



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE



Diseño e Implementación de un SoC basado en *RISC-V* en un *FPGA*

Gianluca Vincenzo
D'Agostino Matute

Profesor guía:

Francisco Rivera Serrano

Miembros de la comisión:

Andrés Caba Rutte

Álvaro Silva Madrid



Conclusiones
del trabajo



Presentación
del tema



Motivación y
objetivos



Resultados
obtenidos



Agenda



Marco
teórico



Metodología
de trabajo

Presentación del tema



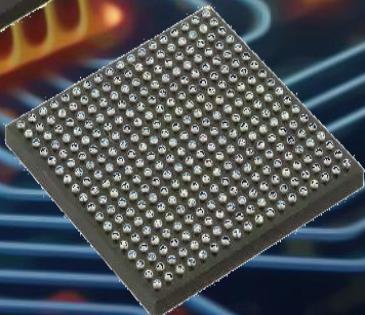
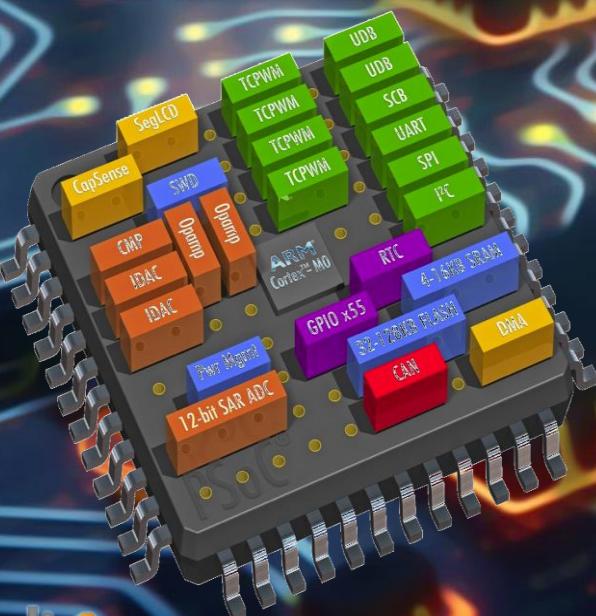
fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

System on a Chip

FPGA

RISC-V



Presentación del tema



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Procesador
basado en
RISC-V

Hardware
mínimo
para un *SoC*
funcional

FPGA

***System
On a
Chip***



Motivación y objetivos



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Objetivos

Diseñar un *System on a Chip* desde la arquitectura de su *CPU*, utilizando el conjunto de instrucciones libre de *RISC-V*, hasta su *hardware* específico e implementarlo correctamente en una tarjeta *FPGA*:

- Utilizar el conjunto de instrucciones *RV32IMF* de *RISC-V* y analizar sus posibilidades
- Diseño de *hardware* mínimo para un *SoC* funcional (entorno de ejecución)
- Utilizar metodología de diseño *Top-Down*
- Utilizar tarjeta de desarrollo *FPGA* y lenguaje *HDL* (System Verilog)
- Implementación de metodologías de verificación



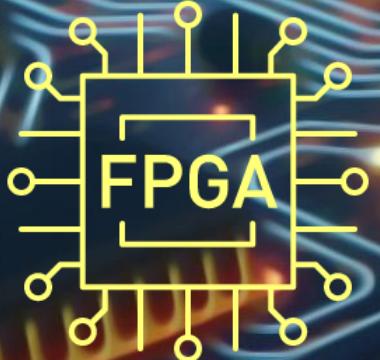
Motivación y objetivos



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE



Contexto mundial y el futuro
del *IoT* junto al *5G* y los *FPGA*





Motivación y objetivos

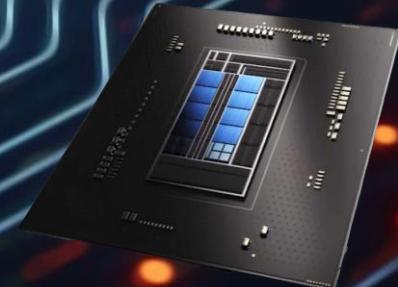


fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE



RISC vs CISC



Motivación y objetivos



die

El impacto de *RISC-V*



European Processor Initiative



Qualcomm

XILINX®

Imagination

SAMSUNG

Raspberry Pi



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE



Motivación y objetivos

RISC-V es la oportunidad de desarrollar tecnología a nivel nacional de forma accesible

Este trabajo busca impulsar el interés académico y crear documentación en español para familiarizar a futuros estudiantes con el proceso de diseño, la metodología e implementación para el desarrollo de este tipo de tecnologías



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE





Marco teórico



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

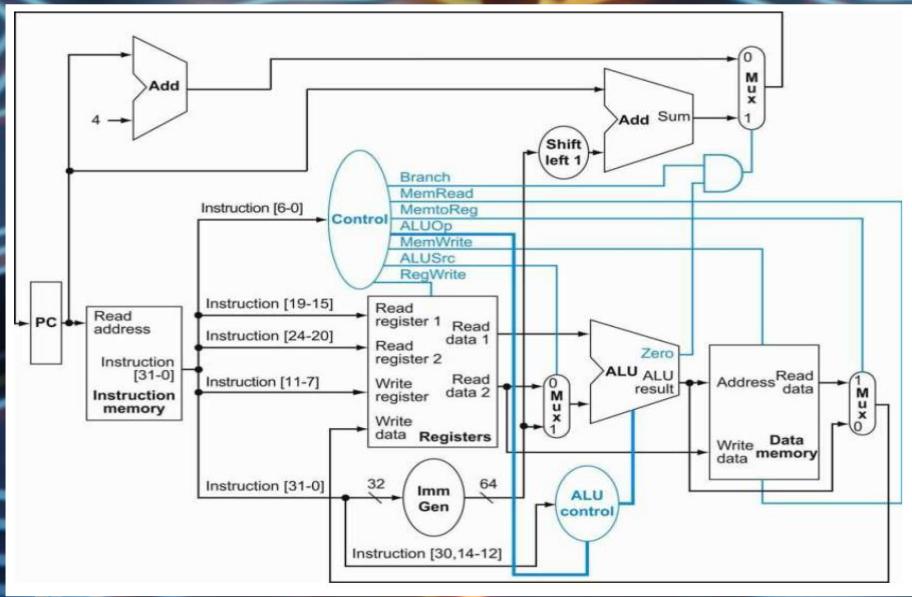
La **CPU** (Central Processing Unit)

- Es el encargado de procesar toda la información en un computador
- Su arquitectura se basa en un **ISA** (*Instruction Set Architecture*)
- Se compone de diferentes unidades funcionales (**datapath**) que son manejadas por un controlador:
 - Las etapas típicas son: **Fetch**, **Decode**, **Execute**, **Memory** y **Write-Back**



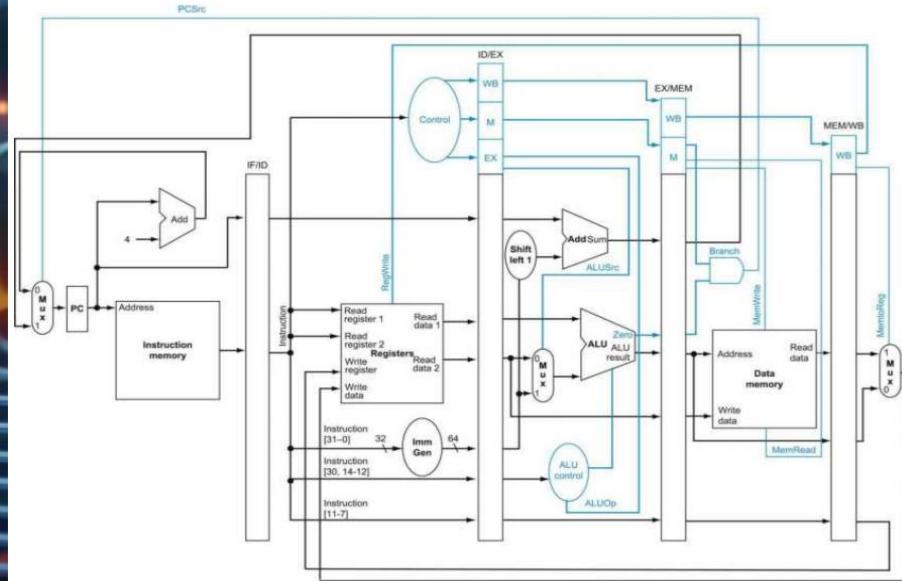
Marco teórico

Single-Cycle



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Pipelined





Marco teórico



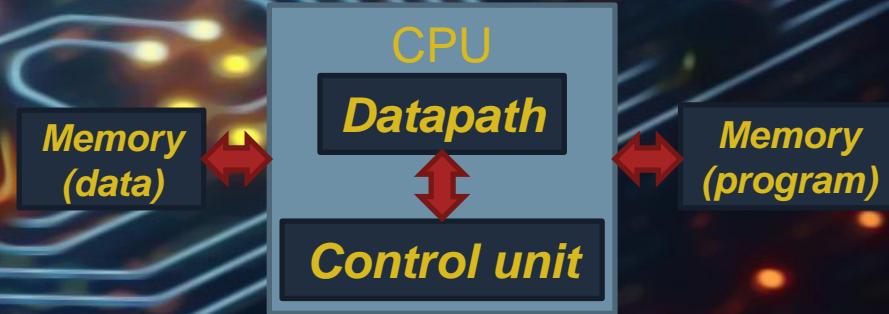
fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Von Neumann



Harvard





Marco teórico

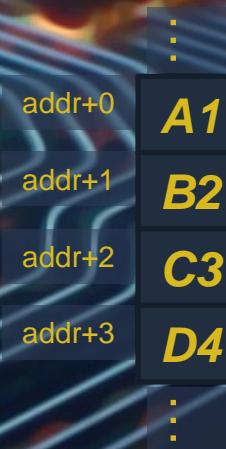


Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Big-Endian



Little-Endian





Marco teórico

El paralelismo multinúcleo

- Por las limitaciones tecnológicas del silicio se han desarrollado arquitecturas más complejas, como las **multinúcleo**
- Debido a estas arquitecturas se conciben los conceptos:
 - **Thread:** hilo de procesamiento a nivel de **software**
 - **Hart:** hilo de ejecución a nivel de **hardware**
- *Performance* de una **CPU** = tiempo de procesamiento medio



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Otros elementos del computador

- **Sistema de memoria:**
 - Almacena la información con la que trabaja el sistema
 - Se jerarquiza debido a las velocidades de los distintos tipos de memoria
- **Dispositivos de E/S:**
 - El medio por el cual se recibe y entrega información (interacción con el sistema)
- **Entorno de ejecución:**
 - Ambiente encargado de administrar los hilos de ejecución de los **Hart** disponibles

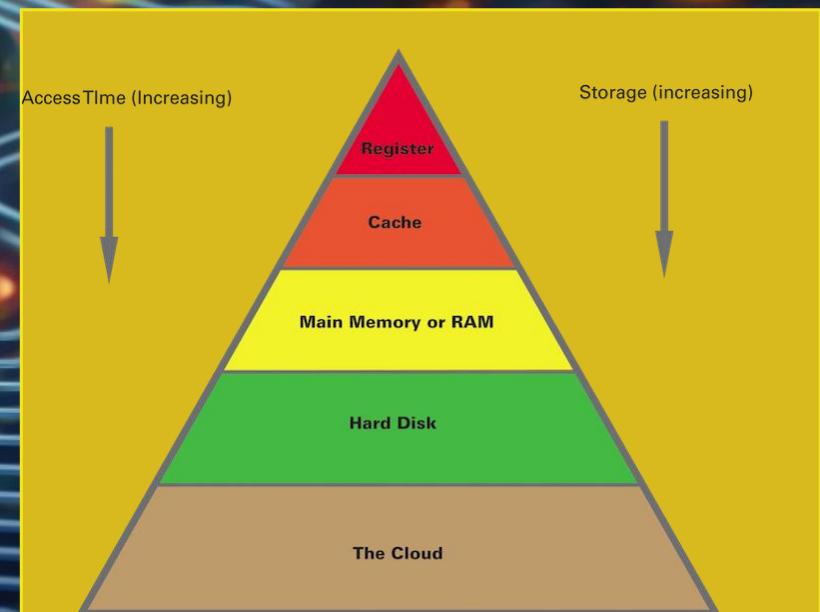


Marco teórico

Dispositivos de E/S



Jerarquización de la memoria



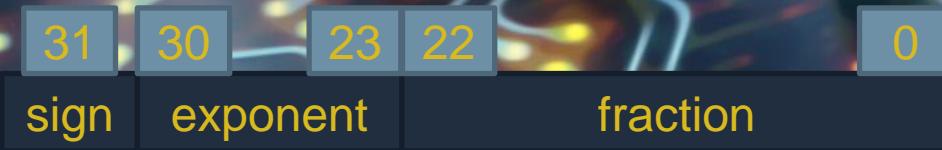


Marco teórico



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Representación binaria en punto flotante de precisión simple



$$X = (-1)^{\text{sign}} * 1.\text{fraction} * 2^{\text{exponent}-127}$$

exponent	fraction	X
= 0	= 0	Cero
= 0	≠ 0	Desnormalizado
= 255	= 0	±Infinito
= 255	≠ 0	NaN



Marco teórico



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Software utilizado

Vivado Design Suite
2020.2 de Xilinx

VIVADO[®]
HLx Editions

2020.2

XILINX[®]

Copyright 1986-2020 Xilinx, Inc.
All Rights Reserved.

die

The screenshot shows the RARS 1.5 assembly debugger interface. On the left is the assembly code window displaying the `fibonacci_inputs.s` file. The code implements a recursive function for calculating Fibonacci numbers using SystemV64 ABI conventions. It includes comments explaining the assembly instructions and their corresponding C code. On the right are two panes: "Control and Status" which shows the state of various registers (zero through t6, pc) and floating-point registers, and "Registers" which lists the same registers with their current values.

```
fibonacci_inputs.s
1 | text
2 |
3 | mmain:
4 |     sw    $0, ($p)           # se guarda $0 en la dirección apunta
5 |     add   $0, $0, $p          # $0 = $p
6 |     addi  $p, $p, -16        # se reserva espacio para 4 words (48)
7 |     li    x17, 5             # syscall 5: GetInt
8 |     ecall                      # Get int: INIT_ARG en a0
9 |     add   a1, x0, a0          # a1 = a0
10 |    sw   a1, -4($0)          # Se guarda a1
11 |    ebreak                     # Simplemente para ayudar a diferenciar
12 |    li    x17, 5             # syscall 5: GetInt
13 |    ecall                      # Get int: MAX_ARG en a0
14 |    add   a2, x0, a0          # a2 = a0
15 |    sw   a2, -8($0)          # Se guarda a2
16 |    # Hay espacio sin utilizar para otra variable en -12($0).
17 |    # Pero el stack pointer debe ser "16-byte aligned" (https://riscv.org)
18 |    loop:
19 |        add   a0, x0, a1          # a0 = a1 -> permite mostrar al (el argumento
20 |        jal   ra, fibonacci      # Se llama la función, recibe arg en a0
21 |        (w   a1, -4($0)          # Se recupera el arg de entrada a la función
22 |        li    x17, 1             # syscall 1: PrintInt
23 |        ecall                      # Print int de la respuesta ya almacenada
24 |        addi  a1, a1, 1           # arg++
25 |        sw   a1, -4($0)          # se guarda en nuevo argumento
26 |        ble   a1, a2, loop       # se guarda en dirección apunta
27 |    done:
28 |        li    x17, -10            # syscall 10: Exit
29 |        ecall
30 |
31 |    # int fibonacci(int n){ if(n <= 2) return 1; else return fibonacci(n-1) + fibonac
32 |    fibonaci:                 # Recibe n en a0 y retorna el resultado en el mismo registro
33 |        # Se almacenan los registros
34 |        sv   $0, ($n)           # se guarda $0 en la dirección apunta
35 |    }
36 |
```

Line: 1 Column: 1 Show Line Numbers

Messages Run I/O

Clear

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000000
\$p	2	0x00000fffc
gp	3	0x000001800
tp	4	0x000000000
t0	5	0x000000000
t1	6	0x000000000
t2	7	0x000000000
\$0	8	0x000000000
s1	9	0x000000000
a0	10	0x000000000
a1	11	0x000000000
a2	12	0x000000000
a3	13	0x000000000
a4	14	0x000000000
a5	15	0x000000000
a6	16	0x000000000
a7	17	0x000000000
s2	18	0x000000000
s3	19	0x000000000
s4	20	0x000000000
s5	21	0x000000000
s6	22	0x000000000
s7	23	0x000000000
s8	24	0x000000000
s9	25	0x000000000
s10	26	0x000000000
s11	27	0x000000000
t3	28	0x000000000
t4	29	0x000000000
t5	30	0x000000000
t6	31	0x000000000
pc		0x000003000

RARS 1.5



Marco teórico



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Hardware utilizado

Nexys 4 FPGA Board





Marco teórico

Investigación del estado del arte



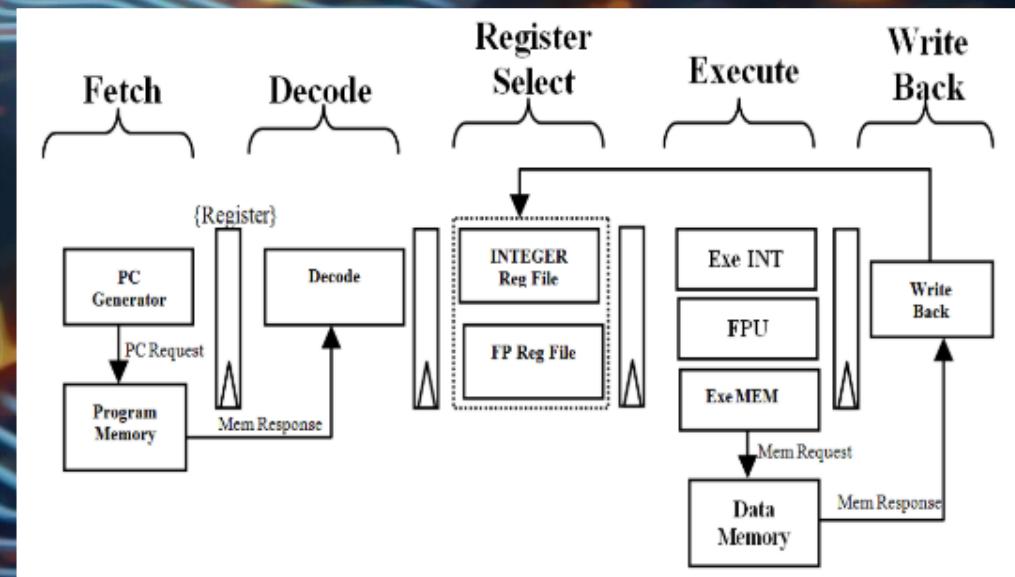
fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

A RISC-V Instruction Set Processor-Micro- architecture Design and Analysis

De este *paper* se destaca:

- La arquitectura de nivel superior
- La metodología de verificación
 - Similar a la utilizada
 - Complementada con *RARS*



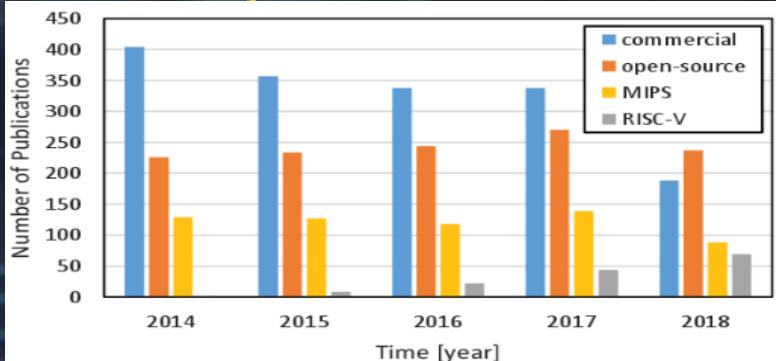


Marco teórico

Investigación del estado del arte

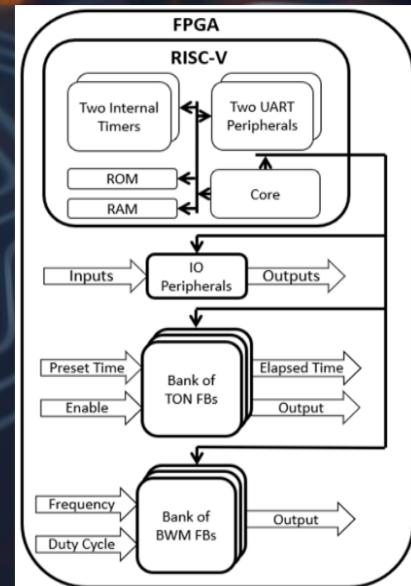
Open-Source RISC-V Processor IP Cores for FPGAs – Overview and Evaluation

Deja en claro la relevancia que ha adquirido *RISC-V* en los últimos años como *hardware libre* y su papel en el desarrollo tecnológico junto a los *FPGA*



RISC-V based implementation of Programmable Logic Controller on FPGA for Industry 4.0

Se presenta la implementación de un *PLC* basado en *RISC-V* (utilizando un *FPGA*) como producto de la industria 4.0 y el *IIoT*





Marco teórico

Investigación del estado del arte

Design and Verification Environment for RISC-V Processor Cores

- Desarrolla un entorno de diseño y verificación orientado a *RISC-V*
- Habla de las ventajas de *RISC-V* y las herramientas disponibles
- El modelo desarrollado consta de:
 - Un emulador **golden reference model**
 - Un **disassembler**
 - Ambos escritos en **Verilog**
 - Si ocurre alguna inconsistencia entre el emulador y el diseño desarrollado el **disassembler** traduce y muestra las instrucciones involucradas



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Co-verification design flow for HDL Languages

- Se destaca el tiempo consumido por la etapa de verificación (60% del tiempo)
- Forma clásica:
 - Se tiene un **System Model** que debe concordar con el modelo *HDL* pero ambos pueden ir cambiando durante el desarrollo
 - Esto propicia la aparición de errores
- Enfoque propuesto:
 - Se enfoca en diseños **fixed-point**
 - Utiliza herramientas y metodologías para obtener el modelo *HDL* a partir del **System Model** junto a un proceso de verificación automática



Marco teórico

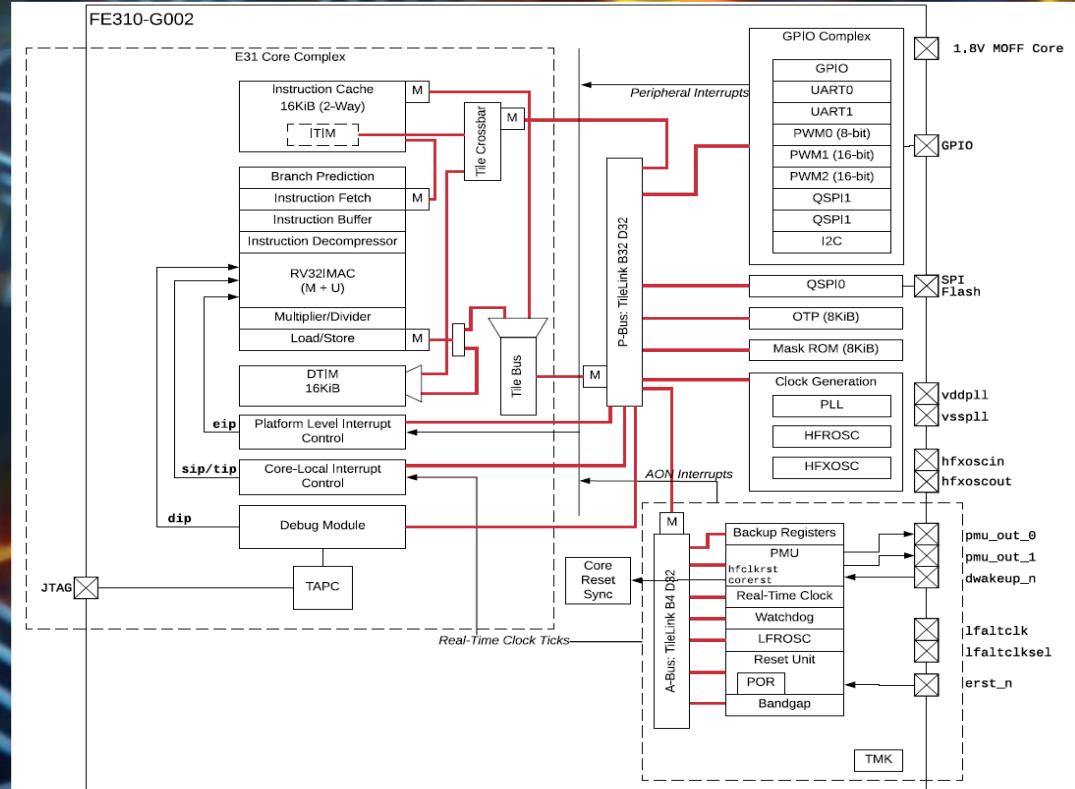
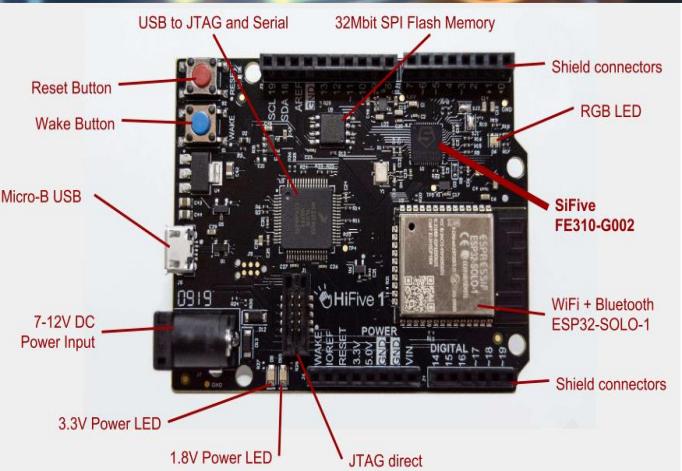
Investigación del estado del arte



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

HiFive Rev B



Metodología de trabajo

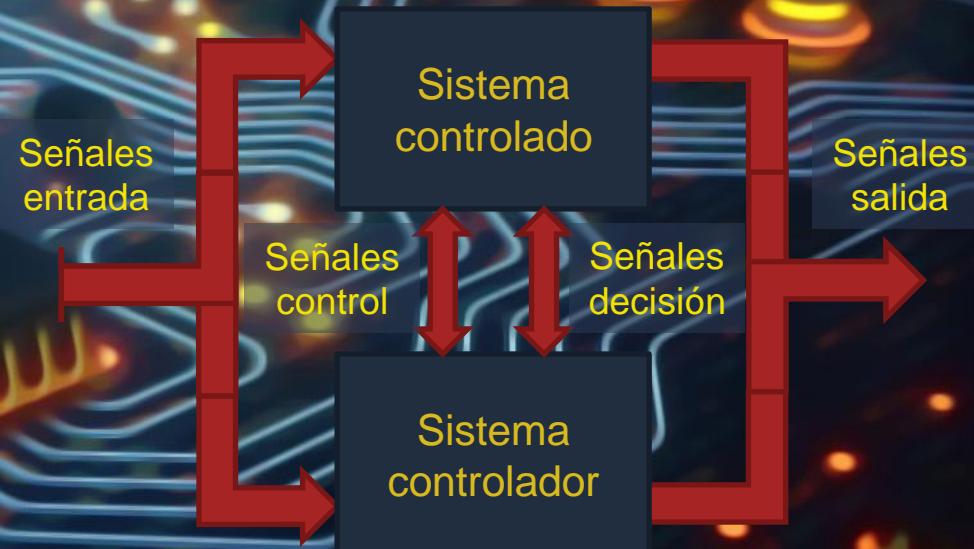
Metodología de diseño *Top-Down*

Características de esta metodología:

- Manejo de diseños grandes y complejos
- Esfuerzos simultáneos e independientes
- Elimina tempranamente problemas serios de la arquitectura
- Identifica tempranamente módulos reusables
- Considera la confección de diagramas de:
 1. Bloques simplificado
 2. Flujo simplificado
 3. Bloques detallado
 4. Flujo detallado
 5. MDS (Mnemonic Documented State)



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE



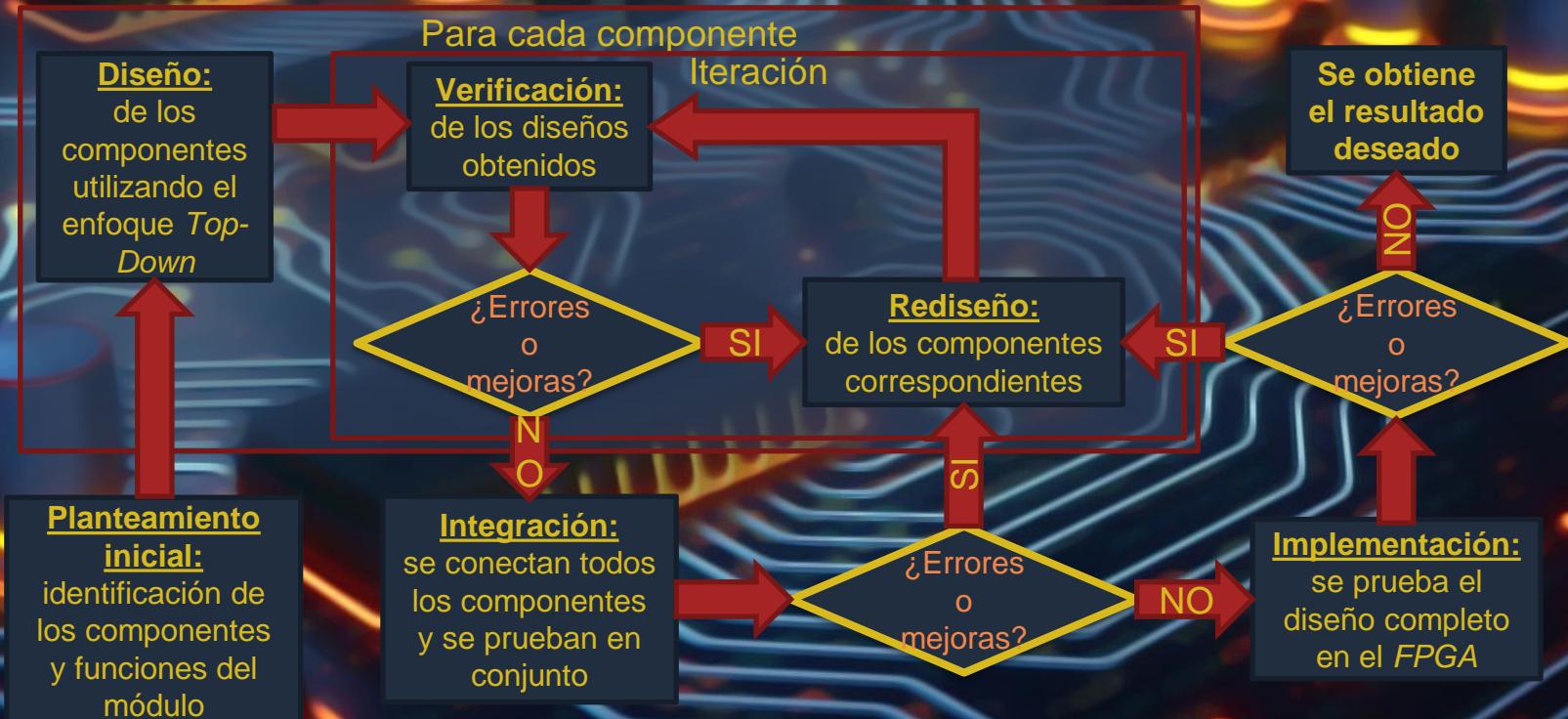
Metodología de trabajo

Flujo de trabajo



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE



Metodología de trabajo

Metodología de verificación

- Se realizan ***Test Bench*** en ***System Verilog*** sobre la ***UUT (Unit Under Test)***
- Cuando es una unidad más compleja (***ALU o FPU***) se generan los ***Test Bench*** mediante un código escrito en ***C*** (entradas pseudoaleatorias)
- Para probar la ***CPU*** se escribe código en lenguaje ensamblador utilizando ***RARS*** para contrastar resultados y realizar la compilación del código en hexadecimal
 - Más la ayuda de un *script* en ***Python*** se carga el programa en la memoria del proyecto en ***Vivado***
 - De todas formas hace falta un ***Test Bench*** para ejecutar las simulaciones del código en ***Vivado***
- En ***hardware*** se comprueba el correcto funcionamiento del ***SoC*** contrastando sus resultados con los de la simulación en ***RARS*** y comprobando la estabilidad del mismo
- Nótese que para poder utilizar ***RARS*** en la verificación, el entorno de ejecución implementado utiliza las mismas llamadas de sistema que dicho simulador

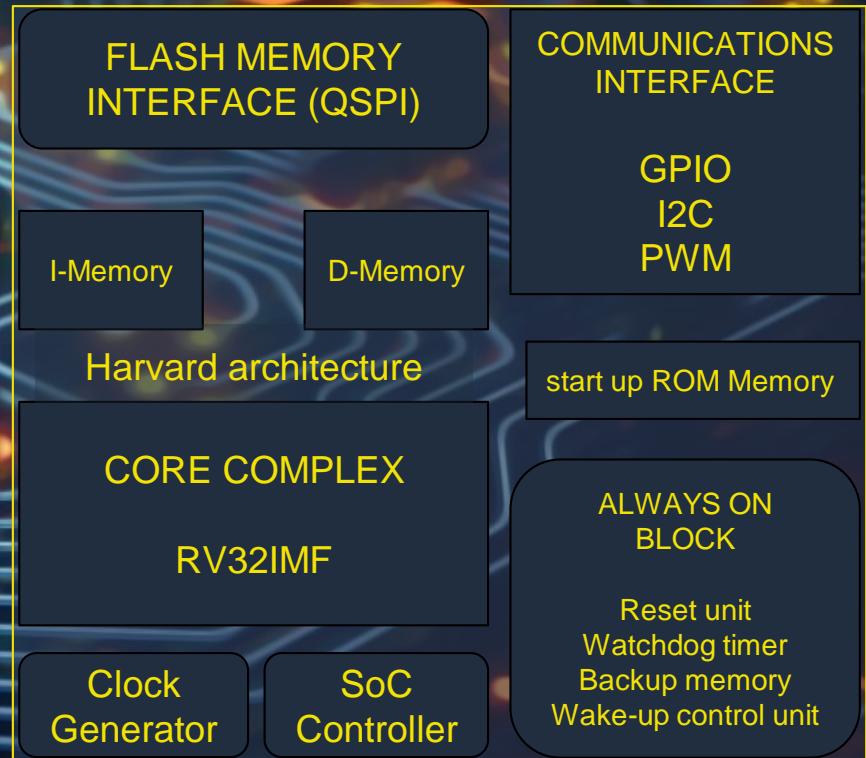


Resultados obtenidos

Diseño inicial propuesto

Siguiendo la metodología de diseño *Top-Down*:

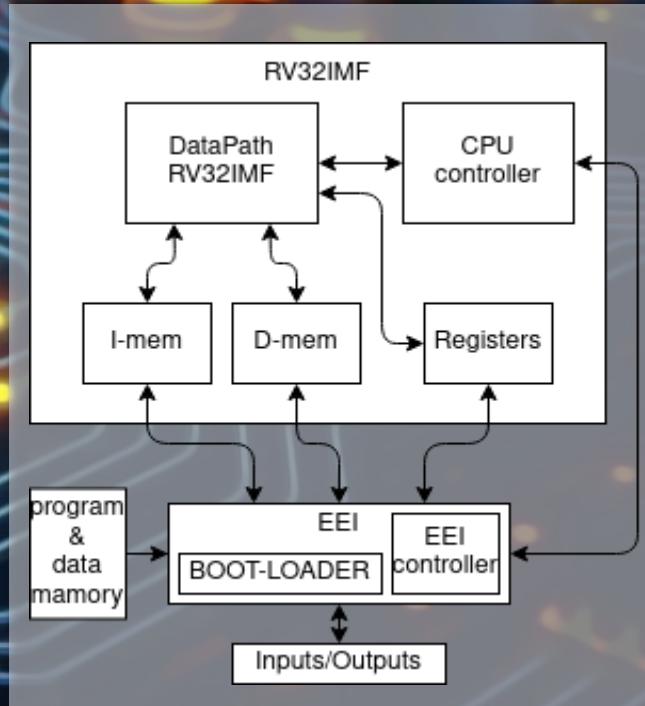
- Primero se plantea un diagrama de bloques de nivel superior del **SoC** a diseñar
- Se identifican los principales bloques
- Se inicia con el diseño del **core-complex** (el resto queda propuesto)
 - Utilizando la metodología *Top-Down*, se realiza una versión **Single-Cycle** y otra **Pipelined**



Resultados obtenidos

Diagrama de bloques simplificado de nivel superior, propuesta final para el *SoC*

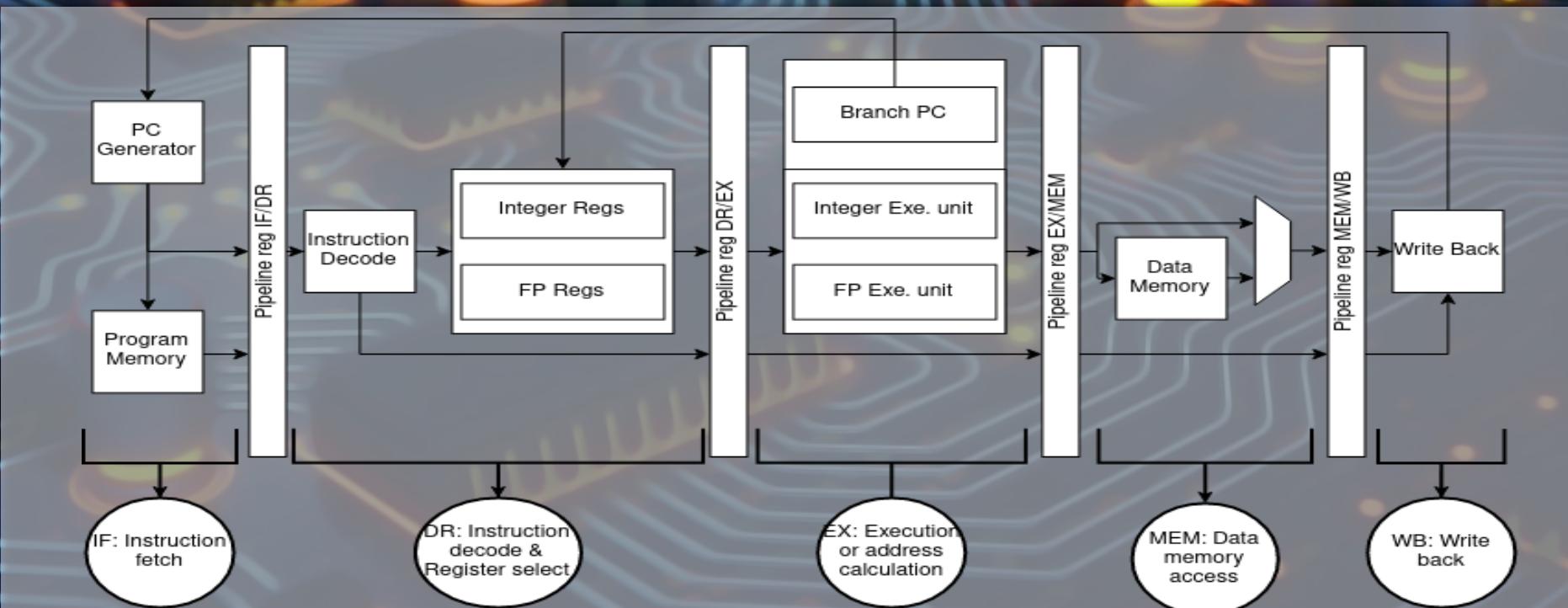
- Debido al tiempo requerido en el desarrollo de la **FPU** se simplifica el diseño propuesto para el **SoC**
- Igualmente, el bloque “**program & data memory**” y “**BOOT-LOADER**” no se implementan





Resultados obtenidos

Diagrama de bloques simplificado del *core complex pipelined*

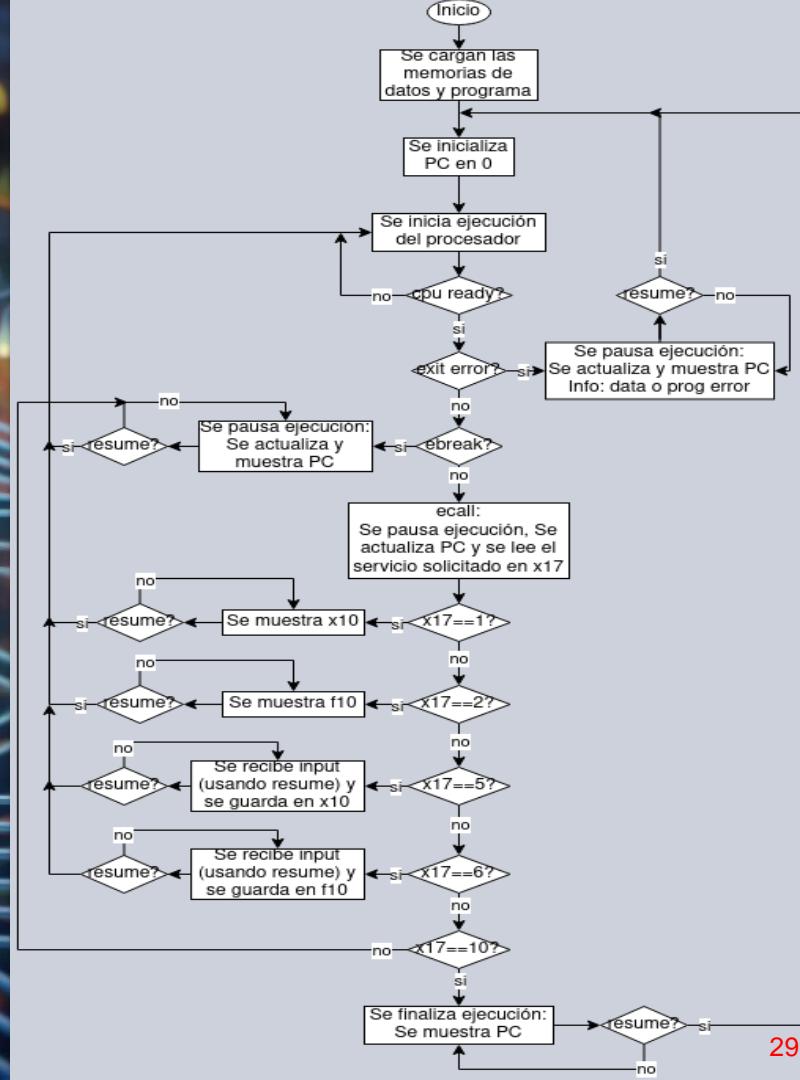




Resultados obtenidos

Diagrama de flujo simplificado del SoC

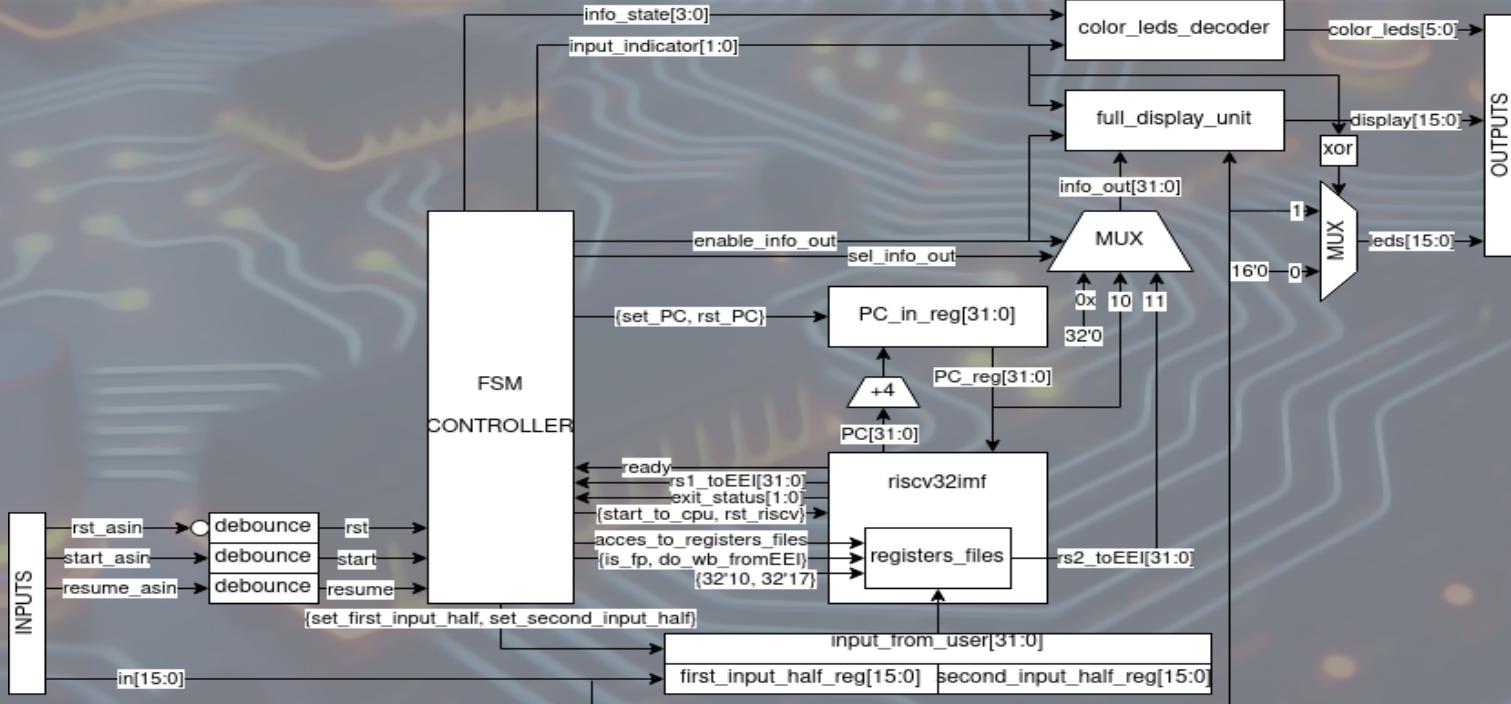
Nótese que este diagrama contempla una fase de carga del programa, sin embargo esta no se implementa debido a que no se agregaron los bloques **"program & data memory"** y **"BOOT-LOADER"** por temas de tiempo





Resultados obtenidos

Diagrama de bloques detallado del SoC

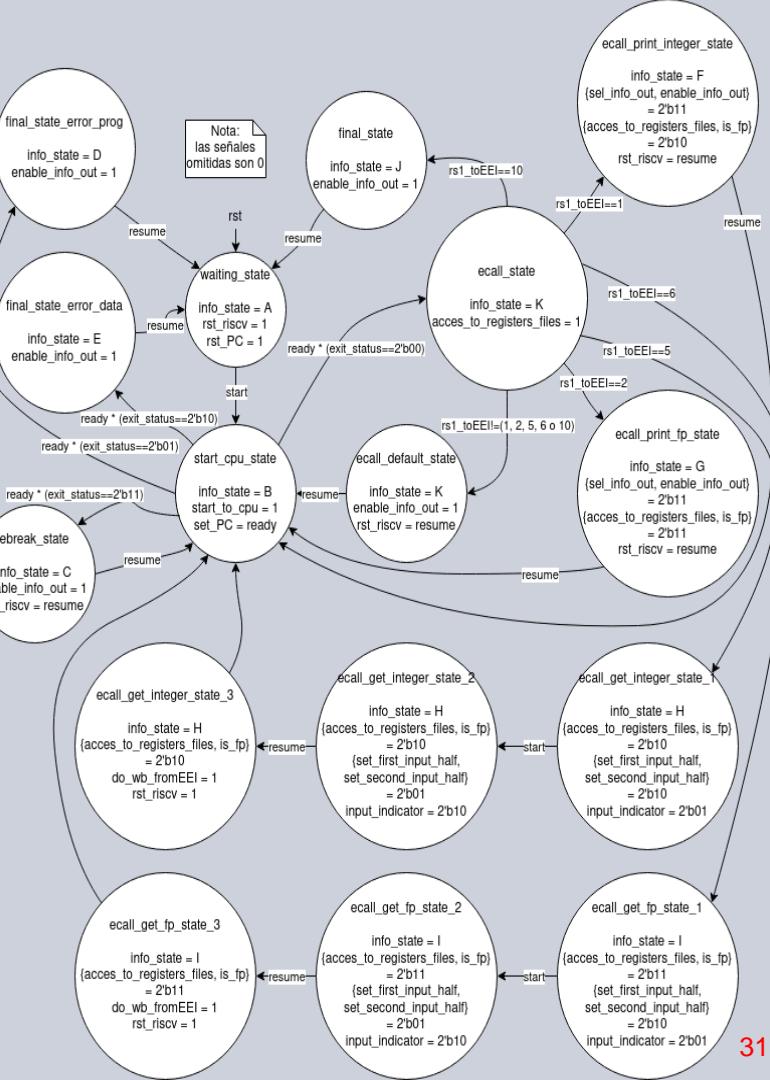




Resultados obtenidos

Diagrama MDS del SoC

- Gracias a la experiencia adquirida durante el trabajo se pasa directamente al diagrama **MDS** sin necesidad de confeccionar el diagrama de flujo detallado
- Para identificar el estado de la ejecución del programa en el *hardware* se utiliza un código de colores (detallado más adelante)





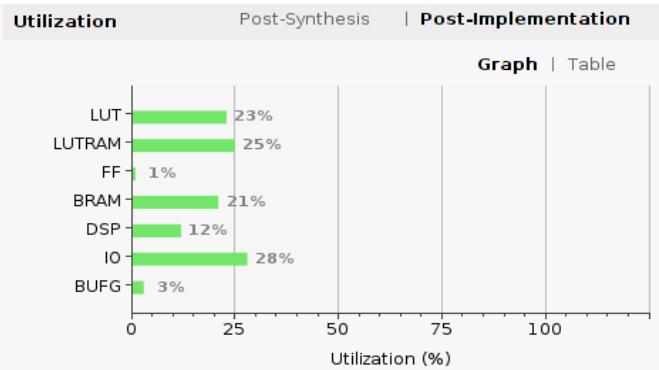
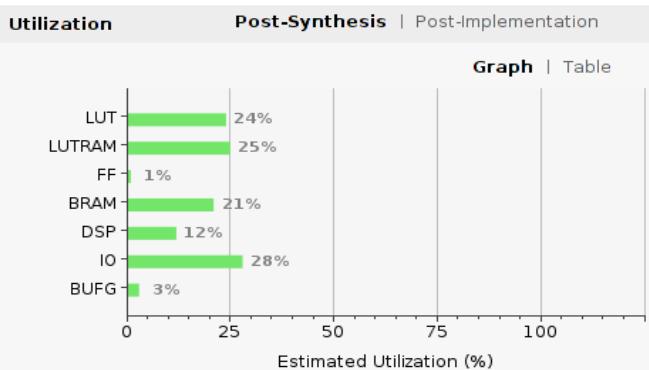
Resultados obtenidos

Recursos del *FPGA*



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE



Utilization | Post-Synthesis | Post-Implementation

Graph | Table

Resource	Estimation	Available	Utilization %
LUT	14978	63400	23.62
LUTRAM	4840	19000	25.47
FF	1313	126800	1.04
BRAM	29	135	21.48
DSP	28	240	11.67
IO	58	210	27.62
BUFG	1	32	3.13

Utilization | Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization %
LUT	14731	63400	23.24
LUTRAM	4840	19000	25.47
FF	1313	126800	1.04
BRAM	29	135	21.48
DSP	28	240	11.67
IO	58	210	27.62
BUFG	1	32	3.13



Resultados obtenidos

Consumo de energía y reloj del diseño

Primer **Warning** (que surge después de la implementación) del proyecto:

- [Power 33-332] Found switching activity that implies high-fanout reset nets being asserted for excessive periods of time which may result in inaccurate power analysis

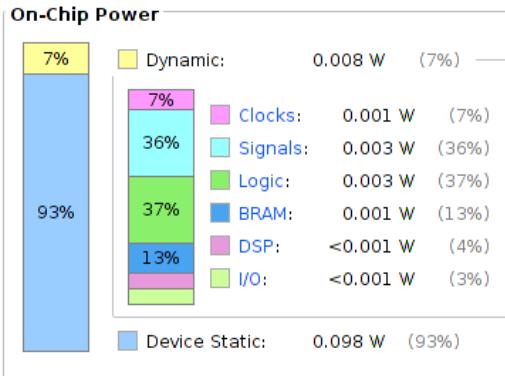
Clock Summary			
Name	Waveform	Period (ns)	Frequency (MHz)
CLK	{0.000 135.000}	270.000	3.704

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	0.106 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	25.5°C
Thermal Margin:	59.5°C (12.9 W)
Effective θJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

[Launch Power Constraint Advisor](#) to find and fix invalid switching activity





Resultados obtenidos



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Sobre la etapa de verificación

Se destacan los *test bench* generados de forma automática:

- **ALU:**
 - Dos números pseudoaleatorios de entrada por *set test*; cada *set test* tiene 22+12 *test*
 - 10 simulaciones de **1000 set tests**
- **FPU – Converter:**
 - Un número pseudoaleatorio de entrada por *set test*; cada *set test* tiene 4*4 *test*
 - 20 simulaciones de **500 set tests**
- **FPU – Arithmetic:**
 - Dos números pseudoaleatorios de entrada por *set test*; cada *set test* tiene 5*4 *test*
 - 20 simulaciones de **500 set tests**

Warnings del diseño

El diseño presenta **93 warning** cuando se genera el *bitstream* y están relacionados con los **multiplicadores** de la **FPU** y **ALU**:

- **37 [DRC DPIP-1] Input pipelining:** se podría mejorar el rendimiento
- **28 [DRC DPOP-1] PREG Output pipelining:** se podría mejorar el rendimiento y el consumo
- **28 [DRC DPOP-2] MREG Output pipelining:** se podría mejorar el rendimiento y el consumo

Resultados obtenidos

Errores hallados en la verificación de la *FPU*



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

```
>>>Test 2758-MUL-RNE <<<
In1: 001010000011110101010110000101100
In2: 00010111101010100110110110111001
Operation: 001
Rounding_Mode: 000
Initial time: 2873690000
Elapsed time: 60000
out: FAILED!!
out should be: 0000000001111100100010110100001
out : 0000000001111100100010110100000
status: PASSED!!
status: 001

>>>Test 3678-DIV-RTZ <<<
In1: 010010011000010100101001000100010
In2: 01010101100010101011110111011111
Operation: 010
Rounding_Mode: 001
Initial time: 2009120000
Elapsed time: 720000
out: FAILED!!
out should be: 00110011011101011011001110100011 out should be: 00110011011101011011001110100011
out : 00110011011101011011001110100110 out : 00110011011101011011001110100110
status: PASSED!!
status: 111

>>>Test 3678-DIV-RDN <<<
In1: 01001001100001010010100100010010
In2: 01010101100010101011110111011111
Operation: 010
Rounding_Mode: 010
Initial time: 2012130000
Elapsed time: 720000
out: FAILED!!
status: PASSED!!
status: 111

>>>Test 3678-DIV-RUP <<<
In1: 01001001100001010010100100010010
In2: 01010101100010101011110111011111
Operation: 010
Rounding_Mode: 011
Initial time: 2015140000
Elapsed time: 720000
out: FAILED!!
status: PASSED!!
status: 111
```



Resultados obtenidos

Algunas simulaciones: testing code imf.s y tb_riscv32imf_top.sv



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Registers		
integer_regs	fp_regs	
00: 00000000	00: 00001ffc	
01: 80001ffe	01: 000017fd	
02: ffffffd	02: 000037f9	
03: 00001800	03: bfe00000	
04: 000007ff	04: bfe00000	
05: 00000535	05: 000081f0	
06: 0029a800	06: 000061f4	
07: fffffffa	07: 80001ffc	
08: 00000ffe	08: 000081f0	
09: 000003fc	09: 7fc00000	
0a: 00000ffe	0a: 80001ffc	
0b: 00001ffc	0b: 00001ffc	
0c: 000017fd	0c: c0440000	
0d: 00001002	0d: c0440000	
0e: 0000007c	0e: 00001ffc	
0f: 00000000	0f: c0670000	
10: 00000ffe	10: c0ab8000	
11: 0000000a	11: c0c00000	
12: ffffffd	12: 4f800000	
13: 7fffffd	13: 00000000	
14: 00000134	14: 00000000	
15: ffffffc	15: 00000000	
16: 0000001f	16: 00000000	
17: 00000000	17: 00000000	
18: 00000020	18: 00000000	
19: 000003fd	19: 00000000	
1a: 0000001b	1a: 00000000	
1b: 00000001	1b: 00000000	
1c: 00001ffd	1c: 00000000	
1d: 00001ffc	1d: 00000000	
1e: 001f4000	1e: 00000000	
1f: 001f40f0	1f: 00000000	

Estado de los
registros del
procesador

Registers		
Floating Point		
Name	Number	Value
zero	0	0x00000000
ra	1	0x80001ffe
sp	2	0xffffffff
gp	3	0x00001800
tp	4	0x000007ff
t0	5	0x00000535
t1	6	0x029a800
t2	7	0xffffffff
s0	8	0x00000ffe
s1	9	0x000003fc
a0	10	0x00000ffe
a1	11	0x00001ffc
a2	12	0x000017fd
a3	13	0x00001002
a4	14	0x0000007c
a5	15	0x00000000
a6	16	0x00000ffe
a7	17	0x0000000a
s2	18	0xffffffff
s3	19	0x7fffffd
s4	20	0x00003134
s5	21	0xffffffff
s6	22	0x0000001f
s7	23	0x00000000
s8	24	0x00000020
s9	25	0x000003fd
s10	26	0x0000001b
s11	27	0x00000001
t3	28	0x00001ffd
t4	29	0x00001ffc
t5	30	0x001f4000
t6	31	0x001f70f0
pc	32	0x000031c4

Resultados obtenidos

Algunas simulaciones: testing code imf.s y tb_riscv32imf_top.sv



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00000080	0x00000000	0x0000ffffd	0x00000000	0x00001ffc	0x00000000	0x00000000	0x00000000	0x3fe00000
0x000000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000000e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Estado de
la memoria
de datos

en	+	data_out.mem
		1000 GB Volume ~/Documents
		00000080: 00000000
		00000084: 0000ffffd
		00000088: 00000000
		0000008c: 00001ffc
		00000090: 00000000
		00000094: 00000000
		00000098: 00000000
		0000009c: 3fe00000
		000000a0: 00000000

Resultados obtenidos

Prueba en hardware: *fibonacci_inputs.s*

```
1 #include <stdio.h>
2 int main(){
3     int a, b;
4     scanf("%d", &a);
5     // Entre cada entrada se agrega un "ebreak", para probar su funcionamiento.
6     scanf("%d", &b);
7     for(int i=a; i<=b; i++)
8         printf(" %d", fibonacci(i));
9     return 0;
10 }
```

```
1 int fibonacci(int n){
2     if(n <= 2)
3         return 1;
4     else
5         return fibonacci(n-1) + fibonacci(n-2);
6 }
```

Versión en C del
código implementado
en *assembly*

Resultado
simulación en
RARS

```
1
9
112358132134
-- program is finished running (0) --
```

In 1	In 2	Dec	1	1	2	3	5	8	13	21	34
		Hex	1	1	2	3	5	8	D	15	22
1	9										



Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

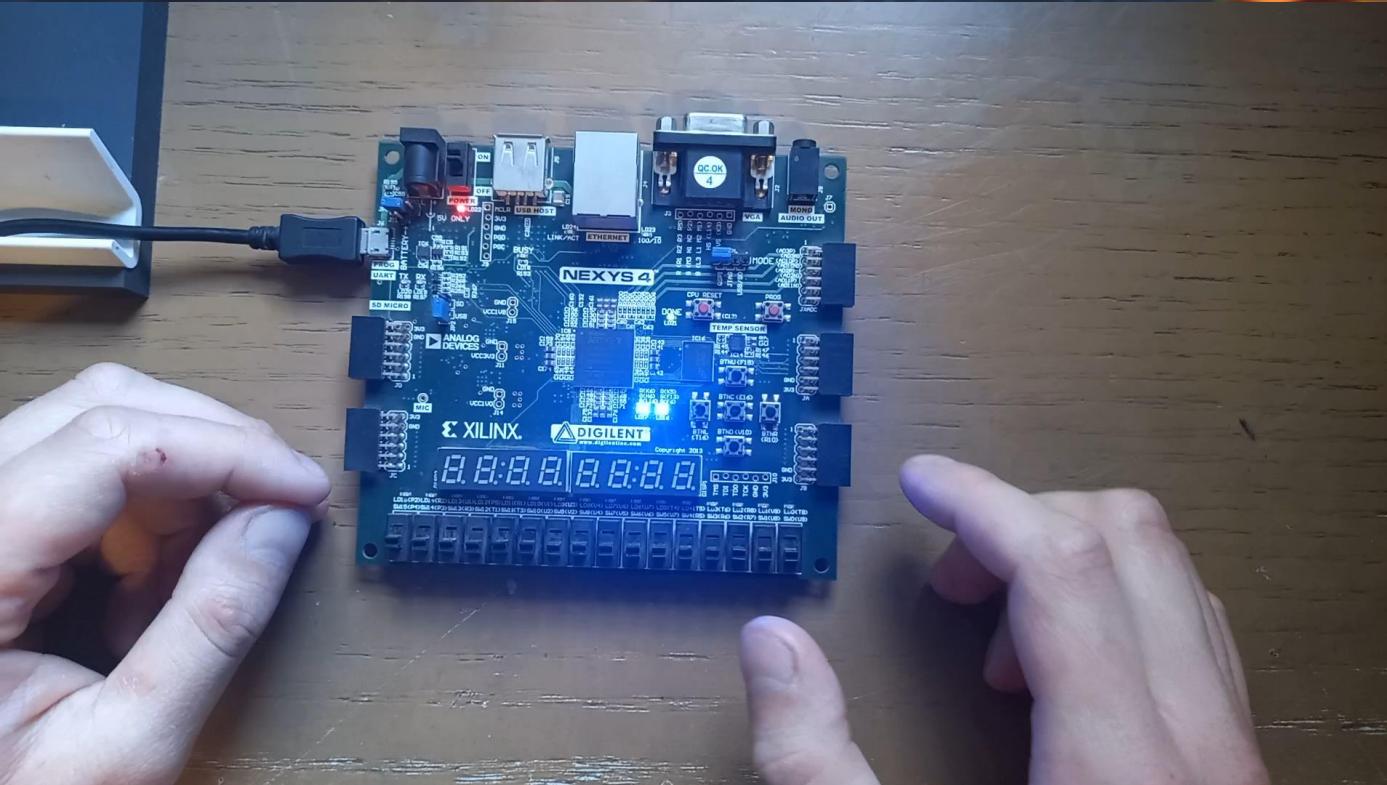
Resultados obtenidos

Prueba en *hardware: fibonacci_inputs.s*



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE



Video de
funcionamiento

★ Conclusiones del trabajo



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

- Se obtiene un **resultado simple y satisfactorio** con respecto a los objetivos planteados
- Ciertamente **no** es un resultado **comercial**, pero **valioso en el ámbito académico nacional**
- Este resultado es un **punto inicial** con **posibilidades de crecimiento**
- Se prioriza la **simplicidad** para **disminuir los errores** y permitir **abordar el reto** en un **tiempo razonable**
- Se ponen en práctica las **metodologías de diseño** de **sistemas digitales** aprendidas en la facultad
- Se **documenta** todo el **proceso de diseño e investigación** de forma amigable para el lector
- Es importante generar **documentación en español** y detallada para la implementación de sistemas basados en el **ISA de RISC-V**

- Se aprecia la **versatilidad** de implementaciones de **RISC-V** y el valor que entrega un **ISA abierto** debido a su impacto en el **mercado** y en la **academia**
- Se adquiere **experiencia** en el diseño de **sistemas digitales complejos** utilizando **HDL**, un mayor **conocimiento** sobre la **arquitectura de computadores** y por sobre todo del rol que juega el **entorno de ejecución**
- Se destaca la **versatilidad** de los **FPGA** para **desarrollar tecnologías**
- Se enfatiza la importancia de revisar el **trabajo previo** de otros autores: ayuda a prever dificultades y **herramientas/metodologías** del trabajo
- Se aprecia la **dificultad** del **tiempo** de **verificación/depuración** por sobre la de **diseño**, en especial para la **FPU**
- El **mayor reto** fue el desarrollo de la **FPU** (donde la unidad aritmética no es 100% satisfactoria) y a su vez la mayor limitante del diseño

★ Conclusiones del trabajo

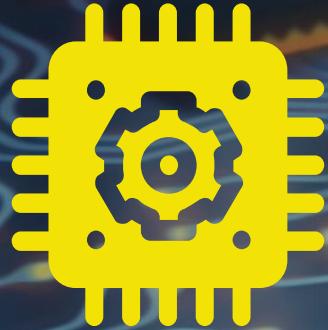
Mejoras propuestas

- Optimizar la **FPU** (modelo ***pipelined***) e intensificar su etapa de verificación
- Complejizar el entorno de ejecución e implementar mayores funciones al ***SoC***
 - considerando los planteamientos iniciales y los bloques desechados del segundo planteamiento, como los bloques “***program & data memory***” y “***BOOT-LOADER***”
- Con lo anterior, aprovechar las memorias externas de la ***Nexys 4***
- Integrar los módulos ***CSR*** y ***A*** para permitir diseños con múltiples núcleos
- Complejizar los ***control hazzards*** y añadir ***multiple issue***
- Mejoras en la memoria ***cache*** (esquema asociativo)
- Añadir una arquitectura con privilegios
- En este tipo de diseños siempre hay cabida a las mejoras



fcfm

Ingeniería Eléctrica
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE



Diseño e Implementación de un SoC basado en *RISC-V* en un *FPGA*

Gianluca Vincenzo
D'Agostino Matute

¡Muchas gracias
por su atención!