Implementación del procesador Cortex™-M0 DesignStart en una FPGA de rango bajo

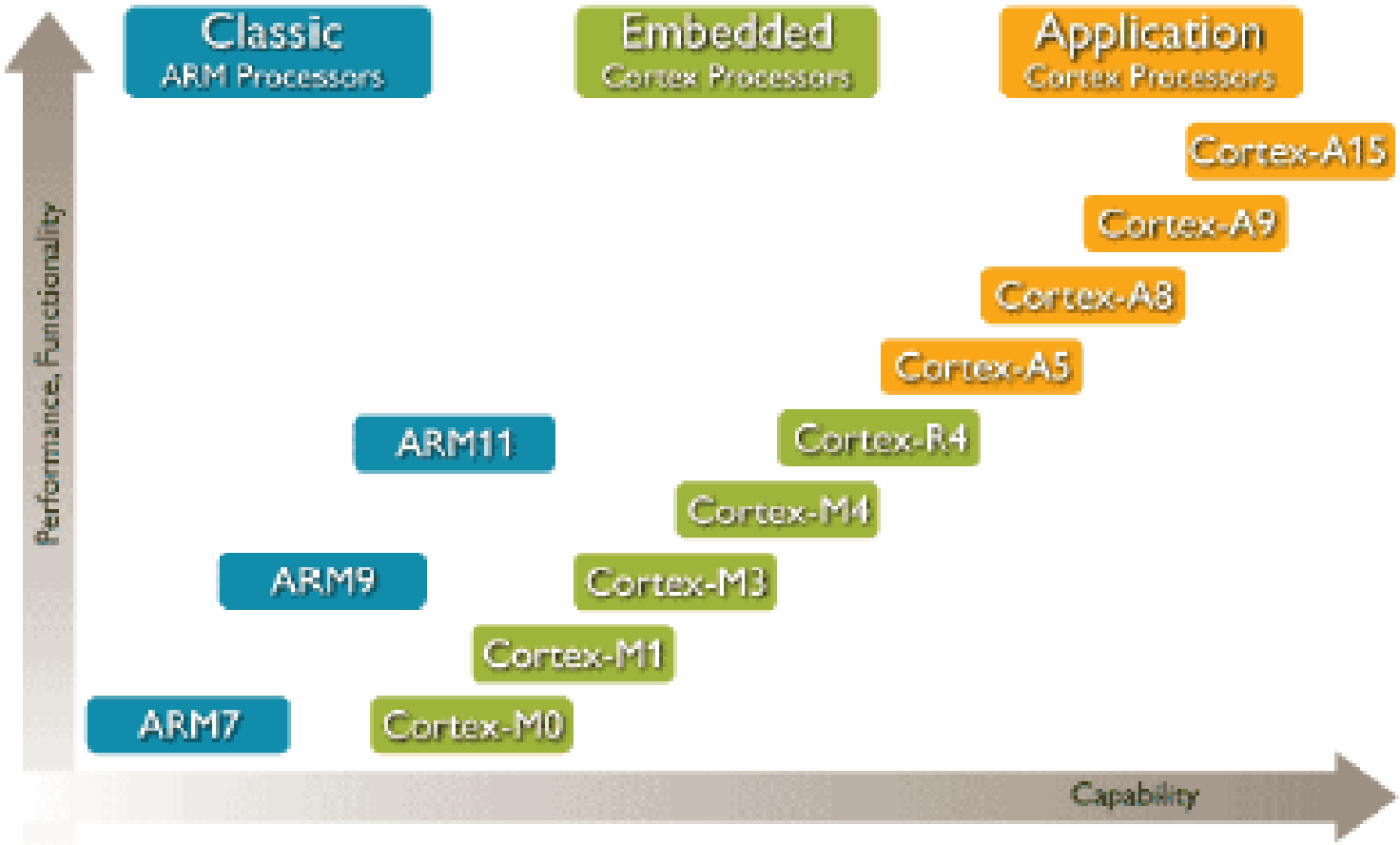
Pedro Ignacio Martos y Fabricio Baglivo

Laboratorio de Sistemas Embebidos, Facultad de Ingeniería – UBA, Buenos Aires, Argentina.

ARM Cortex™-M

- Procesadores de 32 bits
- Bajo costo (~ U$S3 LPC1100)
- Bajo consumo
- Set de instrucciones de alta densidad
- Fácil de usar
- Alta performance

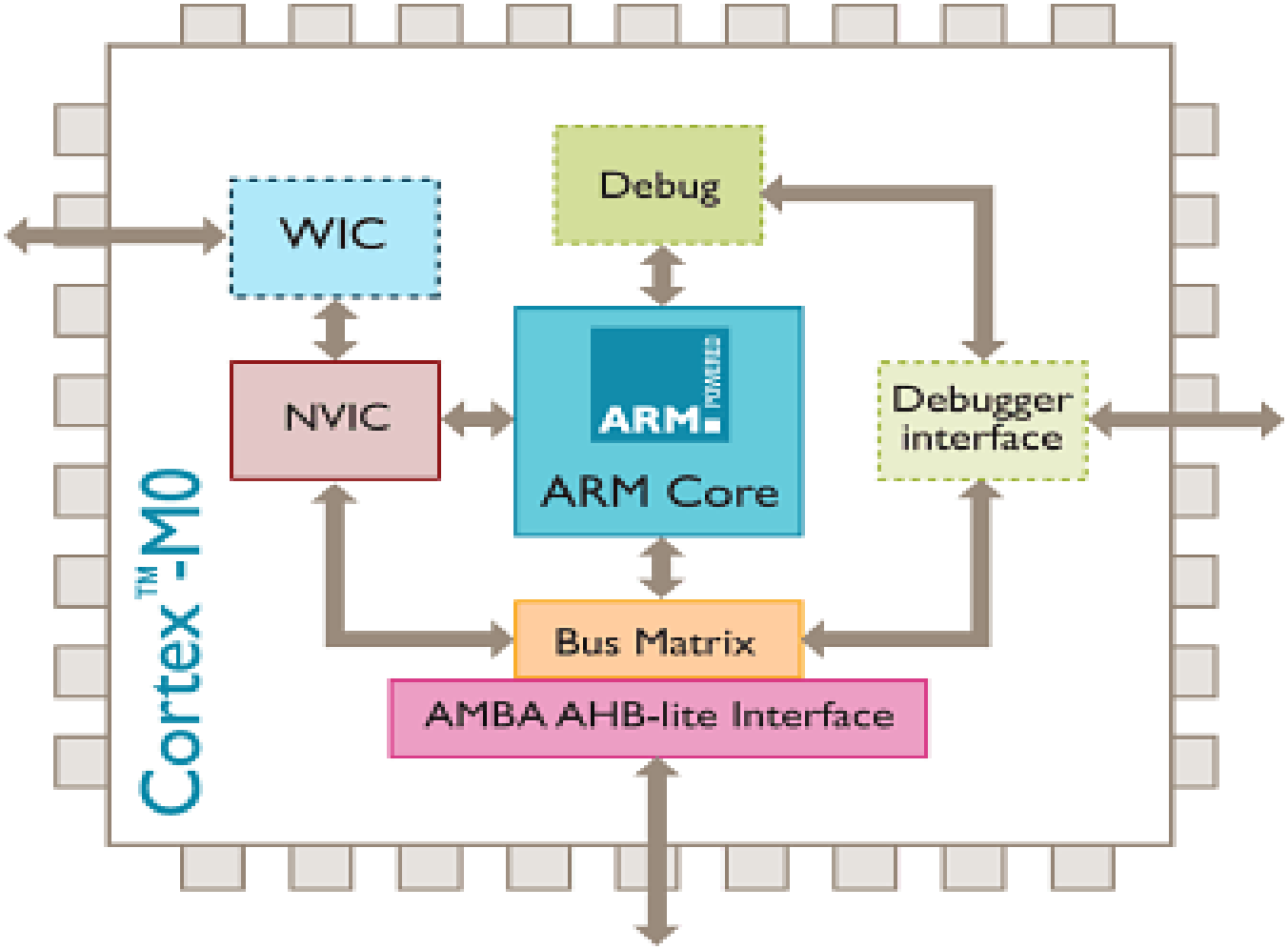
Cortex™ M vs otros Cortex



Cortex™-M0

- Es el más pequeño de la familia Cortex™-M
- consume 85 microwatts/MHz
- Frec. = 40 – 50 MHz

Cortex-M0


Debug
WIC
NVIC
ARM
ARM Core
Debugger interface
Cortex™-M0
Bus Matrix
AMBA AHB-lite Interface

Protocolo AMBA

- AMBA → estandar de interconexión de bloques funcionales
- Características:
  - Flexible
  - Multi capas -> permite trabajar con varios Maestros y Esclavos
  - Compatibilidad
  - Soporte de 3ros

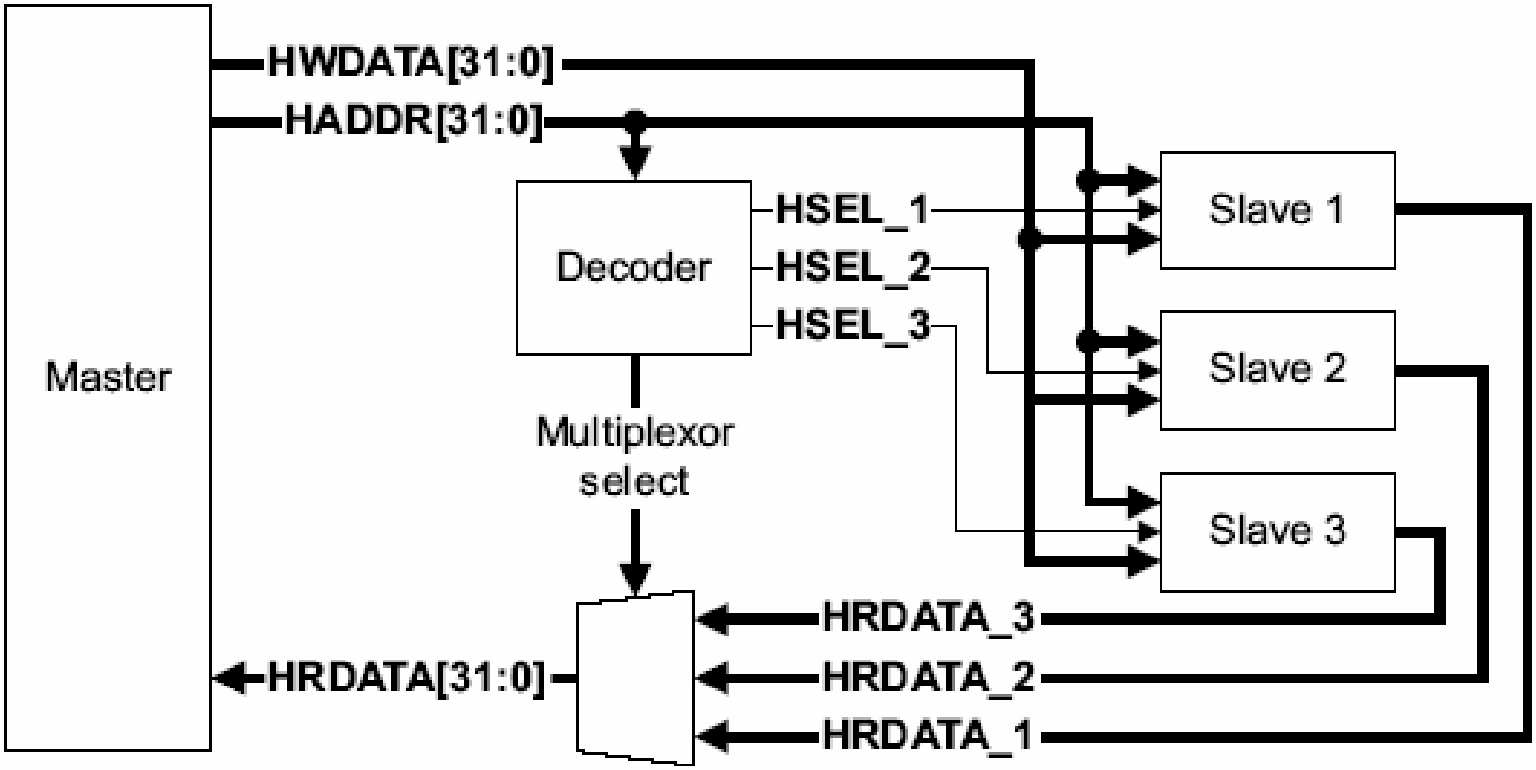
AMBA versión 3

- AXI → interfaz de comunicación con 5 canales de alta velocidad.
- AHB → interfaz de comunicación de 1 solo canal para conexión con periféricos simples.
- APB → interfaz que soporta conexiones con periféricos de bajo ancho de banda, utilizado para configurar registros e intercambio de datos. Es una inerfaz de bajo consumo.
- ATB → interfaz utilizada para debugging

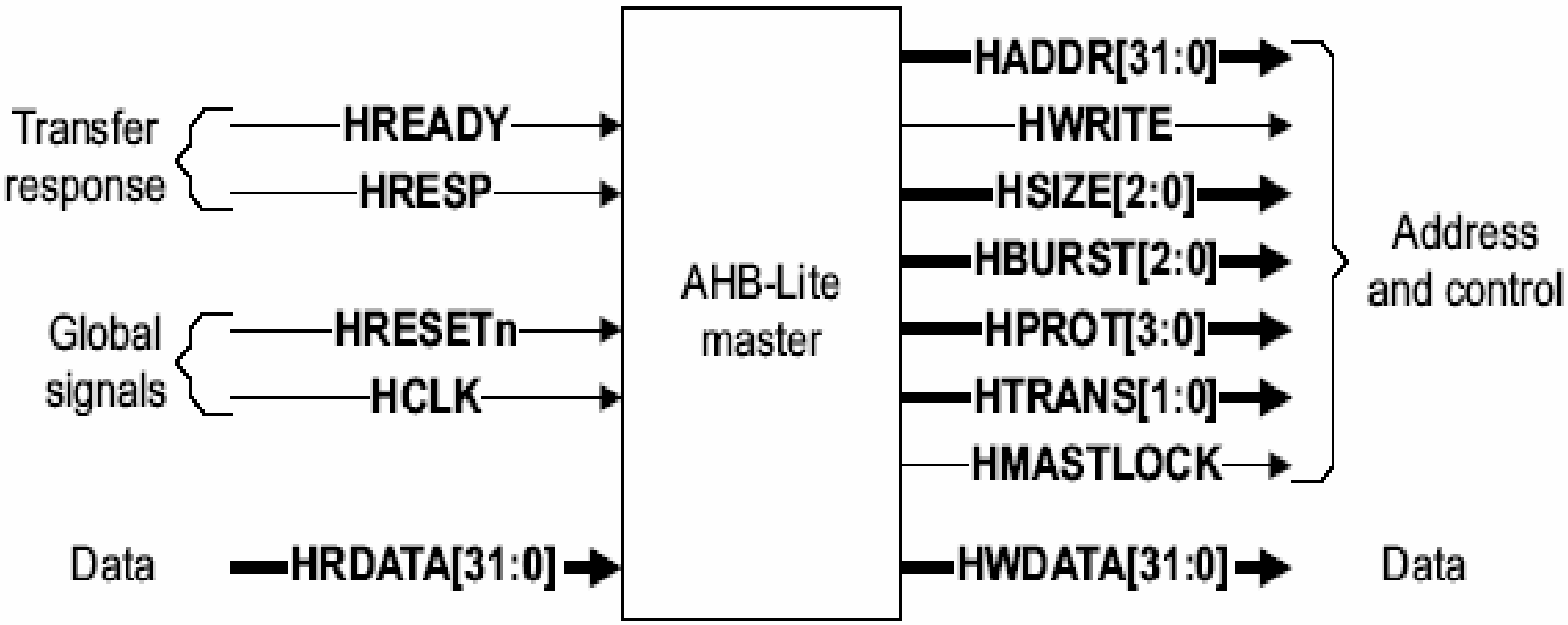
AMBA 3 ABH-Lite

- Es una simplificación de ABH
- Implementa:
  - Comunicaciones de tipo “rafaga”
  - Operaciones por flanco de clock
  - sin tri-state
  - Configuración de ancho de bus en: 64, 128, 512 y 1024 bits

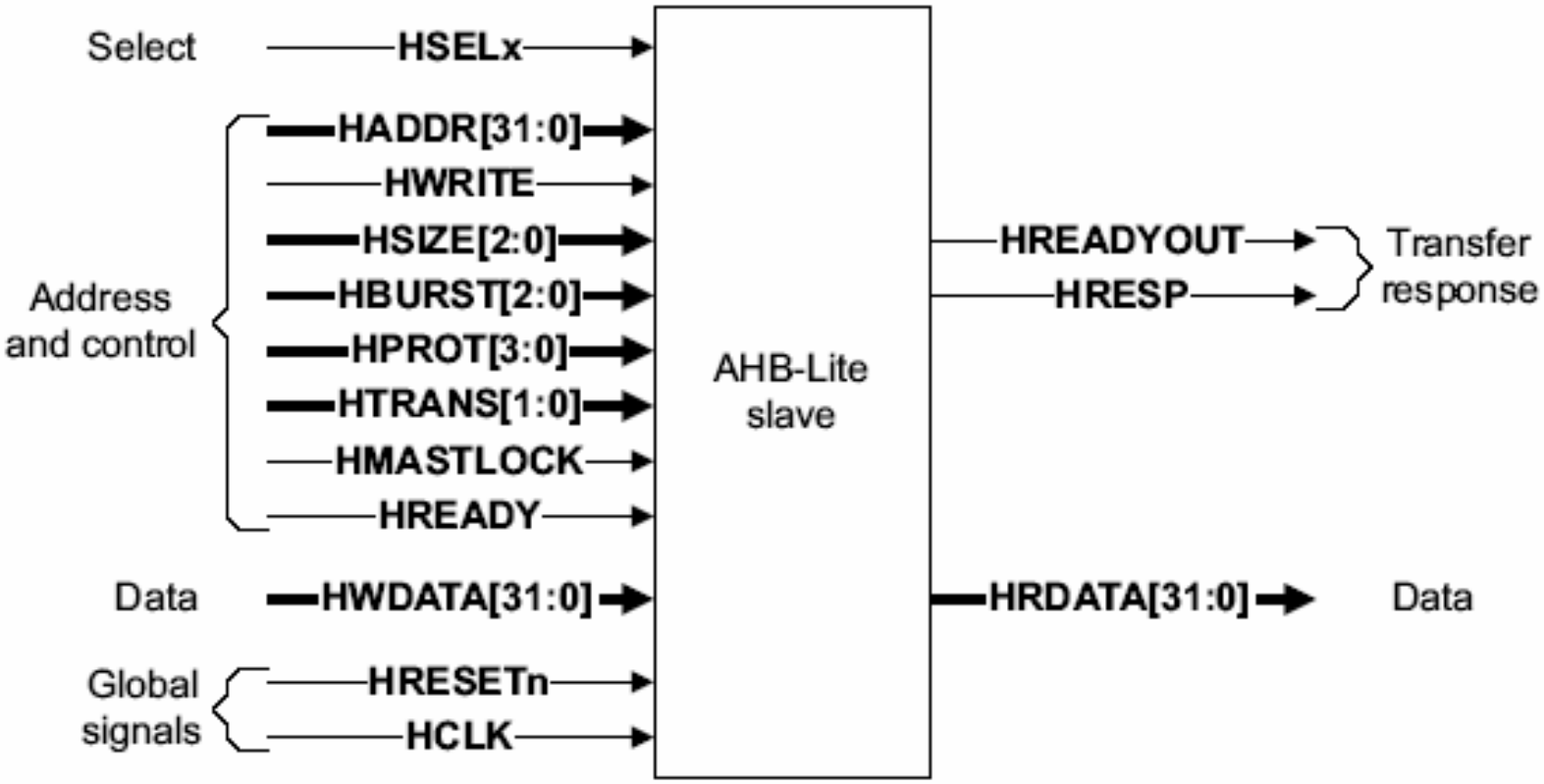
Esquema de conexión: AHB-Lite



Maestro en AHB-Lite



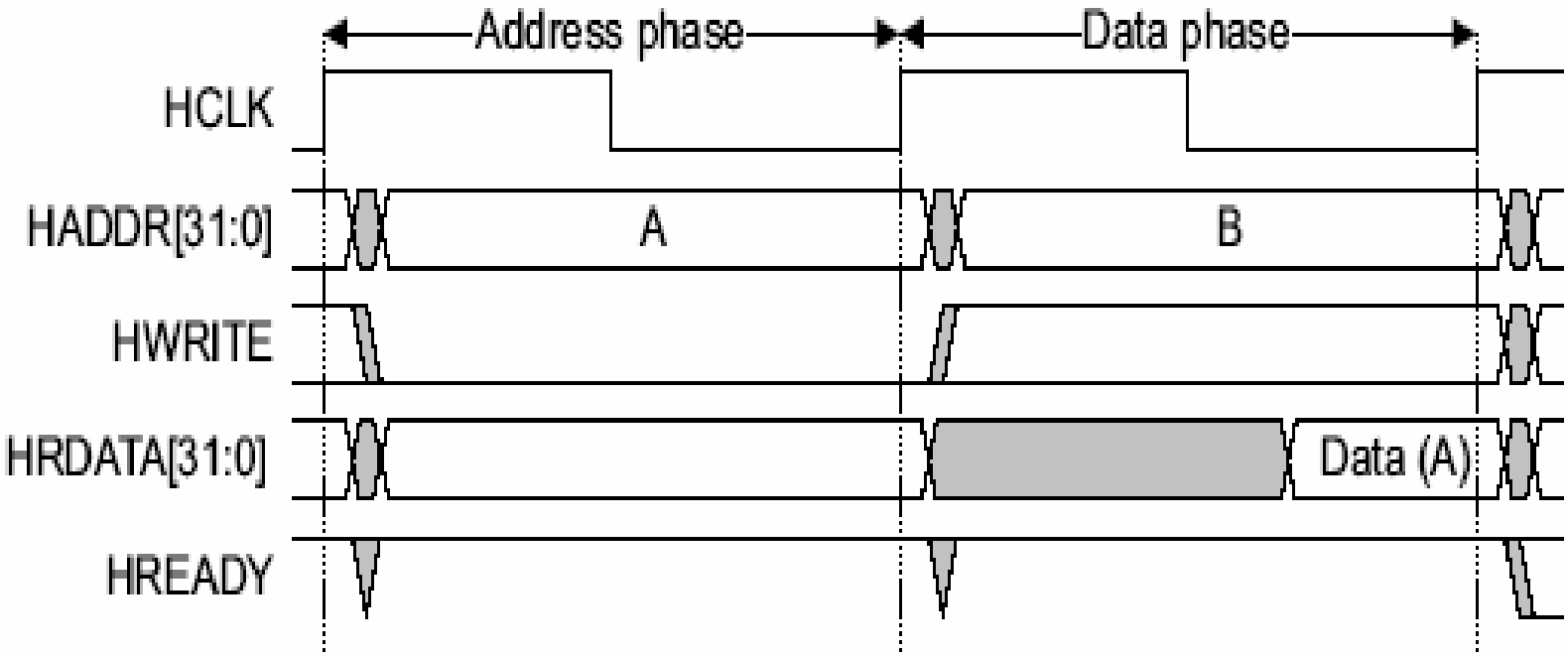
Esclavo en AHB-Lite



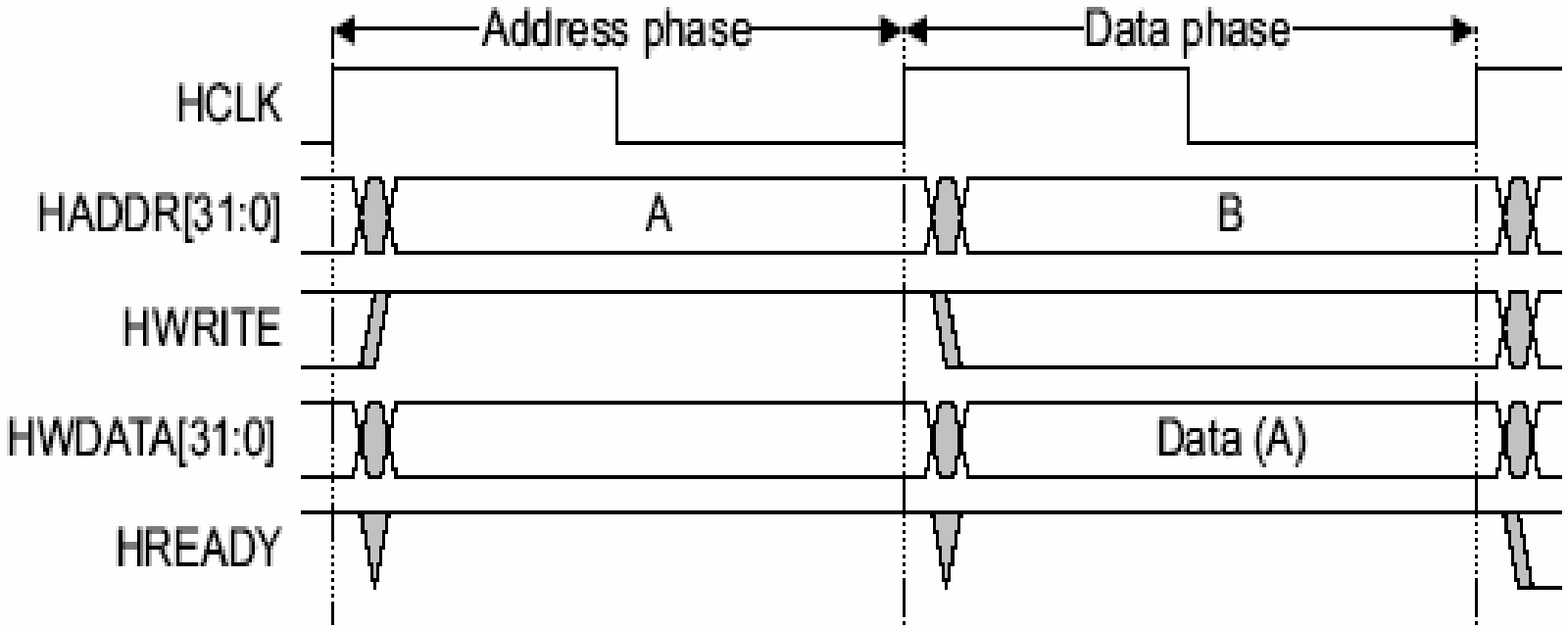
Señales importantes en AHB-Lite

Globales	Maestro a Esclavo	Esclavo a Maestro
HCLK	HADDR[31:0]	HRDATA[31:0]
HReset	HBURST[2:0]	HREADY
	HMASTLOCK	HRESP
	HPROT[3:0]	
	HSIZE[2:0]	
	HTRANS[1:0]	
	HWDATA[31:0]	
	HWRITE	

Ciclo básico de Lectura



Ciclo básico de Escritura



Cortex M0 DesignStart

- Es una configuración fija del Cortex-M0:
  - Implementable tanto en software como en hardware
  - Se distribuye en lenguaje Verilog totalmente sintetizable

Cortex M0 DesignStart

```
AT510-BU-98000-r0p0-00rel0/
  ├── AT510-BU-98000-r0p0-00rel0.lst
  ├── ARM_Cortex-M0_DesignStart_ReleaseNote.pdf
  └── logical/
        └── cortexm0ds/
              ├── verilog/
              │     ├── CORTEXM0DS.v
              │     └── cortexm0ds_logic.v
              └── tbench/
                    ├── cortexm0ds_tb.v
                    ├── helloworld.c
                    ├── Makefile
                    └── ram.bin
```

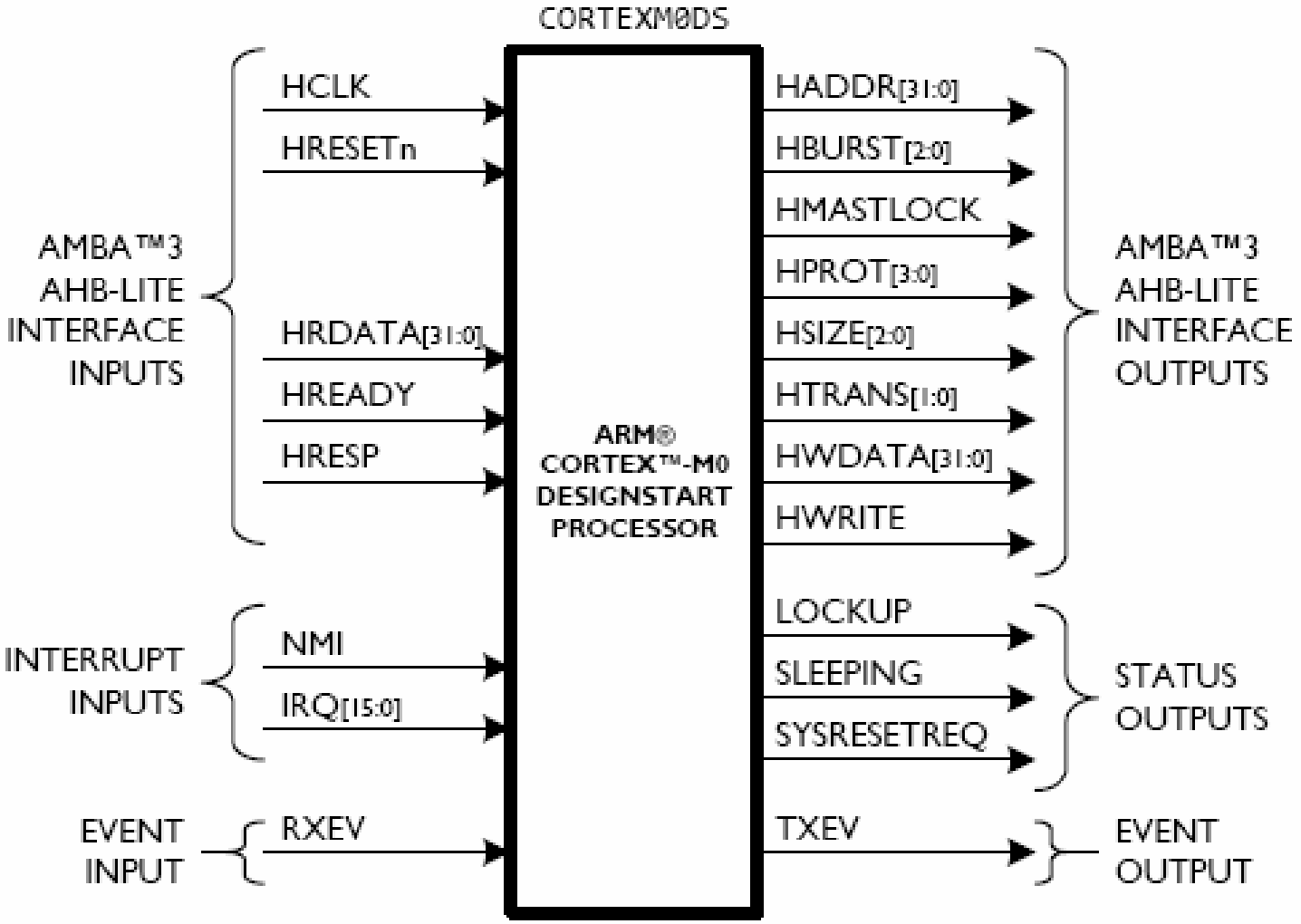
TAR MANIFEST

RELEASE-NOTE (THIS DOCUMENT)

SYNTHESIZABLE VERILOG

TEST-BENCH ENVIRONMENT

Cortex M0 DesignStart

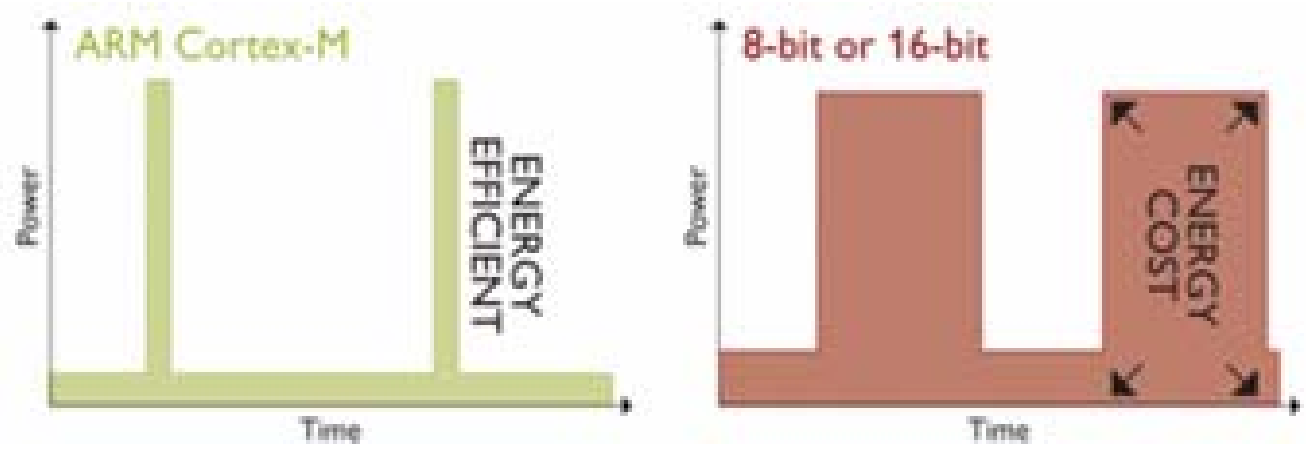


Cortex-M – Características

RISC processor core	Thumb-2® technology
• High performance 32-bit CPU • Deterministic operation • Low latency 3-stage pipeline	• Optimal blend of 16/32-bit instructions • 3x smaller code size than 8-bit devices • No compromise on performance
Low power modes	**Nested Vectored Interrupt Controller (NVIC)**
• Integrated sleep state support • Multiple power domains • Architected software control	• Low latency, low jitter interrupt response • No need for assembly programming • Interrupt service routines in pure C
Tools and RTOS support	CoreSight debug and trace
• Broad 3rd party tools support • Cortex Microcontroller Software Interface Standard (CMSIS) • Maximizes software effort reuse	• JTAG or 2-pin Serial Wire Debug (SWD) connection • Support for multiple processors • Support for real-time trace

Cortex-M – Características

- Tienen mejor manejo de los modos de bajo consumo.
- Trabajan a baja frecuencia y con ciclos de actividad más cortos

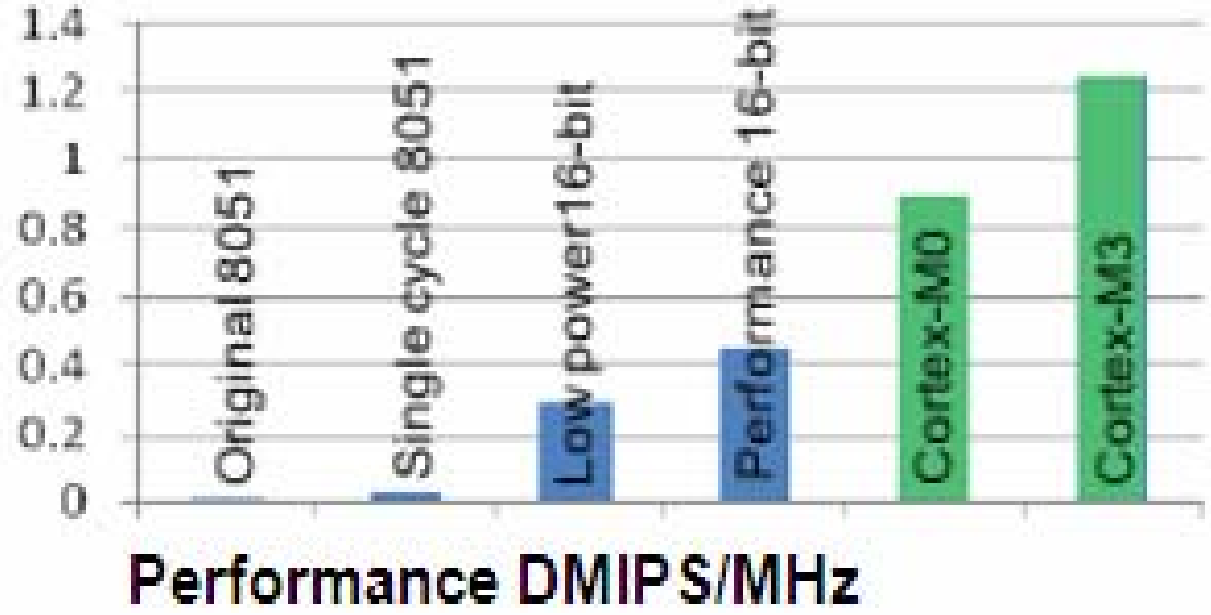


Cortex-M – Características

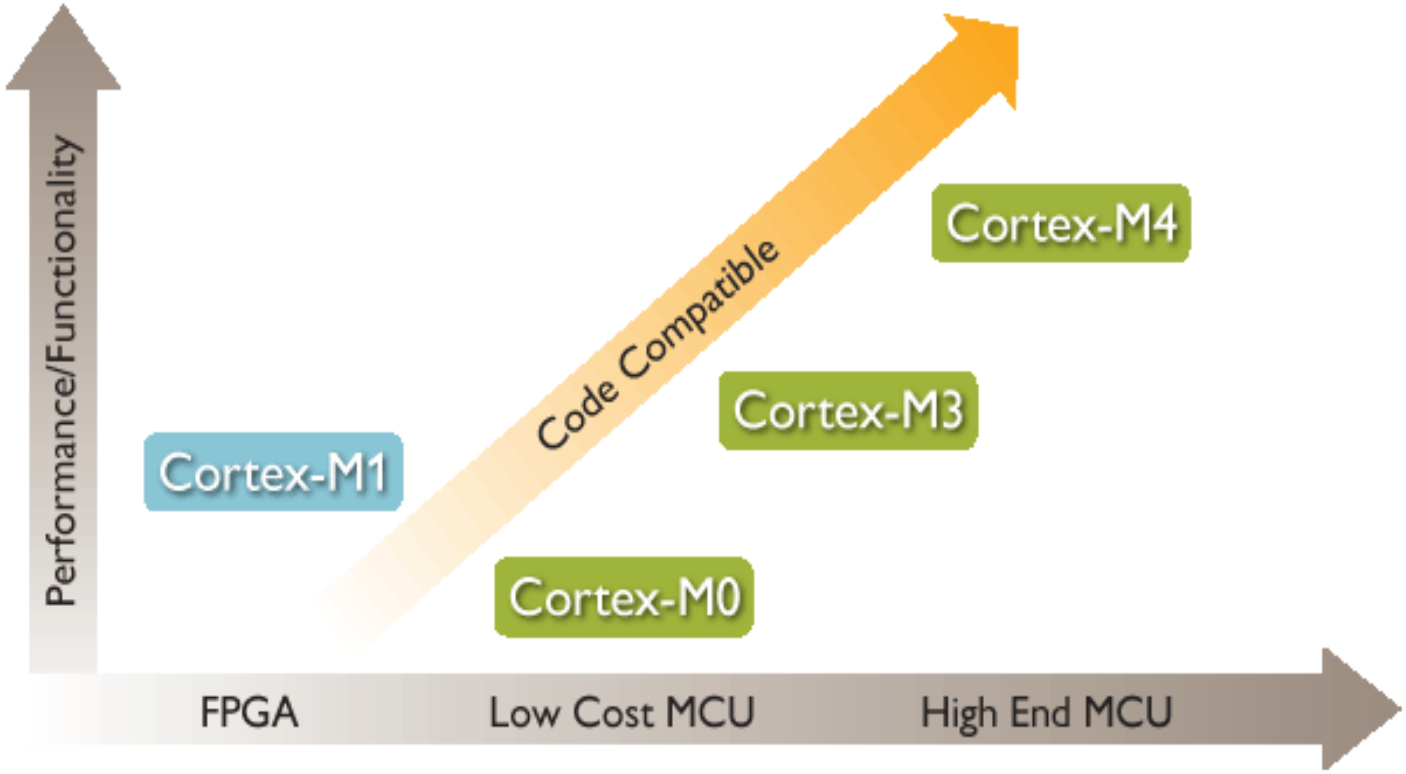
- Fabricado por diferentes empresas
  - Atmel, Actel, NXP, Cypress, Samsung, Texas Instruments,ST, etc
- Compatibilidad de código entre ellos
- Herramientas unificadas y para diversos OS

- volver

Cortex-M – Comparacion de performance



Cortex-M – capacidades relativas de la familia

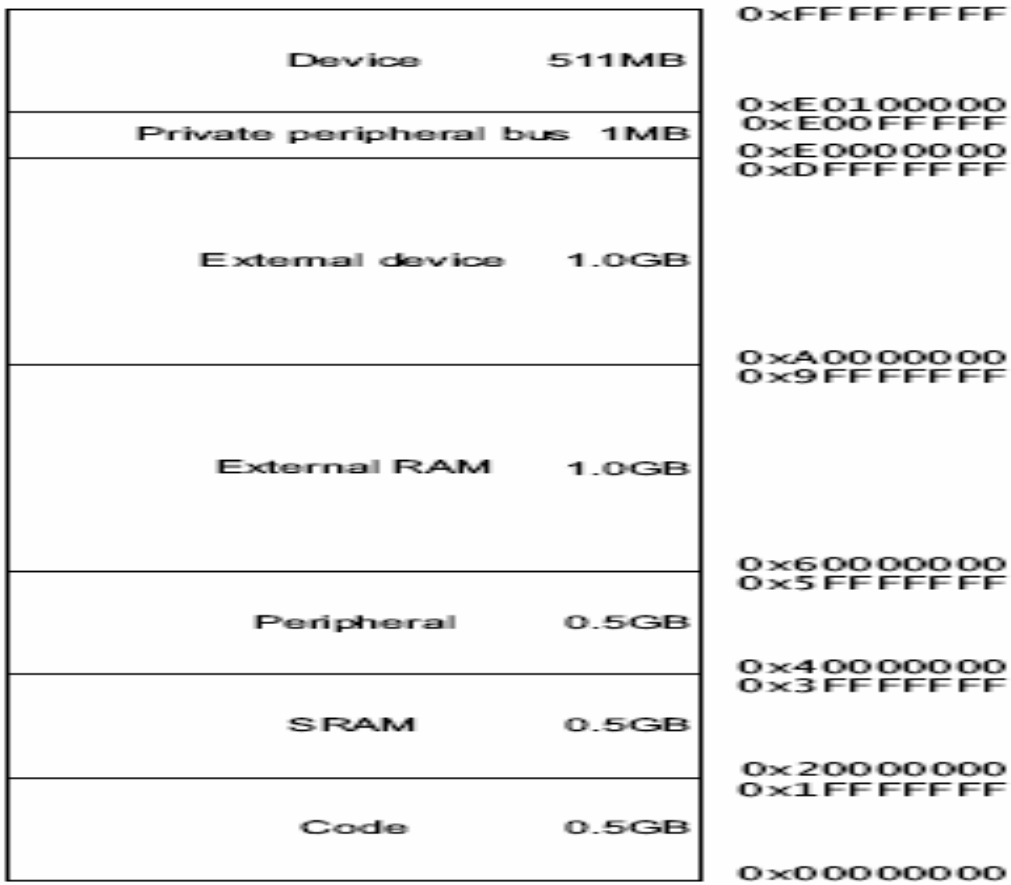


Cortex-M0 – Características

» **ARM Cortex-M0 Features**

Cortex-M0	
Architecture	ARMv6-M (Von Neumann)
ISA support	Thumb®/ Thumb-2 technology*
Pipeline	3-stage
Dhrystone	0.9 DMIPS/MHz
Interrupts	NMI + 1 to 32 physical interrupts
Interrupt latency	16 cycles
Sleep modes	Integrated WFI and WFE instructions Sleep & Deep Sleep Signals Optional Retention Mode with Power Management Kit
Enhanced Instructions	Single-cycle (32x32) multiply
Debug	JTAG or Serial-Wire Debug ports

Cortex-M0 – Mapa de memoria



Implementación en FPGA

Etapas

- Desarrollo del software
- Implementación del sistema
- Simulación Funcional
- Verificación en Hardware

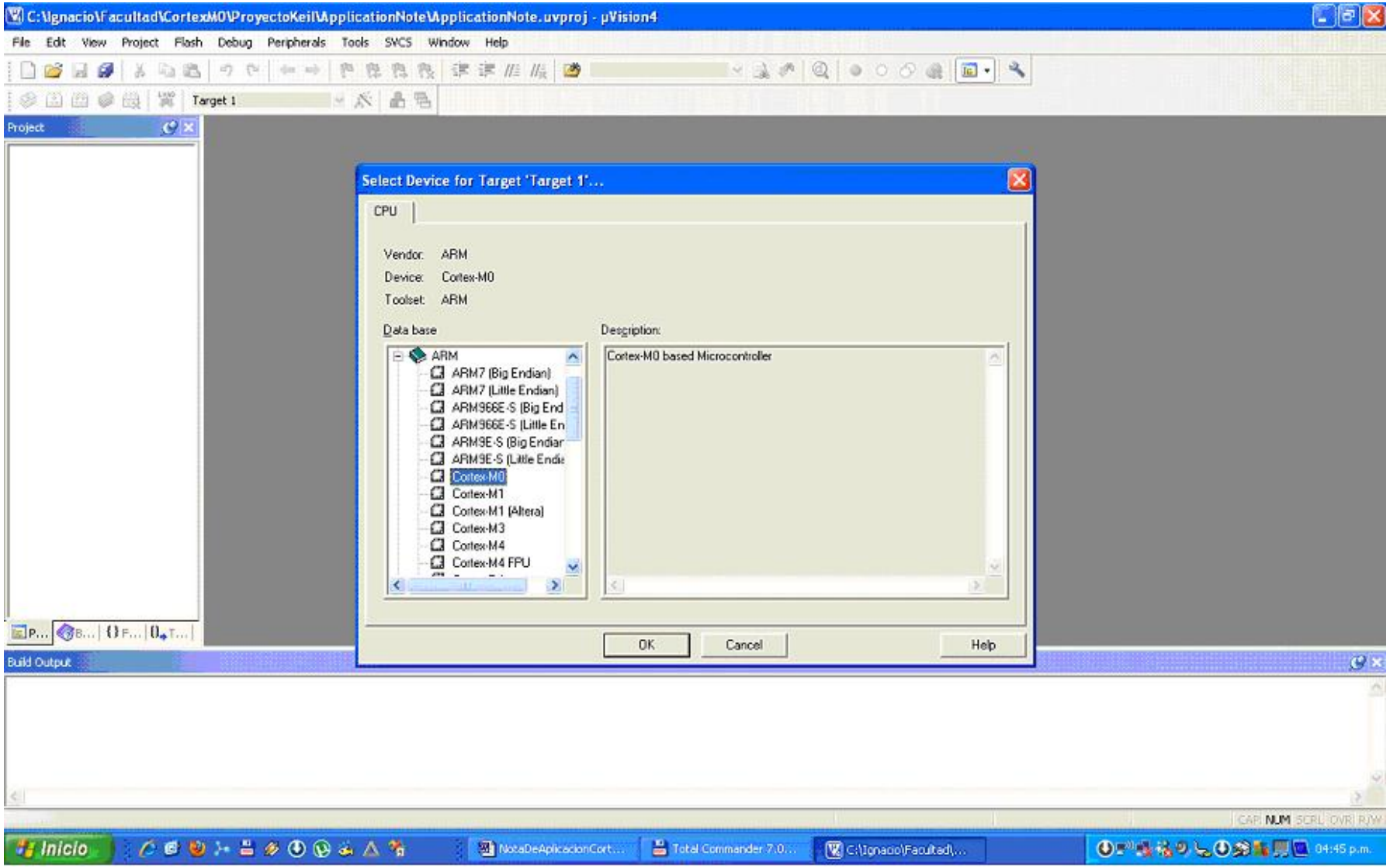
Desarrollo del software

- Entorno de desarrollo: Keil ARM MDK
- Proyecto básico: se accede a constantes predefinidas a intervalos regulares.

Desarrollo de software

- Se crea un proyecto nuevo seleccionando el procesador ARM Cortex-M0

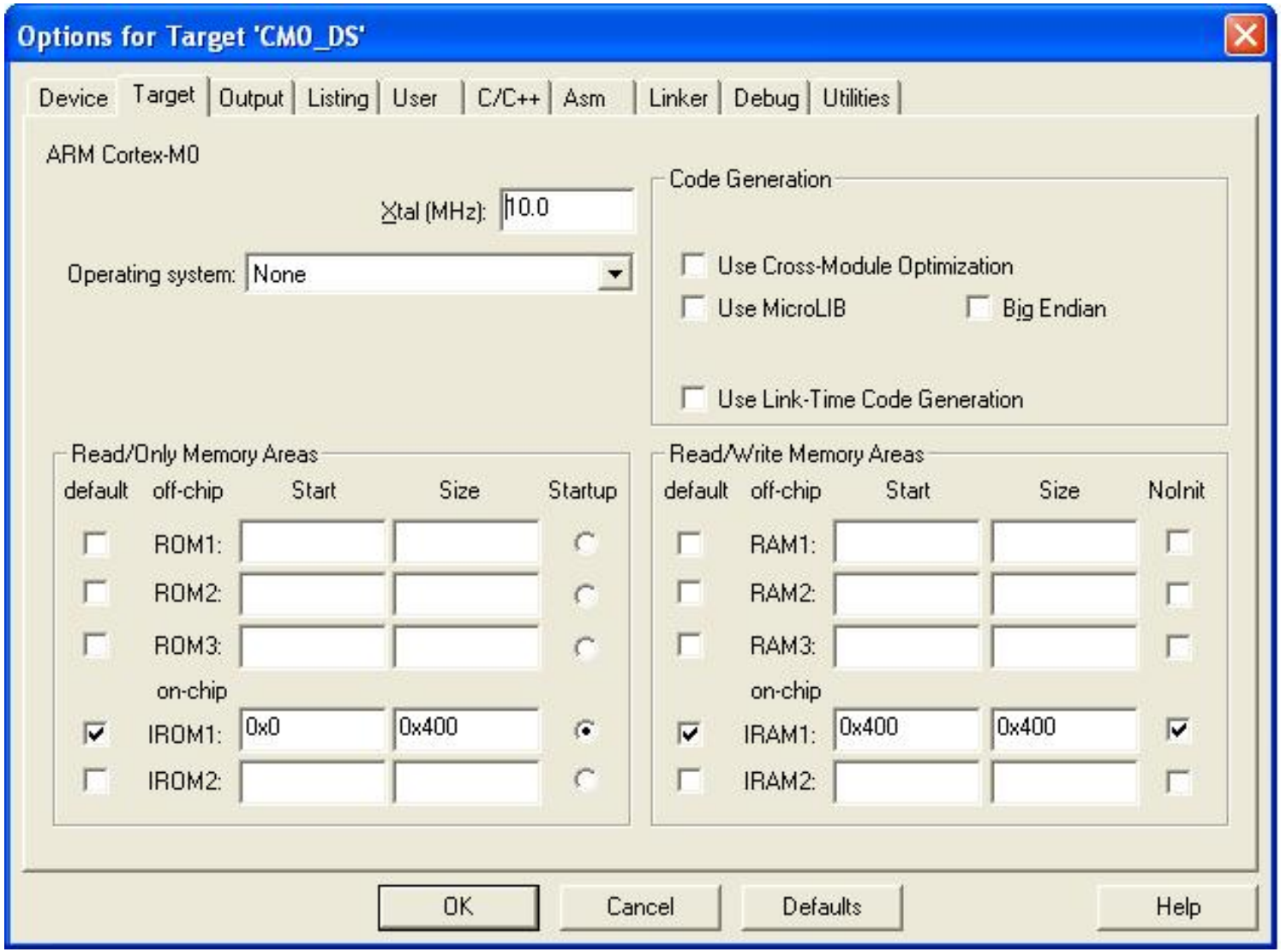
Desarrollo de software



Desarrollo de software

- Se deben configurar las áreas de memoria ROM y RAM en forma contigua a fin de que la implementación se realice con una memoria RAM predefinida

Desarrollo de software



Desarrollo de software

- Hay dos archivos de código fuente:
- Main.c con el programa
- Vectors.c con los vectores de interrupción

Desarrollo de software

```
#define LedOn          0xaaaa5555
#define LedOff         0xf0f0f0f0
int main(void)
{
    unsigned int counter;          // dummy
    unsigned int ii;               // loop iterator
    unsigned int trap;             // memory access pattern receiver
    unsigned int period;           // time interval for memory access
    //period=20000000;  // roughly 3 seconds for a 10MHz osc in CM0_DS
    period=200;  // period for simulations
    while (1)
    {
        counter=0;
        for (ii=0;ii<period;ii++)
            counter++;
        trap=LedOn;            // memory access pattern (turn on)
        for (ii=0;ii<period;ii++)
            counter++;
        trap=LedOff;   // memory access pattern (turn off)
        trap++;        // dummy
    }
}
```

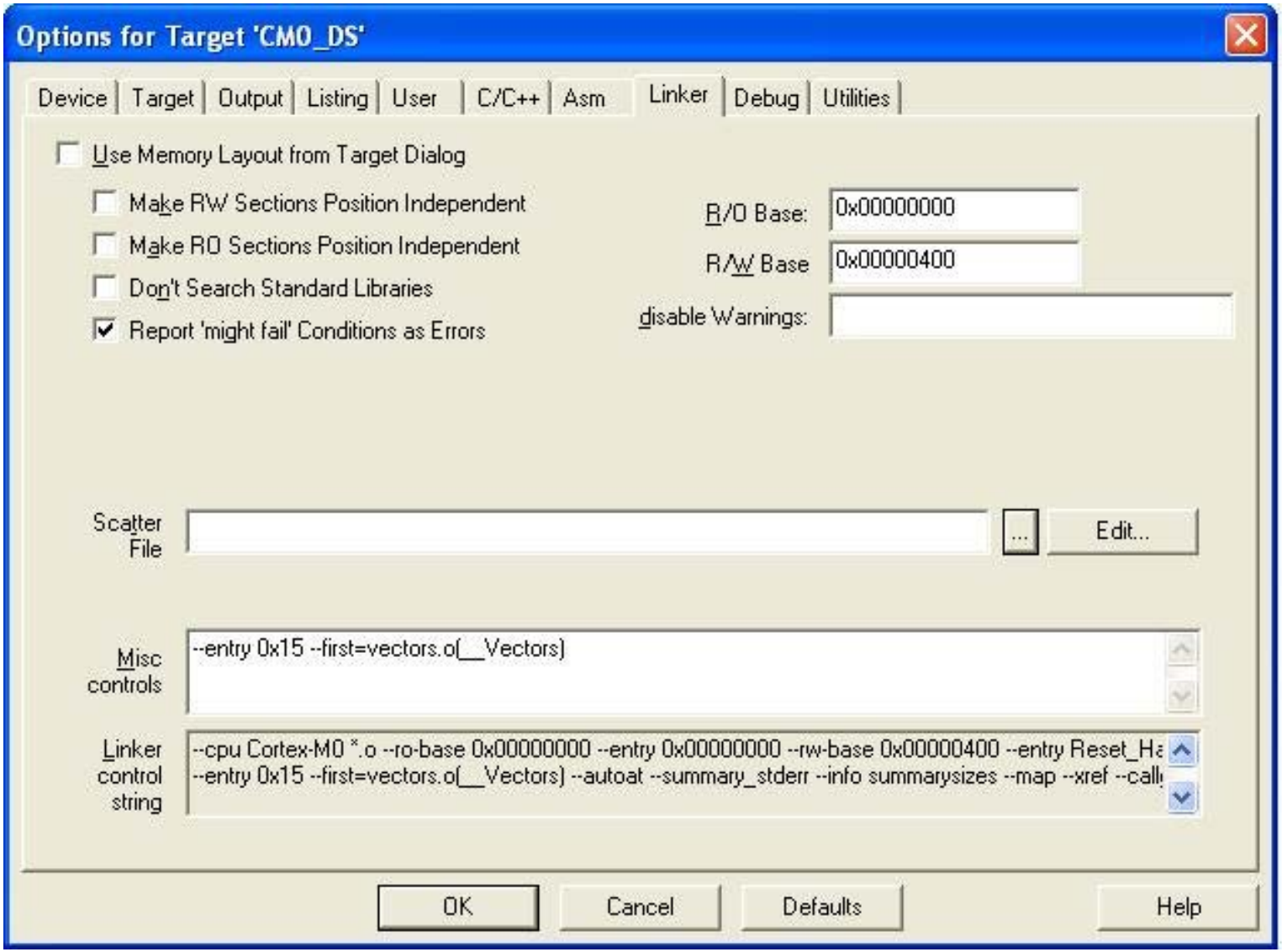
Desarrollo de software

```
// Vectors.c
// Define where the top of memory is.
#define TOP_OF_RAM 0x400U

extern int main(void);     // Use C-library initialization function.

__attribute__ ((section("__Vectors")))
static void (* const vector_table[])(void) =
{
  (void (*)(void)) TOP_OF_RAM, // Initial value for stack pointer.
  (void (*)(void)) main,     // Reset handler is C initialization.
  0,                         // No HardFault handler, just cause lockup.
  0,                         // No NMI handler, just cause lockup.
  0//...                     // Additional handlers would be listed here.
};
```

Desarrollo de software

- Para que el linker tome la información del archivo vectors.c es necesario agregar la opción “--entry 0x15 --first=vectors.o(__vectors)”

Desarrollo de software



Desarrollo de software

- Se realiza el compilado y simulación del código fuente, viéndose en el assembler generado el acceso a las constantes predefinidas (0xaaaa5555 y 0xf0f0f0f0)

Desarrollo de software

C:\Ignacio\Facultad\CortexM0\ProyectoKeil\CM0_DSNexys2BlinkingLed\BlinkingLed.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
Core	
R0	0x0...
R1	0x0...
R2	0x0...
R3	0x0...
R4	0x0...
R5	0x0...
R6	0x0...
R7	0x0...
R8	0x0...
R9	0x0...
R10	0x0...
R11	0x0...
R12	0x0...
R13 (SP)	0x0...
R14 (LR)	0xF...
R15 (PC)	0x0...
xPSR	0x8...
Banked	
System	
Internal	
Mode	Thr...
Stack	MSP
States	473...

Project | Registers

Disassembly

```
    22:         period=20000000;  // roughly 3 seconds for a 10MHz osc in CM0_DS
0x000000A0 4A09      LDR      r2,[pc,#36]  ; @0x000000C8
    23:         while (1)
    24:         {
0x000000A2 E00F      B        0x000000C4
    25:            counter=0;
0x000000A4 2100      MOVS     r1,#0x00
    26:            for (ii=0;ii<period;ii++)
    27:            {
0x000000A6 2000      MOVS     r0,#0x00
0x000000A8 E001      B        0x000000AE
    28:              counter++;
    29:            }
0x000000AA 1C49      ADDS     r1,r1,#1
0x000000AC 1C40      ADDS     r0,r0,#1
0x000000AE 4290      CMP      r0,r2
0x000000B0 D3FB      BCC      0x000000AA
    30:            trap=0xaaaa5555;       // memory access pattern (turn on)
0x000000B2 4B06      LDR      r3,[pc,#24]  ; @0x000000CC
    31:            for (ii=0;ii<period;ii++)
    32:            {
0x000000B4 2000      MOVS     r0,#0x00
0x000000B6 E001      B        0x000000BC
    33:              counter++;
    34:            }
0x000000B8 1C49      ADDS     r1,r1,#1
```

Command

```
Load "C:\\Ignacio\\Facultad\\CortexM0\\ProyectoKeil\\CM0_DSNexys2BlinkingLed\

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 380 Bytes (1%)

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Locals

Name	Value
counter	0x00784F06
ii	0x00784F06
trap	0x00000000
period	0x01312D00

Call Stack | Locals | Memory 1

Simulation | t1: 4.73073140 sec | CAP NUM SCRL OVR R/W

Logros de esta etapa

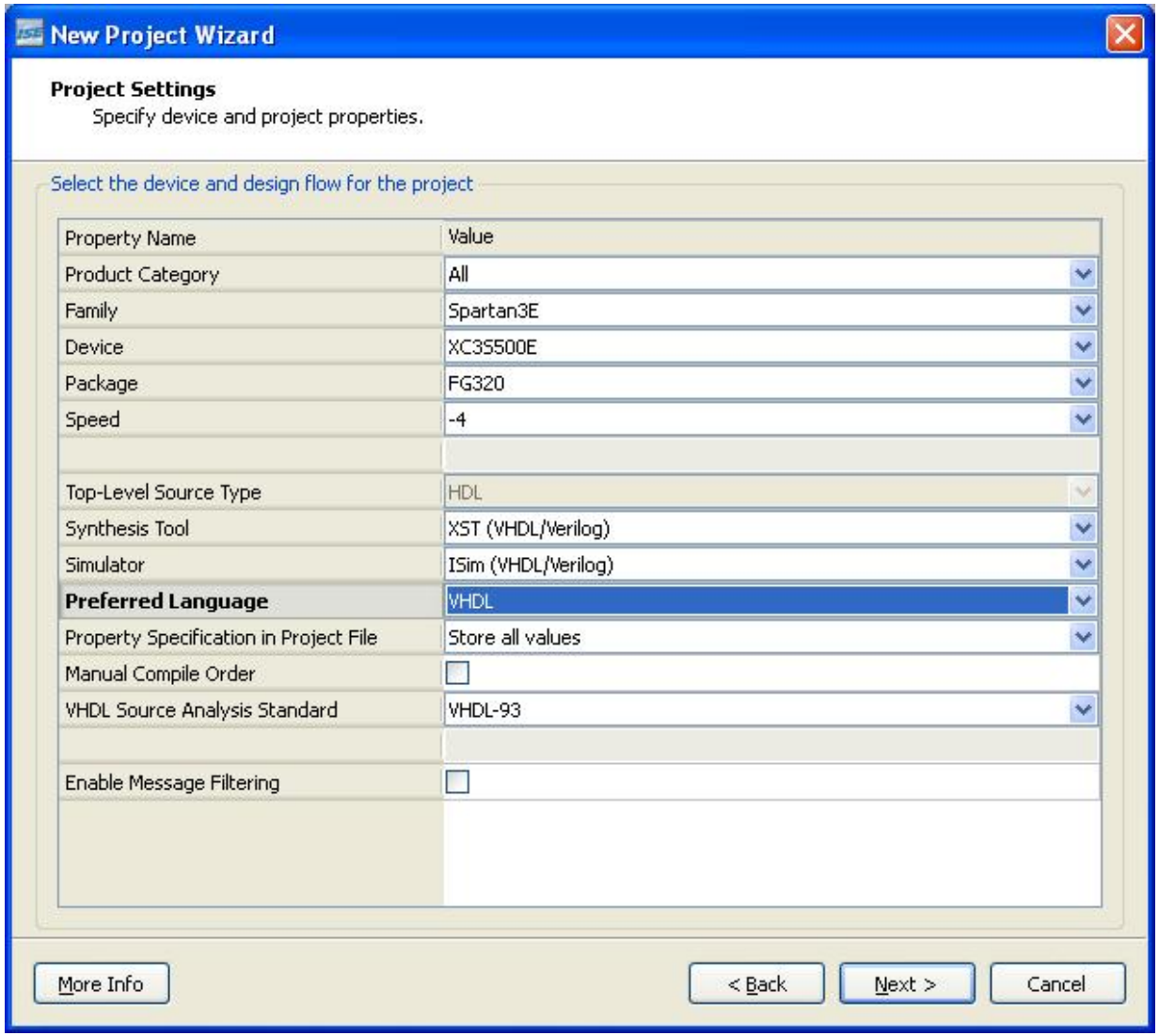
- Un proyecto de software en el entorno de desarrollo ARM MDK correctamente configurado para el procesador Cortex-M0 DesignStart.
- Un código “main.c” y un código “vectors.c” preparados para ejecutar un programa y con los vectores de interrupción y stack configurados.
- Una simulación exitosa del código fuente.

Implementación del sistema

Características:

- Implementación de un sistema mínimo en el entorno Xilinx ISE
- Descripción de hardware mixta (VHDL y Verilog)
- Un detector de patrones en el bus de datos comanda un led.
- Utilización de una FPGA Spartan3E-500 en una placa de desarrollo de bajo costo (Nexys2)

Implementación del sistema

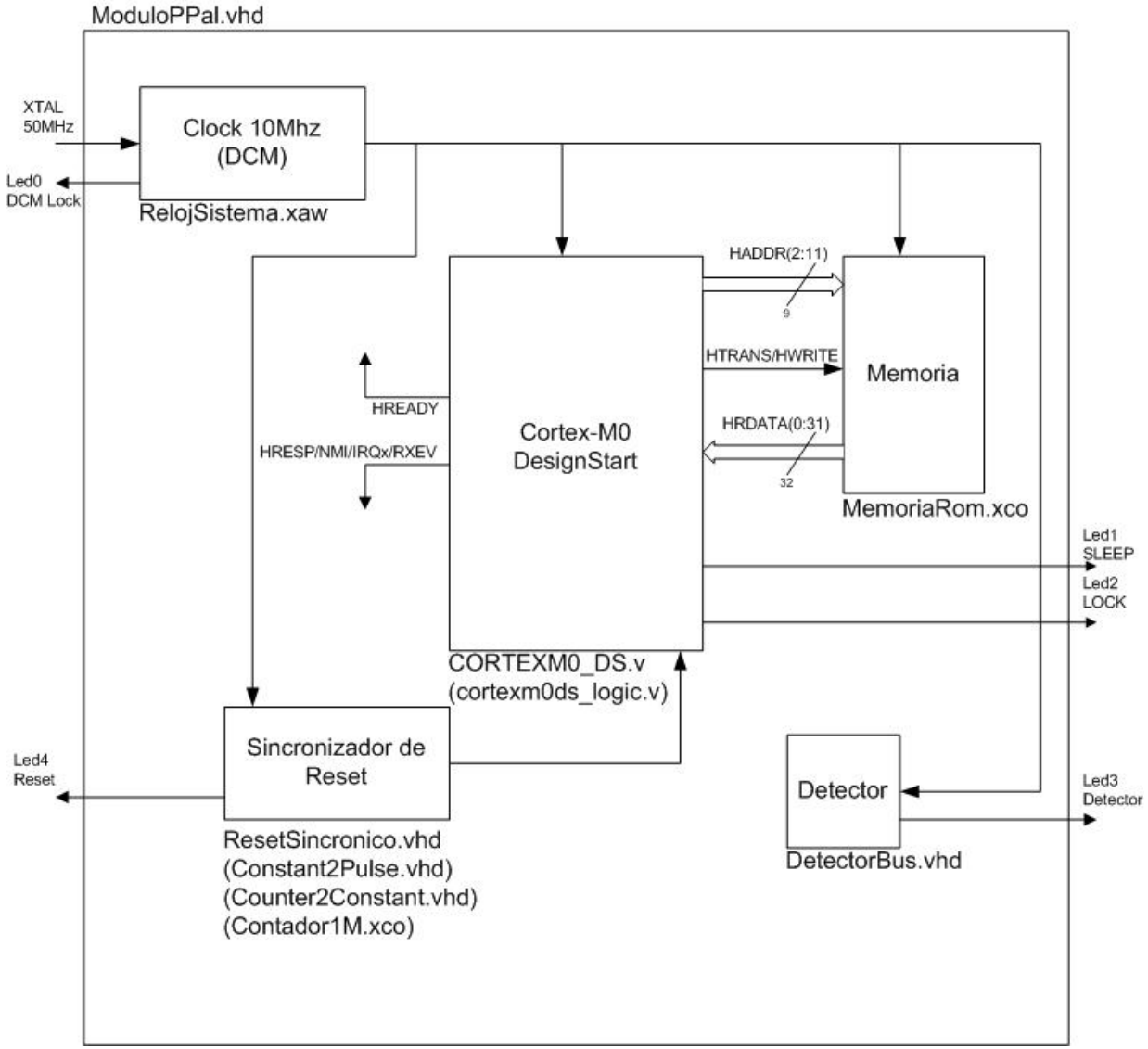


Implementación del sistema

Bloques del sistema:

- Procesador
- Memoria
- Reloj
- Reset Sincrónico
- Detector

Implementación del sistema



Implementación del sistema

Procesador

- Se utilizan las descripciones en verilog provistas en los deliverables:
  - “cortexm0ds_logic.v” (implementación)
  - “CORTEXM0DS.v” (interfase y buses)

Implementación del sistema

Reloj

- Se utiliza un DCM (primitiva de Xilinx) cuya entrada es el oscilador de 50 MHz de la placa Nexys2
- Se lo configura para 10 MHz (se pueden utilizar velocidades de hasta alrededor de 40 MHz)

Implementación del sistema

Implementación del sistema

Detector

- Esta conectado al bus HRDATA del procesador (datos de lectura)
- Configurado para detectar patrones:
  - 0xaaaa5555 Enciende el led
  - 0xf0f0f0f0 Apaga el led

Implementación del sistema

Reset sincrónico

- Se debe generar un único pulso de al menos 2 ciclos de reloj sincronizado con el reloj de sistema
- Se utiliza un contador como base, pero tiene el inconveniente de generar pulsos de un ciclo de reloj de duración y periódicos con el overflow del contador (reset periódico)

Implementación del sistema

Reset sincrónico (cont.)

- Se implementan módulos en cascada con la salida periódica del contador
- Counter2Constant: genera una transición de 0 a 1 en el primer rebalsamiento del contador, manteniendo su salida constante aunque en su entrada se reciban los subsiguientes pulsos de rebalsamiento del contador

Implementación del sistema

Reset sincrónico (cont.)

- Constant2Pulse: genera un pulso de un ciclo de reloj de duración sincronizado con la transición de 0 a 1 de la salida del módulo Counter2Constant

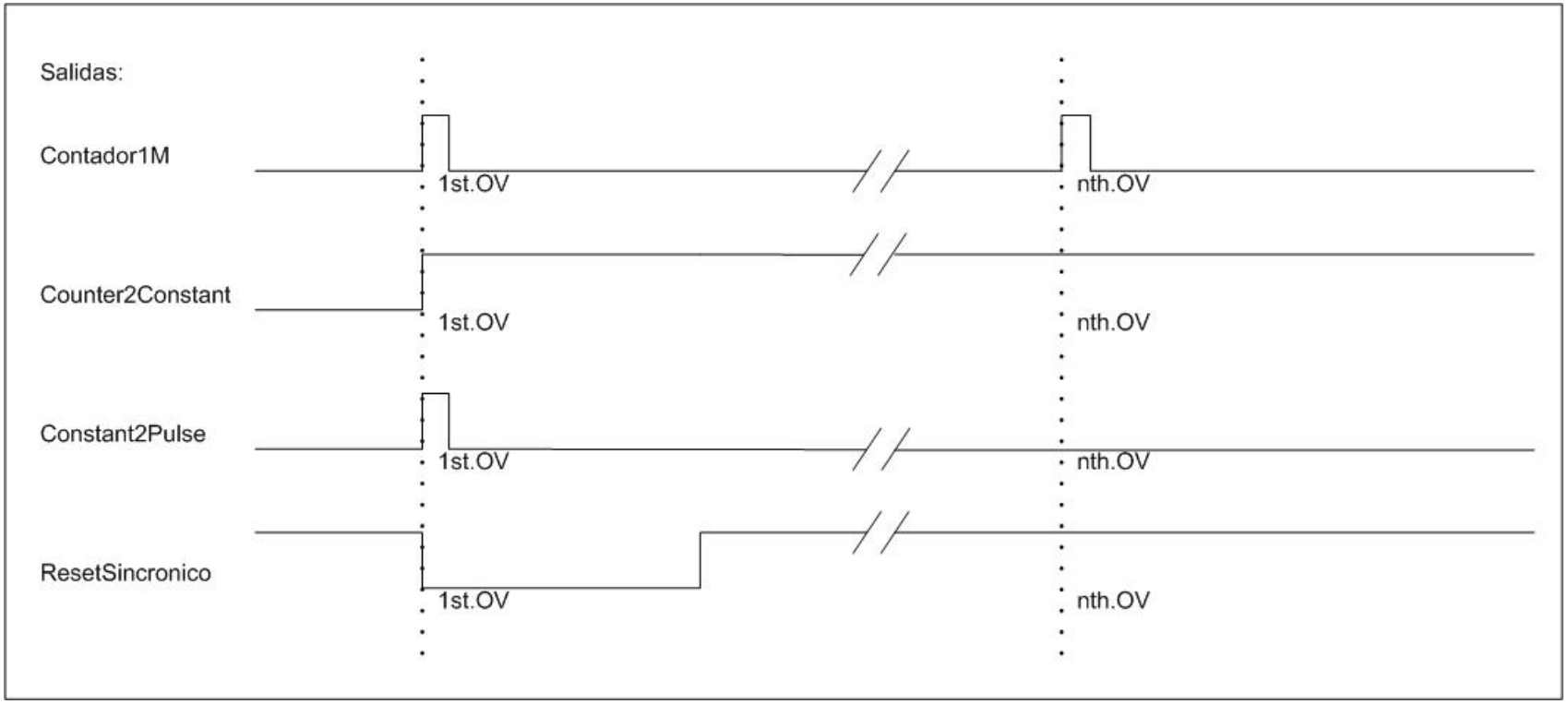
Implementación del sistema

Reset sincrónico (cont.)

- ResetSincronico: recibe como entrada la salida del módulo Constat2Pulse, la invierte, y genera un pulso negativo de 5 ciclos de reloj de duración, que se utiliza como señal de reset del procesador

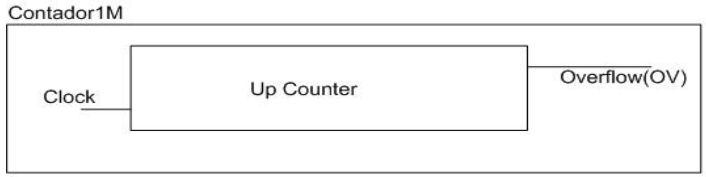
Implementación del sistema

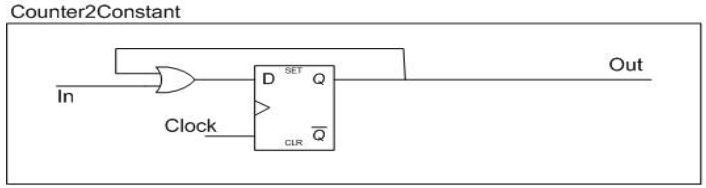
Diagrama de Tiempos

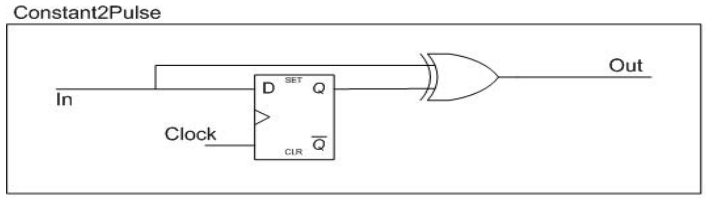


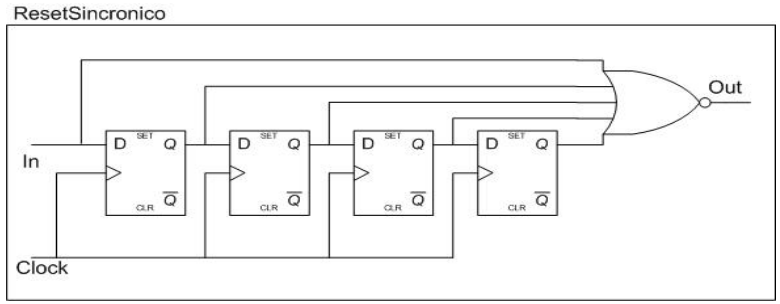
Implementación del sistema

Diagramas circuitales



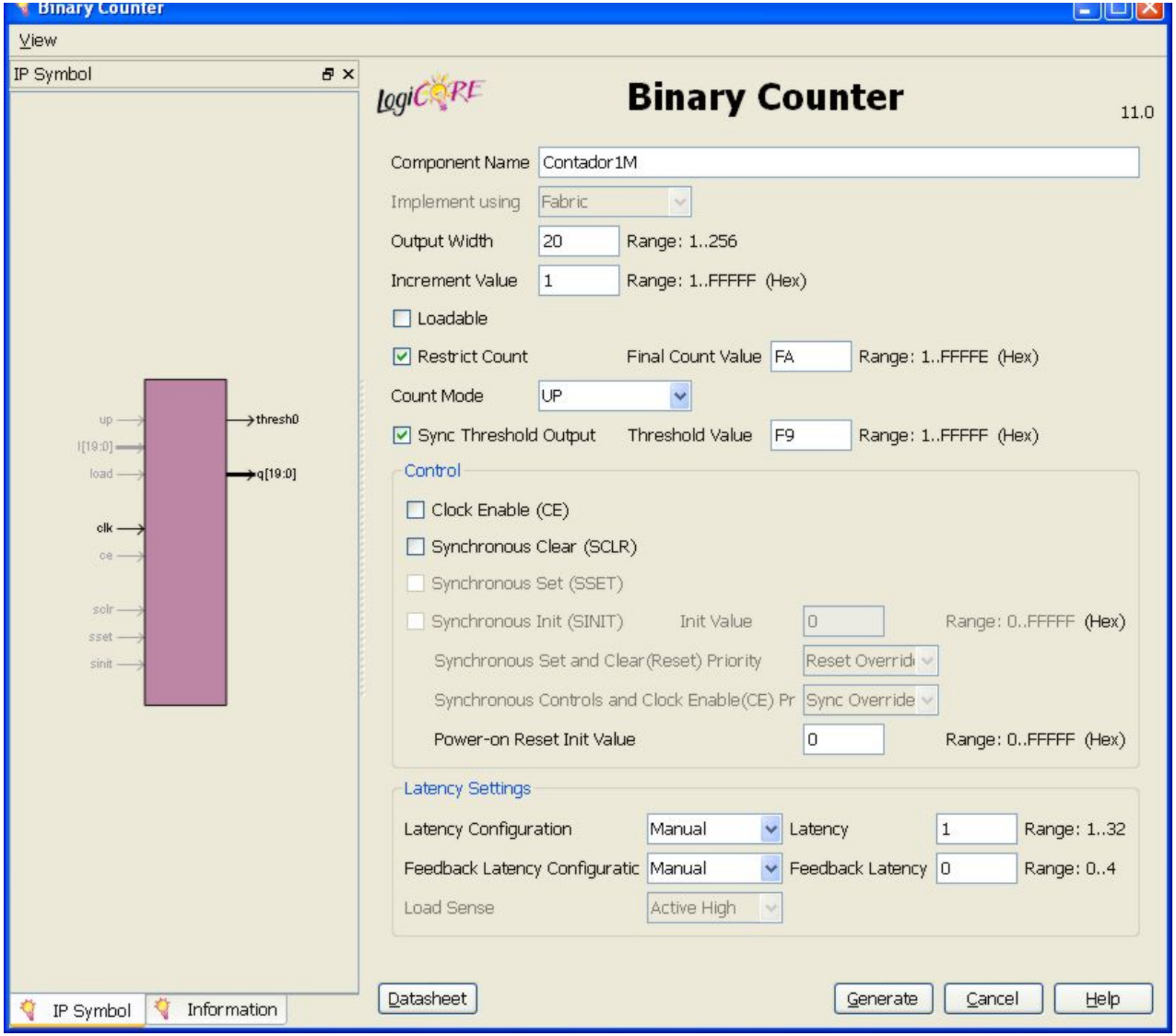






Implementación del sistema

Configuración del contador



Implementación del sistema

Memoria

- Capacidad 512 palabras de 32 bits (2048 bytes)
  - 0-255 para la ROM
  - 256-511 para la RAM
- Realizada a partir de BlockRAM (primitiva de Xilinx)
- Preinicializada con el programa generado en el entorno ARM MDK

Implementación del sistema

Preinicialización:

- Se debe pasar de formato .ELF a una imagen binaria
- Se utiliza la herramienta de línea de comandos “fromelf” provista en ARM MDK

“**fromelf** --bin –o BlinkingLed.bin BlinkingLed.axf”

Implementación del sistema

Preinicialización:

- Del formato binario se debe pasar al formato de inicialización de Xilinx (.COE)
- Para ello se utiliza la herramienta de línea de comandos “bin2coe”, desarrollada especificamente para esta aplicación

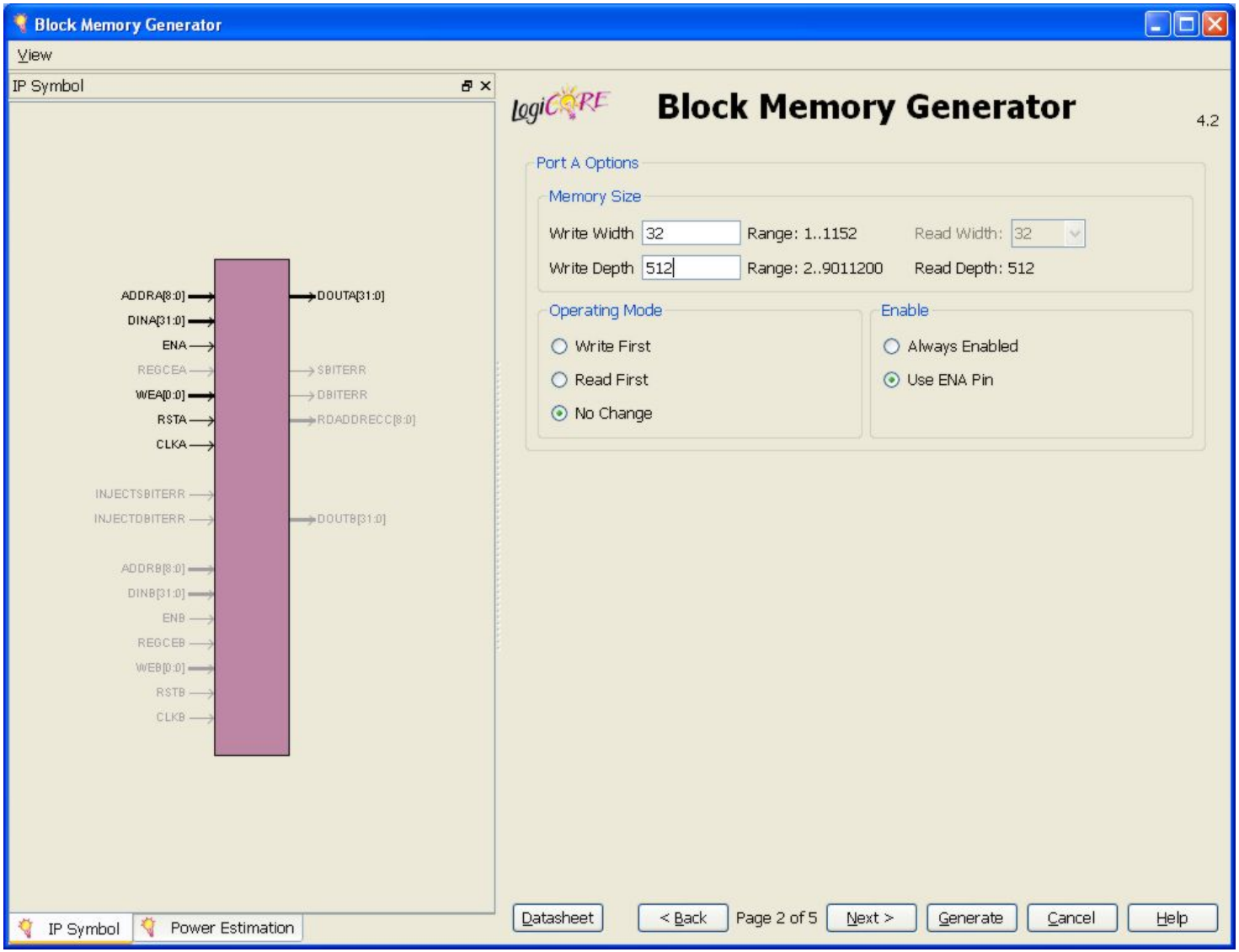
“**bin2coe** BlinkingLed.bin BlinkingLed.coe 512”

Implementación del sistema

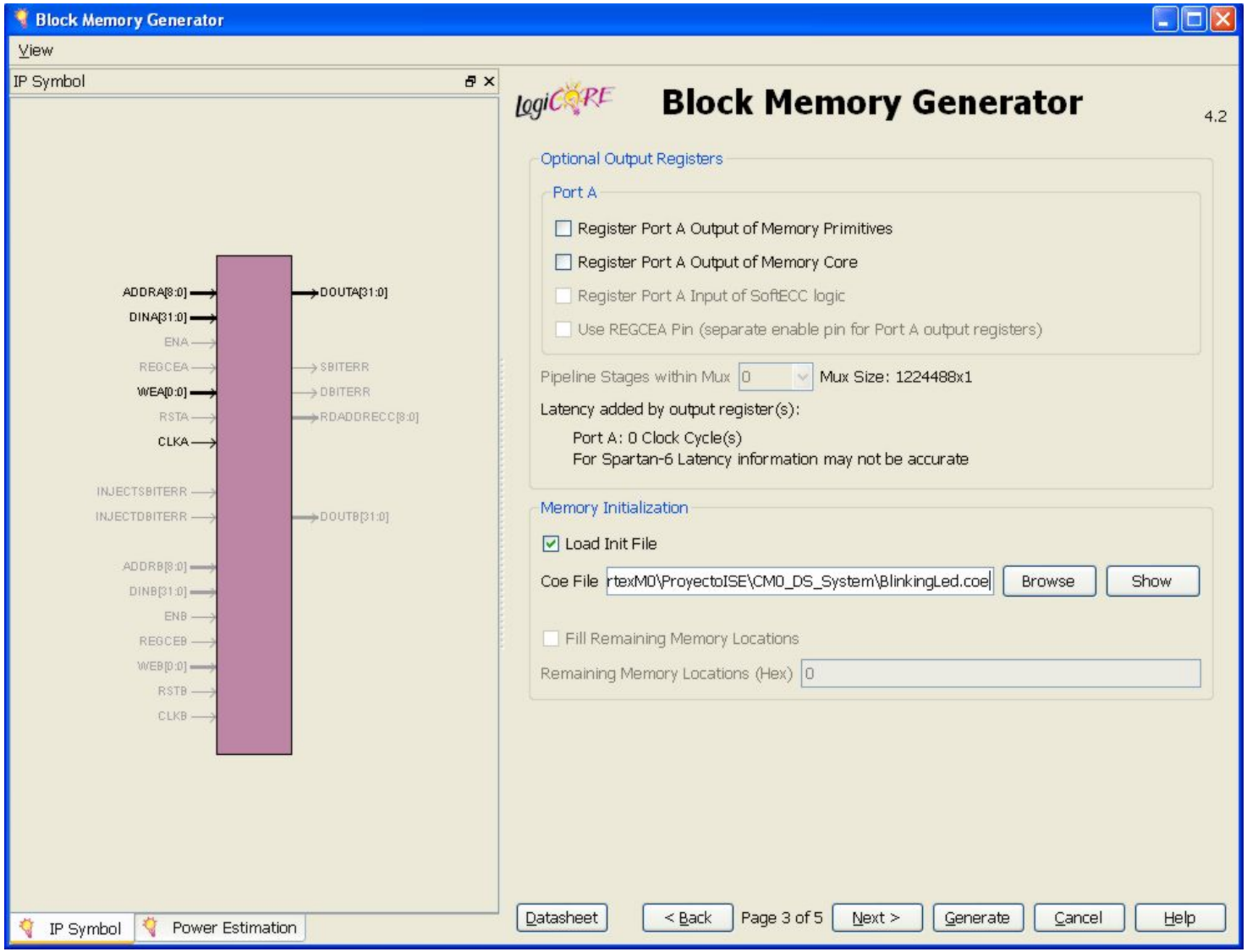
Preinicialización:

- Se utiliza el archivo BlinkingLed.coe generado como archivo de inicialización de contenidos de la memoria

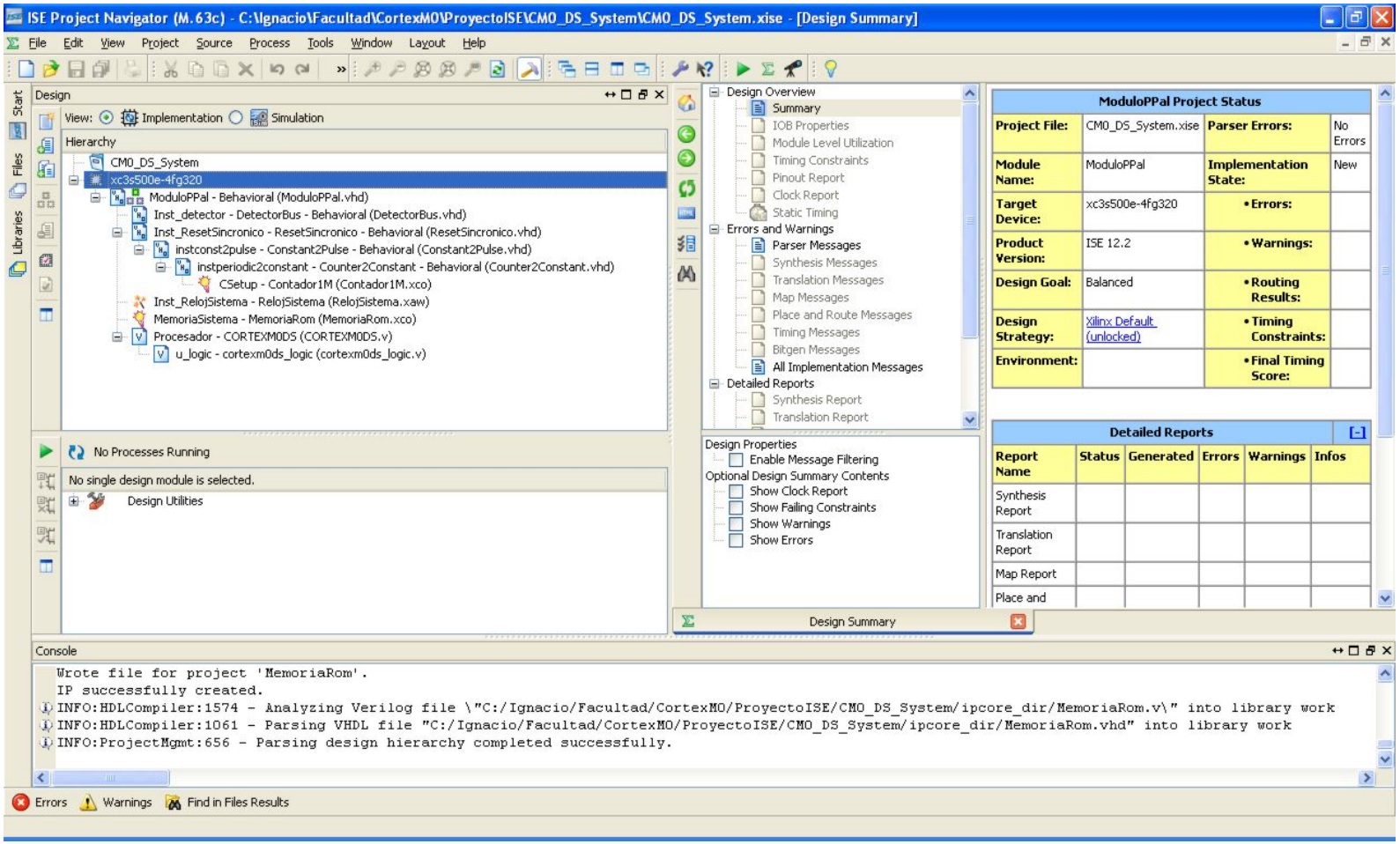
Implementación del sistema



Implementación del sistema



Implementación del sistema



Logros de esta etapa

- Un proyecto en el entorno Xilinx ISE con un sistema basado en el procesador Cortex-M0
- Sistema configurado y con los bloques mínimos con capacidad para ejecutar un programa prediseñado.

Simulación funcional

- Se utiliza la herramienta ISIM, simulador integrado del entorno Xilinx ISE.
- Se debe pasar de la vista de implementación a la vista de simulación:

  Design→View: Simulation

Simulación funcional

ISE Project Navigator (M.63c) - C:\Ignacio\Facultad\CortexM0\ProyectoISE\CM0_DS_System\CM0_DS_System.xise - [Design Summary]

File Edit View Project Source Process Tools Window Layout Help

Design

View: Implementation Simulation

Behavioral

Hierarchy

- CM0_DS_System
- xc3s500e-4fg320
  - ModuloPPal - Behavioral (ModuloPPal.vhd)
    - Inst_detector - DetectorBus - Behavioral (DetectorBus.vl
    - Inst_ResetSincronico - ResetSincronico - Behavioral (Res
    - Inst_RelojSistema - RelojSistema (RelojSistema.xaw)
    - MemoriaSistema - MemoriaRom (MemoriaRom.xco)
    - Procesador - CORTEXM0DS (CORTEXM0DS.v)

No Processes Running

Processes: ModuloPPal - Behavioral

- ISim Simulator
  - Behavioral Check Syntax
  - Simulate Behavioral Model

Design Overview

- Summary
- IOB Properties
- Module Level Utilization
- Timing Constraints
- Pinout Report
- Clock Report
- Static Timing

Errors and Warnings

- Parser Messages
- Synthesis Messages
- Translation Messages
- Map Messages
- Place and Route Messages
- Timing Messages
- Bitgen Messages

Design Properties

- Enable Message Filtering

Optional Design Summary Contents

- Show Clock Report
- Show Failing Constraints
- Show Warnings
- Show Errors

Design Summary

ModuloPPal Project Status (02/28/2011 - 21:02:48)			
Project File:	CM0_DS_System.xise	**Parser Errors:**	No Errors
Module Name:	ModuloPPal	**Implementation State:**	Programming File Generated
Target Device:	xc3s500e-4fg320	**• Errors:**	No Errors
Product Version:	ISE 12.2	**• Warnings:**	46 Warnings (45 new)
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	All Constraints Met
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Device Utilization Summary [-]				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	853	9,312	9%	
Number of 4 input LUTs	4,617	9,312	49%	
Number of occupied Slices	2,917	4,656	62%	
Number of Slices containing only related logic	2,917	2,917	100%	
Number of Slices containing unrelated logic	0	2,917	0%	
Total Number of 4 input LUTs	4,737	9,312	50%	

Console

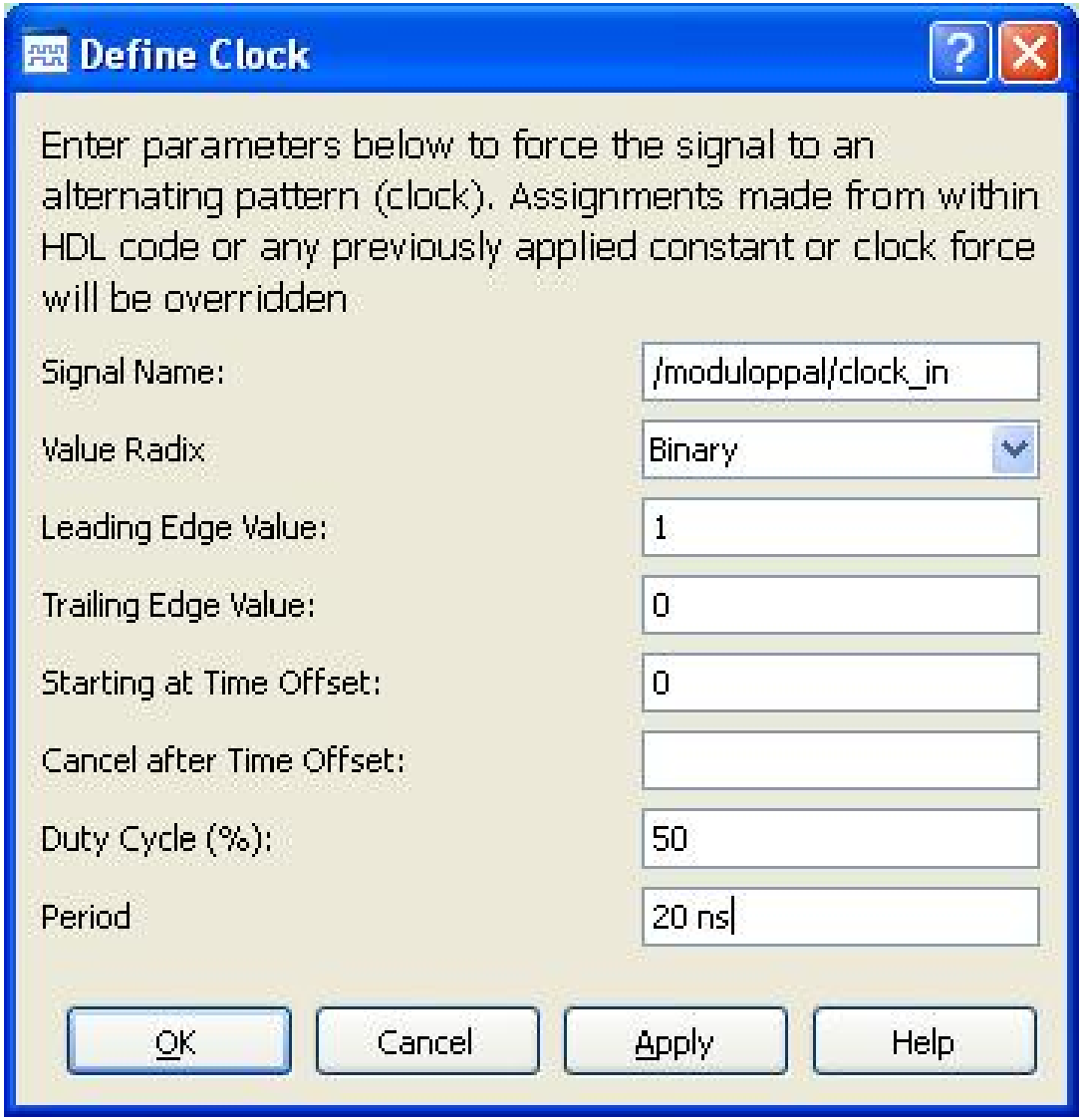
```
INFO:ProjectMgmt:656 - Parsing design hierarchy completed successfully.
INFO:HDLCompiler:1574 - Analyzing Verilog file \"C:/Ignacio/Facultad/CortexM0/ProyectoISE/CM0_DS_System/CORTEXM0DS.v\" into library work
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/Ignacio/Facultad/CortexM0/ProyectoISE/CM0_DS_System/Constant2Pulse.vhd" into library work
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/Ignacio/Facultad/CortexM0/ProyectoISE/CM0_DS_System/Counter2Constant.vhd" into library work
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/Ignacio/Facultad/CortexM0/ProyectoISE/CM0_DS_System/DetectorBus.vhd" into library work
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/Ignacio/Facultad/CortexM0/ProyectoISE/CM0_DS_System/ModuloPPal.vhd" into library work
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/Ignacio/Facultad/CortexM0/ProyectoISE/CM0_DS_System/ResetSincronico.vhd" into library work
INFO:HDLCompiler:1574 - Analyzing Verilog file \"C:/Ignacio/Facultad/CortexM0/ProyectoISE/CM0_DS_System/cortexm0ds_logic.v\" into library work
INFO:ProjectMgmt:656 - Parsing design hierarchy completed successfully.
Launching Design Summary/Report Viewer...
```

Errors Warnings Find in Files Results

Simulación funcional

- Es necesario forzar una entrada de reloj para la simulación, en este caso simularemos la salida del DCM:
  - Seleccionamos la señal clock_in y con el botón derecho del mouse seleccionamos “force clock”
- Luego simulamos 100 uS con esta configuración.

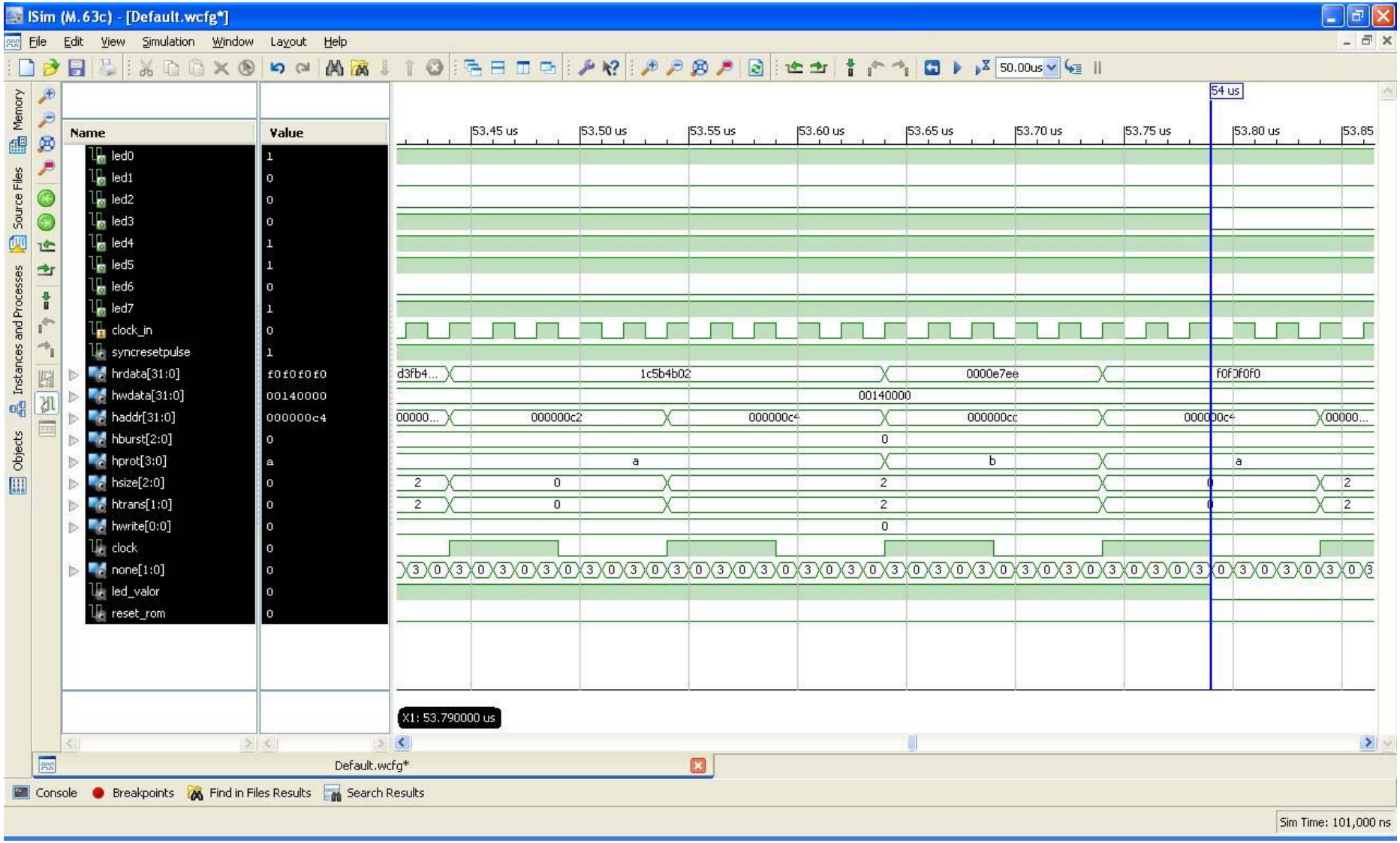
Simulación funcional



Simulación funcional

- Se observa que en el tiempo 53.79 uS se produce la aparición de la secuencia 0xf0f0f0f0 en el bus de datos de escritura (HWRITE), lo que produce la activación de la señal del detector del bus (Led03)
- Esto verifica el correcto funcionamiento del sistema en la simulación.

Simulación funcional



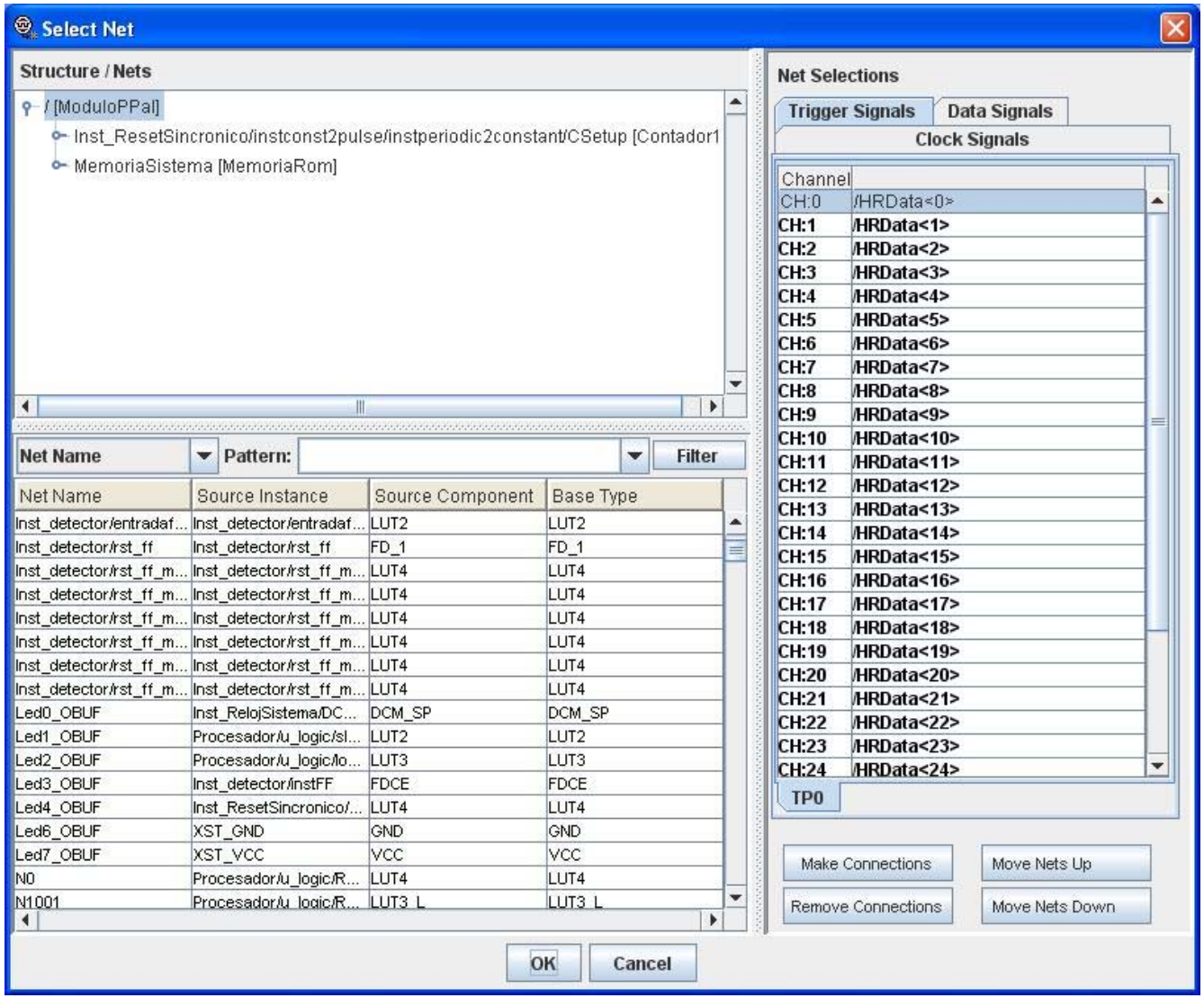
Logros de esta etapa

- Se ha verificado el correcto funcionamiento del sistema mediante una simulación funcional utilizando la herramienta ISIM del entorno Xilinx ISE

Verificación en hardware

- Se utiliza la herramienta ChipScope Pro para visualizar las señales en el interior de la FPGA
- Se debe crear y configurar un módulo CSP:
  - Se establece como condición de disparo la aparición del patrón 0xaaaa5555 en el bus HRDATA
  - Se monitorean las señales HADDR, HRDATA y la salida del detector.
  - Se utiliza como señal de reloj al reloj de sistema (HCLOCK)

Verificación en hardware



Verificación en hardware

- Al darse las condiciones de disparo, se observa en el bus HRDATA la aparición del patrón 0xaaaa5555

Verificación en hardware

ChipScope Pro Analyzer [ModuloPPal]

File View JTAG Chain Device Trigger Setup Waveform Window Help

Trigger Run Mode: Single

Project: ModuloPPal
JTAG Chain
DEV:0 MyDevice0 (XC3S500E)
UNIT:0 MyILA0 (ILA)
Trigger Setup
Waveform
Listing
Bus Plot
DEV:1 MyDevice1 (XCF04S)

Signals: DEV: 0 UNIT: 0
Data Port
Trigger Ports

Trigger Setup - DEV:0 MyDevice0 (XC3S500E) UNIT:0 MyILA0 (ILA)

Match Unit	Function	Value	Radix	Counter
M0:TriggerPort0	==	1010_1010_1010_1010_0101_0101_0101_0101	Bin	disabled

Add / Del	Active	Trigger Condition Name	Trigger Condition Equation
	◉	TriggerCondition0	M0

Type: Window Windows: 1 Depth: 512 Position: 0

Storage Qualification: All Data

Sample Buffer is full IDLE

Waveform - DEV:0 MyDevice0 (XC3S500E) UNIT:0 MyILA0 (ILA)

Bus/Signal	X	O	0	1	2	3	4	5	6	7
/Led3_OBUF	0	0								
/HADDR	029	029	029		02A		02B		02C	
/HRData	AAAA	AAAA	AAAA5555		20002100		1C49E001		42901C40	4B05D

Waveform captured 02/03/2011 22:42:14 X: 0 O: 0 Δ(X-O): 0

```
COMMAND: set_window_capture 0 0 0 1 512 0
COMMAND: set_match_function 0 0 0 0 3 1 10101010101010100101010101010101
COMMAND: set_trigger_condition 0 0 3 1 5555
COMMAND: set_storage_condition 0 0 FFFF
COMMAND: run 0 0
COMMAND: upload 0 0
INFO - Device 0 Unit 0: Sample Buffer is full
INFO - Device 0 Unit 0: Waveform captured 02/03/2011 22:42:14
```

DONE

Logros de esta etapa

- Se verificó la aparición del patrón prestablecido en las señales internas de la FPGA
- Se obtuvo una concordancia entre la simulación de software en el entorno ARM MDK, la simulación funcional con ISIM y la verificación de hardware con ChipScope Pro.

Información de contacto

- Laboratorio de sistemas embebidos (LSE)

  http://laboratorios.fi.uba.ar/lse
- Fabricio Baglivo

  baglivofabricio@gmail.com
- Pedro Martos

  http://www.fi.uba.ar/~pmartos

  pmartos@fi.uba.ar