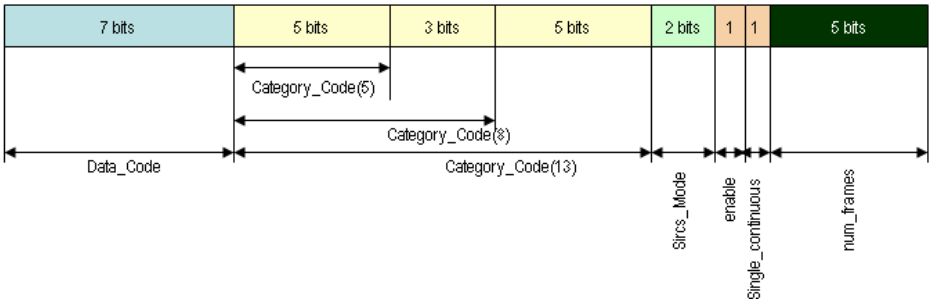


Figura 4.6.- Format dels bits del registre de lectura/escriptura.

4.3.2.2. Interfície Avalon – Wishbone

Per tal de generar els senyals del bus WishBone a partir de senyals del mòdul de l'Avalon, es van nombrar els ports corresponents als senyals d'entrades de la interfície *Avalon Slave* com a sortides, i les seves sortides com a entrades, de manera que els ports que queden a l'esquerra de la figura 4.7 són senyals de sortida i els que queden a la dreta són senyals d'entrada. També es van definir una sèrie de senyals i ports addicionals necessaris per implementar el sistema.

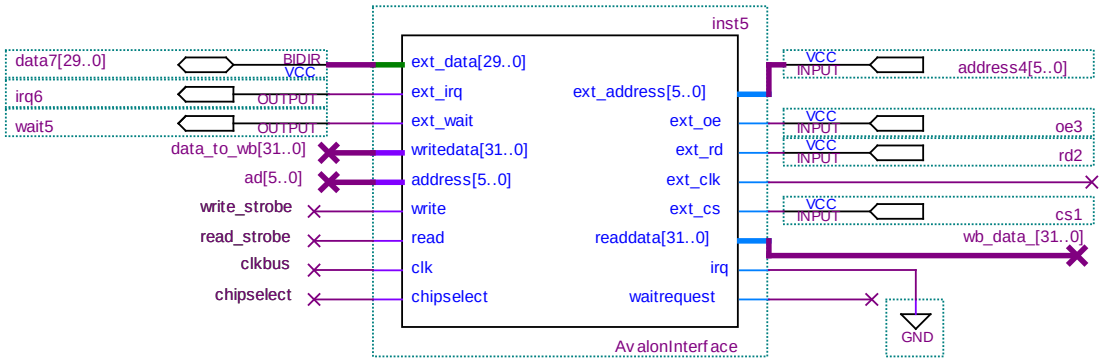


Figura 4.7.- Interfície entre el bus Avalon i el bus Wishbone.

A la secció 2.5.2.2 es detallen les descripcions i funcions dels diferents senyals del bus Avalon. A continuació s'explica breument, les característiques més rellevants de l'adaptació dels ports d'entrada i sortida en senyals i estímuls compatibles amb l'arquitectura del bus WishBone.

L'arquitectura interna de l'*AvalonInterface* uneix directament els següents ports, el prefix *ext_* es refereix ales dades provinents del mòdul del bus Avalon:

chipselect ←*ext_cs, clk* ←*ext_clk, read* ←*ext_rd, write* ←*ext_oe, ext_wait* ←*waitrequest, ext_irq* ← *irq i address* ←*ext_address*.

Per tal d'implementar els cicles d'escriptura i lectura del bus Wishbone amb els senyals Avalon, primerament, s'ha definit el port *ext_data* com a bidireccional, de manera que quan el mòdul del bus Avalon indiqui una transferència de lectura (*read=1*), el port *ext_data* serà d'entrada, i prendrà el valor de *wb_dat_i* (dades disponibles al bus WishBone) per el port *readdata*, en canvi quan s'indiqui un cicle d'escriptura, el port *ext_data* serà de sortida i adreçant el port *writedata* el qual representa el senyal *wb_dat_o.*

El bus Avalon no conté senyals de reconeixement, sinó que té un temps fixat per a realitzar la transferència, després del qual, l'iniciador de la transferència desactiva el senyal *chipselect.* Per implementar els impulsos necessaris per a la comunicació amb WishBone, i activar els senyals *wb_stb_i*, *wb_cyc_i* s'ha usat una porta *and* amb el senyal *chipselect* i un senyal generat amb la multiplexació del senyal *address*. d'aquesta manera la desactivació de la seqüència anirà a càrrec del senyal *chipselect* i podrem comunicar-nos amb el mòdul en qüestió des de la implementació *software* del Nios II, a través de l'adreça. Pel que fa a la generació del senyal *wb_we_i*, n'hi ha prou en utilitzar el senyal *write* de la interfície *AvalonInterface*.

4.3.3. Interconnexió del sistema *Hardware*

A la figura 4.8 es pot apreciar la interconnexió dels blocs descrits anteriorment, s'ha obviat el bloc d'interfície entre el bus Avalon i el bus WishBone, ja que els seus senyals més representatius queden descrits en forma d'entrades i sortides a l'apartat 4.3.2.2. Per a millorar els senyals de rellotge que arriben a la FPGA s'ha utilitzat dos mòduls PLL, de la llibreria del Quartus.

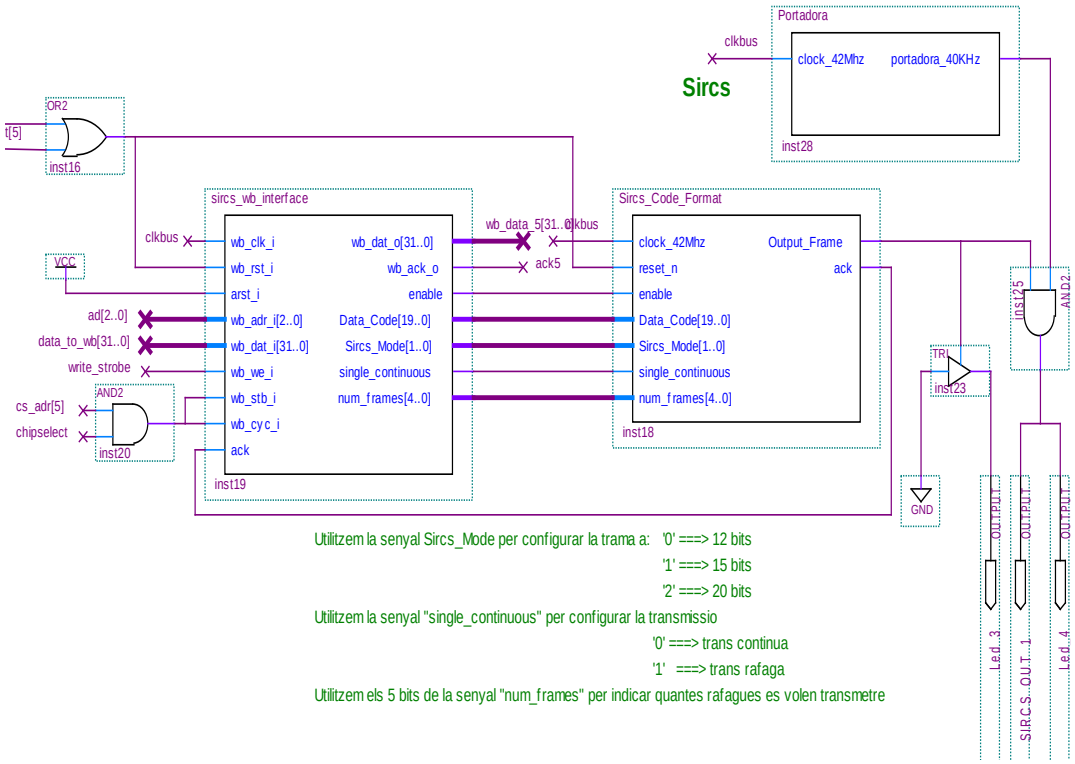


Figura 4.8.- Vista des de del sistema del Quartus II.

Per a poder transmetre les trames a través de un *LED* infraroig, s'ha realitzat una modulació per amplada de pols *BPSK*, multiplicant el senyal de sortida amb una senyal oscil·lant, de freqüència 80KHz, utilitzant una porta *AND*.

La transmissió de les trames està implementada per diversos pins de la FPGA (*PIN_45, PIN_46, PIN_170*). D'aquesta manera podem tenir un control més específic de la seva funcionalitat; s'ha implementat a través d'un pin (*PIN_45*) una sortida sense modular amb un *LED* de la placa ESIICv3 i una modulada a traves d'un altre (*PIN_46*). Finalment des del pin restant, s'envia el senyal modulat a un *LED* infraroig, passant per un circuit excitador de corrent, realitzat amb dispositius SMD de la placa de prototipat. Aquesta última configuració és la que utilitzarem per a verificar la comunicació amb l'aparell de TV remotament.

4.4. Desenvolupament Software

El desenvolupament del *software* implementat, va íntimament lligat a *hardware* desenvolupat, de manera que un sense l'altre no funcionen. Aquest fet provoca que el *software* desenvolupat s'hagi d'entregar conjuntament amb el *hardware*. Per facilitar-ne la reutilització, s'ha creat una llibreria de funcions que simplifica les tasques, de manera que quan un enginyer tingui que usar el *core* i disposi d'aquestes funcions, només necessitarà especificar els paràmetres bàsics que el componen. En aquest cas, l'adreça del dispositiu amb qui es vol realitzar la comunicació (*Catergory_code*), els 7 bits de dades (*Data_code*), la configuració i el mode de transmissió (*sircs_mode i sigle_continuous*) i el nombre de trames que es vol transmetre (*num_frames*).

En aquesta secció s'explica tant les declaracions i funcions que executa el Nios II, com la implementació posterior de una llibreria de funcions.

4.4.1. *Software* executat pel Nios II

El desenvolupament del *software* que executa el Nios II, ha estat restringit a les especificacions del sistema de manera que la tasca que s'ha d'implementar s'ha d'adaptar a la configuració del sistema de test present. Per aconseguir això, sense haver de modificar les altres parts del *software*, en primer lloc, es va definir els paràmetres de configuració especifica dins del fitxer de definicions del *top* del programa, a continuació es va definir la tasca a executar i es van incorporar a les rutines de petició de connexió per Telnet, Finalment es va crear el programa que respon a la tasca de la rutina. A continuació s'exposen les característiques principals de la implementació del *software*.

Per tal d'instanciar la nova tasca dins del fitxer de definicions (*simple_socket_server.h*) al *top* del sistema de verificació de plaques, es van implementar les següents accions:

- Es va declarar la tasca en el sistema de test.
- Es va mapar la codificació de l'adreça d'accés al bloc *hardware* en particular.
- Es van definir les macros per canviar els valors del bus, tan per escriptura com per lectura.
- Definir la prioritat de la tasca dins del sistema de test.

Un cop instanciat el nou sistema dins del projecte de configuració del programa, es van establir les rutines necessàries, per poder crear la tasca depenen de les instruccions transmeses per l'usuari (*simple_socket_sever.c*):

- Definir la petició de connexió per accedir a la rutina
- Crear la rutina d'execució de la tasca, per tal de que quan aparegui una nova transferència de dades vàlida, pugui acceptar la connexió o no depenen d'un paràmetre identificatiu (*sircs).*

En aquest punt es van establir els formats de les comandes de comunicació per *telnet*. Per a poder implementar els diferents tipus de transmissions que especifica l'estàndard *SIRCS*, primerament es va pensar en dos tipus de transmissions; *-single* i *-continuous,* corresponents a transmissions en ràfega o continuades respectivament, però en refinaments posteriors es va determinar que es requeria de una tercera comanda per a poder deshabilitar la transmissió en el mode *-continuous*, de manera que finalment es van classificar en *-single, -enable_continuous* i *disable_continuous*. Els formats finals de les seqüències a executar des de la consola de *telnet* esdevenen així:

- *sircs -enable_continuous -data_code 0xXX -category_code 0xXX -sircs_mode X*
- *sircs -disable_continuous*
- *sircs -single -data_code 0xXX -category_code 0xXX -sircs_mode X -num_frames 0xXX*

Tots aquests formats tenen el paràmetre identificatiu *sircs* com a capçalera, seguit del paràmetre referent al tipus de transmissió. A continuació (en excepció de la comanda de deshabilitació) apareixen els paràmetres específics de les trames; els 7 bits de dades – *data_code*, fins a 13 bits de categoria *-category_code* i el mode de configuració de transmissió – *sircs_mode*. Pel cas de transmissions en rafega també cal especificar el nombre de trames que es vol transmetre, paràmetre *–num_frames*.

Finalment es va implementar la funció específica de la tasca. L'objectiu principal del programa consisteix en interpretar i verificar de manera adient les seqüències introduïdes per *Ethernet,* i carregar els valors dels registres del bus, necessaris per tal d'executar transferències d'escriptura o lectura. A continuació es s'explica, a grans trets, les accions que realitza el programa.(*sircs.c*)

• Un cop verificada la capçalera d'identificació de la tasca (*sircs*), secciona la comanda de la seqüència *telnet*. La primera secció ens identifica el mode de transmissió, i les parelles de seccions restants indiquen el nom del paràmetre i el seu valor.

• La següent acció que realitza es verificar que totes les parelles de "paràmetre-valor" comunes a totes les comandes tinguin el valor adequat i estiguin dins del seu rang d'operació. Per exemple, la parella paràmetre-valor -*category_code 0xXX*, ha de tenir un rang entre 0x00 i 0x1F per un -*sircs_mode 0*, corresponent a una configuració de la trama de 12 bits, un rang màxim de 0x00 a 0xFF per una configuració de 15 bits (-*sircs_mode 1*) i un rang de 0x00 a 0xFFF per una configuració de 20 bits (-*sircs_mode 2*).

• Quan els paràmetres de la trama són validats, els organitza de manera que es corresponguin amb el format desenvolupat a l'entorn *hardware* (veure figura 4.6 del Capítol 4), canvia els valors dels registres del bus i finalment desactiva la tasca.

• Per a la desactivació automàtica en mode -*single*, es determina el temps necessari per enviar el nombre de trames especificat per el paràmetre –*num_frames* multiplicant aquest pel període d'una trama, (45ms * sircs_num_frames) i un cop ha passat aquest temps i es llegeix la senyal d'*ack* es finalitza la tasca.

Val a dir, que la implementació *software* d'aquest projecte ha estat desenvolupada tenint en compte el *hardware* implementat, i s'ha anat validant cadascuna de les fases, refinant paral·lelament el desenvolupament *hardware* i *software*

4.4.2. Llibreria de funcions

L'elaboració d'una llibreria de funcions aporta un valor afegit al sistema, ja que en possibilita la seva reutilització per posteriors dissenys de manera senzilla i pràctica. La implementació de la funció de la llibreria es va desenvolupar des de l'entorn de desenvolupament LabWindows/CVI que proporciona National Instruments i posteriorment es va incloure al sistema. Aquest desenvolupament va fortament lligat a la estructura de test ja present, i s'ha d'incorporar respectant els paràmetres de configuració establerts.

A part de carregar paràmetres específics de configuració de les trames, la funció desenvolupada permet establir la connexió i tancar-la, enviar les comandes al *socket* i capturar el

què en retorna. Aquestes ultimes funcions s'han implementat a través de funcions ja creades de la llibreria, les quals verifiquen la connexió i els temps establerts per a cada transferència.

Les funcions principals que realitza la funció, nombrada *SetSircsESIIC*, de la llibreria són, bàsicament, les implementades pel software que ha d'executar el Nios II:

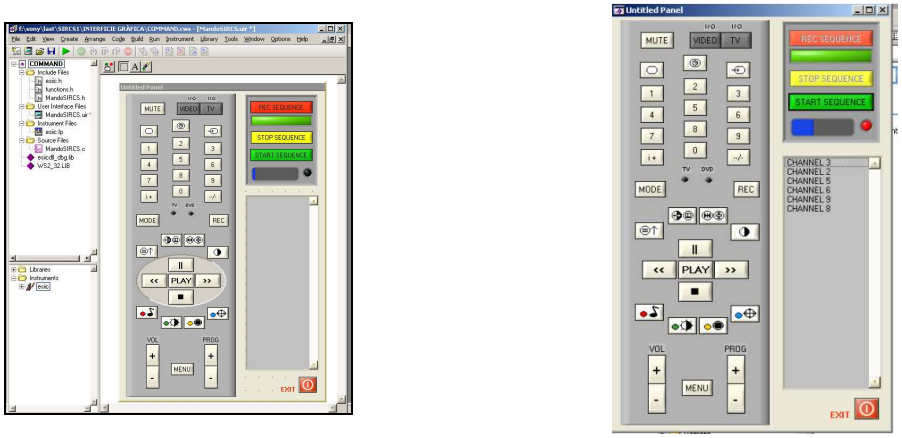
- Verifica que tots els paràmetres introduïts estiguin dins del seu rang
- Canvia els valors dels registres del bus
- Estableix la connexió per *ethernet*
- Envia les dades per el *socket*
- Tanca la connexió.

4.5. Desenvolupament entorn software

El desenvolupament del *software* per a la interfície gràfica, es va realitzar amb l'ajuda de la llibreria de funcions implementada. Amb aquesta llibreria resulta molt senzill realitzar una interfície amb l'usuari, brindant la possibilitat de realitzar proves d'una manera molt simple.

Entre les funcions de la llibreria utilitzades per a realitzar el *software*, hi trobem, la nostra funció, *SetSircsESIIC* i funcions per a realitzar tasques de caràcter general, com per exemple la funció dedicada a establir la connexió amb la plataforma de prototipatge *ConnectWithESIIC* que permet obrir la connexió ethernet especificant la IP i el port d'entrada.

A la figura 4.9 es mostra l'aspecte del la interfície gràfica i la *suite* de desenvolupament LabWindows/CVI de National Instruments.



a) *b)*

Figura 4.9.- a) Entorn de desenvolupament LabWindows/CVI, b) aspecte entorn gràfic en funcionament

L'objectiu d'aquest programa és, essencialment, permetre la possibilitat de realitzar proves de test de manera seqüencial i a nivell d'usuari. El *software* desenvolupat permet executar comandes en mode rafega o contínua, gravar una seqüència determinada i executar-la posteriorment. El *software* també permet establir, verificar i finalitzar la connexió amb la plataforma de prototipat. A la figura 4.9 *b)* mostra una captura del *software* executant una de les seqüències gravades.

La codificació de les comandes s'ha realitzat a través de un fitxer Excel. Aquest fitxer conté una taula indexada on s'especifiquen les funcions de les comandes i els valors de descodificació de les trames (paràmetres *data i category).* D'aquesta manera, si es modifiquen algunes de les codificacions de les comandes, no cal que ens preocupem de canviar el codi del programa, tan sols tindrem que reassignar el nou valor a la funció de la comanda del fitxer Excel.

KeyNo	Function	Data	Category
1	CHANNEL 1	10	0
2	CHANNEL 2	10	40
3	CHANNEL 3	10	20
4	CHANNEL 4	10	60
5	CHANNEL 5	10	10
6	CHANNEL 6	10	50
7	CHANNEL 7	10	30
8	CHANNEL 8	10	70
9	CHANNEL 9	10	8
10	CHANNEL 0	10	48
11	POWER ON/OFF	10	54
12	VIDEO POWER	1A	54
13	MUTING	10	14
14	RED	18	19
15	GREEN	18	59
16	YELLOW	18	39
17	BLUE	18	7A
...	...	...	...
37	VIDEO SELECT	10	52

Figura 4.10.- Fragment de taula del fitxer Excel

Capítol 5.
Test i Resultats

5.1. Entorn de test

En aquest apartat es descriuen les eines utilitzades per a validar el correcte funcionament del nostre sistema. A continuació es descriuen tant les eines *hardware* com *software* utilitzades durant la verificació i test d'aquest projecte.

5.1.1. Eines *hardware*

a) Analitzador lògic Signal Tap II

L'analitzador lògic Signal Tap II, és una eina de test i validació que ens ofereix el propi Quartus II. Per a poder-lo utilitzar cal activar-lo en el nostre disseny i recompilar-lo, donat que s'ha generar lògica addicional dins la FPGA. Cal especificar tots els senyals que vulguem veure i també ens dóna la possibilitat de introduir varis nivells de *trigger*, etc.

b) ModelSim Altera

ModelSim és un entorn complet de simulació de llenguatge de descripció de *hardware*, que ens permet verificar la funcionalitat, els models de *timing* del nostre disseny i el codi font.

c) Oscil·loscop

L'oscil·loscopi es va usar per comprovar, que les captures del senyal de sortida, es corresponguessin als formats de les trames. L'oscil·loscopi es va usar durant tot el desenvolupament *hardware* i *software* i el seu ús va ser essencial per verificar les respostes del sistema de manera real.

d) TV (*Sony Corp.*)

En la verificació final del sistema, s'ha usat un TV convencional de la marca Sony. D'aquests a manera podem veure com respon l'aparell a les comandes generades pel *LED* infraroig.

5.1.2. Eines *software*

a) Nios II IDE

El sistema Nios II permet executar i depurar el sistema dissenyat, sobre el mateix compilador *Nios II IDE*, establint la connexió *ethernet* i permeten la transferència de dades des del PC.

c) Telnet

Telnet (*TELecomunication NETwork*) és el protocol que s'utilitza per accedir, mitjançant una xarxa, a una màquina. Telnet només serveix per accedir en mode terminal. Aquesta eina ha estat de gran utilitat per tal de verificar els missatges d'error implementats durant el desenvolupament del *software.*

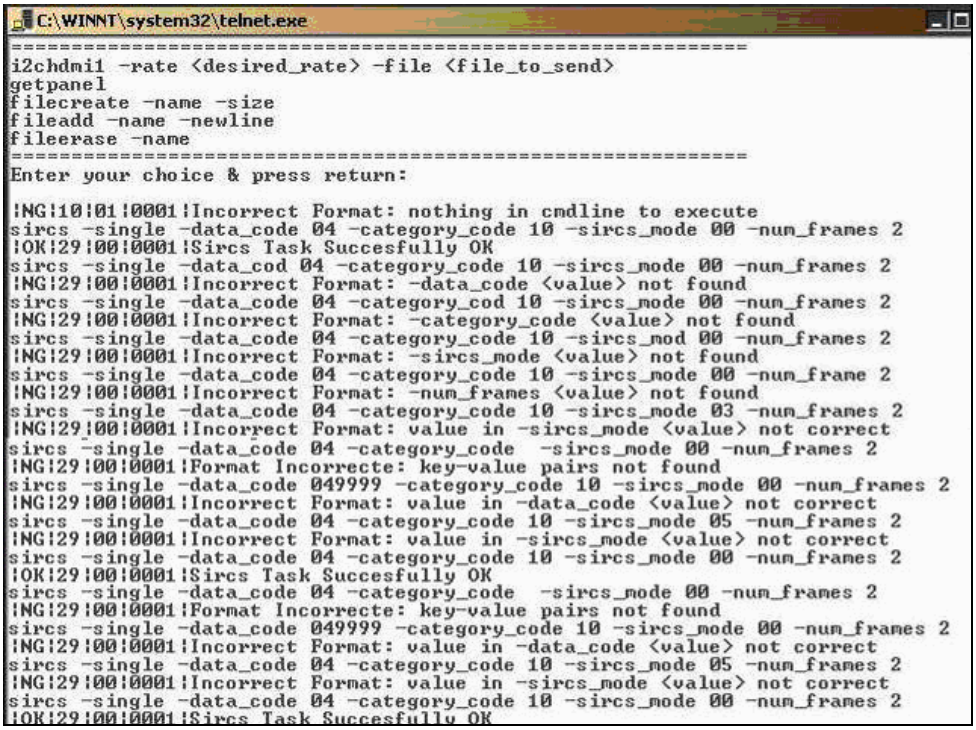


Figura 5.1.- Missatges d'error escrits pel Nios II desde la consola de *Telnet.*

b) LabWindows/CVI

L'entorn de desenvolupament i test LabWindows/CVI permet desenvolupar aplicacions software en C/C++, combinant la funcionalitat especifica per al control d'instrumentació amb la possibilitat de crear interfícies amb l'usuari a partir de una biblioteca d'objectes.

5.1.3. Muntatge del sistema

Com s'ha explicat anteriorment la plataforma de prototipat està formada per dues plaques de desenvolupament; el kit de desenvolupament d'Altera Nios II, i la plataforma complementaria a aquesta, la ESIICv3. A la imatge de sota, podem veure el muntatge del sistema i els seus perifèrics.

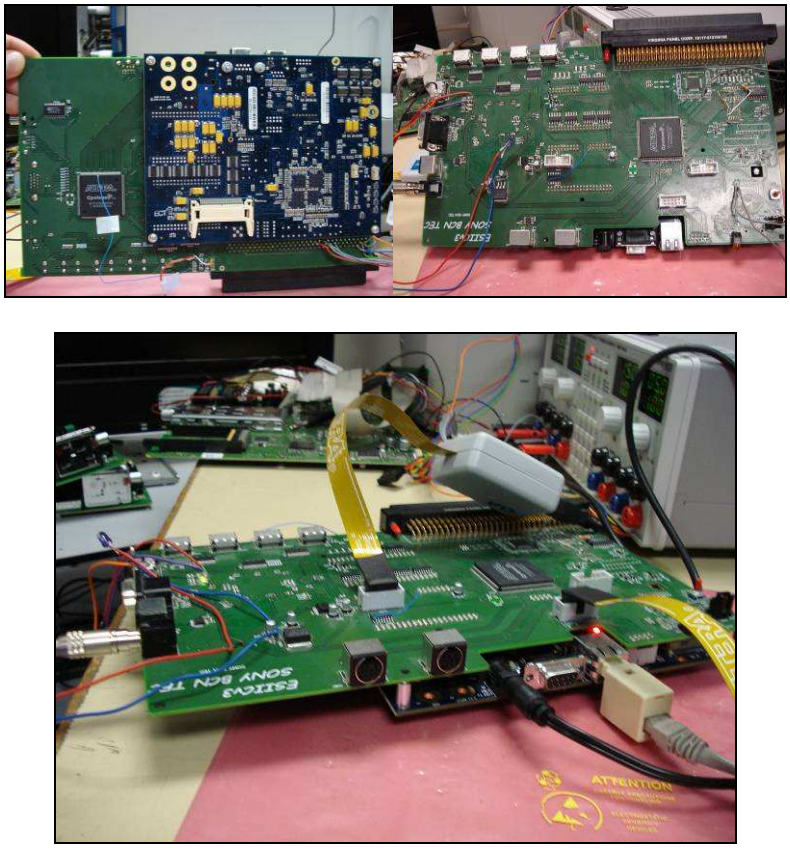


Figura 5.1.- Muntatge del sistema

Pel què respecte a la detecció del senyal, tant si s'utilitza un mòdul detector d'infrarojos genèric o un fotodíode amb una resistència de *pull-up* connectada a +Vcc, es produeix una inversió del senyal, i per tant, provoca que quedi en un nivell alt quan no existeixi transmissió. El circuit excitador de corrent utilitzat per a la transmissió del senyal (figura 3.7) compleix aquesta propietat i inverteix el senyal modulat mantenint un nivell alt en l'estat de repòs. En una primera verificació, abans de realitzar la transmissió per infraroig, es van connectar el senyal de sortida sense modular directament al detector del TV, d'aquesta manera vaig poder comprovar que la trama és detectava correctament sense necessitat de modular el senyal. Posteriorment es va procedir a la verificació del sistema modulat per infraroig.

5.2. Resultats de simulació

En aquest apartat es mostren els resultats obtinguts a partir de simulacions i l'estratègia de test del sistema. Tal com s'ha comentat breument en altres seccions, la validació del codisseny *hardware-software* del sistema s'ha realitzat de manera conjunta, comprovant les funcionalitats implementades per *hardware* paral·lelament amb el codi a executar sobre el Nios II. A continuació s'expliquen detalladament els resultats de simulació obtinguts, que validen la totalitat del disseny implementat.

Per a verificar el sistema un cop descarregat a la PFGA, s'han realitzat diverses captures utilitzant diferents nivells de *trigger* amb l'analitzador lògic SignalTab. En primer lloc s'ha verificat que els senyals dels ports d'entrada i sortida corresponents al mòdul generador de trames, prenguin els valors adients.

A les simulacions de la figura 5.1 podem veure la correcta generació dels senyals d'entrada i sortida, per una transmissió en mode ràfega de quatre trames, de la seqüència de valor; data=0x70 i category=0x10. La simulació superior correspon al inici de l'operació, i la inferior al final, és en aquesta simulació on es pot apreciar que quan el senyal encarregat de contar les rafegues incrementa fins a quatre, el senyal d' *ack* s'activa i en conseqüència es deshabilita el senyal d'*enable* finalitzant la transmissió.

Figura 5.1.- Simulacions amb SignalTab, de l'inici de la transmissió i el final

A les figures següents es mostren les transicions de la màquina d'estats. La figura *a)* mostra la transició de l'estat de repòs *Idel* a l'estat que realitza la capçalera (estat *guide*).

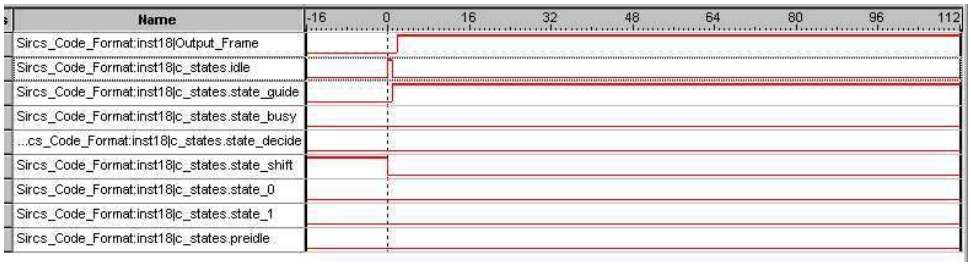


Figura 5.2.- Simulacions amb SignalTab, de les transicions dels estats

A la simulació següent es mostra el final del cicle de l'estat *guide*, i com s'activen l'estat d'espera i l'estat *decide* consecutivament. Depenent del valor del bit a transmetre d'aquest últim estat, s'activaran els estats on es realitzen les codificacions d'un "1" o un "0" (en aquest cas un nivell alt; *state_1*).

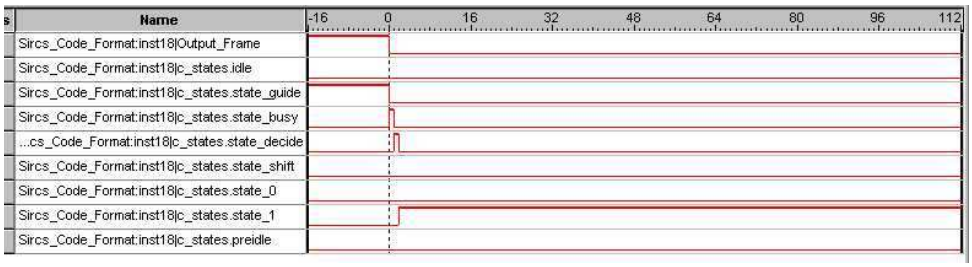


Figura 5.3.- Simulacions amb SignalTab, de les transicions dels estats

A la figura *c)* es pot apreciar les transicions que realitza la màquina d'estats per tal de generar la codificació del senyal de sortida; després de la desactivació de l'*state_1*, desplaça el bit a codificar, passa a l'estat d'espera i altra vegada al de decisió, que en aquest cas, correspon a un transició cap a l'estat encarregat de codificar un "0". (*shift, busy, decide i state_0*).

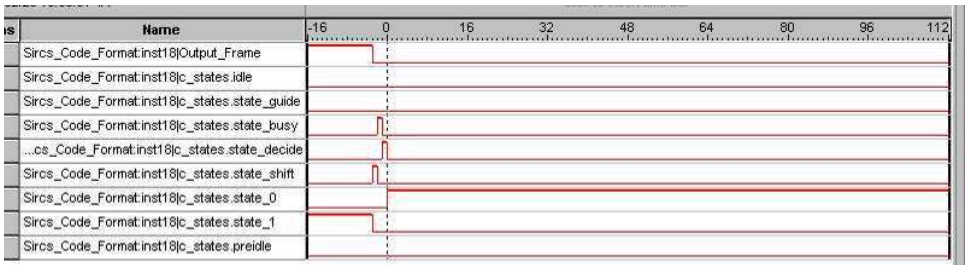


Figura 5.4.- Simulacions amb SignalTab, de les transicions dels estats

En aquest punt les transicions que es realitzen a la màquina d'estats es corresponen a les de figura *c)* fins que s'ha codificat tots els bits de la comanda. Finalment la figura *d)* correspon al final de la generació de la trama (activació de l'estat *Idel*).

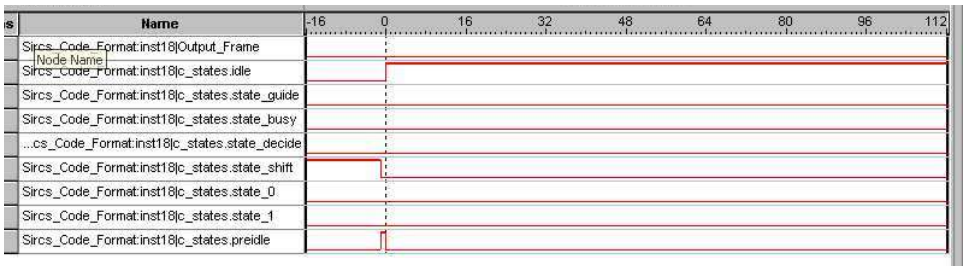


Figura 5.5.- Simulacions amb SignalTab, de les transicions dels estats

Per a validar els senyals d'entrada i sortida de la interfície amb el bus, un cop mapat a la FPGA, s'ha adquirit diferents imatges amb l'analitzador lògic SignalTab on es mostren les diferents operacions que realitza la interfície. A la figura 5.6 podem observar la correcta generació dels senyals d'entrada i de sortida i la assignació dels valors dels senyals interns del bloc. A la simulació inferior podem veure com el senyal *cr* pren els valors adients que a posteriori es descodificaran en cada un dels senyals d'entrada del bloc generador de trames.

El senyal *cr* pren el valor 0x4C00850 (100|1|1|00|0000000010000|1010000), el qual es codificat de la següent manera:

- 1010000=Data_Code
- 0000000010000=Category_Code
- 00=Sircs_mode
- 1=enable
- 1=single_continuous
- 100=num_frames

Figura 5.6.- Simulacions amb SignalTab, de la interfície Wishbone

Amb l'ajuda de l'oscil·loscop, s'ha comprovat que els diversos senyals que formen les trames estiguin dins de les especificacions del disseny. Això s'ha representat enviant el senyal de sortida en diferents modes i per diversos pins de la FPGA, d'aquesta manera podem tenir una visió completa tant de la codificació, com de la modulació de les trames.

La figura 5.7 mostra el detall d'una codificació d'una trama en particular, on es representa una configuració de 12 bits on els paràmetres de dades(*comanda*) i categoria(*adreça del dispositiu*) prenen els valors 0x00 i 0x10 respectivament. Les trames superiors són les invertides i les inferiors són les transmeses en mode normal. Es pot observar, que tant els períodes dels codis com el de la trama, compleix les especificacions de l'estàndard "*SIRCS"* detallat a la taula 2.1.

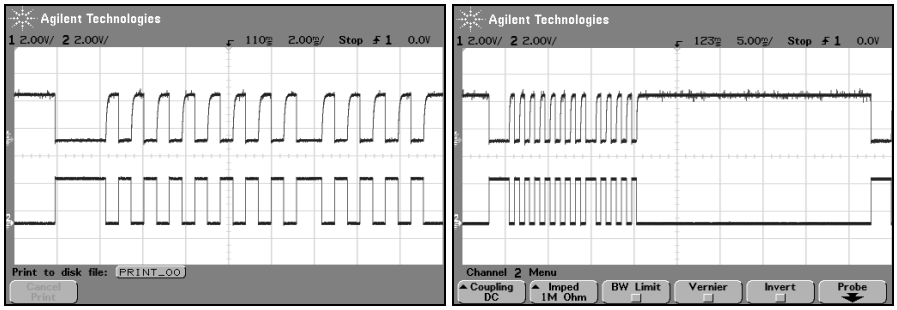


Figura 5.7- *a)* Detall del bits de la trama *b)* Període de transmissió de la trama. Comanda = 0x00, adreça= 0x10, LSB = esquerra

A la figura 5.3 es representen els tipus de transmissions que s'han implementat. La figura *a)* mostra un exemple de transmissió continua d'una comanda, com per exemple, augmentar el volum del televisor. La figura *b)* mostra una transferència en mode rafega (el nombre mínim de trames per l'acceptació d'una comanda és de tres)

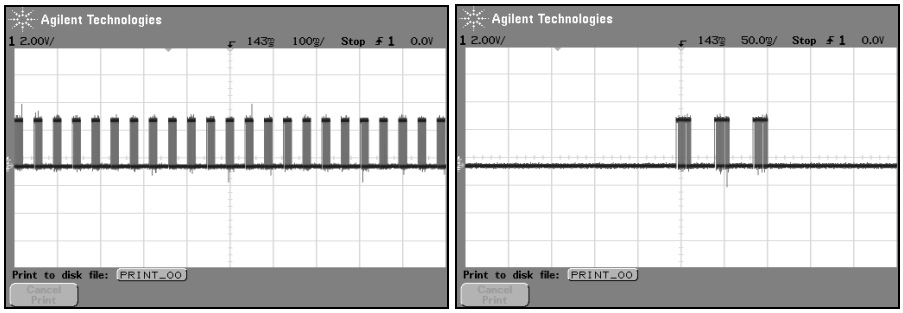


Figura 5.8.- a) Transmissió continua b) transmissió per rafega

A la figura 5.4 *a)* s'han capturat els senyals que formen el sistema de modulació; la portadora, a la part superior, l'oscil·lador de referència al centre i finalment, a la part inferior, el senyal modulat resultant. El senyal superior de la figura *b)* mostra el senyal modulat des d'un dels

terminals del led infraroig havent passat pel circuit excitador de corrent. A la part inferior es pot apreciar la similituds amb el senyal modulat.

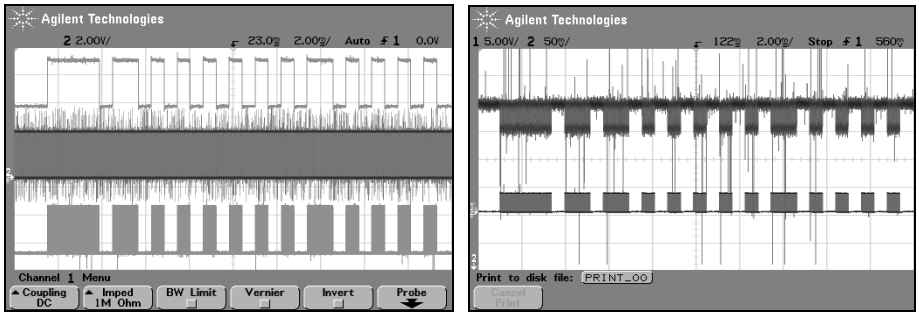


Figura 5.9.- a) Trama, oscil·lador local, senyal modulada, b) Senyal després del circuit excitador, senyal modulada

5.3. Resultats de síntesi

En aquest apartat es mostren els resultats de síntesi del sistema tant per la implementació *hardware* com *software*. Pel què respecta als resultats de síntesi *hardware*, es dona els resultats en percentatges en quan a elements lògics, recursos, pins i nombre de bits de memòria utilitzats. Aquests resultats s'especifiquen tant per la FPGA de la plataforma ESIICv3, com per la FPGA del kit de desenvolupament Nios II.

En quan als resultats de síntesi *software,* es mostra el nombre de línies de codi tant pel software que executa el Nios II, com per l'entorn gràfic dissenyat. També es dona la quantitat de memòria RAM que ocupen respectivament dins del sistema.

5.3.1. Resultats de síntesi *hardware*

Per a realitzar aquesta etapa s'ha utilitzat l'eina de PC Quartus II, amb la que podem sintetitzar el sistema, descarregar-lo a la FPGA i comprovar el seu funcionament en temps real. Quant realitzem una compilació amb el Quartus II, aquest realitza una sèrie d'etapes:

- Anàlisi i síntesi
- Filtrat (Fitter)
- *Linkat* i ensamblat
- Anàlisi de temps

En aquesta compilació el Quartus II ens informa dels possibles errors i "*warnings*" que existeixen al sistema i al codi descrit en VHDL. Un cop depurats els errors, el següent pas és "bolcar" la configuració a la FPGA, aquí s'ha d'especificar quin tipus de configuració utilitzarem. En el nostre cas, per a la implementació *hardware*, s'ha utilitzat una connexió JTAG a través del port USB del PC i una FPGA del tipus *Cyclone I* model EP1C12Q240C8, i un cop s'ha validat la totalitat del sistema s'ha carregat la configuració a una memòria EEPROM externa. Paral·lelament es realitzarà la compilació *software* (apartat 5.3.2).

El Quartus II ens proporciona els resultats de sintetitzar tot el sistema en quan a àrea ocupada i recursos utilitzats a la FPGA. Val a dir que la FPGA utilitzada ja tenia una implementació establerta, aquesta ocupa gran part de la FPGA, per tant, per a poder representar els resultats de síntesi que corresponen a la implementació d'aquest projecte, s'ha usat el fitxer (.fit) que genera el Quartus II on s'especifica la ocupació i recursos utilitzats per a cada un dels blocs del projecte.

La figura 5.10 mostra els resultats de síntesi del sistema global a la FPGA de la ESIICv3, entre aquests, podem veure que la ocupació total de recursos lògics del sistema és del 13%, la qual cosa ens pot permetre introduir més mòduls *hardware* dedicats. En quan als pins d'entrada i sortida, s'ha utilitzat un 73% dels pins disponibles (16 pins d'entrada i 81 pins de sortida), i pel què respecta a la utilització dels bits de memòria, s'han usat 3.968 bits que representen un 2% dels bits de memòria totals de la FPGA.

```
+-----------------------------------------------------------------+
; Flow Summary                                                     ;
+-------------------------+---------------------------------------+
; Flow Status             ; Successful - Mon Feb 25 18:35:49 2008    ;
; Quartus II Version      ; 6.0 Build 178 04/27/2006 SJ Full Version ;
; Revision Name           ; bus2                                  ;
; Top-level Entity Name   ; bus2                                  ;
; Family                  ; Cyclone                               ;
; Device                  ; EP1C12Q240C8                          ;
; Timing Models           ; Final                                 ;
; Met timing requirements ; Yes                                   ;
; Total logic elements    ; 1,552 / 12,060 ( 13 % )               ;
; Total pins              ; 127 / 173 ( 73 % )                    ;
; Total virtual pins      ; 0                                     ;
; Total memory bits       ; 3,968 / 239,616 ( 2 % )               ;
; Total PLLs              ; 1 / 2 ( 50 % )                        ;
+-------------------------+---------------------------------------+
```

Figura 5.10.- Resultats de síntesi de la implementació *hardware* de la FPGA de la plataforma ESIICv3.

D'aquests 13% d'elements lògics utilitzats, el percentatge de LEs que corresponen al sistema de comunicació implementat en aquest projecte és del 3,4%. Aquest percentatge inclou els mòduls descrits en el desenvolupament *hardware* i les portes i recursos necessaris per implementar el sistema.

Pel que fa a la síntesi del sistema Nios II - Avalon localitzat a la FPGA del kit de desenvolupament Nios II d'Altera, a la figura 5.11 es mostren els resultats de síntesi obtinguts de compilar amb el Quartus II el sistema de la FPGA (tipus *Cyclone II* model EP2C35F672C6) on està ubicat el processador Nios II. Aquesta implementació utilitza un nombre d'elements lògics i recursos superiors que els de la configuració de la *Cyclone I*. Això és degut a que en aquesta FPGA està implementat el microprocessador Nios II.

```
+-----------------------------------------------------------------------+
; Flow Summary                                                          ;
+-------------------------+---------------------------------------------+
; Flow Status             ; Successful - Feb 25 18:35:49 2008            ;
; Quartus II Version      ; 6.0 Build 178 04/27/2006 SJ Full Version     ;
; Revision Name           ; PanelSaverCheckv3                           ;
; Top-level Entity Name   ; full_featured                               ;
; Family                  ; Cyclone II                                  ;
; Device                  ; EP2C35F672C6                                ;
; Timing Models           ; Final                                       ;
; Met timing requirements ; Yes                                         ;
; Total logic elements    ; 14,922 / 33,216 ( 45 % )                    ;
; Total registers         ; 7979                                        ;
; Total pins              ; 315 / 475 ( 66 % )                          ;
; Total virtual pins      ; 0                                           ;
; Total memory bits       ; 172,800 / 483,840 ( 36 % )                  ;
; Embedded Multiplier 9-bit elements ; 4 / 70 ( 6 % )                   ;
; Total PLLs                         ; 2 / 4 ( 50 % )                  ;
+-------------------------+---------------------------------------------+
```

Figura 5.11.- Resultats de síntesi de la FPGA del kit de desenvolupament Nios II d'Altera, edició *Cyclone II*.

De la figura 5.11 en podem extreure que el percentatge total d'elements lògics utilitzats és d'un 45%, s'han utilitzat un 66% dels pins de la FPGA, s'utilitzen un total de 7.979 registres, i el nombre de bits de memòria és del 36%.

5.3.2. Resultats de síntesi *software*

Per a realitzar aquesta etapa s'ha utilitzat el *software Nios II IDE* que ens permet executar i depurar el sistema dissenyat, sobre el mateix compilador. Un cop depurats els errors, com a síntesi *software* podem extreure diversos resultats. El *software* que executa el Nios II, només s'especifica els paràmetres relacionats amb la funció *sircs.c*

- 160 Línies de codi.
- 9 KB de memòria.

Els resultats corresponents a la funció *SetSircsESIIC* de la llibreria de funcions són:

- 145 línies de codi.
- 8 KB de memòria.

Finalment la síntesi de la interfície gràfica a nivell d'usuari es correspon a:

- 489 línies de codi.
- 61 KB de memòria.

5.4. Validació a nivell de sistema

Per a comprovar el sistema, recordem que tenim per una banda la FPGA de la plataforma ESIIC amb la implementació hardware que actua com a *slave*, i per altre banda, tenim el Nios II de la FPGA del kit de desenvolupament d'Altera que fa de Master. Ambdós entitats estan connectades entre si a través dels pins d'entrada i sortida (PIO) de les respectives plataformes de prototipatge.

Un cop el sistema es va validar amb les captures amb l'analitzador lògic i l'oscil·loscop es va procedir a realitzar les proves de test directament amb un televisor. Amb l'ajuda de la interfase gràfica dissenyada és va verificar que l'aparell de TV respongués correctament a les comandes executades. També es van realitzar diferents proves per diferents valors del potenciòmetre que hi ha connectat a la sortida modulada, d'aquesta manera podem simular l'efecte de la distància sobre la transmissió de les trames.

El temps mínim perquè una transmissió sigui interpretada correctament pel televisor és equivalent a l'enviament de tres trames, és a dir, 135ms (45ms * 3).

Capítol 6.
Conclusions

6.1. Conclusions.

Un cop finalitzat aquest Projecte Final de Carrera podem dir que s'han assolit els objectius i que el disseny desenvolupat funciona correctament.

S'ha realitzat sobre un arquitectura flexible Nios II; un soft-IP amb VHDL basat en l'estàndard *SIRCS*, una interfície de comunicació entre els busos Avalon i WishBone, un programa de control i reconeixement a executar pel Nios II i finalment s'ha implementat un programa per a la llibreria de funcions amb la qual s'ha desenvolupat un entorn gràfic a nivell d'usuari.

La implementació del sistema s'ha simulat, sintetitzat i comprovat el seu funcionament en temps real i avui en dia forma part del sistema de test per a circuits impresos de l'empresa Sony Bcn Tec.

La incorporació d'aquest tipus d'aplicació, a l'aparell de test automàtic "*VITAM"*, ha aportat un grau de control addicional a la producció de circuits integrats. En el camp de la producció *SMD,* poder determinar els components i àrees afectades de les plaques defectuoses, és un factor de vital importància per tal d'obtenir un control de qualitat elevat. Amb aquest projecte s'ha contribuït a la automatització d'aquest tipus de control i ha proporcionat l'avantatge de localitzar les plaques amb problemes de comunicació per infraroig. Aquest avantatge permet realitzar un estudi dels components i circuits afectats i aplicar les mesures adients per pal·liar aquests problemes. D'altra banda, aquesta implementació a eliminat totalment l'aparició de circuits impresos amb problemes de comunicació per infraroig en les etapes de muntatge posteriors a la producció automàtica, disminuint així el percentatge de qualitat extern del departament

Per concloure podem dir que l'aplicació final d'aquest projecte dins d'un entorn de caracterització amb FPGA, només té sentit dins del marc de la verificació i test per plaques de

televisors, ja que la implementació d'un comandament a distància es podria implementar amb un microprocessador convencional.

6.2. Experiència personal i professional.

El desenvolupament d'aquest projecte m'ha permès ampliar els meus coneixements tan teòrics com pràctics respecte a aspectes molt variats.

Des de un principi en la realització d'aquest projecte vaig tenir la oportunitat de complementar els coneixements obtinguts durant el transcurs de la carrera. Durant les primeres etapes d'aquest projecte en les que vaig dedicar-me a la recerca d'informació, vaig poder aprofundir en els mètodes emprats en el desenvolupament de sistemes digitals, ampliant i refrescant conceptes sobre *IP-Cores* i *SoC's*, passant després per les diferents arquitectures dels busos *on-Chip* utilitzats en aquest projecte, i les seves similituds. També he tingut la oportunitat de treballar amb sistemes de comunicació per control remot, que actualment aporten un gran ventall de possibilitats.

Durant l'etapa de desenvolupament vaig intensificar els meus conceptes sobre llenguatge de descripció de *hardware*. Alhora que anava aprenent a implementant *hardware* en VHDL, vaig aprendre a fer servir les eines d'ajuda i de compilació. La utilització del ModelSim m'ha ajudat a trobar, quan el sistema no funcionava, les pistes suficients per saber on estava l'error i poder trobar una solució. A part d'aquest programa en el desenvolupament *hardware* també vaig usar el Quartus II i les eines que incorpora, com són el *SOPC Builder* i el *SignalTap.* Per al desenvolupament *Software* es va usar el *Nios II IDE,* que no és més que l'entorn de desenvolupament *Eclipse* adaptat pel desenvolupament de *Software* per al microprocessador Nios II d'Altera.

A nivell professional, la realització d'aquest projecte conjuntament amb l'empresa Sony Bcn Tec, m'ha aportat la possibilitat de poder participar en un projecte a nivell empresarial, i poder treballar al costat de professionals que porten anys treballant en el ram de l'electrònica, dels quals he pogut aprendre moltes coses i he passat molts bons moments i alguns de no tant bons.

6.3. Evolució futura.

L'evolució futura d'aquesta aplicació en concret no té gaires besants, s'ha implementat la totalitat de comandes possibles que pot efectuar un comandament a distància, deixant en forma de fitxer Excel les codificacions actuals establertes per a cada funció, de manera que encara que canviïn les codificacions de les comandes, no es tindrà que tocar cap part de la implementació, només el fitxer Excel.

L'evolució futura del sistema global de verificació de plaques és constant. A mesura que van apareixen nous punts crítics en la producció de plaques, s'implementa dins del sistema un nou *hardware* i una nova tasca per tal de comprovar el defecte en qüestió i pal·liar el problema.

Bibliografia i referències

Documentació d'Altera

Bus Avalon

[1] - *Avalon Bus Specification Reference Manual* version 2.3. Altera Co, California, July 2003.

Nios II

[2] - *Nios II Hardware development tutorial*. Altera Co, California, January 2005.

[3] - *Nios II Processor Reference Handbook*, ver. 5.0 Altera Co, California, May 2005.

[4] - *Custom Instructionsfor the Nios Embedded Processor*, ver. 1.2 Altera Co, September 2002.

Quartus II

[5] **-** *Introduction to Quartus II Manual* version 5.0. Altera Co, California, April 2005.

[6] **-** *Quartus II Development Software Handbook (Complet Four-Volume Set)* version 5.0. Altera Co, California, May 2005.

SOPC Builder

[7] - *SOPC Builder User Guide* version 1.0. Altera Co, California, June 2003.

Llenguatges de programació

VHDL

[8] - Dissenyo de Sistema digitales con VHDL, Serafin Alfonso Pérez, Enrique Soto y Santiago Fernandez. Editorial Thomson 2002.

[9] - Apunts de l'assignatura de Disseny Microelectrònic II. *UAB*

C - (MircoC/OS – CVI/LabWindows)

[10] - *µC/OS-View and the Nios II Softcore Processor (User's Manual)* Micrium 2004.

[11] – *LabWindows/CVI User Manual.* National instruments Corporation, February 1998 Edition

Estàndard de control remot per infraroig

[12] - SS-00116 - SIRCS *Sony Infrared remot control sistems*. Revised 2007, Sony Corp. Tokyo.

WISHBONE

[13] - *WISHBONE System-on-Chip (SoC) Interconnection Architecture for portable IP Cores*. Silicore and OpenCores.org revision b.3, September 2002

Altres

[14] - D.M. Pozar, *Microwave Enginering.* Addison-wesley, 1993

[15] - Malvino. Principios de Electronica. Mc. Graw Hill (Espanya), 2000

Pàgines Web

[16] - Altera , http://www.altera.com

[17] - OpenCores.org, http://www.opencores.org

[18] - Nios Comunity Forum, http://niosforum.com