Màster Universitari en Enginyeria de Telecomunicació

Treball de Final de Màster

CONTROLADOR ELECTRÒNIC PROGRAMABLE PER VEHICLES DE MAQUINARIA PESADA

Sergi Gómez Areste

Abstract

- **Motivació**: La empresa Deep Sea Electronics PLC empren un nou projecte per desenvolupar un controlador electrònic programable per vehicles de maquinaria pesada que gestioni tots els sistemes hidràulics.

- **Problema:** Els fabricants de vehicles de maquinaria pesada són especialistes en el desenvolupament de components mecànics i no desenvolupen els elements electrònics. Aquest projecte desenvolupa el producte del subconjunt de processament digital que realitza el control de tots els elements hidràulics i neumàtics del vehicle.

- **Metodologia:** S'ha analitzat quina és la tecnologia mes adient per aquest producte, incloent tipus de tecnologies, marques i maquinari concret. S'ha seleccionat el maquinari, s'ha realitzat el disseny del sistema, la seva implementació i la validació de resultats.

- **Resultats:** El funcionament és satisfactori d'acord amb els requisits del producte. S'ha realitzat una validació del funcionament del sistema mitjançant la simulació de senyals d'entrada i la comprovació dels senyals de sortida

- **Conclusió:** El projecte ha conclòs amb el lliurament del producte tal i com s'havia planificat, incloent el codi font de programació, el projecte i les simulacions.

Resum

El projecte desenvolupat és un subconjunt d'un producte de control industrial per controlar els sistemes electrònics de vehicles de maquinaria pesada. L'aplicació del producte final és el control de les funcionalitats electròniques, hidràuliques i neumàtiques de vehicles com camions de bombers, excavadores, maquinària d'obres públiques, etc.

El subconjunt desenvolupat en aquest projecte és el processament digital s'encarrega de la majoria d'entrades i sortides digitals del producte final. Queda fora de l'abast d'aquest projecte el subconjunt del microcontrolador, les interfícies humanes (teclat, pantalla, etc), les comunicacions amb l'exterior (Ethernet, USB, etc) i les fonts d'alimentació, que son o bé reaprofitades d'altres dissenys o desenvolupades per separat en d'altres projectes.

Els requeriments d'aquest projecte son el desenvolupament del subconjunt de processament digital amb la màxima flexibilitat possible per poder ser reutilitzat en d'altres productes, permetent la seva actualització de funcionalitats en un futur i minimitzant el cost del desenvolupament. Les funcionalitats requerides principalment son:

- Es comunica amb el bloc microcontrolador mitjançant un bus SPI bidireccional
- Ofereix 4 ports de 16 bits de senyals digitals de sortida configurables des del microcontrolador a través del bus SPI i un protocol de comunicació ja definit previ a aquest projecte
- Ofereix 4 ports de 16 *bits* de senyals digitals d'entrada configurables des del microcontrolador a través del bus SPI i un protocol de comunicació ja definit previ a aquest projecte
- Ofereix 8 senyals de sortida digitals de PWM configurables des del microcontrolador a través del bus SPI i un protocol de comunicació ja definit previ a aquest projecte

El projecte comença analitzant les tecnologies existents en el mercat, des del desenvolupament d'un maquinari específic no configurable fins a les tecnologies reprogramables com CPLDs, PSoC o FPGA, analitzant quines son les característiques específiques de cadascuna d'elles i el seu encaix amb les especificacions del projecte. Un cop seleccionada la tecnologia més adequada, la FPGA, s'ha entrat en detall en l'anàlisi dels diferents fabricants existents actualment en el mercat, així com les famílies més significatives de cadascun d'ells per acabar seleccionant el fabricant i la peça en concret que s'ha utilitzat en aquest projecte. Per arribar a aquestes conclusions s'ha tingut en compte també les eines de desenvolupament ofertes per cada fabricant.

Un cop seleccionat el maquinari adequat pel producte, el següent pas d'aquest projecte és el disseny dels components que hauran de realitzar les funcionalitats requerides. En aquesta etapa de disseny s'ha analitzat quina és la millor estructura de desenvolupament del producte dividint-lo en unitats bàsiques i independents que realitzin les funcions de forma autònoma, independitzant els processos de comunicació, de descodificació, i de

gestió independent de les funcions. El disseny del producte s'ha realitzat dividint el sistema en els següents blocs: bloc de rellotge, bloc d'entrades, bloc de sortides, bloc SPI, bloc PWM, bloc de selecció i bloc multiplexor

La següent etapa del projecte ha estat la implementació dels blocs descrit en l'etapa de disseny del producte. Pel bloc de rellotge s'ha utilitzat un bloc IP pre-definit per Xilinx, un *core* MMCM. La resta de blocs s'han implementat en codi VHDL específicament per aquest projecte. En el cas dels blocs d'entrada, de sortida i de PWM, al ser necessaris múltiples blocs iguals (8 per les entrades, 8 per les sortides i 8 pels PWM) s'han implementat només un cop i s'han instanciat 8 cops mitjançant bucles de instanciació per facilitar la lectura del codi, el seu manteniment i la seva simplicitat.

Al finalitzar la implementació s'ha desenvolupat un protocol de proves per la validació del funcionament del producte. La validació s'ha realitzat mitjançant un codi en VHDL de test que genera senyals d'entrada del sistema i analitza els senyals de sortida. Per validar el màxim nombre de casos s'han generat senyals que ofereixin les màximes garanties de validació, com generar senyals als pins d'entrada diferents per garantir que la seva lectura es realitza correctament. En el cas dels senyals de PWM s'han definit diferents períodes i valors de PWM per la seva validació.

1. Índex

2. Índex de figures

3. Índex de taules

4. Glossari

ARM: *Advanced RISC Machine*. Processador d'arquitectura RISC llicenciat a molts fabricants per fabricar circuits integrats de microprocessadors

Bitstream: Seqüència de *bits*. En aquest cas referida al conjunt de dades que configuren una dispositiu programable

CPLD: *Complex Programmable Logic Device*. Dispositiu electrònic programable basat en blocs lògics amb memòria no volàtil

DVB: *Digital Video Broadcast*. Estàndard de difusió del sistema de video digital adoptat per la EU (ETSI).

ECU: *Engine Control Unit*. És una unitat de control electrònic dels motors d'explosió per garantir la seva seguretat i gestionar el seu rendiment

EMC: *Electromagnetic Compatibility*. Capacitat d'un equipament a ser immune a les emissions existents en un ambient determinat de treball. La legislació europea legisla aquests límits a traves de la recomanació EN61000

EMI: *Electromagnetic Interference*. Quantitat de senyals emeses per un dispositiu al seu entorn. La legislació europea legisla aquests límits a traves de la recomanació EN61000

DLL: *Delay-locked up*. Es un rellotge digital en llac tancat controlat per una línia de retard. Molt freqüent en circuits de lògica programable

DSP: *Digital Signal Processing*. Processador digital programable mitjançant llenguatge de programació molt adequat per realitzar funcions matemàtiques complexes en molt poc temps i en paral·lel.

FFT: *Fast Fourier Transform*. Mecanisme computacional senzill per calcular transformades de fourier en dispositius electrònics

FPGA: *Field Programmable Gate Array*. Dispositiu electrònic configurable format per elements lògics digitals.

IC: *Integrated circuit*. Dispositiu electrònic que integra diferents elements basics per conformar un bloc funcional, analògic o digital

LSB: *Least Significant Bit*. Significa la part de menys pes específic a dins una variable binaria

LVDS: *Low-voltage differential signalling*. Estàndard que defineix les característiques elèctriques de un bus de comunicació diferencial de comunicacions sèrie

MPEG: *Moving Picture Experts Group*. Estàndard que cobreix la estructura de les compressions de video i àudio digitals utilitzada internacionalment per la radiodifusió (ISO/IEC).

MSB: *Most Significant Bit*. Significa la part de mes pes específic a dins una variable binaria

NV: *Non-volatile*. Característica de les memòries per retenir el seu contingut encara que no estigui alimentades

PCB: *Printed circuit board*. Es un suport físic format per capes de coure i substrats on es munten components electrònics per formar un conjunt funcional.

PsOC: *Programmable System-on-Chip*. Circuit integrat que es composa per un microcontrolador i d'altres funcionalitats analògiques o digitals fabricades en el mateix component.

PWM: *Pulse-width modulation*. Tecnologia per codificar un missatge en l'amplada d'un pols periòdic. Utilitzada normalment per conversos digital a analògic a traves de circuits integradors posteriors

RTOS: *Real Time Operating system*. Es un sistema operatiu per microcontroladors o microprocessadors que té la característica d'oferir control al programador sobre el moment quan s'executen les funcions.

SPI: *Serial Peripheral Interface*. Bus de comunicació asíncron sèrie.

SRAM: *Static random-access memory*. Dispositiu de memòria volàtil format per bascules que no necessiten refresc per mantenir les dades.

VHDL: *VHSIC Hardware Description Language*. Llenguatge de descripció de maquinari utilitzat en la programació de lògica programable

5. Introducció

5.1. Necessitats del producte

El subconjunt desenvolupat en aquest projecte és una part d'un producte controlador de maquinaria pesada orientada a aplicacions industrials mòbils, com maquinaria específica d'obres públiques, maquinaria agrícola, vehicles especials del sector militar o emergències civils. Els fabricants d'aquests vehicles son especialistes en el disseny i la industrialització dels elements mecànics però no disposen (i no els interessa) el desenvolupament dels elements electrònics. Aquests situació propicia que les empreses orientades al disseny i la industrialització de equipament electrònic tinguin una oportunitat de negoci en aquest sector que d'altra banda no seria accessible. L'objectiu del desenvolupament d'aquest tipus de producte és la venda als fabricants de vehicles per que integrin l'equipament electrònic en els seus vehicles de sèrie, com un component més del seu producte. Aquesta situació ja es va iniciar amb els controladors ECU que governen actualment els motors, i que normalment no son dissenyats pel fabricant del vehicle ni del motor, ja que ambdós estan centrats en el desenvolupament mecànic i la integració.

5.2. Descripció del producte

El producte per cobrir les necessitats del mercat descrits anteriorment és un controlador que governa tota la mecànica relacionada amb el vehicle, exceptuant el control del motor del vehicle que se n'ocupa la ECU. La resta de mecànica relacionada amb el vehicle, depenent del tipus d'aplicació, pot ser molt variada. Per exemple en un vehicle de bombers pot incloure totes les bombes de bombeig d'aigua, tant de càrrega de dipòsits com de sortida d'aigua de les mànegues, els motors hidràulics de control de les escales, els motors elèctrics de remolc, etc. En les aplicacions agrícoles poden controlar tot tipus de hidràulics, gestió del elements de rec, etc.
Aquests equips electrònics necessiten una interfície humana per la seva configuració i control de forma gràfica. És per això que el producte incorpora una pantalla i un teclat de control amb un sistema operatiu amb entorn gràfic (Linux). Això és degut a la facilitat del desenvolupament en Linux per integrar una interfície gràfica i d'usuari atractiva i ràpida de desenvolupar, a més de la garantir la qualitat del sistema operatiu. El desenvolupament de la part del microprocessador i interfície humana queda fora de l'abast d'aquest projecte.

5.3. Descripció del subconjunt desenvolupat en aquest projecte

Aquest projecte cobreix tota l'electrònica i la programació dels sistemes de control electrònic d'entrades i sortides que ofereix el producte i que son governades per la part de control (microprocessador amb interfície humana). El microcontrolador governa la electrònica de control comunicant-se amb la electrònica mitjançant un bus de comunicació que determina quins senyals s'enviaran de sortida i es capturaran d'entrada

5.4. Motiu del projecte

El sistema de control (microprocessador) s'ha dissenyat amb un sistema operatiu Linux que, com és conegut, no és un sistema operatiu en temps real. Per aquest motiu és necessari el desenvolupament del subconjunt que s'ha realitzat en aquest projecte de control industrial dels sistemes mecànics del vehicle. En un sistema on no calgués un OS en Linux es podria haver integrat molta d'aquesta funcionalitat en el propi microprocessador, realitzant les tasques d'entrada i sortida de senyals així com la generació de PWM i PWMi. Existeix una gran varietat de microprocessadors que inclouen aquestes funcionalitats de forma nativa i podrien ser utilitzades si el microprocessador s'hagués programat utilitzant un RTOS o sense sistema operatiu. Però per necessitats de mercat l'empresa ha considerat que el desenvolupament més atractiu per introduir el producte al mercat és la utilització d'un sistema operatiu gràfic, el que ha forçat la necessitat del desenvolupament d'aquest subconjunt de control utilitzant un maquinari dedicat.

6. Estat de l'art o de la qüestió

Existeixen diferents tecnologies de realització de les funcions digitals descrites per aquest projecte. A continuació es presenten les diferents possibilitats per realitzar les funcions requerides, tenint en compte que la utilització del propi microprocessador ha estat descartada degut a l'ús del sistema. Les possibilitats son desenvolupar un sistema digital no configurable per programari o un de configurable (FPGAs, CPLDs o PSoC).

6.1. Sistema no configurable

Consisteix en realitzar un desenvolupament amb electrònica digital discreta que permeti la realització de les funcions requerides pel projecte. Aquesta solució és probablement la més econòmica de totes les que s'exposaran en aquest apartat, ja que els circuits electrònics digitals son ja molt madurs al mercat i hi ha una ferotge competència entre fabricants. Els principals inconvenients de l'ús d'aquesta tecnologia son:

- La superfície que ocupen en un PCB (molt major que un sol IC de funció específica, programable o no)
- La possibilitat d'un error de disseny
- La major sensibilitat a interferències i tendència a emissions (major exposició a les proves EMC/EMI) degut al major nombre de pistes de PCB i de longituds elevades.
- La disminució de la freqüència de treball degut a la longitud de les pistes de PCB.
- La incapacitat de realitzar una funció alternativa o una millora de característiques al no ser reprogramable.

6.2. CPLDs

Les CPLDs ofereixen una solució força bona a molts dissenys digitals com a alternativa a un sistema no configurable. Solucionen tots els inconvenients dels anteriors, oferint un sistema reconfigurable amb una superfície molt petita, capaç de treballar a freqüències elevades i a un cost major que els sistemes no configurables però força raonable. Tenen també l'avantatge de disposar de flash interna, el que pot ser un valor per a sistemes crítics on es vulgui protegir el sistema al copiat per empreses de la competència. Tot i que el sistema no és infal·lible disposa de una certa seguretat. L'únic desavantatge que té l'ús de les CPLDs és la seva limitació de elements lògics o *flip-flops*: no poden realitzar funcions molt extenses on es necessiti una capacitat de processament elevada, però és adequada per capacitats de processament reduïdes fins i tot en altes freqüències

6.3. FPGAs

Les FPGAs son circuits de lògica programable que milloren les prestacions de les CPLDs. Gaudeixen de totes les seves avantatges exceptuant la forma de allotjament del codi. Mentre que les CPLDs allotgen el codi en flash interna, les FPGA més comunes carreguen el codi des d'una Flash (interna o externa) a una SRAM o bé es programen en tecnologia

antifusible. Són els elements de processament digital més àmpliament utilitzats en l'actualitat.

Degut al creixement elevat de l'ús de FPGAs en molts mercats diferents, aquesta tecnologia ha experimentat un creixement en prestacions molt gran. Algunes de les característiques que s'han anat afegint respecte a les CPLDs en els últims anys son les següents:

- Addicció de pins de distribució de rellotge de velocitats elevades.
- Blocs DLL interns de multiplicació de rellotges per assolir velocitats de processament internes de l'ordre de Gbps.
- Ports d'entrada i sortida configurables en diverses tecnologies i estàndards, amb tensions de funcionament separades que poden anar des d'1.2V a 5V.
- Addició de ports diferencials per estàndards com LVDS
- Ports de transceptors d'alta velocitat per processament de senyals d'enllaços òptics.
- Blocs de processament DSP per funcions matemàtiques avançades (multiplicacions, FFTs, filtres avançats, etc)

A continuació s'expliquen les característiques de les dues tecnologies més habituals utilitzades de FPGAs.

6.3.1. FPGAs SRAM

El mercat principal de FPGAs es basa en la tecnologia SRAM. Consisteix en una estructura interna que es configura mitjançant la descàrrega de un fitxer de configuració a les cel·les internes de la FPGA que son de memòria volàtil. Això significa que cada cop que la FPGA es posa en marxa esta desconfigurada i s'ha de configurar. Aquest procés es pot fer de diverses maneres:

- Configuració mitjançant una memòria estàtica externa (Flash, PROM, EEPROM, etc)
- Configuració mitjançant una memòria estàtica interna
- Configuració a través de la descàrrega del fitxer per un processador extern (o intern en algunes peces).

Aquesta tecnologia de fabricació de FPGAs té una amplia acceptació en la majoria de mercats. No obstant això, existeixen alguns mercats on té un ús limitat o inexistent, com per exemple aplicacions militars, d'aeronàutica o d'espai. Els van de de la seva dependència de elements externs per que es configurin (en espai per exemple no es pot permetre que una FPGA no desenvolupi la seva funció si la Flash externa falla) fins a la seguretat de la càrrega del bitstream de forma externa en aplicacions militars.

6.3.2. Antifusible

La tecnologia antifusible té la principal característica que només es programa un sol cop i el codi enviat es queda gravat permanentment a la FPGA. Internament consta de unes connexions d'alta impedància i quan es grava es formen curtcircuits interns que formen les connexions.

Aquesta tecnologia és més cara que la tecnologia SRAM però ofereix més seguretat a sectors determinats. Per exemple en aplicacions militars no li cal un element extern que li transmeti el codi (que pot ser interceptat per un analitzador lògic, per exemple) ja que està internament codificat. Evidentment la seguretat absoluta no existeix i amb un microscopi i aconseguint obrir la peça es podrien descodificar les connexions que s'han fos i les que no, deduint quina és la seva funcionalitat, però cal un esforç molt més gran que per descodificar el codi d'una FPGA SRAM.

6.4. PSoC

Son sistemes combinats de diferents tipus de circuits electrònics. Existeixen alguns que combinen circuits analògics i digitals. Alguns d'altres combinin funcionalitats específiques d'algun mercat especial, com per exemple descodificadors MPEG per aplicacions de DVB amb microcontrolador integrat. N'hi ha d'altres que inclouen un microprocessador i una FPGA integrats en un sol xip, que son el tipus de PSoC que ens interessa avaluar per la nostra aplicació. Tenen l'avantatge de reducció de costos per la seva integració (son més barats que el mateix processador i la mateixa FPGA comprades per separat)

6.5. Anàlisis

Inicialment qualsevol de les tecnologies anterior seria acceptable per realitzar aquest disseny. Per analitzar la tecnologia més adequada és necessari descriure quines son les necessitats d'aquest disseny:

El producte ja disposa d'un microcontrolador extern que funcionarà amb OS Linux, realitzant la interfície d'usuari

- Ha de ser reconfigurable per afegir funcionalitats, a poder ser dinàmicament
- Ha de tenir un cost el mes baix possible
- Ha de ser reutilitzable per ser usat en d'altres productes futurs de la mateixa gama.

La primera condició ja elimina la possibilitat de l'ús d'un PSoC. Es un tipus de dispositiu que està incrementant la seva presencia al mercat i, en d'altres condicions probablement seria l'escollit en aquest disseny. La realitat d'aquest producte que s'està desenvolupant es que l'empresa ja disposa d'un desenvolupament de referencia amb un microcontrolador dual *core* ARM A9 que s'utilitza en d'altres productes, i per economies d'escala s'ha decidit el seu us en aquest nou producte. Això ja descarta la solució de PSoC ja que no tindria sentit la tria d'un segon microcontrolador per realitzar les funcions. Té més sentit l'ús del microcontrolador existent i decidint un maquinari extern per realitzar les funcions del projecte.

La segona condició elimina la possibilitat de l'ús d'un sistema no configurable. Tot i que un sistema tradicional d'electrònica digital compliria amb les funcions necessàries, i a més seria probablement la solució més econòmica, es requereix d'un sistema configurable per poder realitzar actualitzacions amb noves prestacions, i paral·lelament es desenvolupa un sistema que pot ser reutilitzat en d'altres dissenys, complint així el quart requeriment del disseny

Com a conseqüència del tipus de requeriment i de la tercera condició, hauríem de descartar l'ús de la tecnologia CPLD. Si bé es cert que podria ser adequat per aquest disseny, probablement afegiríem limitacions de capacitat per revisions futures (tot i que encara no sabem si la nostra aplicació hi cabria en una CPLD, podria ser que les prestacions futures no hi cabessin). A més es requereix, si és possible, la seva reconfiguració dinàmicament, el que vol dir que el microcontrolador hauria d'allotjar el bitstream del dispositiu programable i podria ser actualitzar remotament. En una CPLD això és molt complicat, tot i ser possible, caldria un gravador de CPLDs al PCB.

El disseny queda reduït a l'ús d'una FPGA, però encara seria necessari la tria de la tecnologia a utilitzar: SRAM o antifusible.

La tecnologia antifusible és més cara que la SRAM, i només aporta com a benefici la més alta protecció davant copies al no necessitar d'un bolcat de dades des de la memòria NV al dispositiu SRAM. En aplicacions on es desenvolupi una tecnologia innovadora i s'hagi invertit un pressupost molt elevat en aconseguir els algoritmes és molt important implementar mecanismes anti-còpia. Com aquest requeriment no es imprescindible en aquesta aplicació l'increment de preu degut a la tecnologia antifusible no esta justificada.

Per tot l'exposat anteriorment es selecciona la tecnologia FPGA SRAM.

7. Solució tecnològica

7.1. Fabricants de FPGA

Existeixen molts fabricants de FPGAs al mercat però ens hem centrat en les més populars. Entre els quatre fabricants analitzats a continuació s'emporten una quota de mercat superior al 95%.

7.2. Altera[i]

Un dels fabricants mes grans de FPGAs emportant-se aproximadament un 40% de la quota de mercat. Dissenya FPGAs i CPLDs però es fabriquen externament (TSMC). Ha estat recentment comprada per Intel

7.2.1. Xilinx[ii]

Es el primer fabricant mundial de FPGAs amb una quota de mercat aproximadament del 50%. Dissenya i fabrica FPGAs, CPLDs i PSOC que integren ARM A9 amb una FPGA.

7.2.2. Lattice[iii]

Dissenya CPLDs i FPGAs principalment pel mercat de les telecomunicacions. Utilitzen tecnologia SRAM amb Flash interna.

7.2.3. Microsemi[iv]

Abans anomenada Actel dissenya FPGAs i PSoCs principalment per mercats mes petits amb característiques especials com per exemple l'aviació o les aplicacions d'espai. Ofereix tecnologies SRAM i antifusible.

1.1. Entorns de desenvolupament

7.2.4. Quartus (Altera)[v]

Es un entorn de desenvolupament per les FPGAs d'Altera que ofereix diverses versions en funció de quines peces es volen utilitzar. La última versió disponible és la 16.0. Existeixen diverses versions, una d'elles gratuïta i altres de pagament. Totes les versions ofereixen la possibilitat de síntesis, implementació, simulació i programació de les FPGAs. Té versions per Windows i per Linux

La versió Lite Edition bàsicament cobreix les peces Cyclone, MAX i Arria II.

La versió Standard cobreix el mateix que la versió Lite més el suport de la família Arria 10 a part d'algunes altres prestacions avançades. Té un cost de $2995 (*fixed*) i $3995 (*float*)

La versió Professional cobreix el mateix que la versió Standard més alguna altra prestació com la optimització incremental $3995 (*fixed*) i $4995 (*float*)

7.2.5. ISE/Vivado (Xilinx)[vi]

L'entorn de desenvolupament de Xilinx s'ofereix amb diferents packs, un de gratuït i d'altres de pagament. La versió actual és la 2016.3. Té versions per Windows i per Linux

La versió Vivado HL Webpack és gratuïta inclou pràcticament totes les peces excepte la Virtex.

Les versions de pagament son la versió Vivado HL Design Edition, la versió Vivado HL System Edition i la versió Vivado HL Lab Edition. Els preus comencen als $2995.

7.2.6. Lattice Diamond (Lattice)[vii]

Disposa de llicencies gratuïtes i de pagament. Es poden descarregar per instal·lar en Windows i Linux.

Les versions gratuïtes disponibles son la Lattice Diamond Free License i la iCEcube Software Free License

Les versions de pagament son la iCEcube Software Free License i la ispLEVER Classic Software

7.2.7. Libero (Microsemi)[viii]

La plataforma Libero de Microsemi ofereix tres possibilitats de llicència: una gratuïta i dues de pagament.

La versió gratuïta s'anomena Gold i està limitada en el nombre de peces que suporta. Excepte per a la família RTG4, suporta algunes peces de cadascuna de les altres famílies, normalment les menys 'potents

Les versions de pagament son la Platinum i la Standalone.

Disposa de versions per Windows, Linux i Solaris.

7.2.8. Anàlisis

Totes les plataformes de desenvolupament ofereixen una versió gratuïta per poder realitzar desenvolupament en un nombre limitat de peces, que normalment exclouen les peces de les games més elevades, amb més prestacions o nombre d'entrades i sortides. A més

també ofereixen una prova (normalment d'un mes) per provar la versió completa del producte.

Totes integren les eines bàsiques per poder realitzar un disseny, com son el sintetitzador, el *place and route*, eines de simulació, editors de FPGA per posicionament manual, i alguns d'ells inclouen versions limitades de analitzadors lògics per realitzar anàlisis i depuració amb senyals reals.

També inclouen eines per descarregar el bitstream a la peça i gravar memòries de configuració tant internes com externes.

1.1. Famílies de FPGA

En aquest apartat es descriuen les principals famílies de les FPGAs dels quatre fabricants analitzats anteriorment. Disposen de més famílies de les mostrades a continuació, però son famílies que no s'adapten clarament a les necessitats del disseny o bé estan en procés d'obsolescència o ho estaran en breu al ser llençades noves versions de les mateixes peces pel fabricant (Spartan 6 vs Sparan 7, Cyclone V vs Cyclone IV, etc). Per no complicar l'anàlisi, les opcions que es descriuen son les més noves dins de cada família. Com la tecnologia de miniaturització va avançant cada cop més ràpidament i els preus dels circuits tenen una forta vinculació amb la superfície que ocupen, el preu d'un dispositiu determinat que es fabriqui en tecnologia de 45nm costarà pràcticament el doble que el mateix dispositiu fabricat en tecnologia de 28nm. Per aquest motiu s'han descartat les peces que ja tenen substitut en un a altra de nivell d'integració superior.

En l'anàlisi posterior també s'ha tingut en compte les característiques del disseny que es vol realitzar, de manera que s'ha analitzat només les característiques que estan relacionades amb els requeriments. En aquest context es mostra només una comparativa del nombre de lògica que inclou cada peça així com el rang de pins d'entrada i sortida dels que disposa. No s'ha entrat a valorar altres aspectes com el nombre de transceptors d'alta velocitat, ni el nombre de blocs DSP ni la versatilitat dels estàndards de *pins* diferencials que es poden configurar, ja que el disseny que es demana no té cap vinculació amb aquestes característiques de les FPGAs.

7.2.9. Altera:

- Stratix[ix]: La versió actual de la família és la Stratix 10. Es fabrica en tecnologia de 10nm. Disposa de models des de 344 I/O *pins* fins a 1640 pins, i des de 484 KLE fins a 5510 KLE
- Arria[x]: La versió actual és la Arria 10. Es fabrica en tecnologia de 10nm. Disposa de models des de 224 I/O *pins* fins a 992 pins, i des de 160 KLE fins a 1150 KLE
- Cyclone[xi]: La versió actual és la Cyclone V. Es fabrica en tecnologia de 28nm. Disposa de models des de 128 I/O *pins* fins a 560 pins, i des de 25 KLE fins a 301 KLE

7.2.10. Xilinx

- Virtex[xii]: La versió actual és la Virtex 7. Es fabrica en tecnologia de 28nm. Disposa de models des de 300 *pins* a 1100 pins i de 582720 a 876160 *Logic Cells*
- Kintex[xiii]: La versió actual és la Kintex 7. Es fabrica en tecnologia de 28nm. Disposa de models des de 185 *pins* a 500 pins i de 65600 a 477760 *Logic Cells*
- Artix[xiv]: La versió actual és la Artix 7. Es fabrica en tecnologia de 28nm. Disposa de models des de 106 pins a 500 *pins* i de 12800 a 215360 *Logic Cells*
- Spartan[xv]: La versió actual és la Spartan 7. Es fabrica en tecnologia de 28nm. Disposa de models des de 72 *pins* a 400 pins i de 6000 a 102400 *Logic Cells*

7.2.11. Lattice

- ECP[xvi]: Disposa de models des de 118 *pins* a 365 pins i de 24000 a 84000 LUTs
- iCE[xvii]: Disposa de models des de 640 a 3520 LUTs

7.2.12. Microsemi:

- IGLOO2[xviii]: Disposa de models des de 28 a 574 *pins* i des de 6060 a 146124 LE
- RTG4[xix]: Disposa d'un model de 720 *pins* i 151824 LE
- SmartFusion4[xx]: Disposa de models des de 28 a 574 *pins* i des de 6060 a 146124 LE

7.2.13. Anàlisis:

Lattice s'ha estat especialitzant en dispositius portàtils i també en equips de telecomunicacions. Aquesta estratègia, sobretot en el mercat dels dispositius mòbils, té com a conseqüència dos desavantatges importants per l'aplicació al mercat industrial. El primer és que la vida útil de les peces és menor ja que els mercats dels dispositius mòbils evolucionen molt ràpidament. El segon és que aquests mercats son molt demandants de volums elevats, i si bé això aporta un preu més baix de la peça, també marca una clara prioritat de la producció cap a aquells clients. Moltes de les empreses que utilitzen elements utilitzats en mercats de dispositius mòbils s'han trobat en ocasions en problemes de subministrament per que els clients principals dels fabricants acaparen tota la producció quan hi ha un pic de demanda.

Microsemi està especialitzada en desenvolupaments de PSoC i Mixed Signal. També s'orienta a mercats específics i els seus dispositius no pretenen atacar el mercat general.

Les dues marques que s'enduen el 90% del mercat al ser més genèriques, aplicació independent i amb un preus molt competitius son Altera i Xilinx. La seva alternança de predomini al mercat ha anat variant amb els anys, però últimament sembla que Xilinx està guanyant la cursa pel lideratge al mercat. No obstant això, aquesta posició de domini no ha influït en la decisió de l'ús d'un dispositiu o un altre per aquest projecte. Les prestacions de les peces de les dues famílies son molt semblants i qualsevol de les dues seria vàlida per a aquest projecte. Les famílies que serien més adequades son la Spartan 7 o Artix 7 si es selecciones Xilinx i la Cyclone V si es seleccionés Altera. El motiu és que dins el ventall de possibilitats de FPGAs de SRAM aquestes dues famílies son les més econòmiques i tenen

potència i prestacions molt sobrades per l'aplicació que es vol desenvolupar i per possibles actualitzacions futures. A més disposen d'un disseny que permet la compatibilitat *pin* a *pin* amb les següents peces de la mateixa família, de forma que si es necessités créixer no caldria canviar el disseny del PCB.

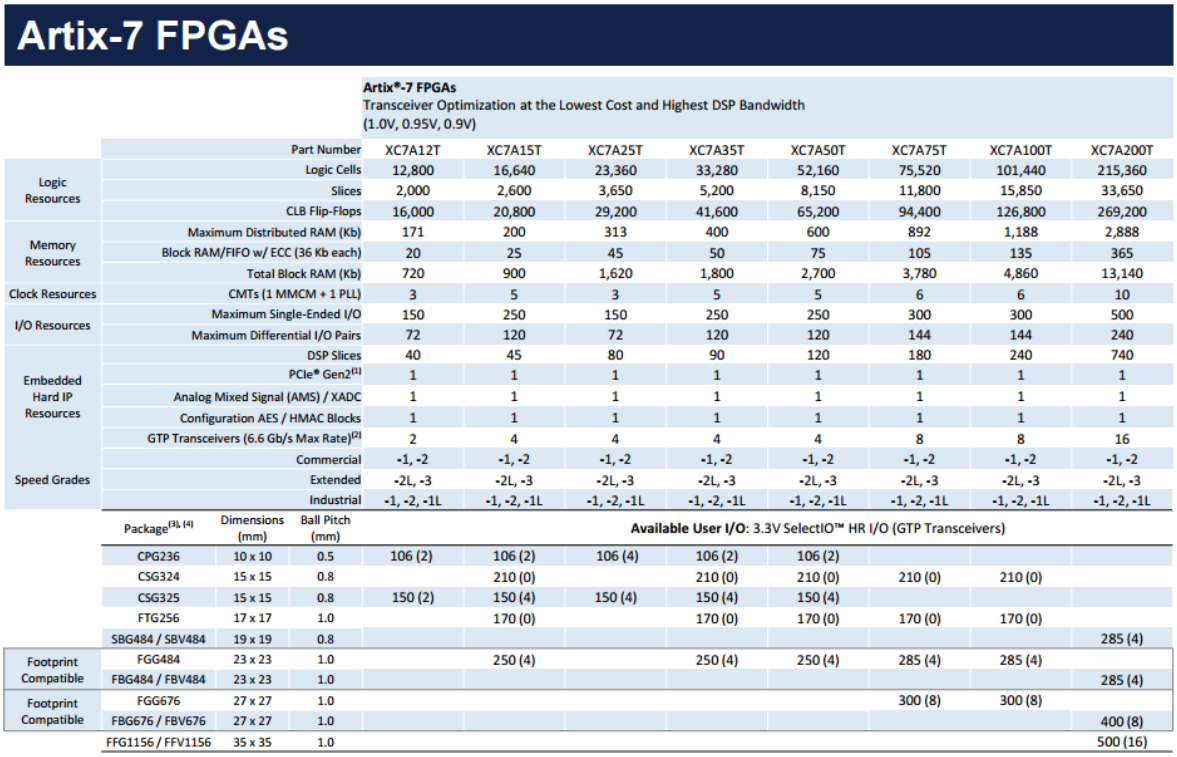
La decisió final de l'ús de Altera o Xilinx s'ha pres per motius externs a les prestacions o característiques de les famílies aptes per aquest disseny. Al ser una aplicació específica per a una empresa que fabrica volums elevats d'altres productes, la decisió final ha recaigut només en la tria de la peça representada pel distribuïdor que ofereix millor servei. El distribuïdor en concret representa Xilinx i ha demostrat al llarg dels anys un suport excel·lent en d'altres productes d'altres marques. Com les característiques i preus tant de Xilinx com d'Altera eren vàlids per aquest disseny la tria s'ha determinat per motius de qualitat del suport.

Un cop seleccionat el fabricant, s'ha procedit a analitzar quina és la família de productes i la peça especifica mes adequada per cobrir les necessitats del disseny. Els requeriments de ports d'entrada i sortida del producte és el següent: 64 ports d'entrada (16x4), 64 ports de sortida (16x4), 8 ports PWM, 4 ports de SPI, 1 de *reset* i 1 de rellotge, total 142 ports. Com cap dels altres requeriments (velocitat de rellotge, ports d'alta velocitat, blocs DSP, etc) no suposa cap restricció a l'hora de triar la peça, ja que totes son adequades, caldrà seleccionar la família i peça més barata que ofereixi 142 ports d'entrada i sortida. La família mes barata és la Spartan-7 i després la Artix-7. La peça mes petita de la família Spartan-7 (XC7S6) disposa nomes de 100 ports, i per arribar als 142 ports hauríem d'anar a la XC7S25. La peça d'Artix-7 que pot oferir 142 ports és la XC7A15T, més barata que l'anterior. Per tant la peça que cobreix les necessitats de ports d'entrada sortida mes barata que s'ha seleccionat per aquest disseny és la XC7A15T de la família Artix-7.

Spartan-7 FPGAs

		Spartan®-7 FPGAs I/O Optimization at the Lowest Cost and Highest Performance-per-Watt					
Part Number		**XC7S6**	**XC7S15**	**XC7S25**	**XC7S50**	**XC7S75**	**XC7S100**
Logic Cells		6,000	12,800	23,360	52,160	76,800	102,400
Slices		938	2,000	3,650	8,150	12,000	16,000
CLB Flip-Flops		7,500	16,000	29,200	65,200	96,000	128,000
Max. Distributed RAM (Kb)		70	150	313	600	832	1,100
Block RAM/FIFO w/ ECC (36 Kb each)		5	10	45	75	90	120
Total Block RAM (Kb)		180	360	1,620	2,700	3,240	4,320
Clock Mgmt Tiles (1 MMCM + 1 PLL)		2	2	3	5	8	8
Max. Single-Ended I/O Pins		100	100	150	250	400	400
Max. Differential I/O Pairs		48	48	72	120	192	192
DSP Slices		10	20	80	120	140	160
Analog Mixed Signal (AMS) / XADC		0	0	1	1	1	1
Configuration AES / HMAC Blocks		0	0	1	1	1	1
Commercial Speed Grade		-1,-2	-1,-2	-1,-2	-1,-2	-1,-2	-1,-2
Industrial Speed Grade		-1,-2,-1L	-1,-2,-1L	-1,-2,-1L	-1,-2,-1L	-1,-2,-1L	-1,-2,-1L
Package[1]	Body Area (mm)	**Available User I/O**: 3.3V SelectIO™ HR I/O					
CPGA196	8x8	100	100				
CSGA225	13x13	100	100	150			
CSGA324	15x15			150	210		
FTGB196	15x15	100	100	100	100		
FGGA484	23x23				250	338	338
FGGA676	27x27					400	400

Figura 1. Família Spartan-7[xxi]



Artix-7 FPGAs

Artix®-7 FPGAs
Transceiver Optimization at the Lowest Cost and Highest DSP Bandwidth
(1.0V, 0.95V, 0.9V)

	Part Number	XC7A12T	XC7A15T	XC7A25T	XC7A35T	XC7A50T	XC7A75T	XC7A100T	XC7A200T
Logic Resources	Logic Cells	12,800	16,640	23,360	33,280	52,160	75,520	101,440	215,360
	Slices	2,000	2,600	3,650	5,200	8,150	11,800	15,850	33,650
	CLB Flip-Flops	16,000	20,800	29,200	41,600	65,200	94,400	126,800	269,200
Memory Resources	Maximum Distributed RAM (Kb)	171	200	313	400	600	892	1,188	2,888
	Block RAM/FIFO w/ ECC (36 Kb each)	20	25	45	50	75	105	135	365
	Total Block RAM (Kb)	720	900	1,620	1,800	2,700	3,780	4,860	13,140
Clock Resources	CMTs (1 MMCM + 1 PLL)	3	5	3	5	5	6	6	10
I/O Resources	Maximum Single-Ended I/O	150	250	150	250	250	300	300	500
	Maximum Differential I/O Pairs	72	120	72	120	120	144	144	240
Embedded Hard IP Resources	DSP Slices	40	45	80	90	120	180	240	740
	PCIe® Gen2[1]	1	1	1	1	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1	1
	GTP Transceivers (6.6 Gb/s Max Rate)[2]	2	4	4	4	4	8	8	16
Speed Grades	Commercial	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L

	Package[3], [4]	Dimensions (mm)	Ball Pitch (mm)	Available User I/O: 3.3V SelectIO™ HR I/O (GTP Transceivers)							
	CPG236	10 x 10	0.5	106 (2)	106 (2)	106 (4)	106 (2)	106 (2)			
	CSG324	15 x 15	0.8		210 (0)		210 (0)	210 (0)	210 (0)	210 (0)	
	CSG325	15 x 15	0.8	150 (2)	150 (4)	150 (4)	150 (4)	150 (4)			
	FTG256	17 x 17	1.0		170 (0)		170 (0)	170 (0)	170 (0)	170 (0)	
	SBG484 / SBV484	19 x 19	0.8								285 (4)
Footprint Compatible	FGG484	23 x 23	1.0		250 (4)		250 (4)	250 (4)	285 (4)	285 (4)	
	FBG484 / FBV484	23 x 23	1.0								285 (4)
Footprint Compatible	FGG676	27 x 27	1.0						300 (8)	300 (8)	
	FBG676 / FBV676	27 x 27	1.0								400 (8)
	FFG1156 / FFV1156	35 x 35	1.0								500 (16)

Figura 2. Família Artix-7[xxii]

1.1. Llenguatges de descripció del maquinari

7.2.14. VHDL

El llenguatge de descripció de maquinari VHDL va esdevenir un estàndard de la IEEE (1076) al 1993. Deriva del llenguatge de programació ADA.

7.2.15. Verilog

Verilog va ser llençat al 1983 per Gateway, més tard comprat per Cadence que el va obrir al domini públic al 1989.

7.2.16. Comparativa VHDL vs Verilog

El llenguatge Verilog té una estructura similar a la programació en C. Com a avantatge presenta la facilitat d'aprenentatge per programadors de C (el llenguatge probablement més utilitzat en el sector de l'electrònica) per aprendre Verilog d'una forma ràpida. Aquest avantatge es presenta com un desavantatge per la seva naturalesa diferent del C: no es tracta d'un llenguatge de programació de programari si no d'un llenguatge descriptor de maquinari. Verilog és un llenguatge més flexible on és més fàcil cometre errors i no ser

identificats fins a la seva depuració. En canvi VHDL és un llenguatge molt més restrictiu i no permet gaire flexibilitat, generant errors molt freqüentment si no s'escriu amb perfecta precisió. Això li permet tenir un índex d'errors menors ja que qualsevol problema de codificació fa saltar un error de forma prematura i no cal arribar al procés de depurar per detectar molts dels errors. Existeix una variant de Verilog que s'anomena SystemVerilog que tot i aportar alguns avantatges respecte a Verilog no aconsegueix arribar a la fiabilitat del VHDL.

També cal destacar alguns problemes del Verilog en quant a portabilitat. L'ordre de compilació a Verilog és important i pot ocasionar diferencies de funcionament si es canvia l'ordre. També cal destacar que el VHDL permet la possibilitat de crear packages que es puguin reutilitzar en d'altres dissenys.

Els tipus de dades son més senzills en Verilog que en VHDL, molt més senzills d'utilitzar però en contrast també menys orientats al modelat de hardware.

Les funcions i mòduls en VHDL poden ser genèriques i reutilitzables per d'altres mòduls mentre que en Verilog son locals.

VHDL presenta restriccions molt fortes en el processament de tipus i dimensió dels senyals i variables mentre que Verilog és més laxa, i per tant és propens a errors ja que estem parlant de connexionat físic de senyals de hardware que son sensibles a aquestes característiques, per exemple la assignació de senyals o vectors de diferent dimensió. Mentre que això és permès en Verilog (com ho permetria un llenguatge de programació C), no és permès en VHDL ja que no es possible a nivell de hardware connectar dos busos amb diferent dimensió.

A Continuació es mostra una taula amb una comparativa de Verilog, VHDL i SystemVerilog[xxiii] de Mentor Graphics

	VHDL	Verilog (2001)	SystemVerilog
Strong typing	**Yes**	**No** • Bit • bit-vector • wire • reg) • unsigned • signed • integer • real • String in certain contexts only	**Partial** Not strongly typed in areas backward compatible with Verilog **Yes** Enhanced type system is strongly typed (but not as strong as VHDL)
User-defined types	**Yes**	**No**	**Yes**
Dynamic memory allocation (pointer types)	**Yes**	**No**	**Partial** Class objects can be dynamically created/destroyed, but via handles ("safe pointers")
Physical types	**Yes**	**No**	**No**
Named events	**No**	**Yes**	**Yes**
Enumerated types (FSM modeling)	**Yes**	**No**	**Yes**
Records/structs	**Yes**	**No**	**Yes**
Variant/unions	**No**	**No**	**Yes**
Associative/sparse arrays	**Partial** (But can be modeled using access types)	**No**	**Yes**
Class/inheritance	**No**	**No**	**Yes** (single inheritance)
Data packing	**No**	**No**	**Yes**
Bit (vector) / integer equivalence	**Partial** Not built-in but standard package supports	**Yes**	**Yes**
User defined signal/net resolution	**Yes**	**No**	**No**
Subprograms (procedural)	**Yes** Function & procedure always automatic	**Yes** Static and automatic functions and tasks	**Yes** Same as Verilog plus void functions (procedures)
Subprograms (concurrent) aka tasks	**Yes** Concurrent procedure calls	**Yes** Static tasks	**Yes** Static tasks
Methods	**No**	**No**	**Yes** (goes hand-in-hand with classes)
Separate packaging	**Yes** Packages	**Yes** Include files	**Yes** Include files

Taula 1. Comparativa de llenguatges descriptors de maquinari (1)

	VHDL	Verilog (2001)	SystemVerilog
Other hierarchy	**Yes** Separate entity / architecture (Interface / implementation)	**No**	**Yes** Programs, Clocking domains, Interfaces
All-read sensitivity	**No**	**Yes** @(*)	**Yes** Same as Verilog. Plus: always_comb
Reactive region processes	**Yes** Postponed processes	**No**	**Yes** Programs, Clocking domains, Final blocks
Dynamic process creation/deletion	**No**	**Yes** Fork/join. Block/task disable.	**Yes** Same as Verilog.
Conditional statements	**Yes** • If-then-else/elsif (priority) • Case (mux) • Selected assign (mux) • Conditional assign (priority) • No "don't care" matching capability	**Yes** • if-else (priority) • case (mux) • casex (mux) • ?: (conditional used in concurrent assignments)	**Yes** Same as Verilog. Adds priority and unique keywords to infer priority encoding/mux implementation
Iteration	**Yes** • Loop • while-loop • for-loop • exit • next Can name the loop to exit or continue with next	**Yes** • repeat • for • while	**Yes** Same as Verilog, Plus: • do-while • break • continue Only closest enclosing loop can be break or continue
Operators & expressions	**Yes** All expected: • arithmetic • logical • bit-wise • shift • concatenation Overloadable (polymorphism). No unary reduction. No logical scalar/vector.	**Yes** All expected: • arithmetic • logical • bit-wise • shift • concatenation • unary reduction • logical scalar/vector • case (in)equality. • conditional (?:) No rotate left/right	**Yes** Same as Verilog. Plus: • wild (in)equality • increment • decrement • assignment (+=, -=, |=, etc.) No rotate left/right
Gate level modeling	**Yes** VITAL. Very good FPGA library support.	**Yes** Builtin primitives. UDPs. Better availability of ASIC library support	**Yes** Same as Verilog. Except, library support yet to be qualified as vendors won't assume Verilog sign-off = SystemVerilog sign-off
Interface abstraction	**Partial** Component abstracts interface from specific module. Two layer binding allows flexibility in generic/port mapping.	**No**	**Yes** Interfaces are a separate construct in language. Supports multiple abstraction level and eases interface reuse. Can reduce coding.

Taula 2. Comparativa de llenguatges descriptors de maquinari (2)

	VHDL	Verilog (2001)	SystemVerilog
Configuration & Binding	**Yes** Control of instance or component binding to entity. Incremental (re)binding of generics and ports.	**Partial** **Control of module to instance binding.**	**Partial** Same as Verilog.
Conditional & iterative generation	**Yes** • If (conditional) • For (iterative)	**Yes** • **If** • **if-else (mutually exclusive)** • **case** • **for**	**Yes** Same as Verilog.
Attributes	**Yes** Attributes are typed. Attribute values can be specified. Attribute values can be referenced. Anything labeled with a name can be attributed. Groups allow attributes to relate two or more named entities in the design.	**Partial** **Not-typed.** **Can be placed virtually anywhere.** **What is attributed is determined by lexical proximity.** **Attribute values cannot be referenced.**	**Partial** Same as Verilog.
Verification targeted capabilities	**Partial** • Access types • Recursive subprograms • Extensive File I/O • Postponed processes • Standard package for random number generation	Limited • File I/O • Random number generation • Recursive subprograms • Fork/join	**Yes** Same as Verilog. Plus: • Random and constrained random value generation • Programs • Clocking domains • Associative arrays • Semaphores • Mailboxes • Classes
Assertions	**Partial** • Combinatorial (Boolean) assertions • User-defined severity and message control	**No**	**Yes** • Combinatorial and sequential (concurrent) assertions. • Sequence (temporal) expression. • Sequence-local variables. • User-defined severity and message control. • API extensions for assertions and coverage information for assertions
Foreign interfaces	Limited • Standard 'Foreign attribute • VhPI defined, but not yet standardized	**Yes** Standard C API (tf, acc, vpi)	**Yes** Same as Verilog. Plus: • Extensions to API for assertions and coverage • Direct C language interface

Taula 3. Comparativa de llenguatges descriptors de maquinari (3)

Tot i que ambdós llenguatges sèrie adequats per aquesta aplicació, per qüestions de garantir la qualitat i la portabilitat del disseny es selecciona el llenguatge de programació VHDL.

8. Disseny

8.1. Selecció la tecnologia

El sistema dissenyat requereix de les prestacions d'un disseny realitzat en maquinari per oferir una bona velocitat de processament i alt nombre de pins de sortida, i alhora també requereix una transportabilitat, flexibilitat i reprogramació que ofereix la lògica programable. Realitzant el disseny en maquinari suposaria perdre la resta de característiques realitzar-la en programari resultaria en una pèrdua de capacitat de velocitat i de nombre de pins de sortida. Per tant la tecnologia mes adequada per la realització d'aquest disseny és la lògica programable.

Dins de les opcions de la lògica programable, com s'ha analitzat en anteriors PACs, la mes adequada és la FPGA per la seva versatilitat, flexibilitat, preu i implantació en el mercat.

8.2. Selecció del maquinari

El maquinari seleccionat degut a les necessitats del sistema combinades amb la relació amb proveïdors i beneficis en preu dels components, ha estat la peca Xilinx Artix 7 XA7ATCSG324-2I. La família Artix-7 és la mes petita de les tecnologies actuals Artix-7, Kintex-7, Spartan-7 i Virtex-7.

8.3. Entrades i sortides del sistema

Les entrades del sistema son les següents:

- General:
  - Clk (entrada): Senyal de rellotge d'entrada al sistema. La freqüència de rellotge d'entrada és de 10MHz
  - Reset (entrada): Senyal de *reset* del sistema. Actiu baix.
- SPI:
  - SS (entrada): Senyal de selecció de dispositiu del bus SPI. Actiu baix
  - Spiclk (entrada): Rellotge del bus SPI. Freqüència màxima permesa 1MHz
  - Mosi (entrada): Senyal del bus SPI de sortida del *master* i d'entrada al *slave*
  - Miso (sortida): Senyal del bus SPI de sortida del *slave* i d'entrada al *master*
- Input pins
  - input_pins1 (entrada): *Pins* d'entrades digitals a llegir per SPI (16 bit)
  - input_pins2 (entrada): *Pins* d'entrades digitals a llegir per SPI (16 bit)
  - input_pins3 (entrada): *Pins* d'entrades digitals a llegir per SPI (16 bit)
  - input_pins4 (entrada): *Pins* d'entrades digitals a llegir per SPI (16 bit)
- Output pins
  - output_pins1 (output): *Pins* de sortides digitals a escriure (16 bit)
  - output_pins2 (output): *Pins* de sortides digitals a escriure (16 bit)
  - output_pins3 (output): *Pins* de sortides digitals a escriure (16 bit)
  - output_pins4 (output): *Pins* de sortides digitals a escriure (16 bit)

- pwm1 (output): *Pin* de sortida de PWM
- pwm2 (output): *Pin* de sortida de PWM
- pwm3 (output): *Pin* de sortida de PWM
- pwm4 (output): *Pin* de sortida de PWM
- pwm5 (output): *Pin* de sortida de PWM
- pwm6 (output): *Pin* de sortida de PWM
- pwm7 (output): *Pin* de sortida de PWM
- pwm8 (output): *Pin* de sortida de PWM

8.4. Diagrama de blocs del sistema

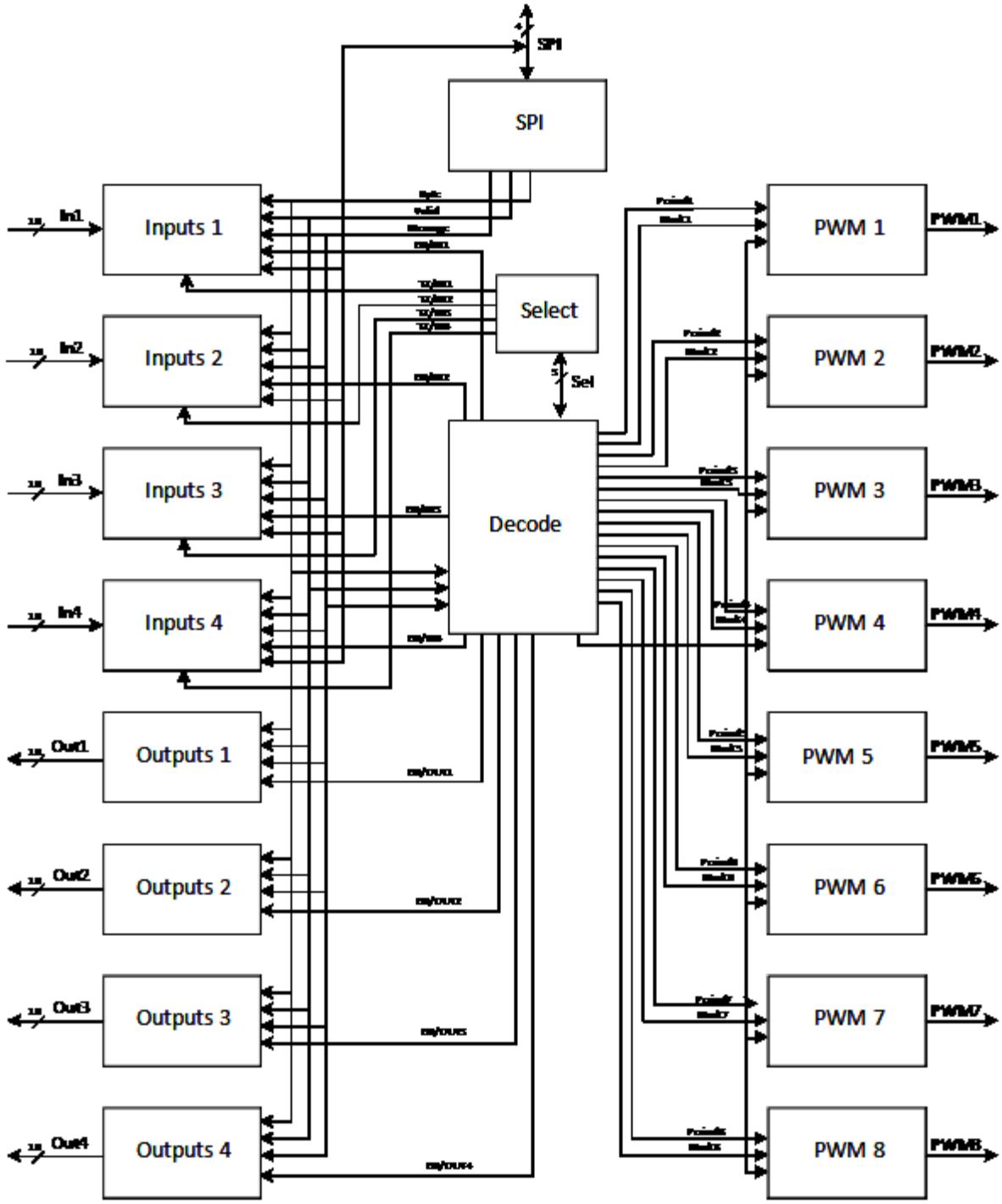


Figura 3. Diagrama de blocs del sistema

8.4.1. Bloc SPI

El bloc SPI realitza la funció de la recepció de les dades a traves del protocol SPI així com la transmissió de les respostes sol·licitades. Com a entrada té els *pins* de la interfície SPI: SS, CLK, MISO i MOSI i com a sortida téun bus paral·lel de dades de 8 bits, un senyal de dada valida al bus i un senyal de missatge vàlid. Aquest bus de sortida s'actualitza a cada recepció de l'últim bit de cada byte transmès pel bus SPI

8.4.2. Bloc Select

El bloc Select serveix per activar les dades de sortida del bloc de *pins* d'entrada d'acord amb la selecció rebuda pel bloc SPI, prèviament descodificada pel bloc Decode. Aquest mecanisme es necessari per que el bus de sortida de dades es compartit amb tots els blocs de entrades de pins, per tant cal un mecanisme d'arbitratge d'accés al bus.

8.4.3. Bloc Decode

El bloc Decode realitza tasques de descodificació de les dades rebudes pel bloc SPI. Realitza una descodificació completa de les dades i activa als senyals adequats de sortida. En el cas de rebre una comanda de escriptura en pins de sortida o de lectura de pins d'entrada el bloc realitza la selecció del perifèric a mes de generar el senyal pel bloc Select que gestiona l'accés al bus de dades de tornada del SPI

8.4.4. Bloc Inputs X

El bloc d'Inputs realitza la lectura de dades d'un bloc de pins d'entrada de longitud 16bits. Aquest bloc torna les dades cap al bloc SPI per la seva serialització. Es defineixen quatre blocs d'entrades de 16 bits (64 pins d'entrada).

8.4.5. Bloc Outputs X

El bloc d'Outputs realitza la escriptura de dades d'un bloc de *pins* de sortida de longitud 16bits. Aquest bloc escriurà les dades enviades pel bloc SPI. Es defineixen quatre blocs de sortides de 16 bits (64 pins de sortida).

8.4.6. Bloc PWMX

El bloc de PWM de sortida s'ha implementat de manera que es permeti una màxima flexibilitat de configuració. S'han definit dos paràmetres basics de configuració i un *pin* de sortida que realitza la tasca de PWM. Els dos paràmetres son el període de PWM, que és variable (amplada 16 bits) i la marca de PWM, que és el nombre de cicles que el *pins* de sortida estan a '1', canviant a '0' per la resta de cicles fins arribar al període.

El bloc pren les dades d'entrada del bloc Decode que li lliura els dos paràmetres de configuració rebuts pel bloc SPI.

S'han definit 8 blocs de PWM independents, tots ells configurables per separa amb diferents paràmetres. Els 8 PWM s'habiliten al mateix temps quan es rep la ultima dada del vuitè PWM. Aquesta característica ve definida externament per l'aplicació.

El rellotge del PWM es constant a 10MHz i es generat internament per gestor de rellotges de la FPGA. Per tant el cicle de rellotge es de 100ns. El període del PWM pot anar des de 2 fins a 65535, per tant des de 200ns a 6,5535ms. La precisió màxima es de 100ns.

8.5. Protocol de comunicació

La interfície de comunicació entre els elements externs (microcontrolador) i el sistema desenvolupat és l'SPI. Per sobre la interfície de comunicació s'ha definit un protocol de comunicació on s'estableixen les característiques de la informació transmesa, definint quin contingut té cadascun dels bits d'informació i com s'accedeix a cadascun dels perifèrics

El protocol es defineix agrupant la informació transmesa en bytes (8 *bits*), i seqüenciant els bytes en paquets de dades que varien dels 4 als 6 bytes, depenent de la informació que es vulgui transmetre. Les dades de escriptura l'equip son sempre de 4 bytes i en cas de lectura de dades de la FPGA s'afegeixen els dos bytes de tornada de la lectura

La comunicació s'estableix sempre des del *master* que serà l'element extern (microcontrolador) de forma que la FPGA actuarà sempre com a *slave*.

Algunes de les comandes s'estructuren de forma que es permeti el creixement del producte incorporant d'altres característiques futures, per tant algunes de les comandes o grups de comandes s'han previst per poder suportar aquest creixement.

El primer byte serà sempre la comanda a executar: *command*

- *Command*: Es defineixen 3 tipus de comandes possibles
    - Escriptura de PWM (0x00)
    - Lectura de pins (0x01)
    - Escriptura de pins (0x02)

A continuació s'enviarà en grup de comanda, que pot variar en funció de la comanda enviada.

- Escriptura de PWM (0x00):
    - Grup (*high*) (0x05)
    - Grup (*low*) (0x05): període PWM1 (*high*)
    - Grup (*low*) (0x06): període PWM1 (*low*)
    - Grup (*low*) (0x07): marca PWM1 (*high*)
    - Grup (*low*) (0x08): marca PWM1 (*low*)
    - Grup (*low*) (0x09): període PWM2 (*high*)
    - Grup (*low*) (0x0A): període PWM2 (*low*)
    - Grup (*low*) (0x0B): marca PWM2 (*high*)
    - Grup (*low*) (0x0C): marca PWM2 (*low*)

    - Grup (*low*) (0x0D): període PWM3 (*high*)
    - Grup (*low*) (0x0E): període PWM3 (*low*)
    - Grup (*low*) (0x0F): marca PWM3 (*high*)
    - Grup (*low*) (0x10): marca PWM3 (*low*)
    - Grup (*low*) (0x11): període PWM4 (*high*)
    - Grup (*low*) (0x12): període PWM4 (*low*)
    - Grup (*low*) (0x13): marca PWM4 (*high*)
    - Grup (*low*) (0x14): marca PWM4 (*low*)
    - Grup (*low*) (0x15): període PWM5 (*high*)
    - Grup (*low*) (0x16): període PWM5 (*low*)
    - Grup (*low*) (0x17): marca PWM5 (*high*)
    - Grup (*low*) (0x18): marca PWM5 (*low*)
    - Grup (*low*) (0x19): període PWM6 (*high*)
    - Grup (*low*) (0x1A): període PWM6 (*low*)
    - Grup (*low*) (0x1B): marca PWM6 (*high*)
    - Grup (*low*) (0x1B): marca PWM6 (*low*)
    - Grup (*low*) (0x1C): període PWM7 (*high*)
    - Grup (*low*) (0x1D): període PWM7 (*low*)
    - Grup (*low*) (0x1E): marca PWM7 (*high*)
    - Grup (*low*) (0x1F): marca PWM7 (*low*)
    - Grup (*low*) (0x20): període PWM8 (*high*)
    - Grup (*low*) (0x21): període PWM8 (*low*)
    - Grup (*low*) (0x22): marca PWM8 (*high*)
    - Grup (*low*) (0x23): marca PWM8 (*low*)
- Lectura de pins (0x01):
    - Grup (*low*) (0x00): lectura perifèric 1
    - Grup (*low*) (0x01): lectura perifèric 2
    - Grup (*low*) (0x02): lectura perifèric 3
    - Grup (*low*) (0x02): lectura perifèric 4
- Escriptura de pins (0x02):
    - Grup (*low*) (0x00): escriptura perifèric 1
    - Grup (*low*) (0x01): escriptura perifèric 2
    - Grup (*low*) (0x02): escriptura perifèric 3
    - Grup (*low*) (0x02): escriptura perifèric 4

Exemples de comunicació:

- Escriptura de PWM: Escriptura de PWM (0x00), (0x05), període de PWM3 (0x0E) al valor 103 (0x67): 0x00050E67
- Escriptura de *pins*: Escriptura de *pins* (0x02), perifèric 2 (0x01), valor 17767 (0x4567): 0x02014567
- Lectura de *pins*: Lectura de *pins* (0x01), perifèric 3 (0x02): 01020000

9. Implementacio

9.1. Gestió dels rellotges

El rellotge d'entrada al sistema és un oscil·lador de 10MHz que serveix de referencia per a tota la cadena de rellotges interna. S'ha utilitzat el rellotge de 10MHz per l'avantatge en cost d'un oscil·lador de baixa freqüència i la capacitat de la multiplicació de la freqüència de rellotge de la FPGA utilitzant un bloc IP proporcionat per Xilinx del MMCM. La freqüència d'entrada és de 10MHz i té dues freqüències de sortida: una de 100MHz (que és la freqüència de referencia interna de la FPGA) i una de 10MHz (que és la freqüència de referencia del PWM)

Figura 4. Bloc de generació de rellotge

9.2. Transceptor SPI

El transceptor SPI s'encarrega de gestionar la comunicació amb l'exterior a traves del bus SPI.

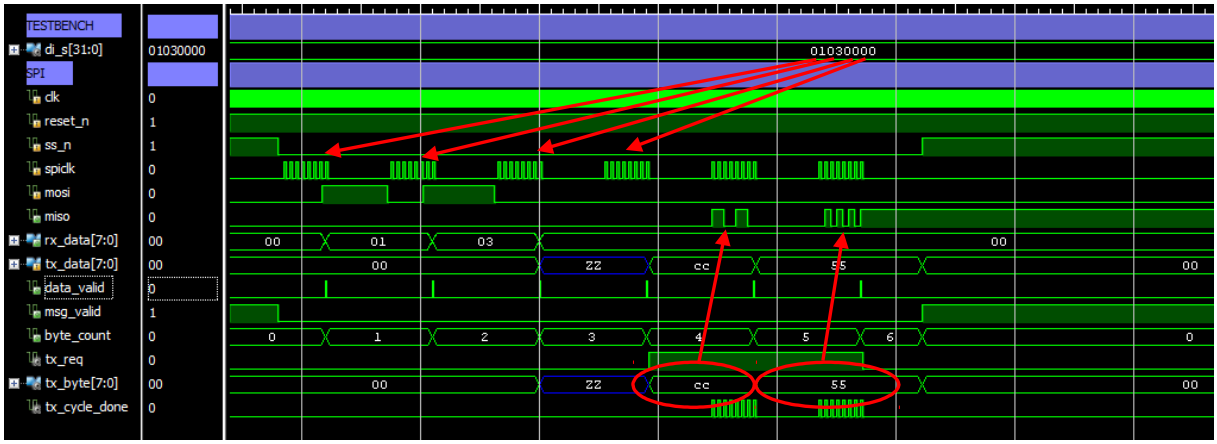


Figura 5. Bloc de transceptor SPI

El transceptor SPI té la funció de rebre pel pin MOSI i enviar pel pin MISO les dades de la interfície SPI. La implementació del bus SPI utilitza els pins estàndards del protocol SS (habilitació de missatge pel destinatari), CLK (rellotge de *bit*), MOSI (dades de sortida del *master* i entrada al *slave*) i MISO (dades de sortida del *slave* i entrada al *master*)

En la captura anterior es mostra un exemple de lectura de pins d'entrada que combina tant la escriptura del *master* en el *pin* MOSI com el retorn de dades pel *pin* MISO. La comanda que entra pel bus SPI (pin MOSI) es: 0x01030000. Aquesta comanda equival a la lectura del port de dades d'entrada #3. Com es pot veure en la captura primer s'envia el *byte* 0x01, després el *byte* 0x03, després el *byte* 0x00 i després el *byte* 0x00. Els següents dos *bytes* son la lectura de la informació sol·licitada, per tant en el *pin* MISO apareix primer el *byte* 0xEE i després el *byte* 0x55.

Descripció dels senyals de la captura:

Di_s[31] : Dada del *testbench* a enviar (*testbench*)

Clk : rellotge del sistema (*testbench*)

Reset_n : *reset* del sistema (*testbench*)

Ss_n : habilitació del dispositiu mentre dura el missatge (*testbench*)

Spiclk : rellotge del bus SPI (*testbench*)

MOSI : dades de sortida del *master* i entrada al slave (*testbench*)

MISO : dades de sortida del *slave* i entrada al master (*testbench*)

Rx_data[7] : Byte rebut (sistema)

Tx_data[7] : Byte a enviar (sistema)

Data_valid : senyal per indicar quan s'ha rebut un byte i esta disponible a rx_data (sistema)

Msg_valid : senyal que indica que estem rebent un missatge (sistema)

Byte_count : comptador de byte rebut a dins un missatge (sistema)

Tx_req : dades disponibles a ser transmeses (sistema)

Tx_byte : dades a transmetre (sistema)

Tx_cycle_done: comptador de bits de dades transmesos (sistema)

La implementació del bloc es realitza en tres processos diferents que succeeixen simultàniament.

Procés 1: Procés de lectura de cada *bit* d'entrada (MOSI) sincronitzat amb el rellotge del sistema i el rellotge d'entrada del SPI i emmagatzematge per construir el *byte*.

Procés 2: Procés de habilitació del senyal de sortida de dada valida quan el *byte* de lectura d'entrada esta complet per permetre la seva lectura externa

Procés 3: Procés de serialitzacio de les dades a transmetre (MISO) pel bus SPI sincronitzat amb el rellotge del sistema i el rellotge del SPI

9.3. Descodificador

El descodificador té la tasca de descodificar la informació rebuda per la interfície SPI i generar senyals de sortida cap a tots els perifèrics.

En el cas de rebre comandes de lectura o escriptura genera el senyal d'activació del perifèric, tant per la descodificació de senyals en cas de sortides com l'accés al bus en cas de lectures

En el cas de rebre comandes de escriptura de PWM genera els senyals adequats per a cada PWM.

En la següent captura es pot veure com el mòdul va rebent seqüencialment les comandes de cadascun dels PWM i va posant els valors rebuts als senyals pwmX_reriod i pwmX_mark.

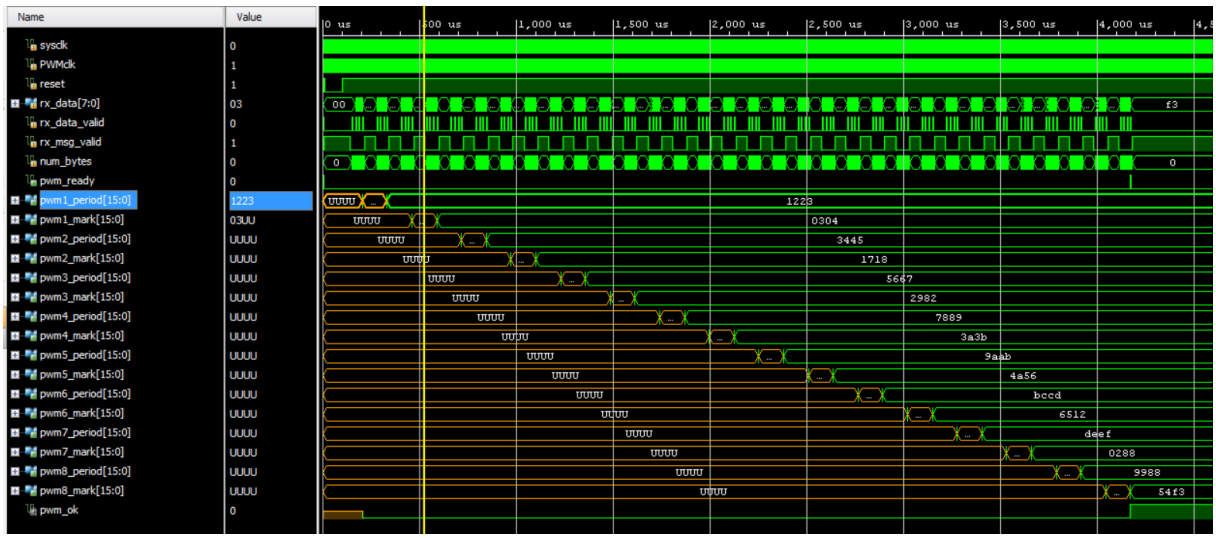


Figura 6. Bloc descodificador

En la captura següent es mostra una secció concreta de la programació del PWM1

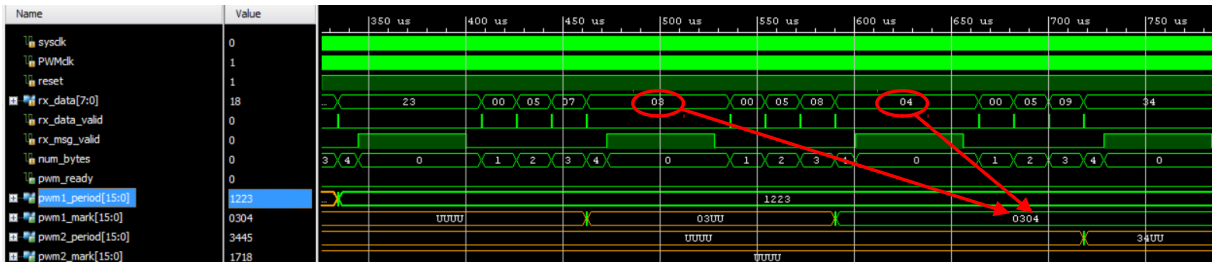


Figura 7. Detall de PWM Del descodificador

La comanda rebuda 0x00050703 indica la escriptura de la part alta del pwm1_mark_h (0x03) i la comanda posterior 0x00050804 indica la escriptura de la part baixa del pwm1_mark_l (0x04), per tant el senyal pwm1_mark esdevé 0x0304.

La implementació d'aquest bloc es realitza mitjançant tres processos que succeeixen simultàniament.

Procés 1: Procés per descodificar cada *byte* de la informació i identificar el *command*, el *group high*, el *group low* i el valor.

Procés 2: Procés per la acció posterior a la descodificació, validant els perifèrics seleccionats o emmagatzemant els valors de PWM

Procés 3: Habilitació de sortida de PWM vàlid en el cas de rebre tota la informació referent als vuit PWMs

9.4. PWM

El bloc de PWM conté la implementació de 8 blocs idèntics de PWM, cadascun d'ells governant una sortida d'un pin. La programació dels PWM es realitza en bloc, seqüencialment des del primer a l'últim. Un cop s'ha realitzat la programació de l'últim es carreguen tots els valors als PWMs. Aquesta implementació es deguda a les especificacions del programari del microcontrolador (no implementat en aquest projecte). Hauria estat mes adequat activar cadascun dels PWM quan es rep la seva informació però la definició del funcionament ha estat externa al projecte. També per especificacions externes no s'ha protegit el PWM contra la definició d'un pols major que el període. Totes aquestes proteccions s'han desenvolupat en el projecte del programari del microcontrolador

Els valors de període i de marca del PWM son descodificats al bloc descodificador i lliurat a cadascun dels blocs de PWM juntament amb el senyal de carrega de valors.

Cadascun dels PWM té dos valors programables de 16 bits: El període del PWM i la duració del pols positiu. D'aquesta manera es dota de gran flexibilitat al PWM, podent tenir precisions màximes de 1/65535.

La implementació del període es realitza fent un comptador de rellotge que incrementa fins al valor emmagatzemat com a període, posant-se a 0 un cop s'arriba a aquest valor.

La implementació del pols es realitza posant el *pin* a 1 quan el comptador esta a 0 i posant el *pin* a 0 quan el comptador arriba a la marca.

La implementació en codi VHDL del component del PWM s'ha realitzat amb bucles iteratius de implementació per aconseguir una millor claredat en el codi.

```
type data16 is array (3 downto 0) of STD_LOGIC_VECTOR(15 DOWNTO 0);
type data8 is array (3 downto 0) of STD_LOGIC_VECTOR(7 DOWNTO 0);
type pwm16 is array (7 downto 0) of STD_LOGIC_VECTOR(15 DOWNTO 0);
type bit4 is array (4 downto 0) of STD_LOGIC;
type bit8 is array (7 downto 0) of STD_LOGIC;

SIGNAL inputs : data16;
SIGNAL outputs : data16;
SIGNAL data2spi : data8;
SIGNAL pwm_period : pwm16;
SIGNAL pwm_mark : pwm16;
SIGNAL tx_req : bit4;
SIGNAL pwm_out : bit8;
```

```
GEN_PWM:
for I in 0 to 7 generate
    Inst_pwm : pwm
    port map (
        sysclk      => sysclk,
        PWMclk      => PWMclk,
        reset       => reset_n,
        period      => pwm_period(I),
        mark        => pwm_mark(I),
        load_data   => pwm_ready,
        pwm_out     => pwm_out(I)
    );
end generate GEN_PWM;
```

Figura 8. Generació d'instanciacions de PWMs en VHDL

9.5. Select (MUX)

El bloc de selecció de perifèric realitza la tasca de l'habilitació del *pin* de sortida del perifèric d'entrada seleccionat per permetre l'accés al bus compartit de dades. Es realitza en un únic procés.

9.6. Lectura de senyals digitals d'entrada

El bloc de multiplexació s'encarrega de la selecció de quin dels quatre blocs de *pins* d'entrada ha estat seleccionat per lliurar les dades al bus (prèviament el bloc descodificador selecciona el perifèric en codi binari).

El bloc de lectura de senyals d'entrada consta de dos processos: un de gestió de *bits* per poder sincronitzar les dades i un altre de devolució del valor dels *pins* d'entrada a traves del bus SPI en el moment adequat.

El procés de sincronització de les dades s'encarrega de comptar la posició de *bit* a dins el *byte* llegint el bus SPI d'entada al perifèric, per així poder determinar quin byte és l'actual. Cal tenir en compte que el bus SPI funciona en sèrie i la dada es retornada en paral·lel al mòdul de SPI que s'encarregarà de serialitzar-la de sortida cap al bus. Aquest procés funciona sempre encara que el perifèric no estigui seleccionat.

El procés de devolució de les dades al bus SPI nomes funciona si el perifèric ha estat seleccionat per evitar conflictes de dades, ja que els quatre perifèrics de lectura de pins d'entrada comparteixen el mateix bus de dades en paral·lel de sortida.

Per explicar el funcionament es mostra la captura següent on s'han creat tres blocs de dades: les generades pel *testbench*, les del bus SPI i les de sortida del bloc de lectura d'entrades

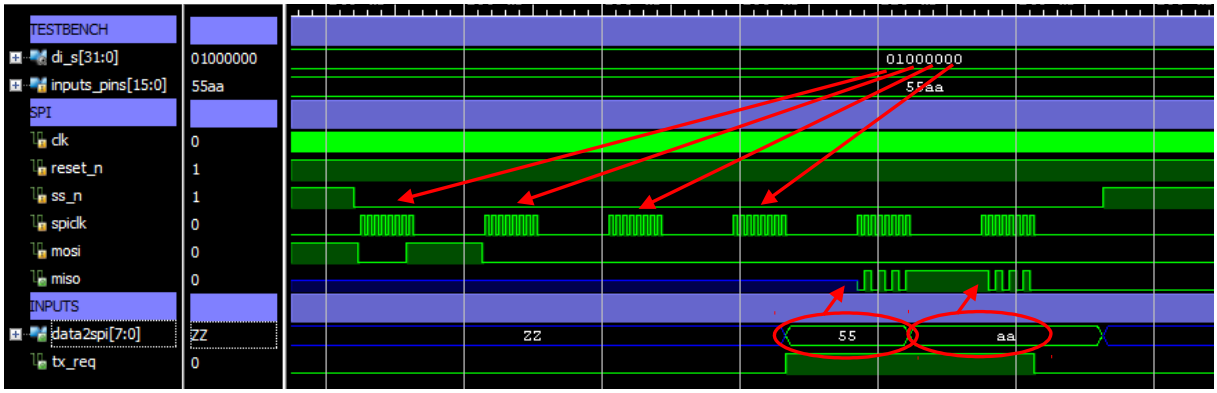


Figura 9. Lectura de senyals digitals d'entrada

Els valors generats pel *testbench* inclouen la comanda que es vol enviar per l'SPI (0x01000000: lectura de pins del bloc 1) i les dades que es posen als *pins* d'entrada de 16 *bits* (0x55aa)

A continuació es veu el missatge SPI, on es pot veure que s'envien els quatre *bytes* serialitzats 0x01000000. A la finalització de l'últim *bit* del quart *byte* el perifèric de lectura d'entrada de dades numero 1 ha estat seleccionat i carrega en el registre data2spi (de 8 bits) el valor dels 8 pins MSB (0x55), activant el senyal de dades disponibles per lliurar al SPI. El Bloc SPI pren les dades del registre data2spi i les serialitza al bus. Al final del vuitè bit del cinquè byte es carrega al registre data2spi el valor dels 8 pins LSB (0xAA). El bloc

SPI serialitza les noves dades. Al final de l'últim *bit* del sisè *byte*, el senyal de tx_req baixa per indicar que ja no té dades valides. Com es pot comprovar cadascun dels blocs de lectura de dades allibera el bus de sortida paral·lel deixant-lo en alta impedància per la següent operació.

La implementació en codi VHDL del component del *Inputs* s'ha realitzat amb bucles iteratius de implementació per aconseguir una millor claredat en el codi.

```
type data16 is array (3 downto 0) of STD_LOGIC_VECTOR(15 DOWNTO 0);
type data8 is array (3 downto 0) of STD_LOGIC_VECTOR(7 DOWNTO 0);
type pwm16 is array (7 downto 0) of STD_LOGIC_VECTOR(15 DOWNTO 0);
type bit4 is array (4 downto 0) of STD_LOGIC;
type bit8 is array (7 downto 0) of STD_LOGIC;

SIGNAL inputs : data16;
SIGNAL outputs : data16;
SIGNAL data2spi : data8;
SIGNAL pwm_period : pwm16;
SIGNAL pwm_mark : pwm16;
SIGNAL tx_req : bit4;
SIGNAL pwm_out : bit8;

GEN_INPUTS:
for I in 0 to 3 generate
    Inst_inputs2spi : inputs2spi
    PORT MAP (
        sysclk => sysclk,
        reset => reset_n,
        msg_valid => rx_msg_valid,
        enable => periph_enable(I),
        spiclk => spiclk,
        inputs_pins => inputs(I),
        data2spi => data2spi(I),
        tx_req => tx_req(I)
    );
end generate GEN_INPUTS;
```

Figura 10. Generació d'instanciació de blocs d'entrada en VHDL

9.7. Escriptura de senyals digitals de sortida

El bloc de escriptura de senyals de sortida consta de dos processos: un de comptador de *bits* dins el *byte* per saber en quin *byte* es troba la comunicació (es el mateix bloc que el de lectura de senyals d'entrada) i un altre procés encarregat de l'escriptura en el port de sortida. Com es tracta d'un port de 16 *bits* l'escriptura es realitza en dos passos, un a la recepció del primer *byte* i l'altre a la recepció del segon.

Per explicar el funcionament es mostra la captura següent on s'han creat tres blocs de dades: les generades pel *testbench*, les del bus SPI i les de sortida dels *pins* del bloc d'escriptura de sortides

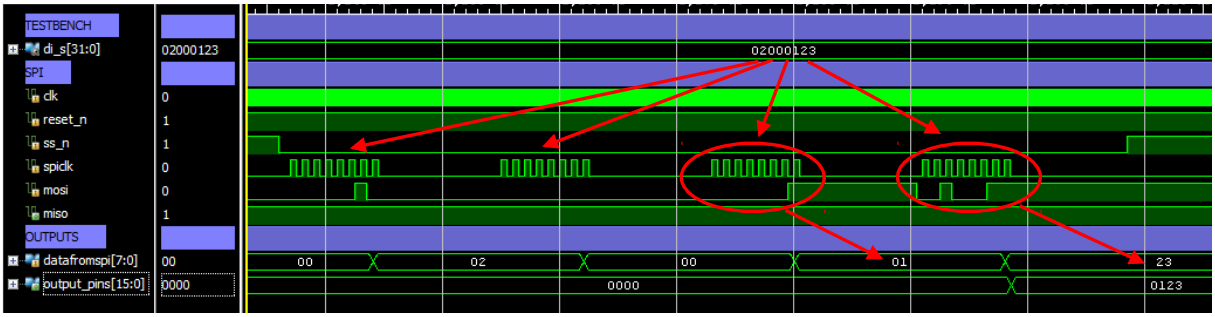


Figura 11. Escriptura de senyals digitals de sortida

Els valors generats pel *testbench* inclouen la comanda que es vol enviar per l'SPI (0x02000123: escriptura de pins del bloc 1).

A continuació es veu el missatge SPI, on es pot veure que s'envien els quatre *bytes* serialitzats 0x02000123. A la finalització de l'últim *bit* del tercer *byte* el perifèric carrega el *byte* MSB al registre datadeomspi (0x01). A la finalització de l'últim bit del quart *byte* es carrega al registre datafromspi el valor dels 8 *pins* LSB (0x23) i es volca el contingut al registre de *pins* de sortida (16 bits) amb el valor dels dos *bytes*: 0x0123.

La implementació en codi VHDL del component del Inputs s'ha realitzat amb bucles iteratius de implementació per aconseguir una millor claredat en el codi.

```
type data16 is array (3 downto 0) of STD_LOGIC_VECTOR(15 DOWNTO 0);
type data8 is array (3 downto 0) of STD_LOGIC_VECTOR(7 DOWNTO 0);
type pwm16 is array (7 downto 0) of STD_LOGIC_VECTOR(15 DOWNTO 0);
type bit4 is array (4 downto 0) of STD_LOGIC;
type bit8 is array (7 downto 0) of STD_LOGIC;

SIGNAL inputs : data16;
SIGNAL outputs : data16;
SIGNAL data2spi : data8;
SIGNAL pwm_period : pwm16;
SIGNAL pwm_mark : pwm16;
SIGNAL tx_req : bit4;
SIGNAL pwm_out : bit8;


GEN_OUTPUTS:
for I in 0 to 3 generate
    Inst_spi2outputs : spi2outputs
    PORT MAP (
        sysclk => sysclk,
        reset => reset_n,
        msg_valid => rx_msg_valid,
        enable => wr_periph_enable(I),
        spiclk => spiclk,
        output_pins => outputs(I),
        datafromspi => rx_data,
        load => '1'
    );
end generate GEN_OUTPUTS;
```

Figura 12. Generació d'instanciació de blocs de sortida en VHDL

10. Validació

10.1. Sistema de validació

La validació del sistema s'ha realitzat amb una simulació de senyals d'entrada realitzada amb el propi entorn de desenvolupament de Xilinx. S'ha creat un fitxer en VHDL que simula uns senyals d'entrada a traves del bus SPI. El disseny realitzat es connecta a aquests senyals d'entrada i mitjançant un analitzador lògic es comprova que el funcionalment del disseny és l'esperat.

10.2. Protocol de proves

10.2.1. Proves del PWM

Les proves dels PWMs es realitzen programant els vuit PWMs amb valors diferents per poder comprovar amb l'analitzador lògic que els valors de cadascun dels PWM responen a la programació.

Els valors de prova dels PWM son els següents:

- PWM1:

Període: 4643 (0x1223) → Missatge: 0x00050512, 0x00050623

Marca: 772 (0x0304) → Missatge: 00050703, 0x00050804

- PWM2:

Període: 13381 (0x3445) → Missatge: 0x00050934, 0x00050A45

Marca: 5912 (0x1718) → Missatge: 0x00050B17, 0x00050C18

- PWM3:

Període: 22119 (0x5667) → Missatge: 0x00050D56, 0x00050E67

Marca: 10626(0x2982) → Missatge: 0x00050F29, 0x00051082

- PWM4:

Període: 30857 (0x7889) → Missatge: 0x00051178, 0x00051289

Marca: 14907 (0x3A3B) → Missatge: 0x0005133A, 0x0005143B

- PWM5:

Període: 39595(0x9AAB) → Missatge: 0x0005159A, 0x000516AB

Marca: 19030 (0x4A56) → Missatge: 0x0005174A, 0x00051856

- PWM6:

Període: 48333 (0xBCCD) → Missatge: 0x000519BC, 0x00051ACD

Marca: 25875 (0x6512) → Missatge: 0x00051B65, 0x00051C12

- PWM7:

Període: (0xDEEF) → Missatge: 0x00051DDE, 0x00051EEF

Marca: (0x0288) → Missatge: 0x00051F02, 0x00052088

- PWM8:

Període: 39304 (0x9988) → Missatge: 0x00052199, 0x00052288

Marca: 21747 (0x54F3) → Missatge: 0x00052354, 0x000524F3

```
-- Write PWM (0x00) - Group 0x05
variable fifo_wrconfig : fifo_wrconfig_type :=
    (X"00050512",X"00050623",X"00050703",X"00050804",X"00050934",X"00050A45",X"00050B17",X"00050C18",
     X"00050D56",X"00050E67",X"00050F29",X"00051082",X"00051178",X"00051289",X"0005133A",X"0005143B",
     X"0005159A",X"000516AB",X"0005174A",X"00051856",X"000519BC",X"00051ACD",X"00051B65",X"00051C12",
     X"00051DDE",X"00051EEF",X"00051F02",X"00052088",X"00052199",X"00052288",X"00052354",X"000524F3");

variable fifo_wrconfig_head : integer range 0 to fifo_wrconfig_size-1;


-- Loop to write PWMs
IF (test_select_config = 1) THEN                        -- Check test enabled
    for cnt in 0 to fifo_wrconfig_size-1 loop
        fifo_wrconfig_head := cnt;                      -- pre-compute next pointer
        wait until byte_clk'event and byte_clk = '1';   -- sync fifo data load at next rising edge
        di_s <= fifo_wrconfig(fifo_wrconfig_head);      -- place data into di_s input bus
        wait until byte_clk'event and byte_clk = '1';   -- sync fifo data load at next rising edge
        wren_s <= '1';                                  -- write data into shift register
        wait until byte_clk'event and byte_clk = '1';   -- sync fifo data load at next rising edge
        wait until byte_clk'event and byte_clk = '1';   -- sync fifo data load at next rising edge
        wren_s <= '0';                                  -- remove write enable signal
    end loop;
END IF;
```

Figura 13. Generació del senyal de test d'escriptura de PWMs

10.2.2. Proves de lectura de pins

Les proves de lectura de pins es realitzen fixant prèviament el valor de les entrades a un valor preestablert i comprovant que la lectura es realitza correctament a traves del bus SPI.

Els valors de lectura seleccionats son:

Perifèric 1: 0101010110101010b (0x55AA) → Missatge: 0x01000000

Perifèric 1: 1010101000110011b (0xAA33) → Missatge: 0x01010000

Perifèric 1: 0011001111001100b (0x33CC) → Missatge: 0x01020000

Perifèric 1: 1100110001010101b (0xCC55) → Missatge: 0x01030000

```
-- Read (0x01)
variable fifo_rd : fifo_read_type := (X"01000000",X"01010000",X"01020000",X"01030000");
variable fifo_read_head : integer range 0 to fifo_read_size-1;

-- Loop to read inputs
IF (test_select_read = 1) THEN                          -- Check test enabled
    for cnt in 0 to fifo_read_size-1 loop
        fifo_read_head := cnt;                          -- pre-compute next pointer
        wait until byte_clk'event and byte_clk = '1';   -- sync fifo data load at next rising edge
        di_s <= fifo_rd(fifo_read_head);                -- place data into di_s input bus
        wait until byte_clk'event and byte_clk = '1';   -- sync fifo data load at next rising edge
        wren_s <= '1';                                  -- write data into shift register
        wait until byte_clk'event and byte_clk = '1';   -- sync fifo data load at next rising edge
        wait until byte_clk'event and byte_clk = '1';   -- sync fifo data load at next rising edge
        wren_s <= '0';                                  -- remove write enable signal
        wait for bit_clk_period*100;
    end loop;

    wait for bit_clk_period*100;
END IF;
```

Figura 14. Generació de senyal de test de lectura de *pins* d'entrada

10.2.3. Proves de escriptura de pins

Les proves d'escriptura de pins es realitzen enviant uns valors a traves del bus SPI i comprovant que els perifèrics adequats envien les dades als ports de sortida.

Els valors d'escriptura seleccionats son:

Perifèric 1: 0000000100100011b (0x0123) → Missatge: 0x02000123

Perifèric 1: 0100010101100111b (0x4567) → Missatge: 0x02014567

Perifèric 1: 1000100110101011b (0x89AB) → Missatge: 0x020289AB

Perifèric 1: 1100110111101111b (0xCDEF) → Missatge: 0x0203CDEF

```
-- Write (0x02)
variable fifo_wr : fifo_write_type := (X"02000123",X"02014567",X"020289AB",X"0203CDEF");
variable fifo_write_head : integer range 0 to fifo_write_size-1;


-- Loop to write outputs
IF (test_select_write = 1) THEN                          -- Check test enabled
    for cnt in 0 to fifo_write_size-1 loop
        fifo_write_head := cnt;                          -- pre-compute next pointer
        wait until byte_clk'event and byte_clk = '1';    -- sync fifo data load at next rising edge
        di_s <= fifo_wr(fifo_write_head);                -- place data into di_s input bus
        wait until byte_clk'event and byte_clk = '1';    -- sync fifo data load at next rising edge
        wren_s <= '1';                                   -- write data into shift register
        wait until byte_clk'event and byte_clk = '1';    -- sync fifo data load at next rising edge
        wait until byte_clk'event and byte_clk = '1';    -- sync fifo data load at next rising edge
        wren_s <= '0';                                   -- remove write enable signal
        wait for bit_clk_period*100;
   end loop;

    wait for bit_clk_period*100;
END IF;
```

Figura 15. Generació de senyal de test de lectura de *pins* de sortida

10.3. Reset

En la captura posterior es pot veure com el senyal de *reset* deixa tots els ports en estat d'alta impedància fins que esdevé "0", on s'inicialitzen tots els valors de tots els ports de sortida. Els valors dels registres de PWM es queden indefinits però els pins de sortida es situen a '1'.

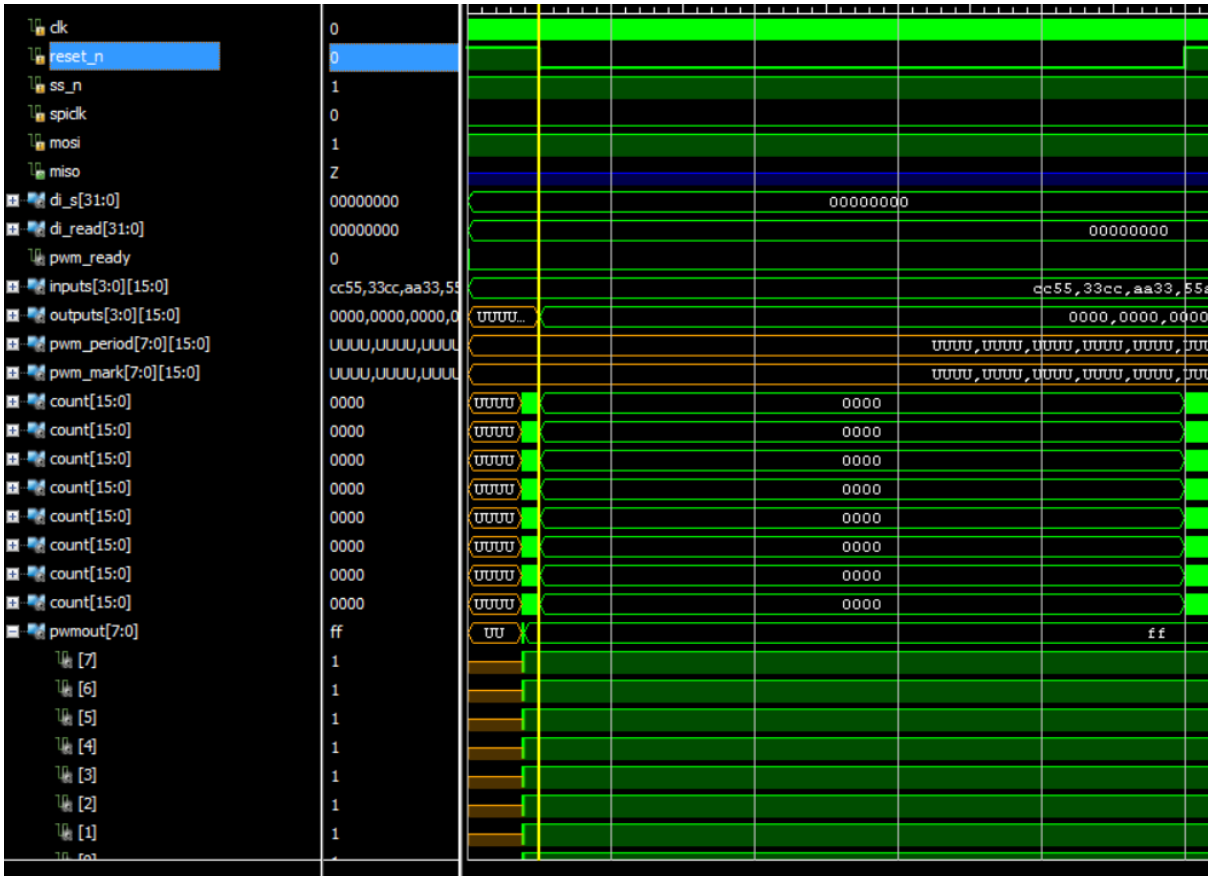


Figura 16. Anàlisi temporal del senyal de reset

10.4. Recepció SPI i descodificació

En aquesta captura (sense entrar en la descodificació particular de cada comanda) es pot veure com es va rebent tota la informació i es van carregant tots els registres de pins de sortida i dels PWM

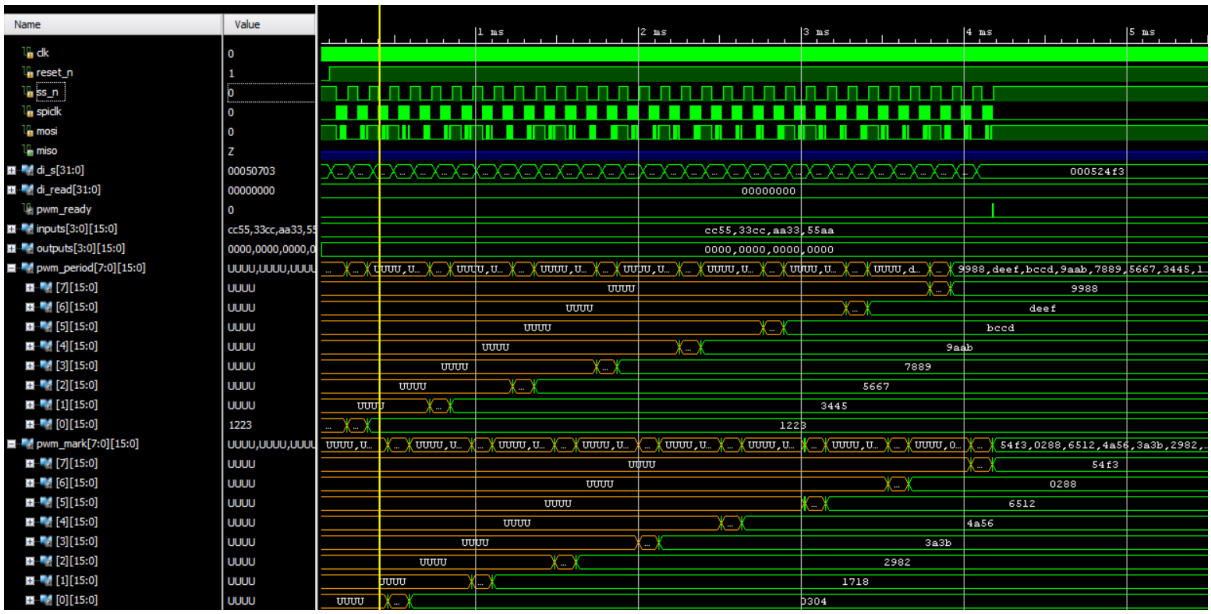


Figura 17. Anàlisi temporal de les dades del bus SPI

10.5. PWM

Tots els PWM tenen dos registres que marquen el seu funcionament i un *pin* de sortida que reflecteix l'estat del PWM. El registre període marca el període en cicles de rellotge de duració del PWM i el registre *mark* marca l'interval del pin on serà positiu en cicles de rellotge de duració. El rellotge utilitzat pel PWM no és el rellotge del sistema si no el rellotge de PWM (10MHz).

El valor de la marca sempre ha de ser menor que el valor del període. El sistema no realitza cap verificació i aquesta ha d'estar implementada al programari del microcontrolador

10.5.1. Proves del PWM1

El període de PWM1 es carrega a 0x1223 (4643d) i la marca es carrega a 0x304 (772d)

Com es pot veure a la captura quan el comptador arriba al valor de la marca el *pin* de PWM es posa a '0'.

Figura 18. Anàlisi temporal de *reset* del comptador del PWM1

Quan el comptador arriba al valor del període el comptador es posa a '0' i el *pin* de sortida del PWM a '1'.

Figura 19. Anàlisi temporal de *set* del comptador del PWM1

10.5.2. Proves del PWM2

El període de PWM1 es carrega a 0x3445 (13381d) i la marca es carrega a 0x1718 (5912d).

Com es pot veure a la captura quan el comptador arriba al valor de la marca el *pin* de PWM es posa a '0'.

Figura 20. Anàlisi temporal de *reset* del comptador del PWM2

Quan el comptador arriba al valor del període el comptador es posa a '0' i el *pin* de sortida del PWM a '1'.

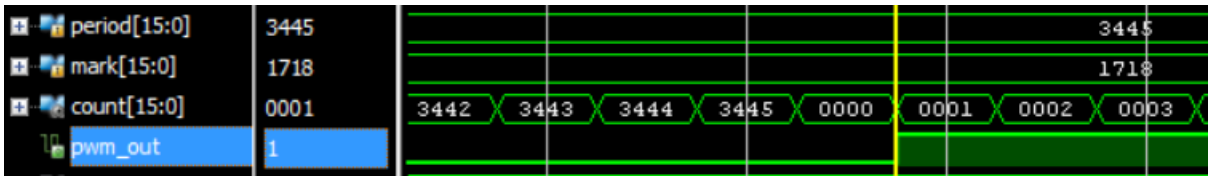


Figura 21. Anàlisi temporal de *set* del comptador del PWM2

10.5.3. Proves del PWM3

El període de PWM1 es carrega a 0x5667 (22119d) i la marca es carrega a 0x2982 (10626d).

Com es pot veure a la captura quan el comptador arriba al valor de la marca el *pin* de PWM es posa a '0'.

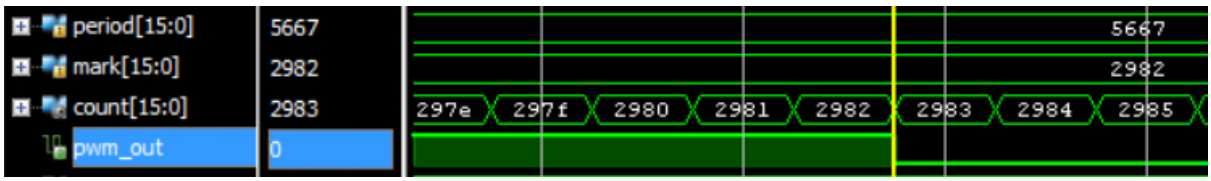


Figura 22. Anàlisi temporal de *reset* del comptador del PWM3

Quan el comptador arriba al valor del període el comptador es posa a '0' i el *pin* de sortida del PWM a '1'.

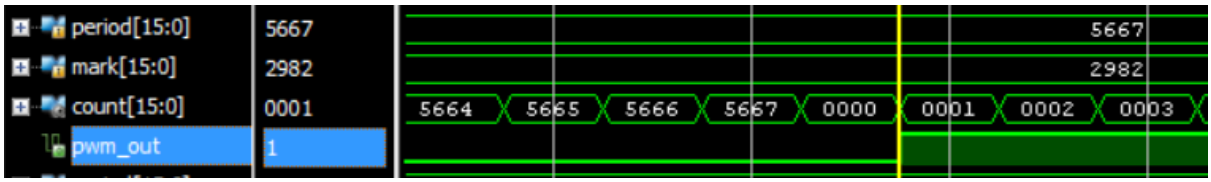


Figura 23. Anàlisi temporal de *set* del comptador del PWM3

10.5.4. Proves del PWM4

El període de PWM1 es carrega a 0x7889 (30857d) i la marca es carrega a 0x3a3b (14907d).

Com es pot veure a la captura quan el comptador arriba al valor de la marca el *pin* de PWM es posa a '0'.

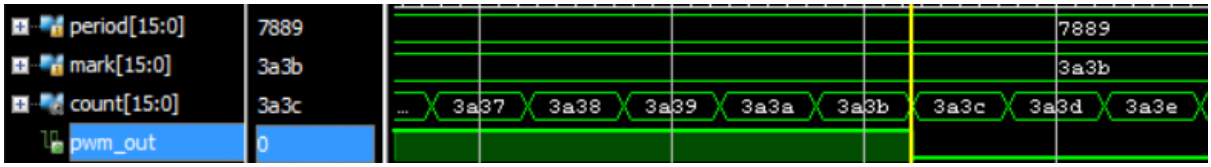


Figura 24. Anàlisi temporal de *reset* del comptador del PWM4

Quan el comptador arriba al valor del període el comptador es posa a '0' i el *pin* de sortida del PWM a '1'.

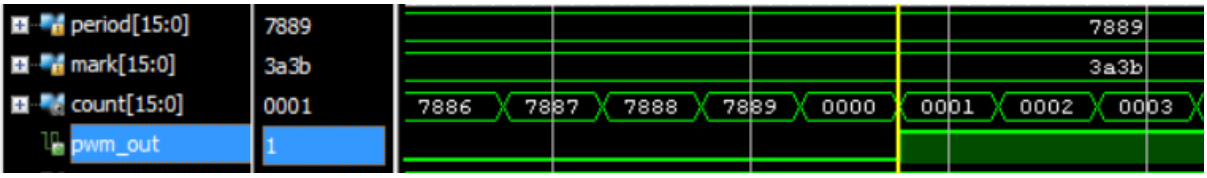


Figura 25. Anàlisi temporal de *set* del comptador del PWM4

10.5.5. Proves del PWM5

El període de PWM1 es carrega a 0x9AAB (39595d) i la marca es carrega a 0x4A56 (19030d).

Com es pot veure a la captura quan el comptador arriba al valor de la marca el *pin* de PWM es posa a '0'.

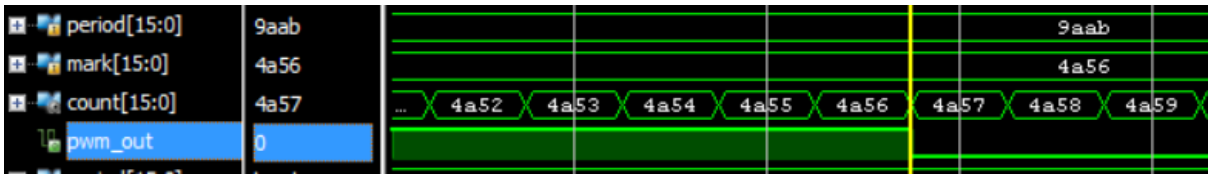


Figura 26. Anàlisi temporal de *reset* del comptador del PWM5

Quan el comptador arriba al valor del període el comptador es posa a '0' i el *pin* de sortida del PWM a '1'.

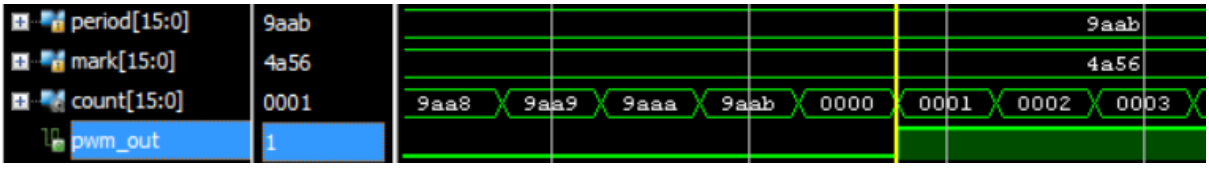


Figura 27. Anàlisi temporal de *set* del comptador del PWM5

10.5.6. Proves del PWM6

El període de PWM1 es carrega a 0xbccd (48333d) i la marca es carrega a 0x6512 (25874d).

Com es pot veure a la captura quan el comptador arriba al valor de la marca el *pin* de PWM es posa a '0'.

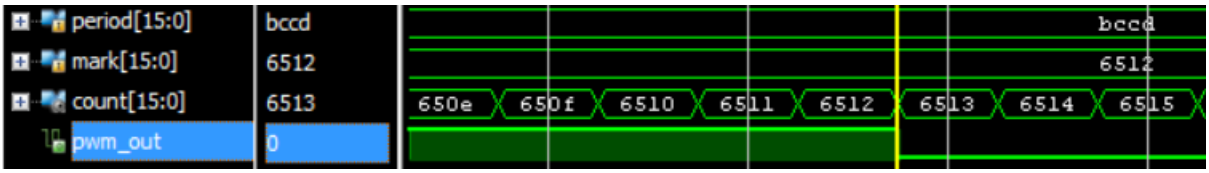


Figura 28. Anàlisi temporal de *reset* del comptador del PWM6

Quan el comptador arriba al valor del període el comptador es posa a '0' i el *pin* de sortida del PWM a '1'.

Figura 29. Anàlisi temporal de *set* del comptador del PWM6

10.5.7. Proves del PWM7

El període de PWM1 es carrega a 0xDEEF (57071d) i la marca es carrega a 0x288 (648d).

Com es pot veure a la captura quan el comptador arriba al valor de la marca el *pin* de PWM es posa a '0'.

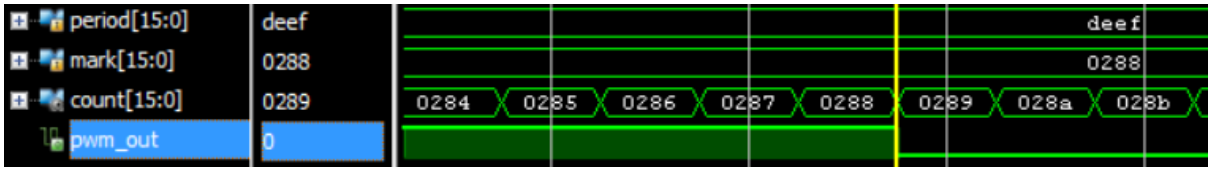


Figura 30. Anàlisi temporal de *reset* del comptador del PWM7

Quan el comptador arriba al valor del període el comptador es posa a '0' i el *pin* de sortida del PWM a '1'.

Figura 31. Anàlisi temporal de *set* del comptador del PWM7

10.5.8. Proves del PWM8

El període de PWM1 es carrega a 0x9988 (38304d) i la marca es carrega a 0x54F3 (21747d).

Com es pot veure a la captura quan el comptador arriba al valor de la marca el *pin* de PWM es posa a '0'.

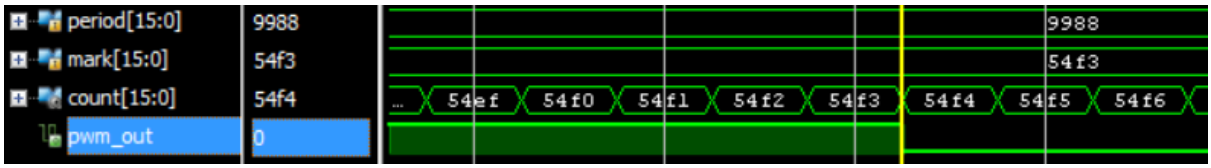


Figura 32. Anàlisi temporal de *reset* del comptador del PWM8

Quan el comptador arriba al valor del període el comptador es posa a '0' i el *pin* de sortida del PWM a '1'.

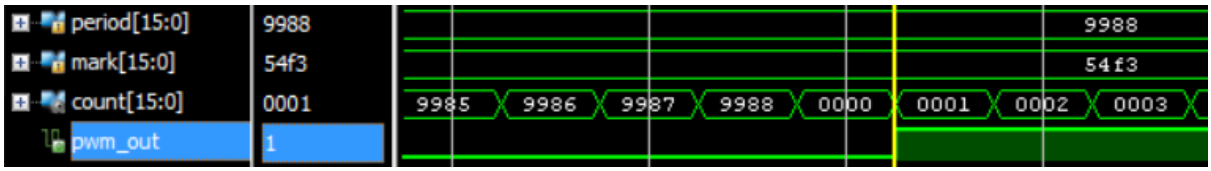


Figura 33. Anàlisi temporal de *set* del comptador del PWM8

10.6. Lectura de senyals digitals d'entrada

10.6.1. *Input* 1:

La comanda de lectura de la entrada de *pins* 1 es rep a través del *pin* MOSI (0x01000000). Els dos *bytes* llegits a continuació de sortida del *pin* MISO son els 0x55 i 0xAA

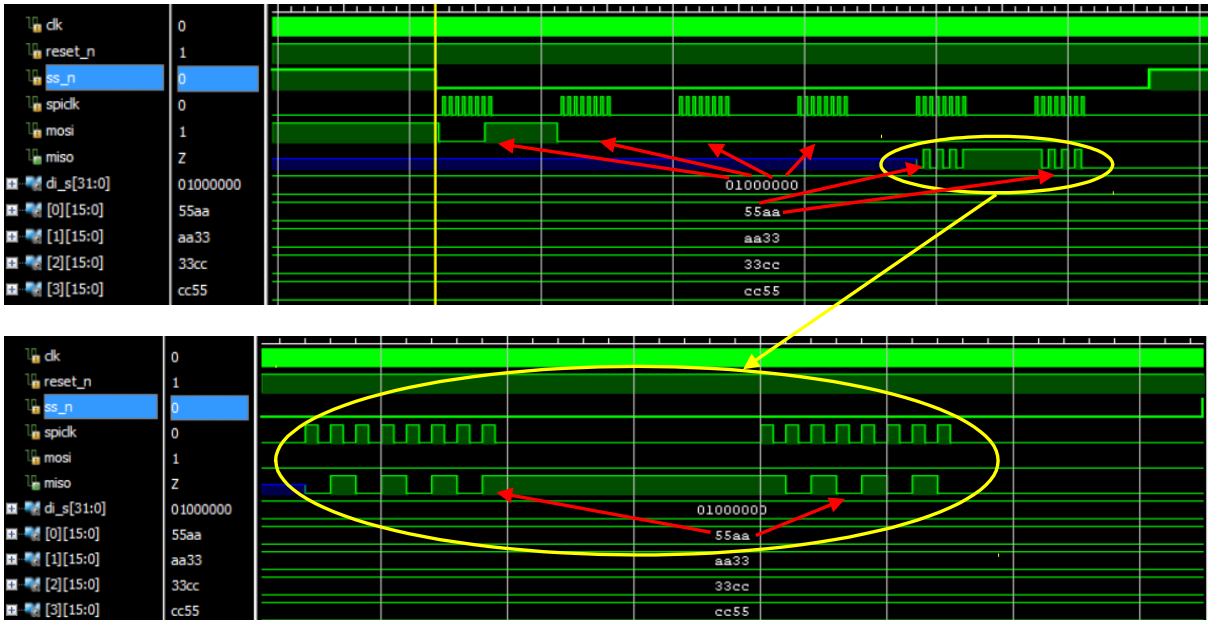


Figura 34. Anàlisi temporal de lectura de senyals d'entrada 1

10.6.2. *Input* 2

La comanda de lectura de la entrada de *pins* 2 es rep a través del *pin* MOSI (0x01010000). Els dos *bytes* llegits a continuació de sortida del *pin* MISO son els 0xAA i 0x33

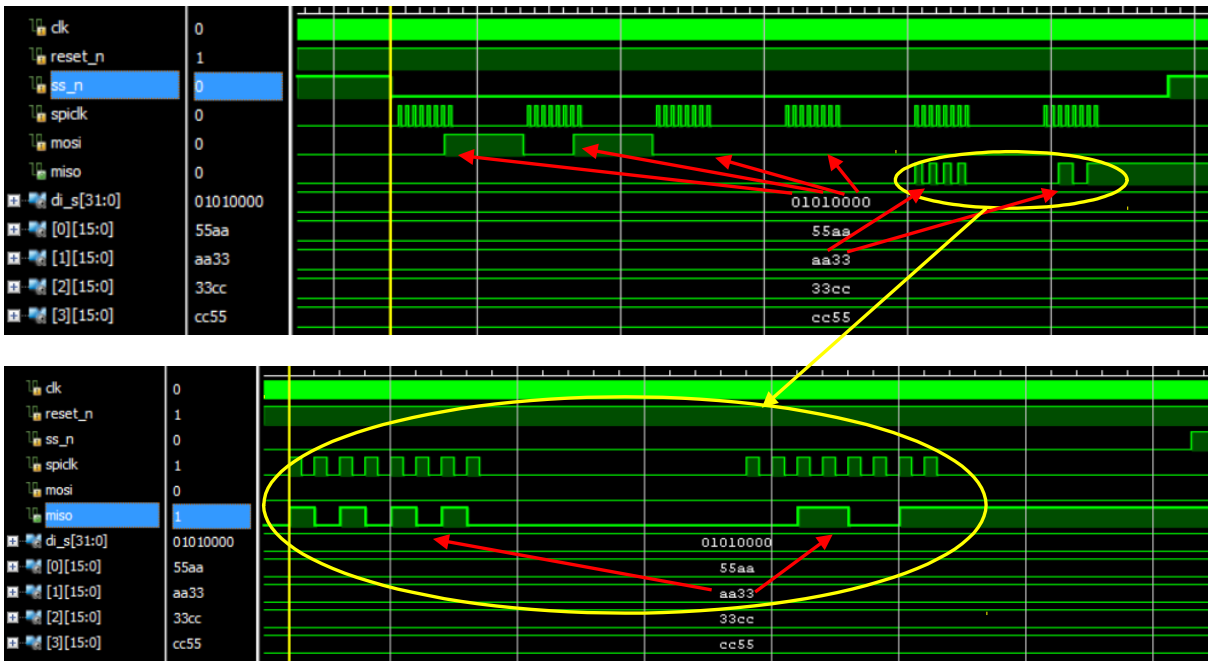


Figura 35. Anàlisi temporal de lectura de senyals d'entrada 2

10.6.3. *Input* 3

La comanda de lectura de la entrada de *pins* 3 es rep a través del *pin* MOSI (0x01020000). Els dos *bytes* llegits a continuació de sortida del *pin* MISO son els 0x33 i 0xCC

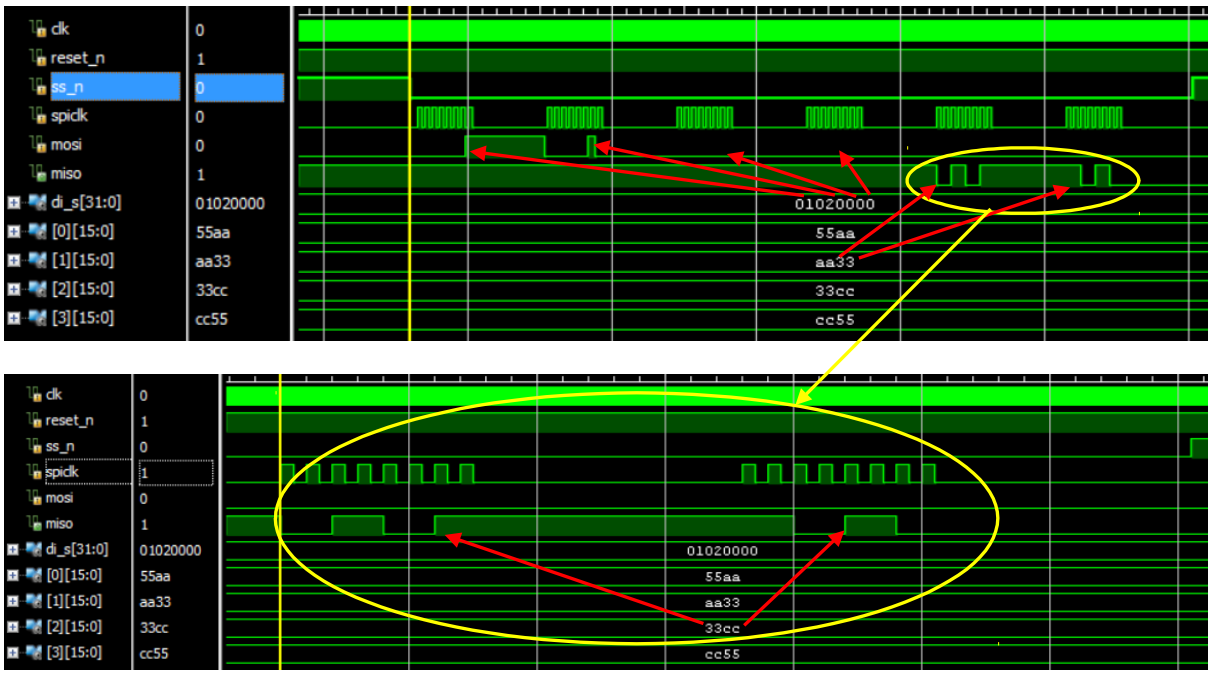


Figura 36. Anàlisi temporal de lectura de senyals d'entrada 3

10.6.4. *Input* 4

La comanda de lectura de la entrada de *pins* 4 es rep a través del *pin* MOSI (0x01030000). Els dos *bytes* llegits a continuació de sortida del *pin* MISO son els 0xCC i 0x55

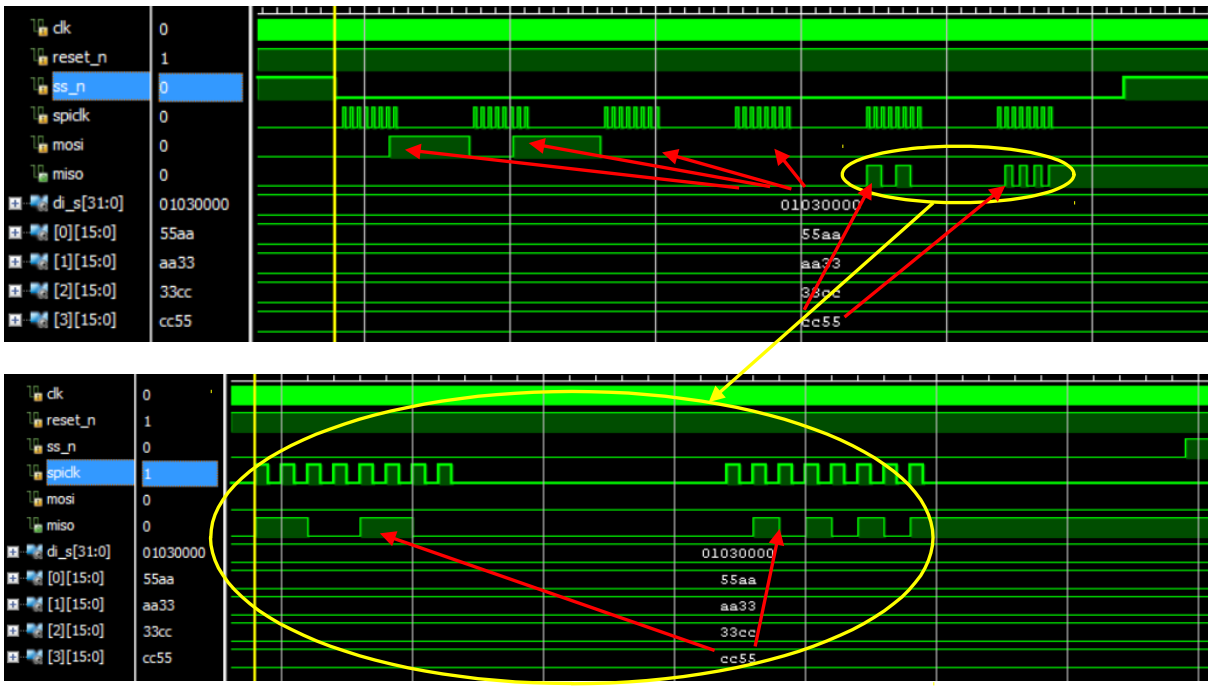


Figura 37. Anàlisi temporal de lectura de senyals d'entrada 4

10.7. Escriptura de senyals digitals de sortida

10.7.1. *Output* 1

La comanda d'escriptura de sortida de *pins* 1 es rep a través del *pin* MOSI (0x02000123). Els dos últims *bytes* corresponen al valor que es desitja en els *pins* de sortida: 0x0123

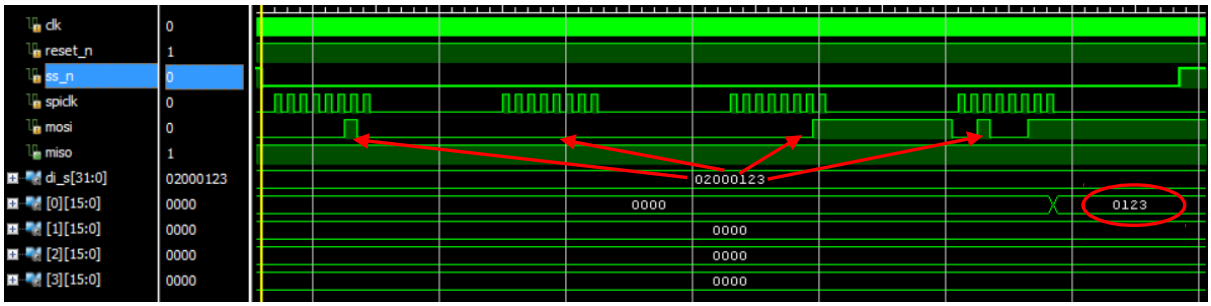


Figura 38. Anàlisi temporal de lectura de senyals de sortida 1

10.7.2. *Output* 2

La comanda d'escriptura de sortida de *pins* 2 es rep a través del *pin* MOSI (0x02014567). Els dos últims *bytes* corresponen al valor que es desitja en els *pins* de sortida: 0x4567

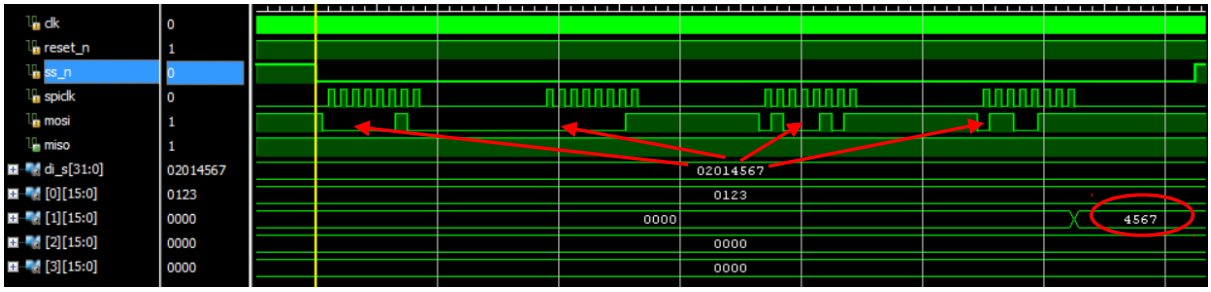


Figura 39. Anàlisi temporal de lectura de senyals de sortida 2

10.7.3. *Output* 3

La comanda d'escriptura de sortida de *pins* 3 es rep a través del *pin* MOSI (0x020289AB). Els dos últims *bytes* corresponen al valor que es desitja en els *pins* de sortida: 0x89AB

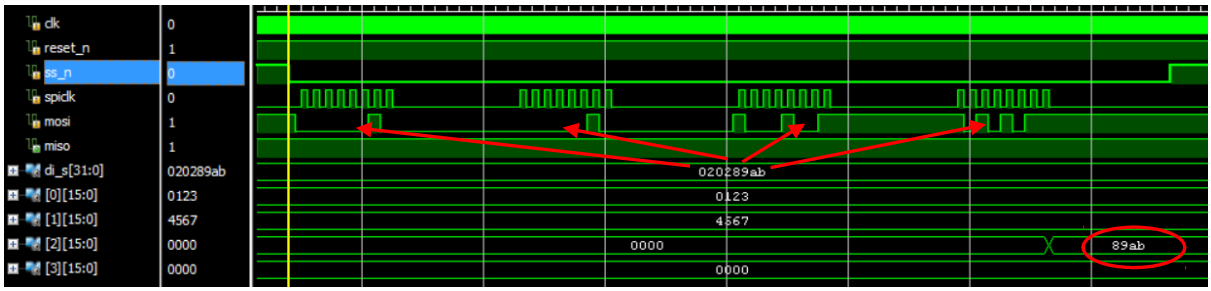


Figura 40. Anàlisi temporal de lectura de senyals de sortida 3

10.7.4. Output 4

La comanda d'escriptura de sortida de *pins* 4 es rep a través del *pin* MOSI (0x0203CDEF). Els dos últims *bytes* corresponen al valor que es desitja en els *pins* de sortida: 0xCDEF

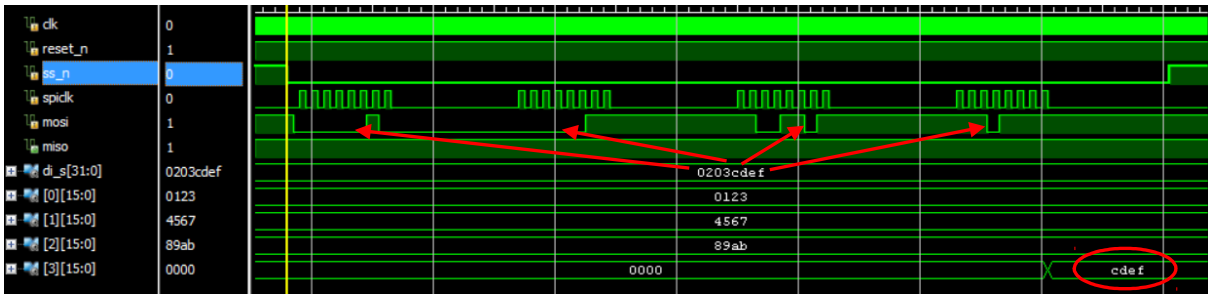


Figura 41. Anàlisi temporal de lectura de senyals de sortida 4

11. Annexes

11.1. Recursos disponibles de codi

Opencores[xxiv]: Es una plataforma online de gestió de continguts per professionals o estudiants sobre dissenys realitzats per FPGAs i per dissenys de ICs. La seva missió principal és l'allotjament de projectes amb aplicacions concretes de codi font de FPGA. Fomenta els principis del Programari Lliure aplicats al llenguatge descriptor de maquinari. Normalment els treballs es publiquen amb llicencies GPL o LGPL. Cal crear un usuari (gratuït) per poder crear un projecte o descarregar-ne un. La seva orientació és el sector professional i els estudiants

FPGACenter[xxv]: Es una plataforma de divulgació informativa sobre aspectes de les FPGAs, des de tecnologies, fabricants, llenguatges de programació, eines de desenvolupament i programació. La seva orientació és el sector professional i els estudiants

fpga4fun[xxvi]: Es una plataforma divulgativa de projectes i aspectes relacionats amb les FPGAs. La seva orientació és més a estudiants o no iniciats en FPGAs.

12. Referències i Annexes

i https://www.altera.com/
ii https://www.xilinx.com/
iii http://www.latticesemi.com/
iv http://www.microsemi.com/
v https://www.altera.com/downloads/download-center.html
vi https://www.xilinx.com/products/design-tools/vivado.html
vii http://www.latticesemi.com/latticediamond
viii http://www.microsemi.com/products/fpga-soc/design-resources/design-software/libero-soc
ix https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/pt/stratix-10-product-table.pdf
x https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/pt/arria-10-product-table.pdf
xi https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/pt/cyclone-v-product-table.pdf
xii https://www.xilinx.com/support/documentation/selection-guides/virtex7-product-table.pdf
xiii https://www.xilinx.com/support/documentation/selection-guides/kintex7-product-table.pdf
xiv https://www.xilinx.com/support/documentation/selection-guides/cost-optimized-product-selection-guide.pdf
xv https://www.xilinx.com/support/documentation/selection-guides/cost-optimized-product-selection-guide.pdf
xvi http://www.latticesemi.com/en/Products/FPGAandCPLD/ECP5.aspx
xvii http://www.latticesemi.com/en/Products/FPGAandCPLD/iCE40Ultra.aspx
xviii http://www.microsemi.com/products/fpga-soc/fpga/igloo2-fpga#product-tables
xix http://www.microsemi.com/products/fpga-soc/radtolerant-fpgas/rtg4
xx http://www.microsemi.com/products/fpga-soc/soc-fpga/smartfusion2
xxi https://www.xilinx.com/support/documentation/selection-guides/cost-optimized-product-selection-guide.pdf#S7
xxii https://www.xilinx.com/support/documentation/selection-guides/cost-optimized-product-selection-guide.pdf#A7
xxiii http://www.amos.eguru-il.com/vhdl_info/Comparison_of_VHDL_Verilog_and_SystemVerilog.pdf
xxiv http://opencores.org/
xxv http://www.fpgacenter.com/
xxvi http://fpga4fun.com/