Implementació d'un dispositiu de test i comunicacions satèl·lit.

Pedro José Bauzá Ruiz

Resum– La tecnologia de xarxa SpaceWire està destinada a ser utilitzada per la comunicació a bord de naus espacials. Ha estat dissenyada per connectar sensors, unitats de processament, dispositius de memòria i diferents sistemes de bord. L'objectiu d'aquest estudi és implementar un dispositiu basat en FPGA, en concret mitjançant la placa de desenvolupament AVNET Ultra96-V2, que sigui capaç d'enviar i rebre dades usant el protocol de comunicacions SpaceWire. Descriu les característiques claus d'SpaceWire i verifica que els transmissor i receptors SpaceWire implementats funcionen correctament mitjançant l'eina de desenvolupament SpaceWire Link Analyser Mk3, un analitzador d'enllaç d'Star-Dundee que permet monitoritzar el tràfic per una xarxa SpaceWire. Com resultat, s'aconsegueix un dispositiu apte pel testeig i validació d'un bus de comunicació SpaceWire, que segueix l'estàndar ECCS-E50-12A de la ESA.

Paraules clau– SpaceWire, FPGA, Ultra96-V2, transmissor, receptor, SpaceWire Link Analyser Mk3, ECCS-E50-12A

Abstract– SpaceWire network technology is intended to be used for communication onboard spacecraft. It has been designed to connect sensors, processing units, memory devices and different onboard systems. The objective of this study is to implement an FPGA-based device, specifically using the AVNET Ultra96-V2 development board, that is capable of sending and receiving data using the SpaceWire communications protocol. It describes the key features of SpaceWire and verifies that the implemented SpaceWire transmitter and receiver are working properly using the SpaceWire Link Analyser Mk3 development tool, a SpaceWire link analyzer from Star-Dundee that allows to monitor the traffic over a SpaceWire network. The result is a device suitable for testing and validation of a SpaceWire communications bus, which follows the ESA ECCS-E50-12A standard.

Keywords– SpaceWire, FPGA, Ultra96-V2, transmitter, receiver, SpaceWire Link Analyser Mk3, ECCS-E50-12A

1 Introducció

El 4 d'octubre de 1957 la Unió Soviètica posa en òrbita el Sputnik 1, el primer satèl·lit artificial de la història que entrava en òrbita al voltant de la Terra. Aquest satèl·lit seria el començament de la carrera espacial entre la Unió Soviètica i els Estats Units, que ha portat a la humanitat a explorar fora del seu planeta.

Avui en dia la tecnologia és prou avançada per a tenir en funcionament múltiples naus espacials, tant tripulades com no tripulades amb l'objectiu de l'exploració i obtenció de dades, o per millorar la qualitat de vida de totes les persones, amb, per exemple, comunicacions via satèl·lit. És tal la importància d'aquest camp que, a gener de 2022, trobem a l'espai més de 12000 satèl·lits, 8000 d'ells en òrbita i 4000 no operatius.

Aquestes disposen de dispositius que s'encarreguen del maneig de dades. Aquests dispositius poden ser instruments, dispositius de memòria, subsistemes de telemetria i altres sistemes de bord. Tots aquests dispositius necessiten una xarxa de comunicació entre ells, tant enllaços point-to-point com amb l'ús d'encaminadors, que segueixin un cert protocol de comunicació.

Com és evident, al tractar-se d'un camp tan innovador i en constant desenvolupament, és de preveure que el nombre de satèl·lits i naus que s'enviïn a l'espai incrementi en els pròxims anys. És per això que, per facilitar la construcció de sistemes de maneig de dades a bord d'alt rendiment i per ajudar a reduir els costos d'integració del sistema, és necessari crear protocols de comunicació estàndards, per pro-

• E-mail de contacte: pjbauza@gmail.com
• Menció realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Màrius Montón Macián (departament)
• Curs 2021/22

moure la compatibilitat entre l'equip de maneig de dades i els subsistemes a la vegada que es fomenta la reutilització de l'equip de tractament de dades en diverses missions diferents.

És aquí on entra SpaceWire, una xarxa informàtica dissenyada per connectar sensors, unitats de processament, dispositius de memòria i diferents subsistemes a bord de naus espacials, podent adaptar-la a aplicacions particulars amb l'ús d'enllaços punt a punt o encaminadors.

Les FPGAs (Field-Programmable Gate Array) són molt utilitzades en missions espacials degut a la possibilitat de ser programades per realitzar funcions lògiques específiques i encarregar-se de tasques de processament de dades. No obstant això, cal tenir present que no totes aquestes són aptes pel seu enviament a l'espai. L'espai és un entorn increïblement desafiant per les FPGAs. Si una FPGA no està homologada i qualificada per ser enviada a l'espai, la mínima exposició d'aquesta a una radiació ionitzant provocaria intercanvis de bits en els elements de la memòria i vulneraria la lògica de configuració. L'abast d'aquest projecte no cobreix com afecta aquesta radiació a les connexions SpaceWire, però sí que s'usarà una FPGA per la implementació d'aquest.

1.1 Objectius

Aquest projecte es tracta de la implementació d'un dispositiu de test de comunicacions per satèl·lits, seguint un protocol de comunicació SpaceWire [1].

Es desenvoluparà un dispositiu basat en FPGA[2] i un conjunt d'eines per testejar i validar el bus de comunicacions SpaceWire usat en satèl·lits. D'aquesta manera, es buscarà tenir un dispositiu que sigui capaç d'enviar i rebre paquets SpaceWire i es validarà mitjançant un dispositiu [3] d'Star-Dundee, que permet descodificar aquests paquets.

Tot el projecte s'implementarà mitjançant VHDL en una FPGA Avnet Ultra96-V2, usant el software de Xilinx: Vivado Design Suite[4].

Per tant, els objectius globals són els següents:

- Elegir un IP-Core per la codificació i descodificació de dades SpaceWire.
- Implementar aquest IP-Core en una FPGA.
- Testejar i validar que tant el transmissor com el receptor de dades d'SpaceWire funciona correctament.

2 Estat de l'Art

SpaceWire és un estàndard de comunicació digital d'alta velocitat (2 a 400Mb/s) bidireccional, full-duplex, que es caracteritza per la transmissió de les dades en data-strobe [5], és a dir, un senyal de dades sèrie i un senyal estroboscòpic s'envien en dos parells diferencials. Això fa que es tinguin dues línies de senyal que codifiquen tant les dades com el seu rellotge. El rellotge és recuperat al receptor simplement realitzant una XOR dels dos senyals. [6]

L'estàndard [7] va ser publicat el gener de 2003 per la European Cooperation for Space Stadarization (ECCS-E50-12A), redactat per Steve Parkes, director de STAR-Dundee LTD, amb la col·laboració d'enginyers de tota Europa.

Els objectius principals d'aquest estàndard són definir un protocol de comunicació que faciliti la construcció de sistemes de tractament de dades de bord d'alt rendiment, promovent la compatibilitat entre els equips i subsistemes de tractament de dades ajudant a reduir els costs d'integració del sistema, i fomentar la reutilització dels equips de tractament de dades en diverses missions diferents. [8]

D'aquesta manera, l'estàndard de comunicació SpaceWire té diversos avantatges característics que fan que hagi estat adoptat per més de 100 naus espacials [9]; com la baixa complexitat, que comporta un baix nombre de portes lògiques (5000-8000 portes), podent-se implementar en ASIC o FPGA [10].

Per la realització del projecte s'usarà la placa FPGA Avnet Ultra96-V2 [2], una placa de desenvolupament Xilinx Zynq Ultrascale + MPSoC basada en ARM, i el kit de programari Vivado Design Suite [4], un paquet de software produït per Xilinx per la síntesi i anàlisis de dissenys HDL. Aquest programari serveix tant per integrar els diferents IP cores i codis HDL, com per la implementació de tests i la programació del microcontrolador en C, mitjançant el software Vitis, inclòs en el paquet.

3 Metodologia

La metodologia seguida per la realització del projecte és molt clara i es basa en una metodologia de desenvolupament en cascada i incremental.

En cascada perquè permet organitzar el treball en vertical, duent a terme una activitat per fases seqüències i sense ser possible passar a la següent sense verificar l'anterior; i incremental, ja que moltes de les fases suposen afegir funcionalitats.

En un primer moment, es farà una tasca d'investigació sobre SpaceWire i la FPGA que s'usarà en el desenvolupament. Això permetrà, posteriorment, elegir un IP-Core adient i fer proves sobre aquest. Una vegada testejat i comprovat que el seu funcionament és correcte, es podrà passar a afegir enviament de dades codificades en data-strobe de SpaceWire i la seva posterior descodificació, per verificar que el dispositiu és funcional i que s'envien i reben correctament paquets d'SpaceWire.

3.1 Entorn de treball

Com ja s'ha comentat, el dispositiu que es programarà es tracta d'una placa FPGA Avnet Ultra96-V2. A més, s'usarà el kit de programari Vivado Design Suite per tot el desenvolupament del dispositiu.

Aquest programari també segueix una metodologia d'ús en cascada. El desenvolupament des de zero d'un projecte en vivado es divideix en 7 fases:

- **Fase d'integració d'IP-Cores.** En aquesta fase es crea un bloc de disseny on s'implementen tots els IP-Cores tant prefefinits com amb codi d'implementació pròpia. En aquesta fase sols es té en compte la funcionalitat i l'entrada/sortida però no el mapeig amb pins físics de la placa.
- **Fase de simulació.** En aquesta fase es simula el comportament tant dels IP-Cores com del codi HDL generat.

- Un cop verificat que el comportament és l'esperat, es pot procedir amb l'**anàlisi RTL**, on ja entra el flux de senyals digitals entre registres de hardware i operacions lògiques realitzades en aquests senyals i d'aquesta manera poder crear una representació d'alt nivell d'un circuit. En aquesta fase ja es poden definir les restriccions (constraints) que defineixen el mapeig de cada port d'entrada/sortida a pins físics de la placa, donant a aquests el seu IO Standard.

- **Fase de Síntesi.** Transformació del disseny RTL en una representació a nivell de porta lògica.

- **Fase d'Implementació**. Tots els passos necessaris per col·locar i encaminar la llista connexions (netlist) als recursos del dispositiu, dins de les limitacions físiques, lògiques i de temps de disseny.

- **Programació de l'Aplicació.** Usant el software Vitis, inclòs amb el programari, a partir d'una plataforma de hardware generada prèviament a partir del bitstream de la implementació, es programa l'aplicació que s'executarà en el nucli ARM. En el cas d'aquest projecte, simplement s'usa un programa predefinit amb una implementació FSBL (First Stage Bootloader), que configura la FPGA amb el bitstream i carrega una imatge des de la RAM i l'executa. Aquesta imatge s'emprarà simplement perquè es generi el rellotge del sistema.

- Finalment, la darrera fase es basa en la programació de la placa, carregant a la FPGA la imatge d'inici del Sistema Operatiu.

Per finalitzar, per verificar que el dispositiu funciona correctament, s'usarà tant un oscil·loscopi per revisar els senyals, com un dispositiu SpaceWire Link Analyser [11] per descodificar els paquets d'SpaceWire d'una connexió.

4 DESENVOLUPAMENT

4.1 Elecció de l'IP-Core

Un nucli IP és un bloc de lògica o dades que s'utilitza en FPGAs i ASICs. Aquests són elements essencials per la reutilització de disseny i són part de la creixent tendència de la indústria d'automatització del disseny electrònic cap a l'ús repetit de components dissenyats prèviament.

S'ha realitzat una cerca exhaustiva de l'IP-Core idoni per la implementació del dispositiu de comunicació SpaceWire. Existeixen moltes alternatives, però en el nostre cas tenim moltes limitacions que no ens permeten elegir amb tant marge.

La principal contra que tenim amb la majoria dels IP-Cores és el fet de que siguin de pagament. Busquem IP-Cores gratuïts, de manera que la cerca es limita a aquells amb menys funcionalitats implementades, amb manca de comunicació i molt antics. És per això que s'ha realitzat una taula comparativa entre aquells IP-Cores que compleixen els requisits principals per realitzar la implementació.

Si ens fixem en la taula comparativa entre IP Cores diferents de l'Annex A.2, s'han comparat quatre IP Cores (els més interessants trobats) entre ells:

- **OpenSpaceWire v3** [12]: IP d'SpaceWire implementat en CESR (Center for the Study of Radiation in Space Laboratory), a partir del document de la ESA: ECSS-E-50-12A[7].

- **SpaceWire Light** [13]: IP Core bàsic encoder-decoder amb interfície FIFO per SpaceWire.

- **GRSPW_CODEC SpaceWire Codec** [14]: SpaceWire encoder decoder amb una interfície FIFO de 9 bits en cada direcció. Les dades són transmeses i enviades a través d'una FIFO de 9 bits.

- **SpaceWire GRSPW2 Link** [15]: El nucli GRSPW2 implementa un controlador d'enllaç SpaceWire amb suport RMAP i interfície de host AMBA.

Segons la taula comparativa, veiem que els millors IP Cores, amb més funcionalitats i millor documentació són els de Gaisler. Tantmateix, al no tenir compatibilitat amb la nostra placa, s'implementarà l'IP Core "SpaceWire Light".

Aquest IP Core s'ha triat, a més de per les seves funcionalitats implementades i pels seus tests inclosos, per la seva excel·lent documentació. A més, d'entre tots els projectes no és tan antic (darrera actualització el 2018) i segueix a la perfecció l'estàndard ECCS-E50-12A.

El nom de "Light" ve donat, ja que no implementa funcionalitats avançades com RMAP [16] o Routing, així que, si al final es volen implementar, no es podrà fer de manera senzilla com podria ser amb els IP Cores de Gaisler.

L'IP Core s'ha incorporat al software Vivado, i s'ha generat el seu diagrama de blocs, que es pot consultar en la Figura 1.

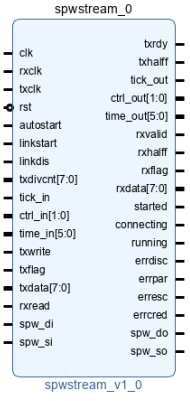


Fig. 1: Diagrama de blocs de SpaceWire Light.

En la Figura 1 es veuen tant els senyals d'entrada com de sortida de l'entitat spwstream, definits a l'Annex A.3.

4.2 SpaceWire Light

SpaceWire Light és un nucli VHDL que implementa un codificador-descodificador SpaceWire, sintetitzable per a objectius FPGA.

Aquest projecte està dirigit a entorns de laboratori, interfícies SpaceWire per a dissenys FPGA personalitzats i interfícies SpaceWire per a ordinadors. L'objectiu del mateix és proporcionar una implementació completa, fiable i ràpida d'un codificador-descodificador SpaceWire segons la norma ECSS-E-ST-50-12. El nucli és lleuger en el sentit que

no proporciona característiques addicionals com RMAP o encaminament de paquets. [13]

4.2.1 Receptor

El receptor descodifica els senyals de data/strobe per a produir una seqüència de caràcters i fitxes de control. La implementació del receptor es divideix en dues entitats. La part frontal (front-end) del receptor detecta les transicions de bits en la línia SpaceWire. Els bits rebuts es transfereixen a la part principal del receptor, que descodifica els patrons de bits en tokens i realitza la comprovació de paritat.

4.2.2 Transmissor

El transmissor pren els caràcters i les fitxes de control i els codifica en senyals de data/strobe. Un enllaç SpaceWire en funcionament mai no és silenciós. Quan no hi ha dades útils a transmetre, el transmissor envia automàticament tokens NULL.

4.3 Codi pel testeig del dispositiu

Mitjançant una entitat programada en VHDL anomenada streamtest, s'ha realitzat un codi pel testeig de l'IP-Core implementat, per comprovar que aquest funciona correctament.

Aquest fitxer es tracta d'una entitat que implementa una instància d'spwstream, una entitat de l'IP-Core que proporciona una interfície amb flux de caràcters i mitjans per l'inici de connexió, l'enviament i la recepció de caràcters i time-codes, a més de la notificació d'errors. Implementa una FIFO de recepció i una FIFO de transmissió, per tant, disposa d'un transmissor i un receptor com els explicats anteriorment i realitza la transmissió dels caràcters prenent aquests de la FIFO i transmetent-los per l'enllaç SpaceWire.

Streamtest es basa a enviar una sèrie de patrons de prova al port de transmissió d'SpaceWire encaminat a la placa i, al mateix temps, realitza un monitoreig de la recepció d'spwstream i verifica que les dades rebudes coincideixen amb el patró de dades transmès. Per tant, aquest streamtest implementa un loopback sempre que connectem els pins de transmissió TX als pins de recepció RX externament.

L'entitat streamtest es basa, principalment, en la inicialització de time-codes i de paquets de dades. La transmissió de time-codes s'activa mitjançant un pols en un senyal "tick_in"en l'entitat spwstream. Aquests codis de temps llavors s'emmagatzemen en el nucli fins que puguin ser transmesos. Per altra banda, un time-code rebut s'anuncia a través d'un pols en el senyal "tick_out".

Per la generació de paquets de dades, s'ha establert una màquina d'estats pel transmissor amb tres estats: idle, on es genera la longitud del paquet; prepare, on es genera el primer byte del paquet; i data, on es generen tots els data bytes i EOP. Aquesta transmissió, com ja s'ha explicat, es fa byte per byte, agafant directament de la FIFO de transmissió. A més, s'ha implementat una manera de "debugar" i tenir cert feedback de què està passant en el dispositiu. Un registre "gotdata", s'activa cada vegada que es rep una dada, per tant, es podrà mapejar aquest registre a un LED perquè realitzi un blinking quan es rebi una dada.

Per la recepció de dades, també s'ha implementat una màquina d'estats de recepció amb dos estats: idle, on s'obté la longitud del paquet esperada; i data, on es llegeixen un a un els bytes de la FIFO de recepció. L'End of Packet (EOP), es marca en la transmissió amb un byte tot a 0 (0x00000000) i EEP (Error End of Packet) es marca amb (0x00000001), per tant, a la recepció de les dades es té en compte, activant un registre "badpacket" en el cas que es produeixi EEP i passant la recepció a estat "idle" en el cas que es rebi un EOP.

4.4 Simulació

A banda d'aquest streamtest, que es tracta d'un testeig de la síntesi mitjançant un loopback, també es disposa d'un testbench "streamtest_tb", pel qual s'ha pogut simular el funcionament de l'entitat streamtest. Aquest testbench realitza les següents proves:

- Una instància de spwstream amb senyals SpaceWire cap a sí mateix.
- Enviament de bytes, paquets i codis de temps a través de l'enllaç SpaceWire.
- Maneig de la desactivació i desconnexió de l'enllaç.

Cal destacar que aquest testbench està destinat a provar la lògica d'emmagatzematge en búfer d'spwstream i no verifica a fons el comportament del receptor, el transmissor i els patrons de senyals. Si es volguessin testejar més a fons tant el transmissor com el receptor s'haurien de realitzar tests especialitzats.

El fitxer de testbench (streamtest_tb) conté diferents processos que simulen diverses funcionalitats. En aquest fitxer es simula un loopback i, per tant, es connecta el senyal de sortida al senyal d'entrada. Això es realitza quan un senyal de control "loopback" s'activa i, per tant, podem tenir un procés que controli en tot moment si, donat un loopback activat, es dóna un error al senyal "linkerror", un pols d'un cicle que s'activa quan es produeix un dels següents errors:

- "errdisc": error de desconnexió detectada en l'estat d'execució que activa un restabliment de l'enllaç i s'esborra automàticament.
- "errpar": error de paritat detectat en l'estat d'execució que activa un restabliment de l'enllaç i s'esborra automàticament.
- "erresc": detecció de seqüència d'escap no vàlida en l'estat d'execució que activa un restabliment de l'enllaç i s'esborra automàticament.
- "errcred": detecció d'error de crèdit que activa un restabliment de l'enllaç i l'esborra automàticament.

També conté un procés que s'encarrega de verificar que es rebin dades regularment quan l'enllaç estigui actiu. Això es realitza mitjançant el monitoratge de la variable "gotdata" cada cert temps (3 ms) quan l'enllaç està obert (senyal de "linkrun" actiu).

A més, un altre procés en paral·lel compta el nombre de caràcters rebuts, monitoritzant la variable "gotdata" i sumant en una variable "s_nrecieved" una unitat cada vegada que "gotdata" s'activa.

El procés principal es tracta d'un testbench general, que testeja el bon funcionament de l'enllaç durant diferents proves activant i desactivant els senyals de "loopback" i canviant el factor d'escala, usat per derivar la taxa de bits de transmissió del rellotge base de transmissió. És a dir, canvia la velocitat de transmissió.

En haver-se realitzat la simulació, aquesta reporta que no s'han produït errors, per tant, estem llestos per realitzar la síntesi i carregar la implementació a la FPGA.

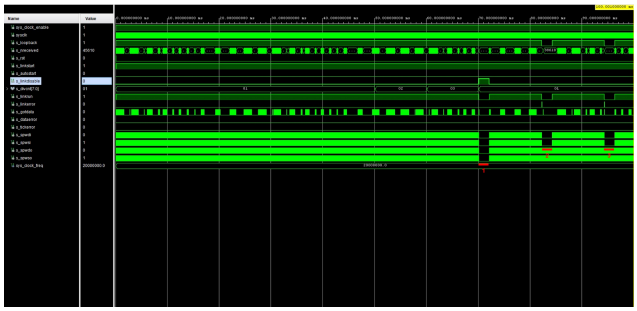


Fig. 2: Simulació streamtest.

Per mostrar una mica els senyals simulats, observem la Figura 2, on es pot veure la simulació d'streamtest completa per 100ms. Com es pot comprovar, tot funciona correctament, el senyal "gotdata" s'activa assíduament i el comptador de dades rebudes continua augmentant. A més, el senyal de "linkerror" no s'activa, cosa que ens indica que no s'estan produint els errors explicats anteriorment.

Malgrat això, ens pot sobtar que la transmissió de les dades es pari (senyals spw a 0), marcat en la imatge amb un "1". Això és degut al fet que el senyal "linkdisable" s'activa, cosa que correspon a un dels tests que es realitzen per comprovar que l'enllaç no falla quan es reactiva. Marcat amb un "2" podem veure com els senyals spw d'entrada estan a 0. El mateix passa en els instants marcats amb un "3". Això és degut al fet que es desactiva el senyal de "loopback", també en el testbench, de manera que es testeja una desconnexió i reconnexió del cablejat en loopback. Si ens fixem, s'activa el senyal de "linkerror", donant a entendre que es detecta l'error de desconnexió en l'estat d'execució (errdisc).

4.5 Programat de la FPGA

Un cop realitzada una simulació del comportament de l'entitat que s'usarà pel test de loopback en la síntesi, podem programar la FPGA per tal de comprovar que el dispositiu es comporta tal com mostra la simulació.

El diagrama de blocs del circuit amb què es programa el dispositiu el trobem en la Figura 3. Aquest es compon de tres blocs principals, corresponents a l'entitat streamtest explicada anteriorment, que inclou l'IP-Core d'SpaceWire Light; a un clock wizard que a partir d'un rellotge de 100MHz treu un rellotge de 20MHz pel correcte funcionament del bloc d'SpaceWire; i un bloc corresponent al sistema multiprocessador en xip (MPSoC) de la placa (Zynq Ultrascale+ MPSoC) usat per generar el rellotge de 100MHz. Algunes de les sortides del bloc d'streamtest s'han mapejat a diferents ports GPIO de la placa per poder analitzarles mitjançant un oscil·loscopi i així realitzar la comparació amb la simulació del comportament.

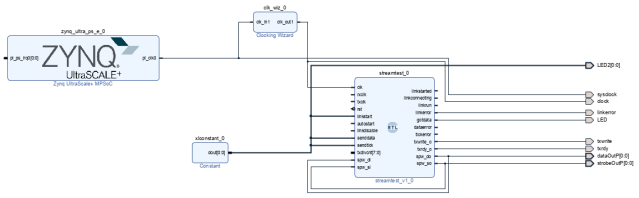


Fig. 3: Diagrama de blocs del dispositiu.

A la Figura 4, mitjançant un oscil·loscopi podem observar la comparació entre un close-up de la simulació i la síntesi. Com veiem, ambdues coincideixen en els 10 cicles de rellotge que es mostren. A més, a la simulació s'afegeix un nou senyal corresponent a la XOR entre el senyal de data i el senyal d'strobe. Com s'ha explicat anteriorment, realitzant aquesta operació lògica, obtenim la xarxa de bits entrants, que correspon a la meitat de la freqüència de rellotge del sistema, com especifica el receptor d'SpaceWire Light.

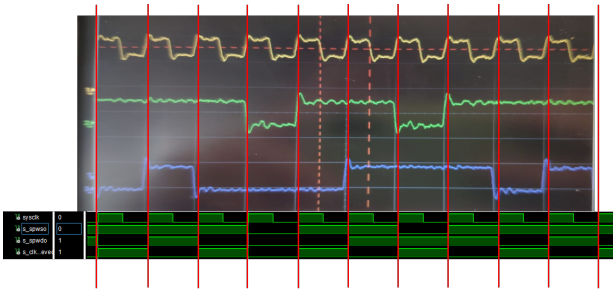


Fig. 4: Comparativa Síntesi - Simulació.

4.6 Enviament de paquets SpaceWire

La informació es transfereix a través d'un enllaç SpaceWire en distints paquets. Els paquets es poden enviar en ambdues direccions de l'enllaç, sempre que existeixi espai en el receptor per més dades.

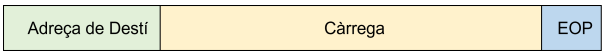


Fig. 5: Format d'un paquet SpaceWire.

L'estructura d'un paquet d'SpaceWire és molt simple i es basa en l'estructura que es mostra en la Figura 5. Aquesta estructura ja s'ha comentat per sobre en les explicacions tant del transmissor com el receptor de l'IP-Core utilitzat,i es compon dels següents camps:

- **Una capçelera amb adreça de destí.** Aquesta és la primera part del paquet que s'envia i és una llista de caracters de dades que representa la identitat del node de destí o la ruta que ha de prendre el paquet a través d'una xarxa SpaceWire per arribar al node de destí. En el cas que un enllaç punt a punt directament entre dos nodes, la direcció de destí no és necessària.
- Un camp per les **dades** que es transferiran des de l'origen al destí, sense limitació de bytes.
- I un **End-Of-Packet** que s'utiltiza per indicar el final del paquet, d'aquesta manera, el caràcter de dades que

segueixi al EOP marcarà el començament d'un nou paquet d'SpaceWire.

En el nostre cas, estem implementant un dispositiu SpaceWire amb connexió punt a punt i connexió directa, és a dir, no disposem d'encaminament ni necessitat d'especificar una adreça lògica ni un camí per l'enviament. Per tant, els nostres paquets d'SpaceWire es tractaran de paquets que simplement disposaran de camp de dades i EOP.

Amb el punt anterior hem pogut comprovar que la síntesi correspon al comportament que s'havia vist a la simulació. Això, no obstant, no ens dóna la seguretat que els paquets d'SpaceWire es codifiquin correctament.

Com tenim la necessitat de comprovar que el dispositiu envia paquets d'SpaceWire correctament codificats i amb el format adient, es modifica l'entitat "streamtest" explicada anteriorment perquè enviï contínuament paquets d'SpaceWire de 8 bytes de dades, corresponent a un seguit de números del 0 al 7 i un byte per l'EOP. Això ens permetrà poder identificar quin caràcter s'està trametent en tot moment tant en la síntesi com en la simulació del comportament, i posteriorment descodificar aquests paquets i comprovar que el dispositiu funciona correctament.

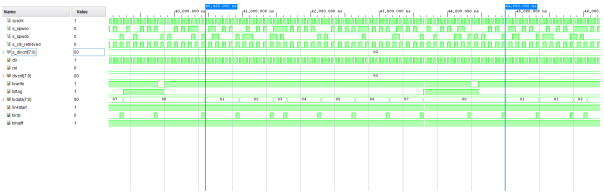


Fig. 6: Simulació del comportament de l'enviament de paquets SpaceWire de la forma (01234567EOP).

Un cop realitzada la simulació del comportament obtenim el waveform de la Figura 6. En aquesta figura podem observar dos marcadors en blau que mostren el començament i final de cada paquet SpaceWire. Com es pot comprovar, cada paquet està compost de 8 bytes de dades (nombres del 0 al 7) i un darrer byte d'EOP. Cal destacar també com es realitza l'enviament de dades en el nostre dispositiu. Per tal d'escriure un byte de dades a la FIFO de transmissió, s'han de complir certes condicions:

- El senyal de "txrdy" ha d'estar en alça. Aquest senyal es tracta d'un senyal de sortida de l'entitat spwstream que indica que l'entitat està llesta per a acceptar un caràcter per a la FIFO de transmissió. És per això que no s'escriu un nou caràcter a la FIFO de transmissió "txdata" fins que txrdy té un flanc de pujada.

- El senyal de "twrite" ha d'estar en alça. Aquest senyal es tracta d'una entrada de l'entitat spwstream i indica que un caràcter es pot emmagatzemar en la FIFO de transmissió sempre que els senyals de "txwrite" i "txrdy" estiguin tots dos alts en el flanc ascendent del rellotge.

Per altra banda, el senyal de "txflag" es tracta d'un senyal de control que indica de quin tipus és el byte que ha d'enviar el dispositiu. Aquest senyal es posa a baixa per enviar un byte de dades, o alta per enviar un EOP o EEP. A la Figura 6 també podem comprovar com aquest senyal "txflag" sempre està a baixa durant l'enviament de dades, però es posa a alça quan s'envia un EOP.

Com s'ha fet en passos anteriors, un cop comprovat que el comportament del dispositiu és l'esperat, podem programar la FPGA i comparar la síntesi amb la simulació anterior.

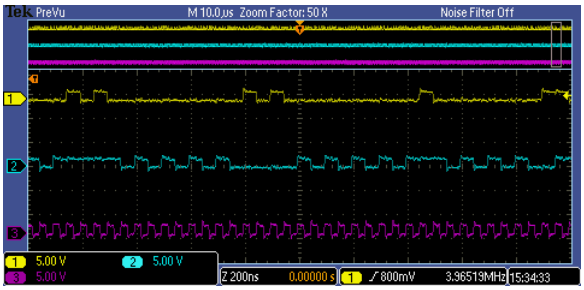


Fig. 7: Senyals single-ended de data, strobe i clock de l'execució de l'enviament de paquets de la forma (01234567EOP).

En la Figura 7 veiem els senyals de data (en groc), strobe (en blau) i el rellotge (en lila) capturats amb l'oscil·loscopi. Si fem un zoom en la zona correcta de la simulació i superposem ambdues execucions (Figura 8), podem comprovar com la síntesi correspon amb la simulació i, per tant, verifiquem l'enviament dels paquets SpaceWire. No obstant això, encara que la simulació i la síntesi coincideixin, no podem assegurar que el transmissor de paquets SpaceWire funciona correctament. Per assegurar-nos haurem de descodificar aquests paquets mitjançant un dispositiu analitzador d'enllaç SpaceWire.

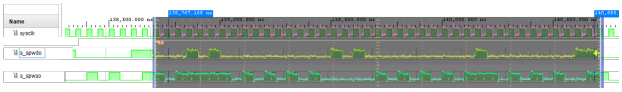


Fig. 8: Comparació simulació-síntesi de l'enviament de paquets.

4.7 Pas a senyals diferencials

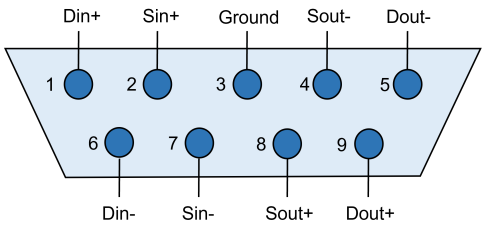


Fig. 9: Distribució de pins del connector SpaceWire.

Els connectors SpaceWire disposen de vuit pins de senyal més un pin de terminació. Aquest tipus de connector tipus D microminiatura de nou pins està qualificat per l'ús espacial[17]. La distribució de pins del connector s'il·lustra en la Figura 9.

Com veiem, el connector disposa de 4 parells diferencials, corresponents als senyals de data i strobe tant d'entrada com de sortida.

Aquest fet ens denota, principalment, que els senyals de data i strobe han de ser diferencials. Això comporta que és necessari passar els senyals que fins ara eren single-ended a senyals diferencials, i mapejar tant les entrades com les sortides a pins de la placa preparats per ser parells diferencials.

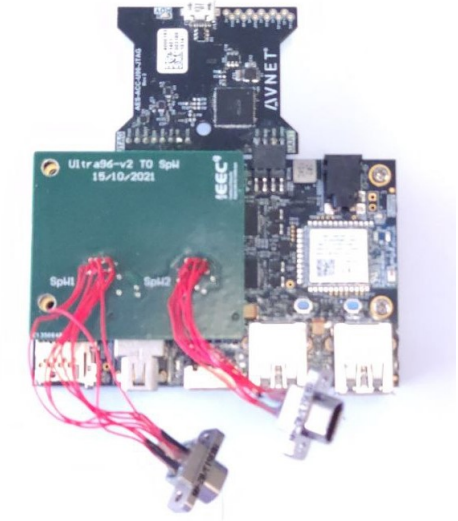


Fig. 10: PCB connectada al mòdul HS del dispositiu.

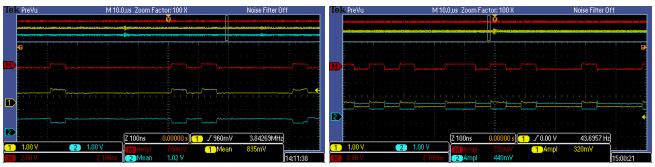
Com SpaceWire es tracta d'un protocol pensat perquè sigui ràpid, usarem el mòdul HS (High Speed) de la placa, on trobem varis parells diferencials. Per aquest motiu s'ha desenvolupat una PCB per connectar a aquest mòdul i poder mapejar les entrades i sortides SpaceWire a un connector Micro-D que compleix l'estàndard, il·lustrat en la Figura 10. L'esquemàtic de la PCB es pot trobar en l'Annex A.1.

Per realitzar el pas a senyals diferencials és necessari afegir un buffer en Vivado que permet fer aquesta conversió. A més, fins ara tots els pins físics del dispositiu tenien el mateix IO Standard (l'estàndard d'entrada/sortida que defineix en quin nivell de voltatge opera la seva interfície i quin tipus de senyal és). Aquest es tractava del "LVCMOS18", l'estàndard per defecte de la FPGA que usem.

Ara, al necessitar senyals diferencials, haurem d'usar iostandards diferents, per tal d'especificar als pins que el senyal és de tipus diferencial. Per aconseguir trobar el tipus correcte, ens hem basat en certes limitacions de voltatge que necessitem per, posteriorment, realitzar l'enllaç SpaceWire amb un analitzador d'enllaç. Segons les limitacions de l'aparell del qual parlarem més endevant, necessitem un nivell de voltatge de l'ordre dels milivolts, aproximàdament una amplitud de 600mV. És per això que, donats els tres iostandards que donen una implementació d'un senyal diferencial en aquest mòdul High Speed comentat, comparem els seus nivells de voltatge:

- **DIFF_HSTL_i_18:** aproximàdament 2.2V d'amplitud.
- **LVDS:** aproximàdament 1.36V d'amplitud.
- **SUB_LVDS:** aproximàdament 720mV d'amplitud.

Podem comprovar que l'iostandard que compleix millor les nostres especificacions és el "SUB_LVDS" i podem veure els senyals diferencials de Data (a la Figura 11a) i Strobe



(a) Data (b) Strobe

Fig. 11: Senyals diferencials

(a la Figura 11b). Cal destacar que el senyal groc correspon al pin positiu del parell diferencial; el senyal blau al pin negatiu; i el senyal vermell és la resta d'amdós senyals.

5 Resultats

5.1 Validació

Per validar que el dispositiu funciona correctament i envia paquets amb seqüències de 8 bytes de dades (01234567) i un byte d'EOP, usem un analitzador de xarxa SpaceWire Link Analyser Mk3 [3]. Aquest dispositiu ha estat dissenyat per monitoritzar el tràfic d'un enllaç SpaceWire. Per tant, és capaç de capturar i mostrar el tràfic bidireccional que es desplaça per un enllaç, podent mostrar estadístiques de l'enllaç. Visibilitzar tot el tràfic d'un enllaç ens dóna la capacitat de testejar i validar que un enllaç SpaceWire funciona correctament.

Fig. 12: SpaceWire Link Analyser Mk3 [18]

El dispositiu que veiem a la Figura 12 disposa de dos connectors entrada/sortida d'SpaceWire. Aquests dos connectors permeten que el dispositiu pugui monitoritzar fins a dos enllaços a la vegada.

El dispositiu està controlat per un ordinador extern, connectat amb una interfície USB 3.0, i totes les dades de l'enllaç es recullen en una aplicació inclosa amb la compra del dispositiu.

5.1.1 Validació del transmissor

Per validar el transmissor, simplement necessitem carregar la síntesi a la FPGA i, mitjançant un cable SpaceWire, connectar el nostre dispositiu amb l'analitzador d'enllaç. Si tot funciona correctament, l'esperat és trobar que es descodifiquen paquets SpaceWire amb la següent estructura: *00 01 02 03 04 05 06 07 EOP*, i que la freqüència del rellotge obtingut de fer la XOR entre el senyal de Data i Strobe sigui

10MHz (la meitat de la freqüència del rellotge del nostre dispositiu).

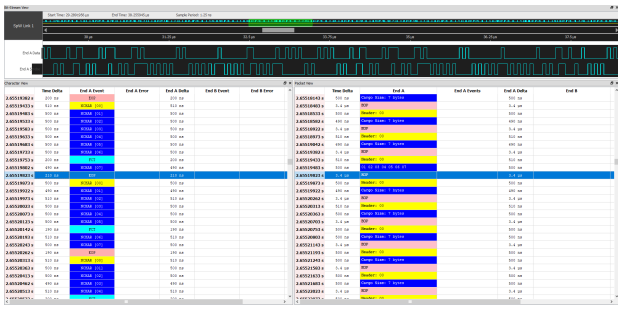


Fig. 13: Vista de caràcter i paquet per un transmissor a 20MHz

A la Figura 13 podem observar tant el senyal de Data com Strobe i unes vistes tant de caràcters com de paquets. A simple vista, es veu que els paquets que arriben corresponen a la seqüència correcta:

- Per començar, en groc, es rep un byte de header corresponent al nostre primer byte de dades *00*.
- A continuació, en blau, res reben 7 bytes de dades corresponents a la seqüència *01 02 03 04 05 06 07*.
- Finalment, en rosa, rebem el byte d'EOP, i, per tant, el paquet rebut verifica que el transmissor funciona correctament.

Un cop comprovats els paquets, comprovem també que la freqüència és la correcta. Per això, calculem el període del rellotge:

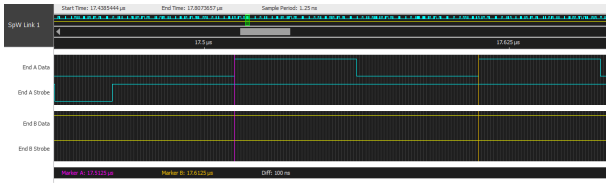


Fig. 14: Període del rellotge recuperat

Com veiem a la Figura 14, el període correspon a 100ns (diferència entre els dos marcadors), i, per tant:

$$f = \frac{1}{T} = \frac{1}{100ns} = 10MHz$$

Així comprovem que la freqüència del rellotge recuperat correspon a la meitat de la freqüencia del rellotge del dispositiu i, per tant, es verifica que és correcta.

5.1.2 Validació del receptor

Un cop validat el transmissor, aprofitarem els dos connectors de la PCB (a la Figura 10), per comprovar que el receptor també funciona correctament i que descodifica els paquets SpaceWire que li arriben.

Per això, realitzem la següent configuració en el nostre dispositiu:

A la Figura 15 trobem la configuració interna final del nostre dispositiu. Ara, com veiem, disposem de dos mòduls diferents, *packet_tx* i *packet_rx*.

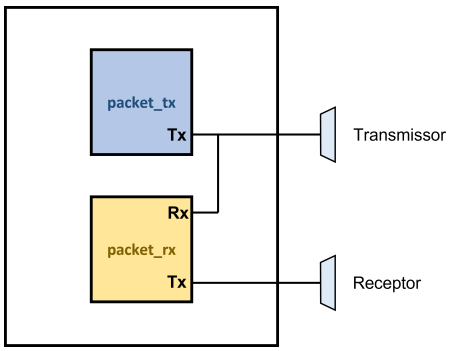


Fig. 15: Configuració final del dispositiu

- **packet_tx**: es tracta del mateix bloc d'streamtest modificat per enviar els paquets SpaceWire amb la seqüència *00 01 02 03 04 05 06 07 EOP*, que ja ha estat validat.
- **packet_rx**: es tracta d'un bloc semblant al transmissor, però que, en lloc de generar paquets de dades i enviarlos, descodifica en paquets els senyals de DataIn i StrobeIn que li arriben, en aquest cas del bloc transmissor, i els reenvia (tornant a codificar els paquets en senyals DataOut i StrobeOut) cap al connector.

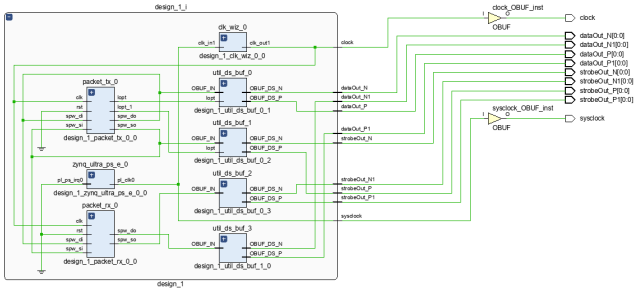


Fig. 16: Esquemàtic de la implementació final del dispositiu

Seguint la configuració de la Figura 15, l'esquemàtic de la síntesi del nostre dispositiu és el corresponent a la Figura 16, i serà aquest el que es carregarà sobre la FPGA.

Amb aquesta configuració, se'ns permetrà validar tant el transmissor com el receptor alhora. Si connectem l'analitzador d'enllaç al connector de *packet_rx* i veiem que arriben paquets amb la seqüència *00 01 02 03 04 05 06 07 EOP*, assegurarem que:

- El transmissor envia aquests paquets codificats en senyals SpaceWire al receptor.
- El receptor rep aquests senyals i els descodifica en paquets SpaceWire.
- El receptor torna a codificar els paquets en senyals SpaceWire i les envia a l'analitzador d'enllaç.

A la Figura 17 trobem la connexió física entre el nostre dispositiu i l'analitzador d'enllaç. D'aquesta manera si tot és correcte, en l'aplicació que ens permet analitzar els paquets rebuts hauríem de poder veure, tant per l'enllaç A com per l'enllaç B, la descodificació correcta de paquets amb la seqüència *00 01 02 03 04 05 06 07 EOP*.

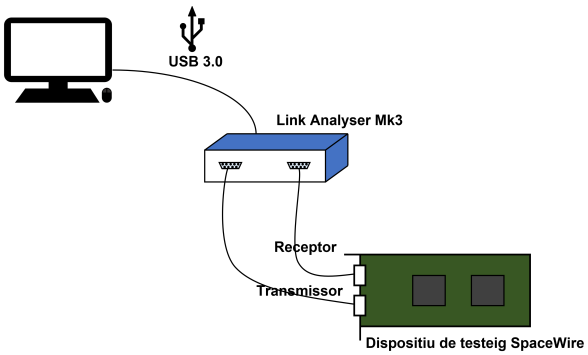


Fig. 17: Connexió PCB - SpaceWire Link Analyser

Fig. 18: Vista de caràcter i paquet per un transmissor i receptor a 20MHz

A la Figura 18 comprovem que els paquets SpaceWire que passen per l'enllaç A (bloc transmissor), com els que passen per l'enllaç B (bloc receptor) corresponen a la seqüència correcta i, per tant, es valida que el receptor implementat també té el comportament esperat.

5.1.3 Transmissió a altres freqüències

Per tal de testejar si el nostre transmissor funciona a diferents freqüències, provem d'alimentar el bloc transmissor amb la freqüència màxima del rellotge del sistema, 100MHz. De la mateixa manera que hem validat el bon funcionament del bloc transmissor, connectem el nostre dispositiu a l'analitzador d'enllaç SpaceWire i comprovem que es reben correctament els paquets.

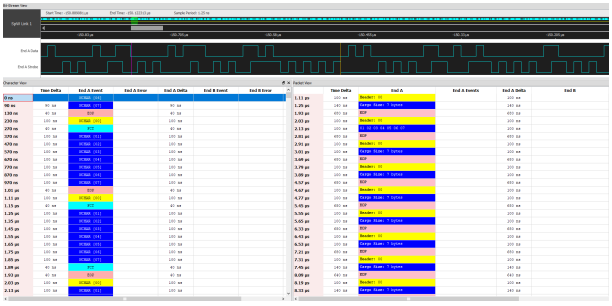


Fig. 19: Vista de caràcter i paquet per un transmissor a 100MHz

En la Figura 19 validem que els paquets rebuts són correctes i, per tant, el transmissor funciona a diferents freqüències. A més, igual que s'ha fet a la Figura 13, podem treure la freqüència de rellotge a partir del seu període.

Com veiem a la Figura 20, el període correspon a 20ns (diferència entre els dos marcadors), i, per tant:

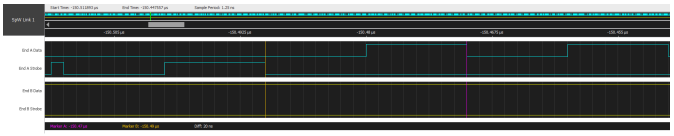


Fig. 20: Període del rellotge recuperat

$$f = \frac{1}{T} = \frac{1}{20ns} = 50MHz$$

Si ens fixem en el camp de "Time Delta", tant de la Figura 13, per 20MHz, com de la Figura 19, per 100MHz, comprovem que, per 20MHz es rep un byte cada 500ns i, per una freqüència de 100MHz, és 5 vegades més ràpid i es rep un byte cada 100ns.

Això ens dóna una velocitat de transmissió màxima (amb 100MHz), d'aproximadament **80Mbit/s**. Això entra dins els barems presentats a l'estat de l'art, però encara està lluny de les velocitats de transmissió més altes (200 - 400 Mbit/s).

6 Conclusions

Com s'ha vist al llarg de l'explicació del projecte, s'han aconseguit complir els objectius principals que s'havien proposat.

En primer lloc, s'han presentat diferents raons per seleccionar un IP Core determinat d'entre diferents trobats mitjançant una cerca exhaustiva.

En segon lloc, s'ha implementat aquest IP Core a la FPGA mitjançant el paquet de programari Vivado/Vitis, realitzant codis HDL amb VHDL que ens permetessin simular i testejar els diferents mòduls.

Per últim, s'ha aconseguit testejar i verificar que el dispositiu funciona correctament i que és capaç de transmetre i rebre senyals SpaceWire, a la vegada que permet codificar paquets en senyals o descodificar senyals en paquets SpaceWire.

Tot això ha estat realitzat fent servir nombroses eines diferents, com poden ser els diferents programaris, l'analitzador d'enllaç SpaceWire Mk3, oscil·loscopis per comprovar que les sortides eren les correctes i inclús la pròpia FPGA.

Al haver acabat el projecte satisfactòriament, puc destacar que el desenvolupament i testeig de hardware és un procés lent, en cascada i pel que has de passar per moltes fases fins a arribar a la implementació final, però que, quan ho aconsegueixes, és molt reconfortant.

6.1 Següents passos

Una vegada complerts els objectius del treball de final de grau, i partint de la base que ja es té un dispositiu amb un transmissor i receptor SpaceWire funcionals, es podrien implementar certes funcionalitats i millores als mòduls ja validats.

Com ja s'ha comentat en l'explicació de l'IP Core utilitzat, SpaceWire Light, a la Secció 4.2, té el nom de Light degut al fet que no implementa funcionalitats més complexes com RMAP o encaminament. RMAP (Remote Memory Access Protocol) [16] és un protocol que proporciona un medi perquè un node SpaceWire escrigui i llegeixi des de

la memòria dins d'un altre node SpaceWire. Aquest protocol podria usar-se per, per exemple, configurar una càmara perquè escrigui dades d'imatge en àrees de memòria assignades en un dispositiu de memòria massiva.

Un **possible pròxim pas** que podria prendre el projecte seria implementar aquest protocol RMAP en el dispositiu. D'aquesta manera, s'hauria de configurar, ja sigui mitjançant un nou IP Core que l'implementi, o mitjançant un codi propi, un mòdul a Vivado que aporti aquesta funcionalitat als blocs que ja han estat validats. El procediment seria el mateix:

- Trobar o implementar els codis HDL que aportin aquesta funcionalitat.
- Simular i testejar el nou mòdul.
- I verificar que la implementació total és correcta mitjançant l'analitzador d'enllaç SpaceWire Mk3, que també permet veure paquets que usen el protocol RMAP [11].

Una altra **possible millora** seria integrar aquest hardware dins un sistema operatiu Linux. Aprofitant els nuclis ARM de la nostra FPGA, es podria integrar Hardware i Software de manera que es pugui veure el dispositiu transmissor/receptor SpaceWire com un dispositiu extern dins del sistema operatiu, i poder ser capaços d'enviar i rebre paquets SpaceWire mitjançant una interfície gràfica.

AGRAÏMENTS

Agrair al meu tutor Màrius Montón i a la resta d'integrants de l'ICE que m'han ajudat amb els dubtes que em sorgien durant el desenvolupament. A l'Institut de Ciències Espacials per dotar-me d'un espai per treballar i del material usat en el desenvolupament i tests. I a la meva família i amics, per encertar quan deien que ho aconseguiria.

REFERÈNCIES

[1] S. Parkes and P. Armbruster, "Spacewire: a spacecraft onboard network for real-time communications," in *14th IEEE-NPSS Real Time Conference, 2005.* IEEE, 2005, pp. 6–10.

[2] "Ultra96-v2 product brief," https://cutt.ly/vAneDCm.

[3] "Spacewire link analyser mk3 - datasheet," https://cutt.ly/iAnePfn.

[4] "Vivado design suite user guide," https://cutt.ly/eAneZDP.

[5] C. McClements, S. Parkes, and A. Leon, "The spacewire codec," in *International SpaceWire Seminar, ESTEC Noordwijk, The Netherlands*, 2003.

[6] B. Cook and C. Walker, "Spacewire on fpga - challenges and solutions," pp. 13–, 08 2008.

[7] "Links, nodes, routers and networks, in ecss-e-50-12a, 2003, pp. 1–8." https://cutt.ly/rIB2tSZ.

[8] S. Parkes, C. McClements, D. McLaren, B. Youssef, M. S. Ali, A. F. Florit, and A. G. Villafranca, "Spacewire and spacefibre on the microsemi rtg4 fpga," in *2016 IEEE Aerospace Conference*, 2016, pp. 1–8.

[9] D. Roberts and S. Parkes, "Spacewire missions and applications," in *2010 SpaceWire International Conference*, vol. 13, 2010, pp. 431–436.

[10] S. Parkes and P. Armbruster, "Spacewire: Spacecraft onboard data-handling network," *Acta Astronautica*, vol. 66, no. 1-2, pp. 88–95, 2010.

[11] S. Parkes and S. Mills, "Developing and testing spacewire devices and networks," *DASIA 2014-Data Systems In Aerospace*, vol. 725, p. 22, 2014.

[12] "Openspacewire v3," http://spacewire.irap.omp.eu/index.html.

[13] "Spacewire light," https://opencores.org/projects/spacewire_light.

[14] "Grspw_codec spacewire codec," https://gaisler.com/index.php/products/ipcores/71-spacewire.

[15] "Spacewire grspw2 link," https://gaisler.com/index.php/products/ipcores/71-spacewire.

[16] "Rmap explained," https://www.star-dundee.com/spacewire/getting-started/rmap-explained/.

[17] B. M. Cook and C. P. H. Walker, "Spacewire and ieee 1355 revisited," in *Presented at the International Spacewire Conference*, vol. 17, 2007, p. 19.

[18] "Spacewire link analyser mk3," https://cutt.ly/WAneNkT.

APÈNDIX

A.1 Esquemàtic PCB

En la Figura 21 es mostra l'esquemàtic de la PCB corresponent a l'adaptador a connectors SpaceWire.

A.2 Taula Comparativa IP Cores

En la Figura 22 es mostra la taula comparativa entre els diferents IP Cores trobats.

A.3 Taula Senyals spwstream

En la Figura 23 es mostra la taula amb l'explicació de tots els senyals d'entrada/sortida de l'IP Core implementat.

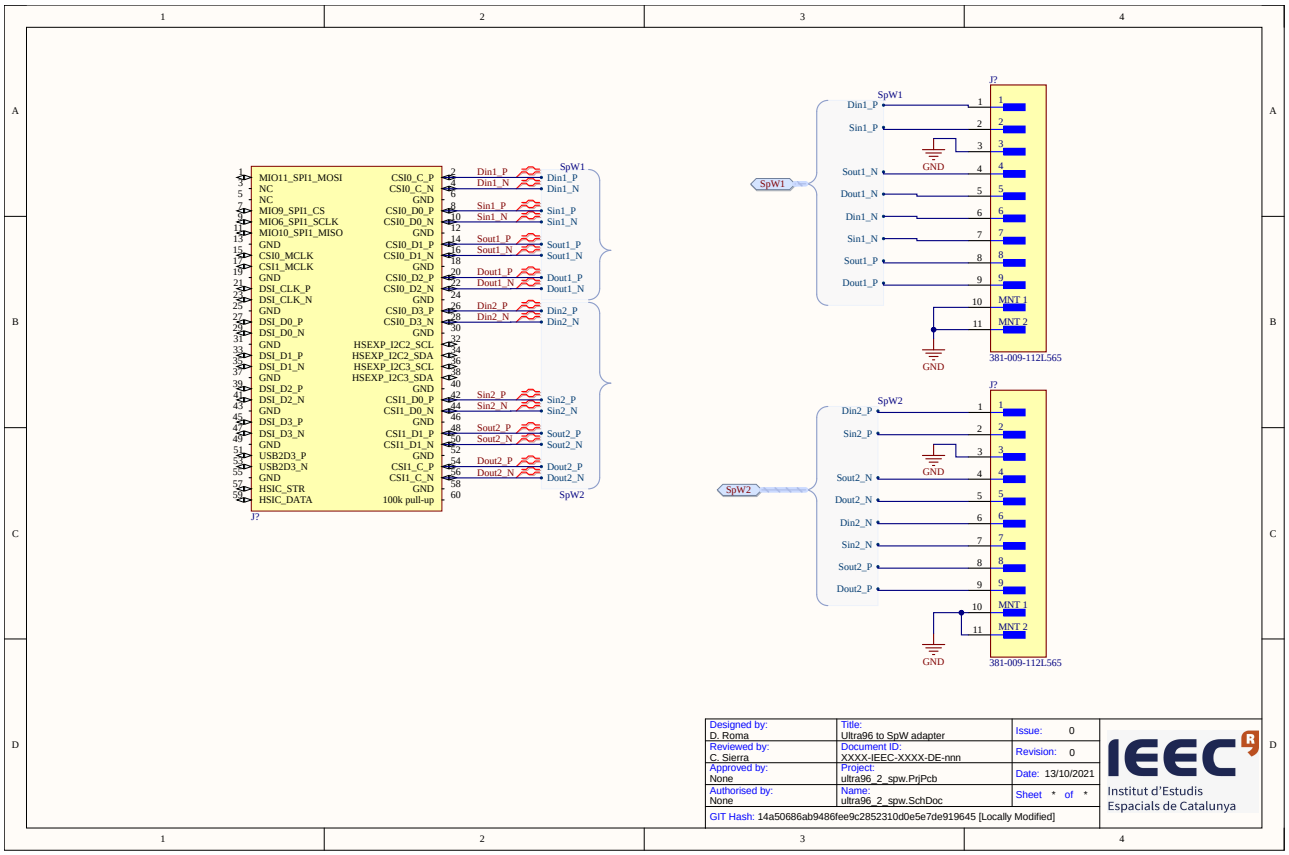


Fig. 21: Esquemàtic de l'adaptador a connectors SpaceWire

IP CORE	Descripció	Avantatges	Desavantatges
OpenSpaceWire v3	IP d'SpaceWire implementat en CESR (Center for the Study of Radiation in Space Laboratory), a partir del document de la ESA: ECSS-E-50-12A	- Gratuït, té llicència CeCILL-C - Disposa de diferents funcionalitats ja implementades com: o TX : transmitter. o RX : receptor. o TX fifo : first in first out buffer for the transmission. o RX fifo : first in first out buffer for the reception. o Two domain clock : interface the signal between different clock. o Time-id buffer : buffer for the time id seeded. o FSM : state machine manage IP. - Incorpora programa de testbench - Bona documentació	- No incorpora funcionalitats avançades com RMAP o ROUTING - Projecte molt antic (2003)
SpaceWire Light	IP Core bàsic encoder-decoder amb interfície FIFO per SpaceWire.	- Projecte relativament actualitzat (Dec. 2018) - Molt bona documentació - Accés FIFO a RX/TX - Interfície AMBA bus - Completament funcional - Gratuït amb GNU General Public License - Testejat en Xilinx Spartan-3 i Virtex-5 FPGA	- No incorpora funcionalitats avançades com RMAP o ROUTING
GRSPW_CODEC SpaceWire Codec	SpaceWire encoder decoder amb una interfície FIFO de 9 bits en cada direcció. Les dades són transmeses i enviades a través d'una FIFO de 9 bits.	- Existeix molta documentació i manuals d'usuari, així com guies per la instal·lació de les llibreries per diferents models de FPGA - Gratuït, GNU - IP Core de Gaisler i, per tant, de bona qualitat - Implementa les funcionalitats bàsiques per transmetre i rebre, també inclou en FIFO	- No s'ha trobat la compatibilitat amb la nostra placa - No incorpora funcionalitats avançades com RMAP o ROUTING
SpaceWire GRSPW2 Link	El nucli GRSPW2 implementa un controlador d'enllaç SpaceWire amb suport RMAP i interfície de host AMBA.	- Existeixen també molts manuals d'usuari i guies per la instal·lació - Gratuït, GNU - És el successor de GRSPW, es basa en el nucli antic i conté totes les seves funcions i característiques, i n'agrega de noves - Suport RMAP i interfície de host AMBA - IP core de Gaisler	- No s'ha trobat la compatibilitat amb la nostra placa

Fig. 22: Taula Comparativa entre els diferents IP Cores

name	dir	description
clk	I	Rellotge de Sistema, actiu en flanc ascendent.
rxclk	I	Rellotge de mostreig del receptor (només quan rximpl = impl_fast).
txclk	I	Rellotge base de transmissió (només quan tximpl = impl_fast).
rst	I	Synchronous reset (active en alt).
autostart	I	Permet l'inici automàtic de l'enllaç en rebre un caràcter NULL.
linkstart	I	Habilita l'inici de l'enllaç una vegada que s'aconsegueix l'estat Llest (anul·la l'autostart).
linkdis	I	No iniciar l'enllaç; desconnectar un enllaç en curs (anul·la el linkstart i l'autostart).
txdivcnt<7:0>	I	Factor d'escala menys 1, utilitzat per a derivar la taxa de bits de transmissió del rellotge base de transmissió. El rellotge base és el rellotge del sistema quan tximpl = impl_generic, o txclk quan tximpl = impl_fast. El rellotge base es divideix per (unsigned(*txdivcnt) + 1). Durant la configuració de l'enllaç, la taxa de bits de transmissió és sempre de 10 Mbit/s independentment d'aquest senyal.
tick_in	I	Pols alt durant un cicle de rellotge per a sol·licitar la transmissió d'un codi de temps.
ctrl_in<1:0>	I	Bits de control que s'envien amb el codi de temps. Han de ser vàlids quan tick_in està en alta.
time_in<5:0>	I	Valor del comptador que s'enviarà amb el codi de temps. Ha de ser vàlid quan tick_in és alt.
txwrite	I	L'aplicació ho posa enlaire per a escriure un caràcter en el FIFO de transmissió. Un caràcter s'emmagatzema en la FIFO sempre que txwrite i txrdy estiguin tots dos alts en el flanc ascendent de clk. Aquest senyal s'ignora mentre txrdy està a baixa.
txflag	I	Bandera de control que s'envia amb el següent caràcter. Es posa baixa per a enviar un byte de dades, o alta per a enviar EOP o EEP. Ha de ser vàlida mentre txwrite estigui en alta.
txdata<7:0>	I	Byte de dades a enviar (txflag=0), o 0x00 per a EOP o 0x01 per a EEP (txflag=1). Ha de ser vàlid mentre txwrite està en alta.
txrdy	O	Alt si l'entitat està llesta per a acceptar un caràcter per a la FIFO de transmissió.
txhalff	O	Alt si la FIFO de transmissió està almenys mig plena.
tick_out	O	Alt durant un cicle de rellotge si s'acaba de rebre un codi de temps.
ctrl_out<1:0>	O	Bits de control de l'últim codi de temps rebut.
time_out<5:0>	O	Valor del comptador de l'últim codi de temps rebut.
rxvalid	O	Alt si rxflag i rxdata contenen dades vàlides (és a dir, quan la FIFO de recepció no està buida).
rxhalff	O	Alt si la FIFO de recepció està almenys mig plena.
rxflag	O	Alt si el caràcter rebut és EOP o EEP; baix si el caràcter rebut és un byte de dades. Vàlid si rxvalid és alt.
rxdata<7:0>	O	Byte rebut (rxflag=0) o 0x00 per a EOP o 0x01 per a EEP (rxflag=1). Vàlid si rxvalid és alt.
rxread	I	Es posa a nivell alt per l'aplicació per a acceptar un caràcter rebut. S'elimina un caràcter de la FIFO de recepció sempre que rxvalid i rxread estiguin alts en el flanc ascendent de clk.
started	O	Alt si la màquina d'estat d'enllaç està en l'estat Iniciat.
connecting	O	Alt si la màquina d'estat d'enllaç està en l'estat Connectant
running	O	Alt si la màquina d'estat de l'enllaç està en l'estat Run, indicant que l'enllaç està operatiu. Si els estats d'inici, connexió i funcionament són tots baixos, l'enllaç es troba en un estat inicial amb el transmissor desactivat.
errdisc	O	Desconnexió detectada en l'estat Run. Activa un restabliment de l'enllaç; s'esborra automàticament.
errpar	O	Es detecta un error de paritat en l'estat d'execució. Activa un reinici de l'enllaç; auto-esborrat.
erresc	O	Es detecta una seqüència de fuita no vàlida en l'estat d'execució. Activa un reinici de l'enllaç; auto-esborrat.
errcred	O	S'ha detectat un error de crèdit. Activa un reinici de l'enllaç; auto-esborrat.
spw_di	I	Senyal d'entrada de dades de l'enllaç SpaceWire al nucli.
spw_si	I	Senyal d'entrada de Strobe des de l'enllaç SpaceWire al nucli.
spw_do	O	Senyal de sortida de dades del nucli a l'enllaç SpaceWire.
spw_so	O	Senyal de sortida de Strobe del nucli a l'enllaç SpaceWire.

Fig. 23: Definició entrades i sortides spwstream [13].