Implementació d'un Sistema RISC-V en *FPGA* amb suport per *Linux*

Informe de seguiment II

Oscar Lostes Cazorla

Desembre 2021

1 Comparativa d'objectius respecte l'informe de seguiment anterior

A l'anterior informe es van exposar les següents fites a curt termini:

- Determinar si el *SoC* per defecte compleix els requisits de recursos com per a cabre a les nostres plaques, i fer els canvis de configuració necessaris si s'escau.
- Creació del component *Qsys* a partir del *SoC*.
- Integració del *SoC* amb la resta de dispositius del sistema.
- Programació efectiva de la *FPGA*, on el sistema fos capaç d'executar algun software (encara que sigui *baremetal* directament des de la BootROM)

D'aquests objectius, només s'han aconseguit de forma efectiva els 2 primers, ja que la integració del *SoC* està comportant alguns problemes.

1.1 Configuració i ús de recursos del *SoC*

Després de les primeres proves de síntesi a partir de la generació per defecte del *RockeChip*, s'observa un ús desmesurat de recursos *LE (Logic Elements)* (de l'ordre del 300%). Un anàlisis de l'ús individualitzat dels recursos per cada component del disseny revela que els components amb més contribució eren les memòries *cache* (tant de primer com de segon nivell). Aquesta observació es confirma experimentalment amb l'eliminació de la *cache* L2 i la reducció al mínim de la *cache* L1 (ja que *Chipyard* no permet eliminar fàcilment la L1). Aquests canvis de configuració porten el disseny a unes dimensions acceptables (de l'ordre del 70%).

Tot i així, aquesta solució dista de ser òptima i no explica per què les memòries disparen tant les dimensions del disseny. Després d'analitzar el codi Verilog generat per *Chipyard*, així com consultant la documentació, s'arriba a la conclusió de que les memòries del disseny no s'estan reconeixent com a tal: *Quartus* està intentant sintetitzar aquests mòduls Verilog a partir d'una descripció comportamental, que no permet que l'eina infereixi que es tracten de memòries, i per tant en fa una síntesi literal amb lògica. Aixo es tradueix en memòries que utilitzen gran quantitat de recursos, ignorant els blocs de memòria disponibles a la *FPGA*.

Preguntant a la *mailing list* del projecte, es conclou que el mètode de generació emprat no és l'adeqüat pel cas d'ús, i es troba l'alternativa correcta. Amb aquest *workflow* de generació, la resta de components es mantenen pràcticament inalterats, mentre que les memòries es generen amb descripcions de *flip-flops* que *Quartus* pot inferir com a memòries correctament. Això resulta en un disseny amb dimensions acceptables (de l'ordre del 50%), fins i tot després d'haver reactivat la L2 i retornat la L1 a quantitats raonables, ja que ara les *caches* utilitzen recursos de memòria (inferior al 10%) en comptes de *LEs*.

Les característiques més importants de la configuració actual (que probablement no serà la definitiva) són:

- 1 RocketCore gran (versió amb totes les capacitats)
- *Cache* L1 associativa de 1 *Kib* i 2 vies (una de dades, i una altra d'instruccions)
- *Cache* L2 associativa de 512 *KiB* i 8 vies
- Un únic port extern (*AXI4*, en aquest cas): el de memòria.

1.2 Component *Qsys*

Importar el codi Verilog al dissenyador de plataformes de *Quartus* ha estat un procés bastant directe: indicar els arxius Verilog involucrats en el disseny, seleccionar la *top-level entity* (*ChipTop*, en aquest cas) i obtenir el component. Després, aquest component es pot portar a *Quartus* sense gaire dificultat i obtenir una síntesi correcta.

Cal destacar una part del procés, que no ha estat tant directa: la generació de les interfícies del component. *Qsys* intenta inferir automàticament les interfícies a partir d'un estàndard de noms dels senyals del mòdul del qual es genera el component. *Chipyard* no utilitza aquests noms estàndard, i per tant aquesta inferència falla, obtenint interfícies incorrectes. Per tant, s'han de re-mapejar els senyals manualment.

1.3 Integració del *SoC*

Obtenir un component *Qsys* amb el *SoC* no és suficient per a tenir el sistema complet. Al voltant d'aquest component s'hi han de connectar una sèrie d'elements externs. El primer que s'intenta integrar (i probablement el més important) és la memòria principal *RAM* del sistema.

Aquesta tasca, al contrari que la generació del component *Qsys*, està resultant problemàtica i és l'actual escull pel progrés del projecte. Unes quantes de les plaques *FPGA* de les que es disposa pel projecte ofereixen accés a memòries SDRAM integrades. Idealment, es vol emprar aquests dispositius com a memòria principal del sistema (en comptes d'emprar recursos de memòria interna de la *FPGA*, com es fa amb les *caches*). El problema és que les plaques que disposen d'aquestes memòries també disposen d'un processador integrat, paral·lelament amb la *FPGA* (*HPS, Hard Processor System*), i la memòria es troba connectada directament amb aquest HPS.

Per defecte, no es pot accedir al recursos del *HPS* des de la *FPGA*. Per a poder utilitzar aquest recursos, s'ha d'emprar el component *Qsys* corresponent al *HPS*. Normalment aquest component s'utilitza quan es farà ús directe del *HPS*, però també permet configurar la cessió de recursos per a la *FPGA*. Amb aquest component, podem obrir (entre d'altres) l'accés a la *RAM* des de la *FPGA*.

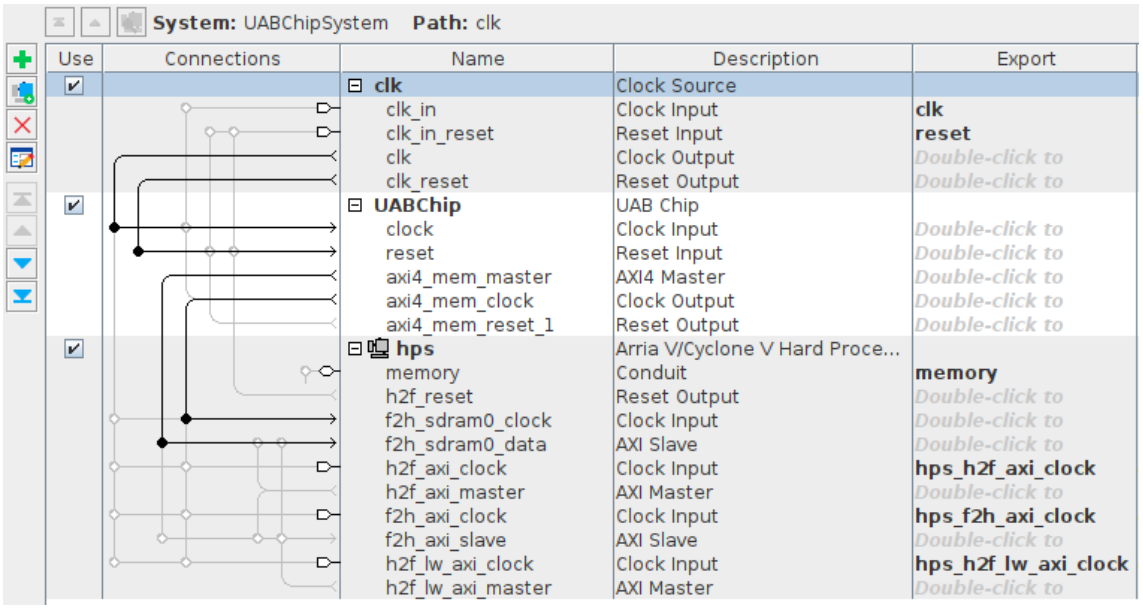

Figura 1: Projecte de `Qsys`

Si bé semblaria que això hauria de ser suficient, el sistema resultant no és sintetitzable. S'ha trobat un exemple de projecte proporcionat per *Terasic*, on queda clar que s'han d'emprar més components i algunes configuracions addicionals. Ara bé, encara no s'ha intentat integrar aquest exemple amb el projecte, i està com a tasca pendent.

2 Següents passos

En base als punts exposat a l'informe, s'actualitza la llista de fites a curt termini:

- Configurar l'accés a la *SDRAM* des de la *FPGA*

- Programació efectiva de la *FPGA*, on el sistema sigui capaç d'executar algun software (es proposa un programa *baremetal* que encengui els *LEDs* de la placa)
- Configurar l'accés a la resta de perifèrics (*UART* i *SD*, com a mínim) des de la *FPGA*

Si s'aconsegueixen aquestes fites, es podrà passar de forma efectiva a la següent fase del projecte, on s'intenti executar *Ubuntu* sobre aquest sistema, tal i com es va plantejar a l'informe anterior.