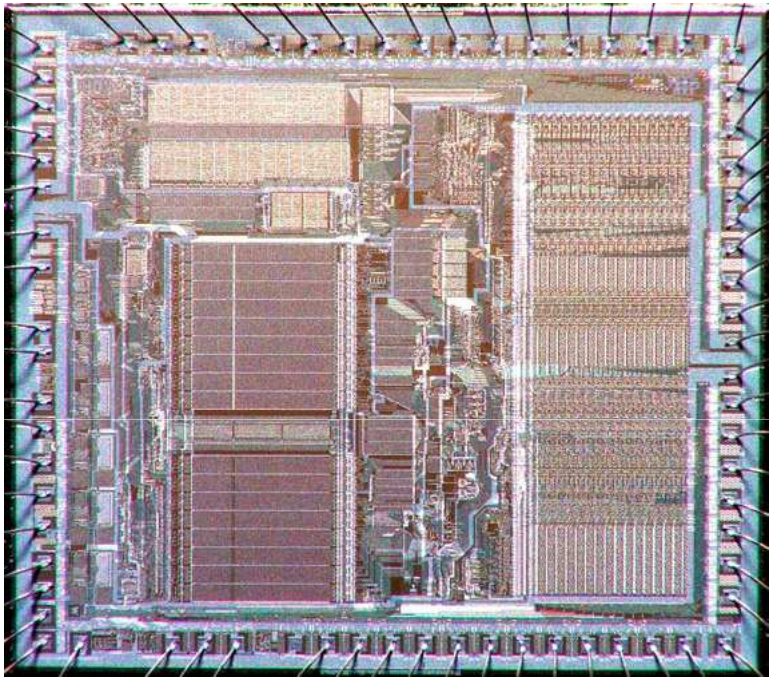
Universidad
Carlos III de Madrid

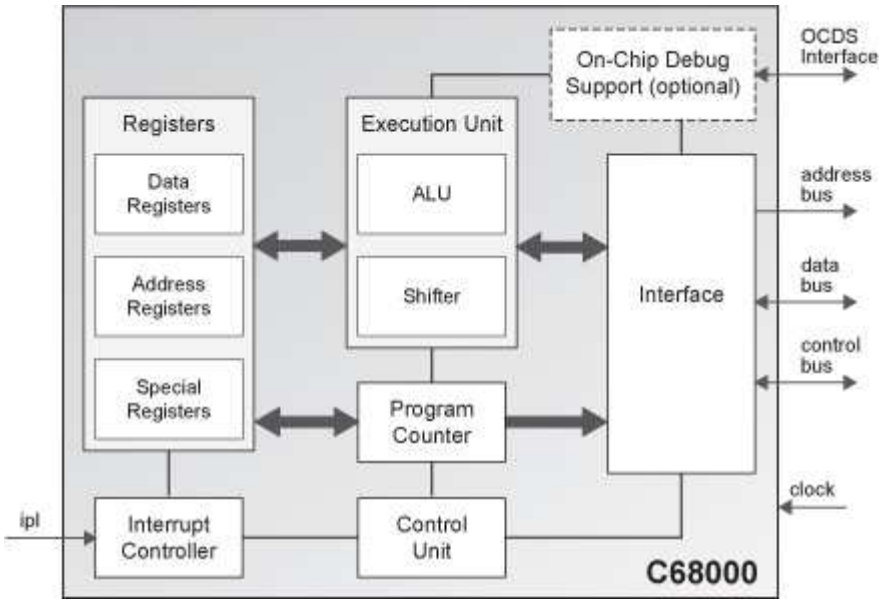
Sistemas embebidos basados en FPGAs para instrumentación

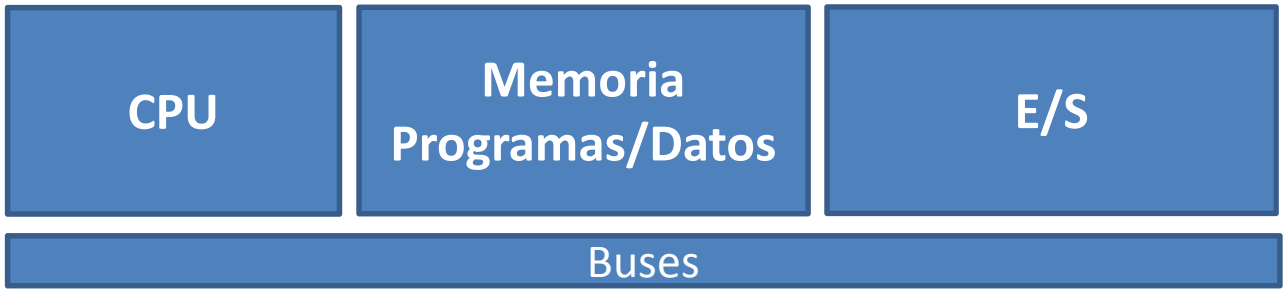
Introducción a los procesadores empotrados en FPGAs. PicoBlaze

Guillermo Carpintero del Barrio

Arquitectura de un Procesador







Soft Cores

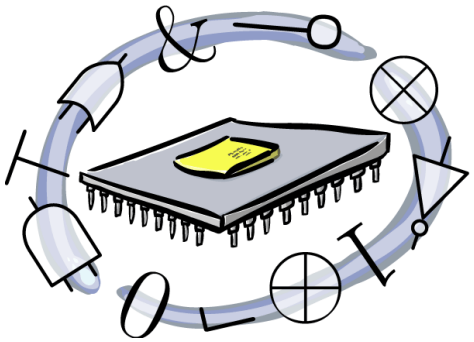
Implementación sobre FPGA

Requiere la descripción hardware de varios elementos nuevos:

Procesador (Soft Core)
- PicoBlaze (gratuito)
- MicroBlaze (IP)

Interfases E/S
- LEDs y 7-segmentos
- LCD
- RS232

IP CORE



For different applications, avoids design of customized FSM hardware in each by keeping the same hardware (processor core) and using customized software


PicoBlaze™

Características Básicas

Datos CPU de 8-bits
16 registros de datos
64 posiciones de memoria de datos (dirección de 6-bit)

ALU de 8-bits con flags C (carry) y Z (zero)

256 puertos de entrada y **256** puertos de salida

Instrucciones

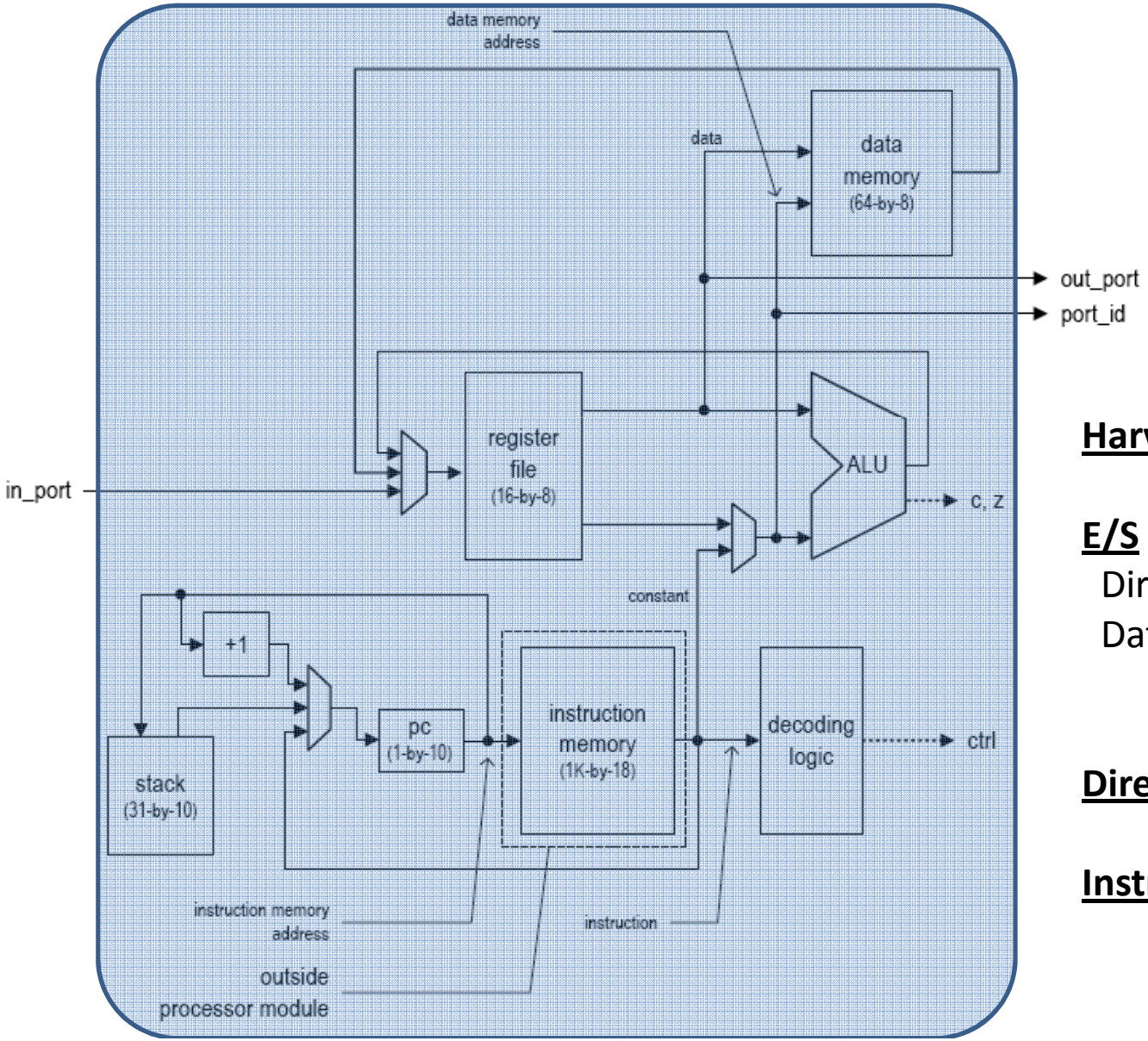
57 Instrucciones de 18-bits
1024 posiciones de memoria de instrucciones (dirección de 10-bits)
2 CPI (ejecución) y 5 CPI (una fuente de interrupción)

@ 50 MHz,
25 MIPS

Procesador sencillo para aplicaciones:

Procesado sencillo de datos (en operaciones no críticas)
Interfases E/S
Escáner de teclado

Diagrama de Bloques (I)



Harvard

E/S

Dirección: port_id

Data: out_port
in_port

Direccionamiento Inmediato

Instrucciones Call/Return

Bloques de síntesis

KCPSM = Constant Coded Programable State Machine

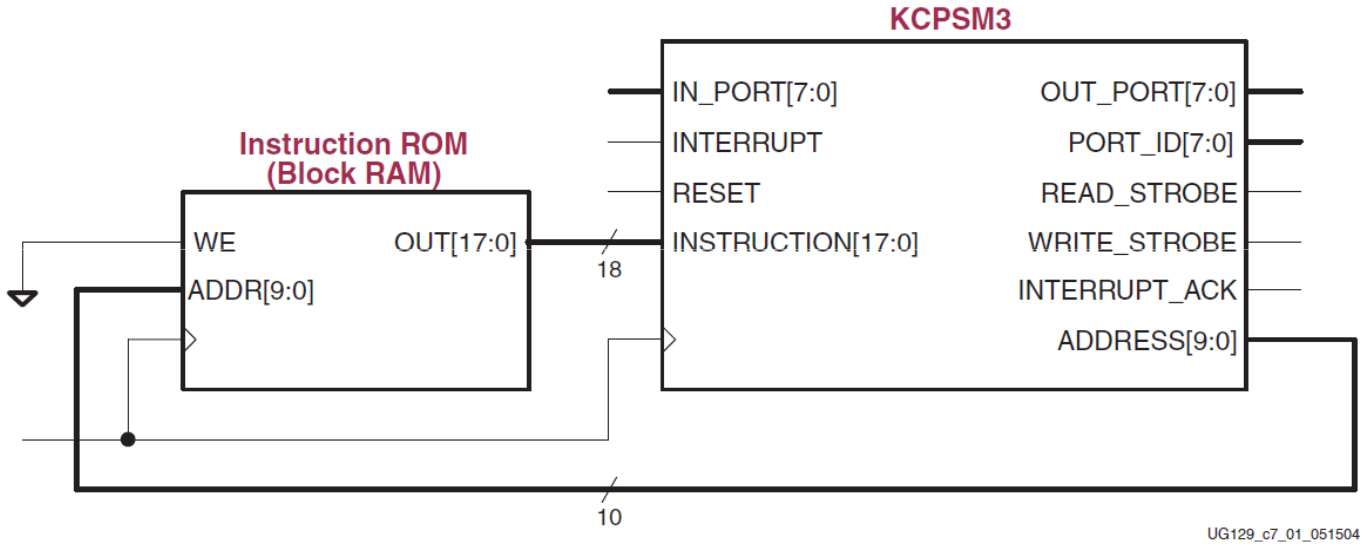
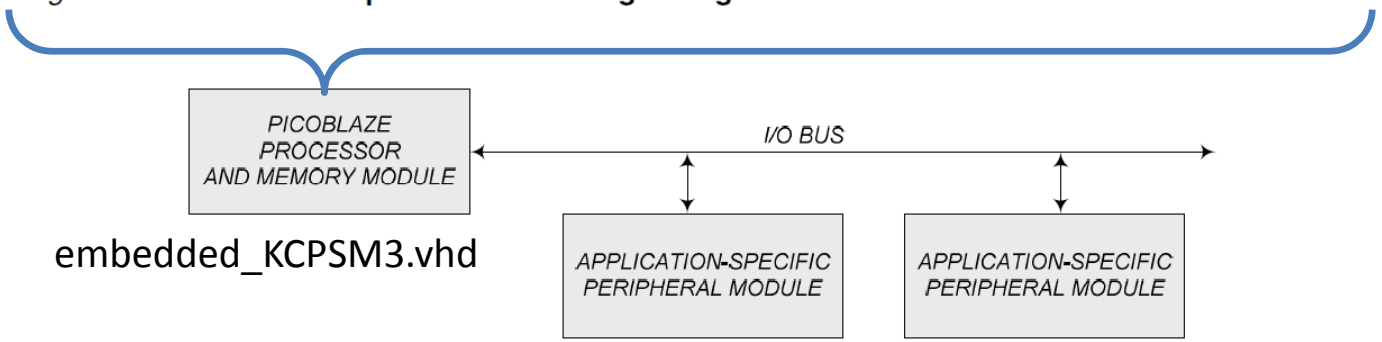


Figure 7-1: **Standard Implementation using a Single 1Kx18 Block RAM as the Instruction Store**



embedded_KCPSM3.vhd

```
component KCPSM3
port (
      address      : out std_logic_vector( 9 downto 0);
      instruction  : in  std_logic_vector(17 downto 0);
      port_id      : out std_logic_vector( 7 downto 0);
      write_strobe : out std_logic;
      out_port     : out std_logic_vector( 7 downto 0);
      read_strobe  : out std_logic;
      in_port      : in  std_logic_vector( 7 downto 0);
      interrupt    : in  std_logic;
      interrupt_ack: out std_logic;
      reset        : in  std_logic;
      clk          : in  std_logic
     );
end component;
```

Figure 9-1: **VHDL Component Declaration of KCPSM3**

```
processor: kcpsm3
    port map(
                   address => address_signal,
               instruction => instruction_signal,
                   port_id => port_id_signal,
              write_strobe => write_strobe_signal,
                  out_port => out_port_signal,
               read_strobe => read_strobe_signal,
                   in_port => in_port_signal,
                 interrupt => interrupt_signal,
             interrupt_ack => interrupt_ack_signal,
                     reset => reset_signal,
                       clk => clk_signal
            );
```

Figure 9-2: **VHDL Component Instantiation of the KCPSM3**

```
component prog_rom
port (
      address     : in  std_logic_vector( 9 downto 0);
      instruction : out std_logic_vector(17 downto 0);
      clk         : in  std_logic
     );
end component;
```

Figure 9-3: **VHDL Component Declaration of Program ROM**

```
program: prog_rom
port map(    address => address_signal,
         instruction => instruction_signal,
                 clk => clk_signal
        );
```

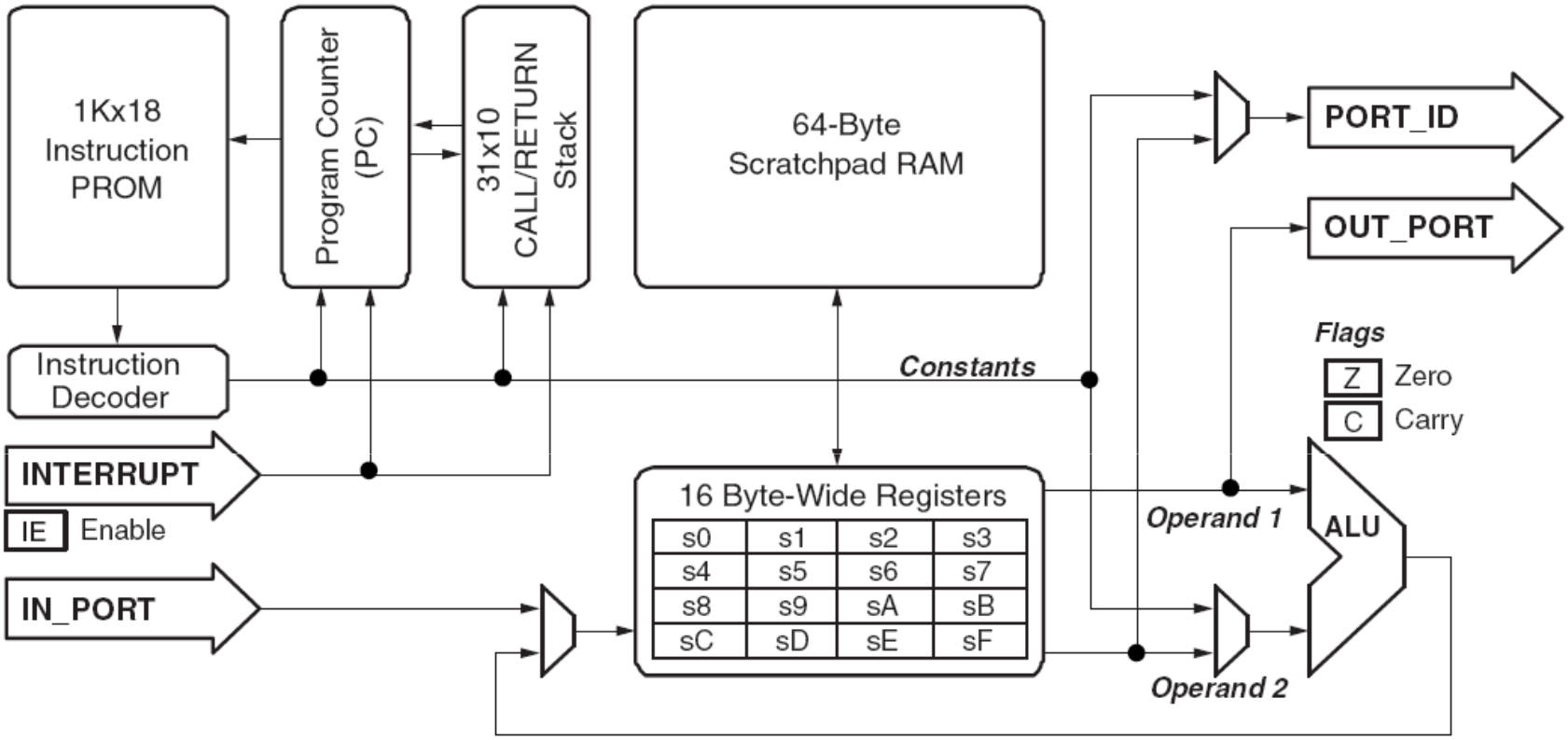
Figure 9-4: **VHDL Component Instantiation of Program ROM**

El ensamblador de PicoBlaze genera un fichero VHDL en el que se define un bloque de 1K de RAM y sus contenidos iniciales (el programa).

Este fichero VHDL se puede utilizar en la síntesis y en la simulación.

Si nuestro programa se llama **test.psm**, el ensamblador genera un fichero ROM que en vez de ser **prog_rom.vhd** es **test.vhd**

Diagrama de Bloques (II)



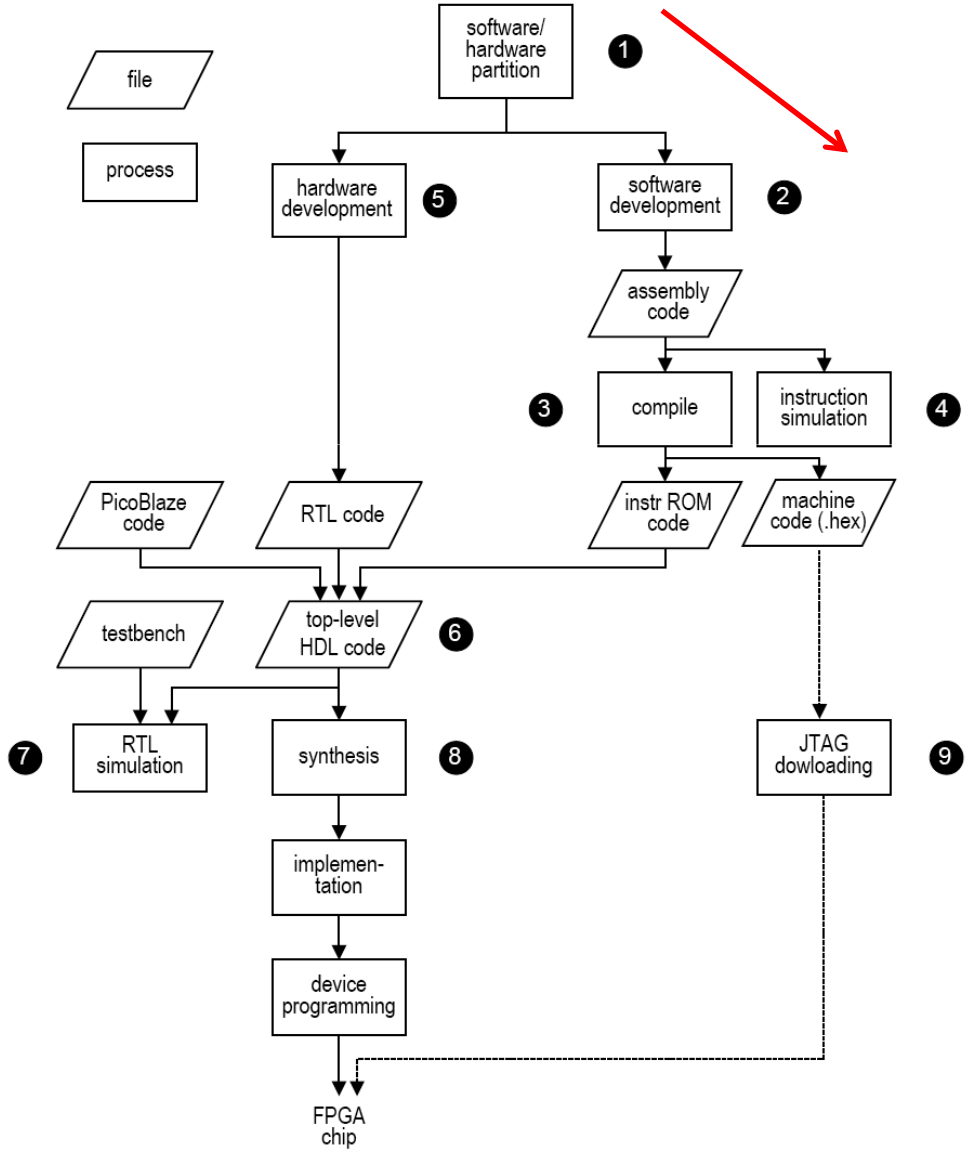
CLAVES

sX, sY	– registros internos del procesador (X, Y = 0x00 a 0x0F)
pc	– contador de programa
c, z, i	– carry, zero, interrupt flags

Interface del Procesador

Name	Direction	Size	Function
clk	input	1	System clock signal.
reset	input	1	Reset signal.
address	output	10	Address of the instruction memory. Specifies address of the instruction to be retrieved.
instruction	input	18	Fetched instruction.
port_id	output	8	Address of the input or output port.
in_port	input	8	Input data from I/O peripherals.
read_strobe	output	1	Strobe associated with the input operation.
out_port	output	8	Output data to I/O peripherals.
write_strobe	output	1	Strobe associated with the output operation.
interrupt	input	1	Interrupt request from I/O peripherals.
interrupt_ack	output	1	Interrupt acknowledgment to I/O peripherals

Flujo de diseño del sistema embebido



Juego de Instrucciones - Claves

Sintaxis	Rango	Ejemplo	Definition -
sX	0 a f	s7	Valor en registro 7
KK	0 a ff	ab	Valor hex ab (0xAB)
PORT(KK)	0 a ff	PORT(2)	Valor de entrada en puerto 2
PORT((sX))	0 a f	PORT((sa))	Valor de entrada en puerto indicado por registro a
RAM(KK)	0 a f	RAM(4)	Valor en dirección 4 de la RAM

Juego de Instrucciones – Repertorio (I)

Instruction	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD sX,kk	0	1	1	0	0	0	x	x	x	x	k	k	k	k	k	k	k	k
ADD sX,sY	0	1	1	0	0	1	x	x	x	x	y	y	y	y	0	0	0	0
ADDCY sX,kk	0	1	1	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
ADDCY sX,sY	0	1	1	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
AND sX,kk	0	0	1	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
AND sX,sY	0	0	1	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
CALL	1	1	0	0	0	0	0	0	a	a	a	a	a	a	a	a	a	a
CALL C	1	1	0	0	0	1	1	0	a	a	a	a	a	a	a	a	a	a
CALL NC	1	1	0	0	0	1	1	1	a	a	a	a	a	a	a	a	a	a
CALL NZ	1	1	0	0	0	1	0	1	a	a	a	a	a	a	a	a	a	a
CALL Z	1	1	0	0	0	1	0	0	a	a	a	a	a	a	a	a	a	a
COMPARE sX,kk	0	1	0	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
COMPARE sX,sY	0	1	0	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
DISABLE INTERRUPT	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ENABLE INTERRUPT	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FETCH sX, ss	0	0	0	1	1	0	x	x	x	x	0	0	s	s	s	s	s	s
FETCH sX,(sY)	0	0	0	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
INPUT sX,(sY)	0	0	0	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
INPUT sX,pp	0	0	0	1	0	0	x	x	x	x	p	p	p	p	p	p	p	p

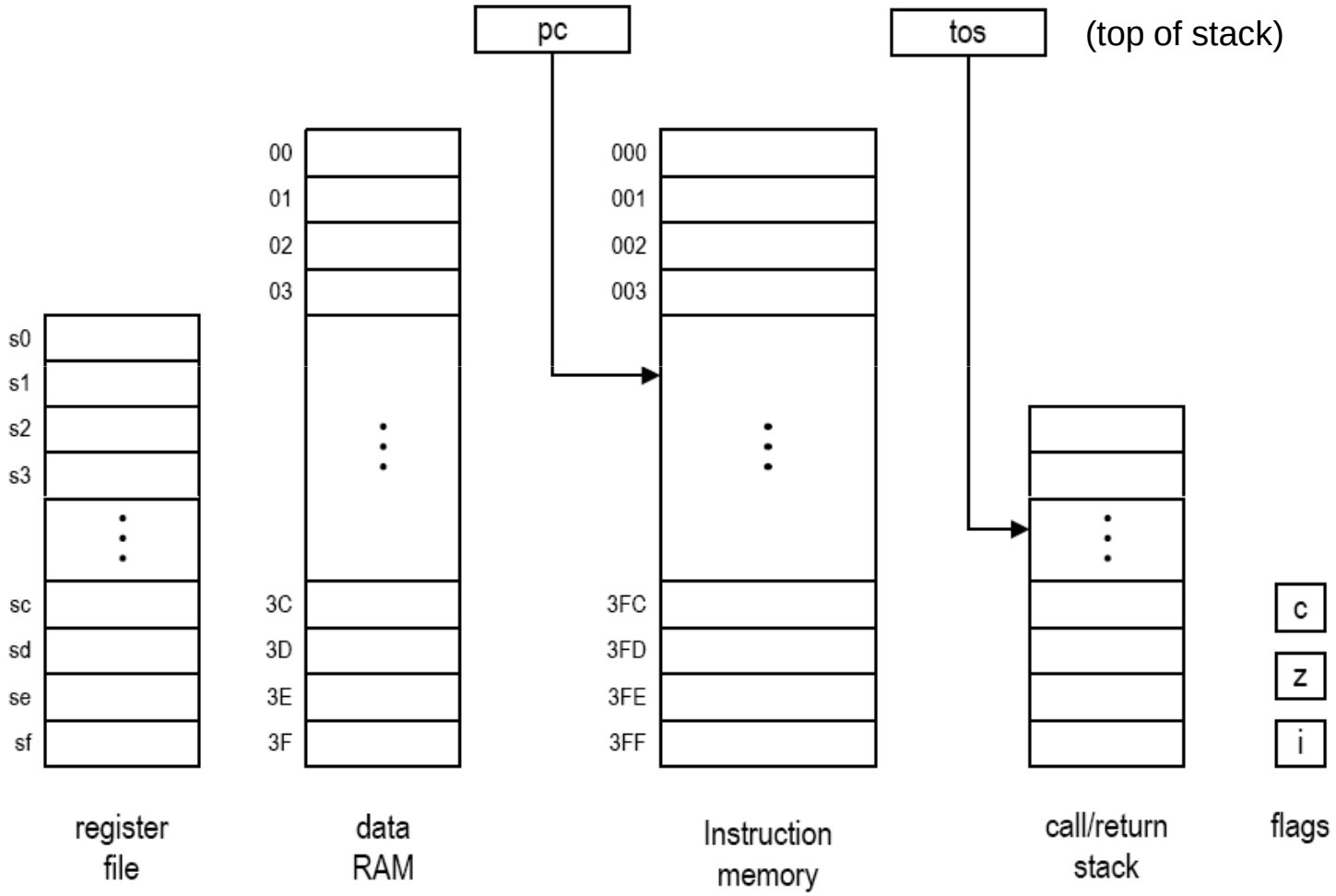
Juego de Instrucciones – Repertorio (II)

JUMP	1	1	0	1	0	0	0	0	a	a	a	a	a	a	a	a	a	a
JUMP C	1	1	0	1	0	1	1	0	a	a	a	a	a	a	a	a	a	a
JUMP NC	1	1	0	1	0	1	1	1	a	a	a	a	a	a	a	a	a	a
JUMP NZ	1	1	0	1	0	1	0	1	a	a	a	a	a	a	a	a	a	a
JUMP Z	1	1	0	1	0	1	0	0	a	a	a	a	a	a	a	a	a	a
LOAD sX,kk	0	0	0	0	0	0	x	x	x	x	k	k	k	k	k	k	k	k
LOAD sX,sY	0	0	0	0	0	1	x	x	x	x	y	y	y	y	0	0	0	0
OR sX,kk	0	0	1	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
OR sX,sY	0	0	1	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
OUTPUT sX,(sY)	1	0	1	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
OUTPUT sX,pp	1	0	1	1	0	0	x	x	x	x	p	p	p	p	p	p	p	p
RETURN	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
RETURN C	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
RETURN NC	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
RETURN NZ	1	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0
RETURN Z	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
RETURNI DISABLE	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RETURNI ENABLE	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

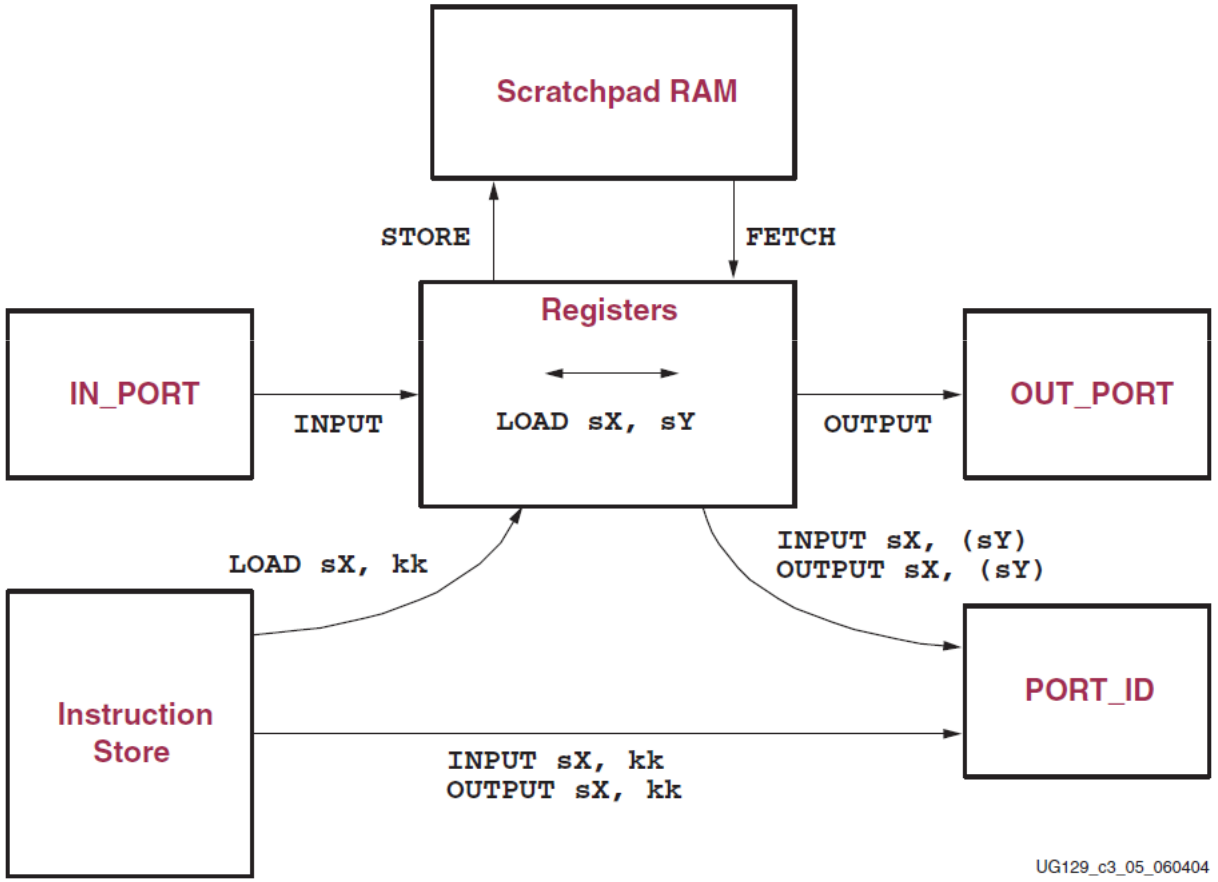
Juego de Instrucciones – Repertorio (III)

RL sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	0	1	0
RR sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	1	0	0
SL0 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	1	1	0
SL1 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	1	1	1
SLA sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	0	0	0
SLX sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	1	0	0
SR0 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	1	1	0
SR1 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	1	1	1
SRA sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	0	0	0
SRX sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	0	1	0
STORE sX, ss	1	0	1	1	1	0	x	x	x	x	0	0	s	s	s	s	s	s
STORE sX,(sY)	1	0	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
SUB sX,kk	0	1	1	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
SUB sX,sY	0	1	1	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
SUBCY sX,kk	0	1	1	1	1	0	x	x	x	x	k	k	k	k	k	k	k	k
SUBCY sX,sY	0	1	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
TEST sX,kk	0	1	0	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
TEST sX,sY	0	1	0	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
XOR sX,kk	0	0	1	1	1	0	x	x	x	x	k	k	k	k	k	k	k	k
XOR sX,sY	0	0	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0

Modelo del Programador

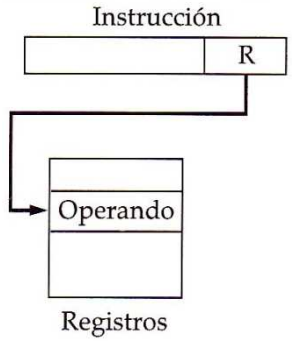


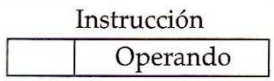
Juego de Instrucciones – Movimiento de Datos

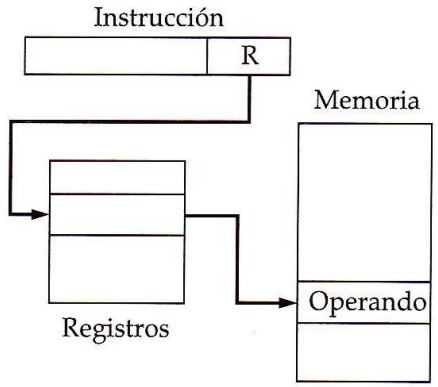


Juego de Instrucciones – Modos de Direccionamiento

Modo Directo	
INPUT s5, 2a	PORT(2a) → s5
ADD sa, sf	sa + sf → sa
Modo Inmediato	
ADDCY s2, 08	s2 + 08 + C → s2
SUB s7, 7	s7 – 7 → s7
Modo Indirecto	
INPUT s9, (s2)	PORT((s2)) → s9
STORE s3, (sa)	s3 → RAM((sa))







Juego de Instrucciones – Users Guide Information (I)

1. Información completa sobre el Juego de Instrucciones

Instruction	Description	Function	ZERO	CARRY
ADD sX, kk	Add register sX with literal kk	sX ← sX + kk	?	?
ADD sX, sY	Add register sX with register sY	sX ← sX + sY	?	?
ADDCY sX, kk (ADDC)	Add register sX with literal kk with CARRY bit	sX ← sX + kk + CARRY	?	?
ADDCY sX, sY (ADDC)	Add register sX with register sY with CARRY bit	sX ← sX + sY + CARRY	?	?

2. Información sobre operaciones no implementadas en el Juego de Instrucciones

Complementar un Registro XOR sX,FF

Complementar (Toggle) un Bit *XOR sX, <bit_mask> ; 1 en bit a complementar*

Puesta a 0 (Clear) un Registro *XOR sX, sX ; sets ZERO flag*
LOAD sX,00 ; ZERO flag unaffected

Set Bit *OR sX, <bit_mask> ; 1 en bit a activar*

Clear Bit *AND sX, <bit_mask> ; 0 en bit a desactivar*

Nop *LOAD sX, sX*

Juego de Instrucciones – Users Guide Information (II)

```
Negate:
    ; invert all bits in the register performing a one's complement
    XOR sX,FF
    ; add one to sX
    ADD sX,01
    RETURN
```

Figure 3-12: **Destructive Negate (2's Complement) Function Overwrites Original Value**

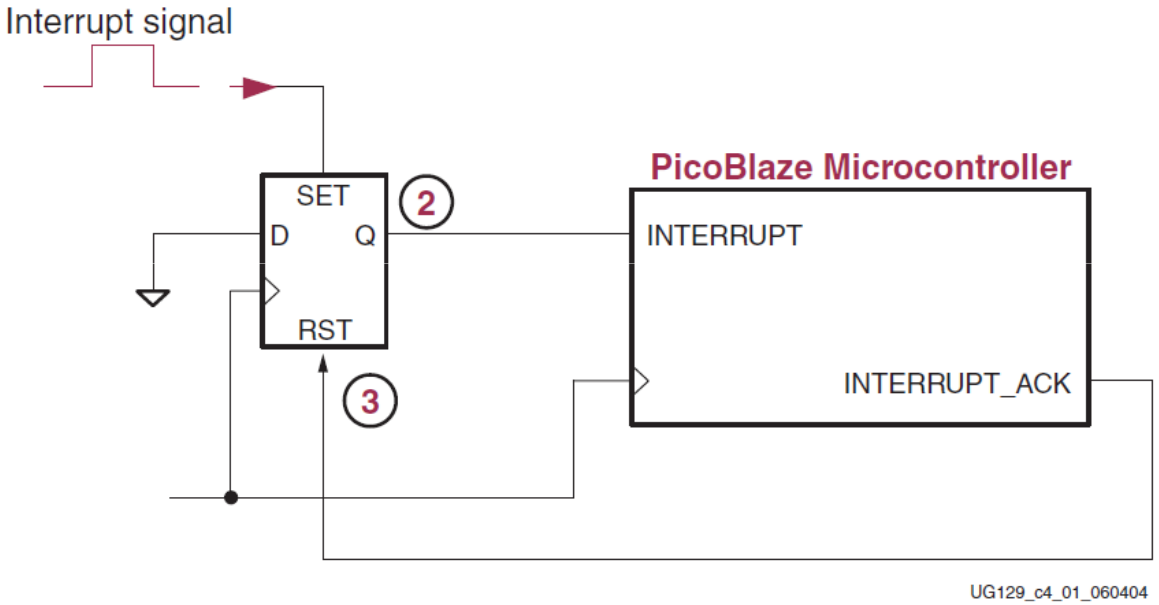
```
Negate:
    NAMEREG sY, value
    NAMEREG sX, complement
    ; Clear 'complement' to zero
    LOAD complement, 00
    ; subtract value from 0 to create two's complement
    SUB complement, value
    RETURN
```

Figure 3-13: **Non-destructive Negate Function Preserves Original Value**

Interrupciones

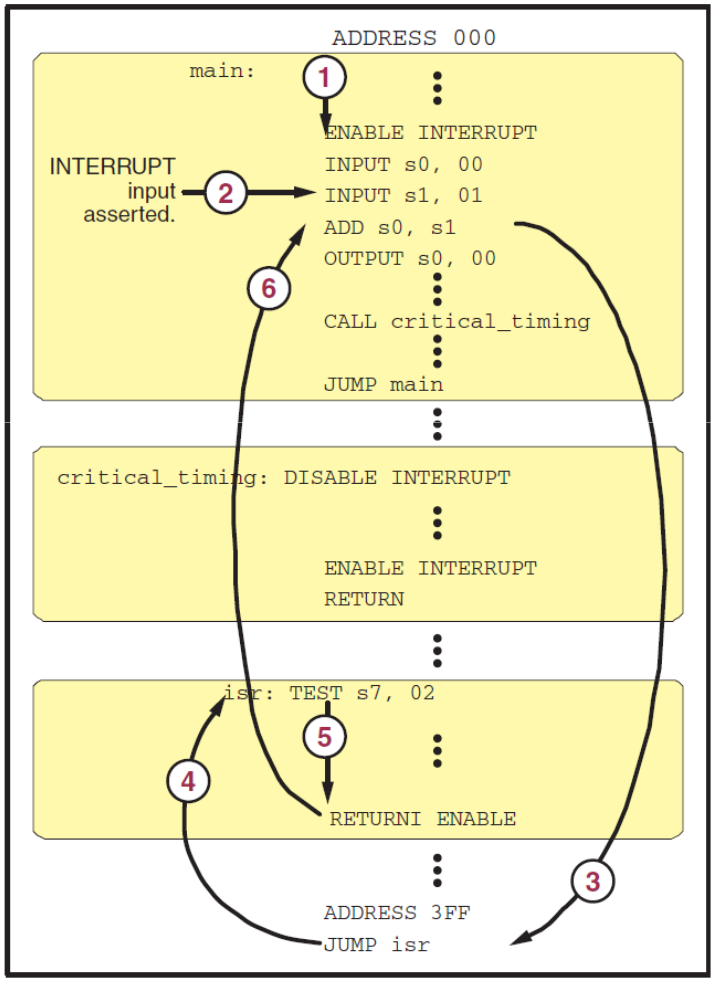
Hay una única línea de solicitud de interrupción.

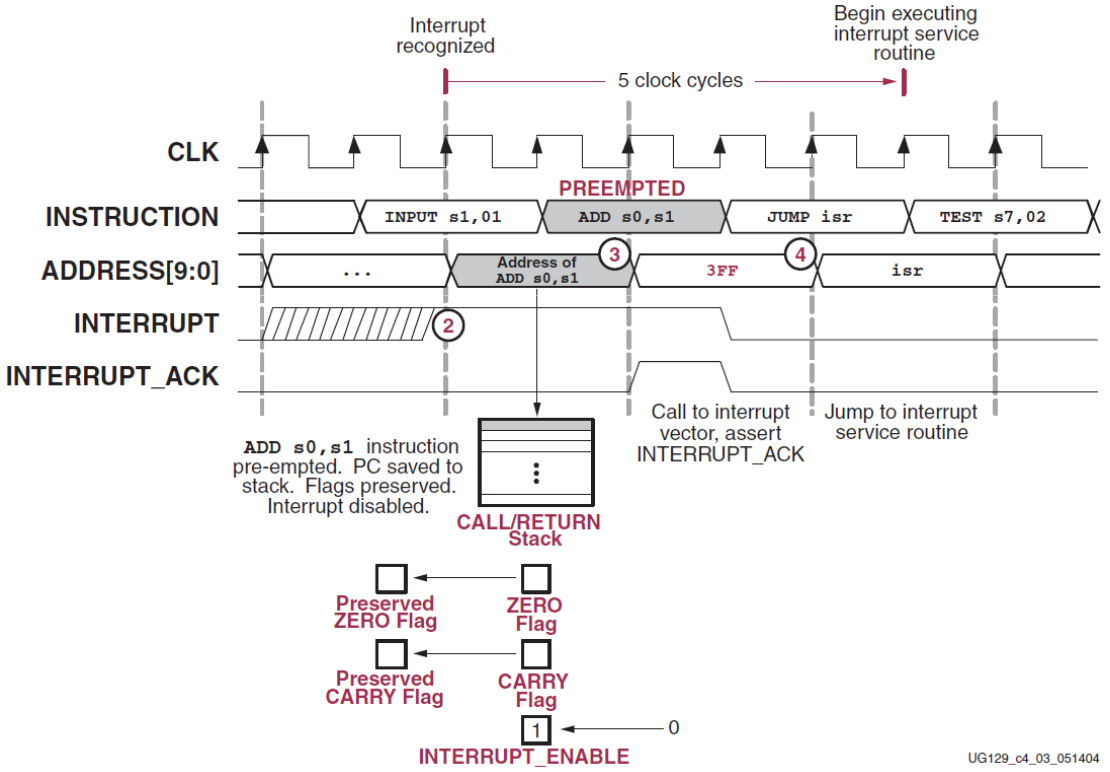
Si necesitamos múltiples fuentes, podemos utilizar la FPGA para construir un sistema de gestión.



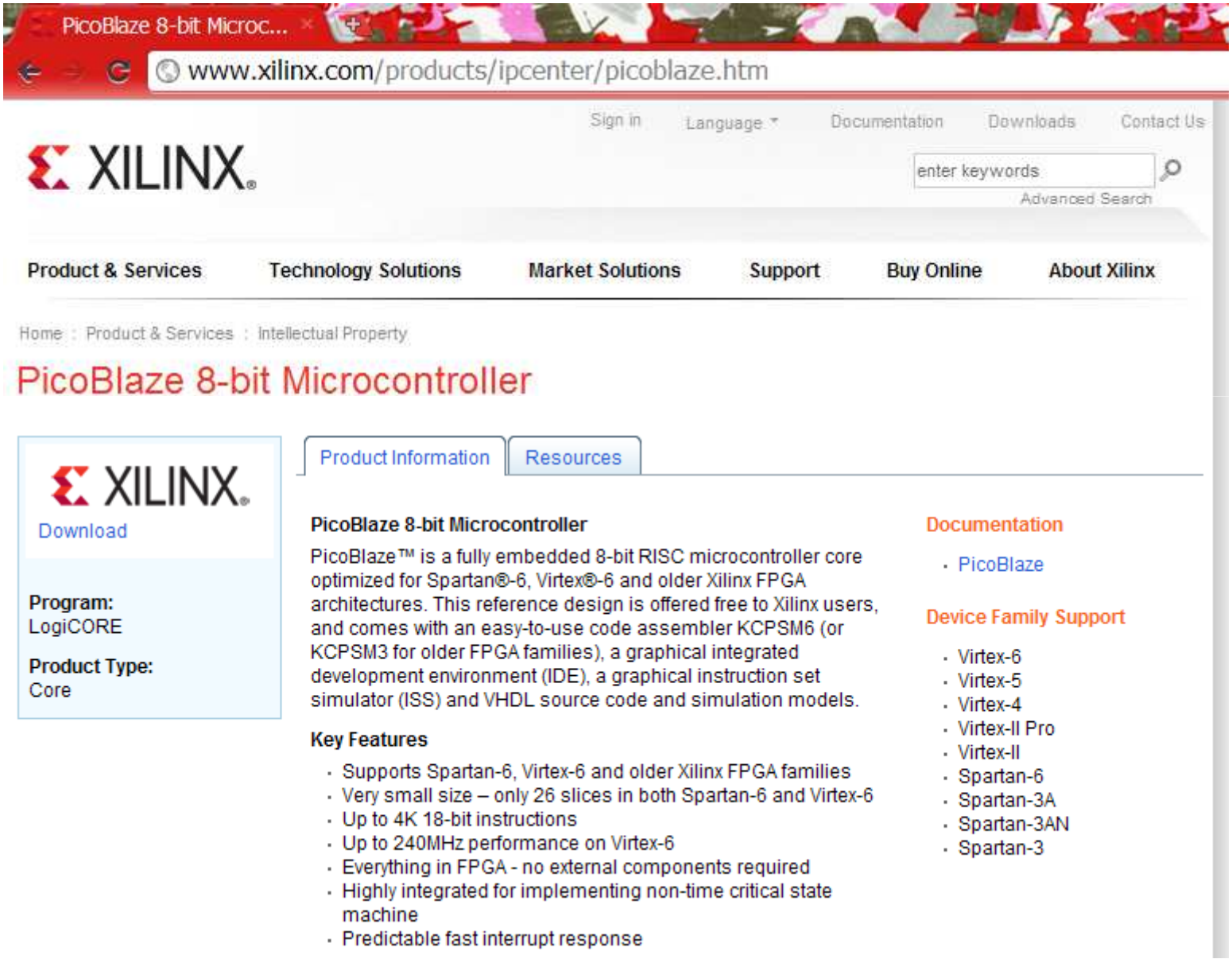
La activación de la interrupción, causa la ejecución de CALL 3FF (última posición memoria prog).

Interrupciones





www.xilinx.com/picoblaze



Registered PicoBlaze Lounge @ www.xilinx.com

Home : Technology Solutions : Embedded Processing : PicoBlaze Soft Processor : PicoBlaze Lounge

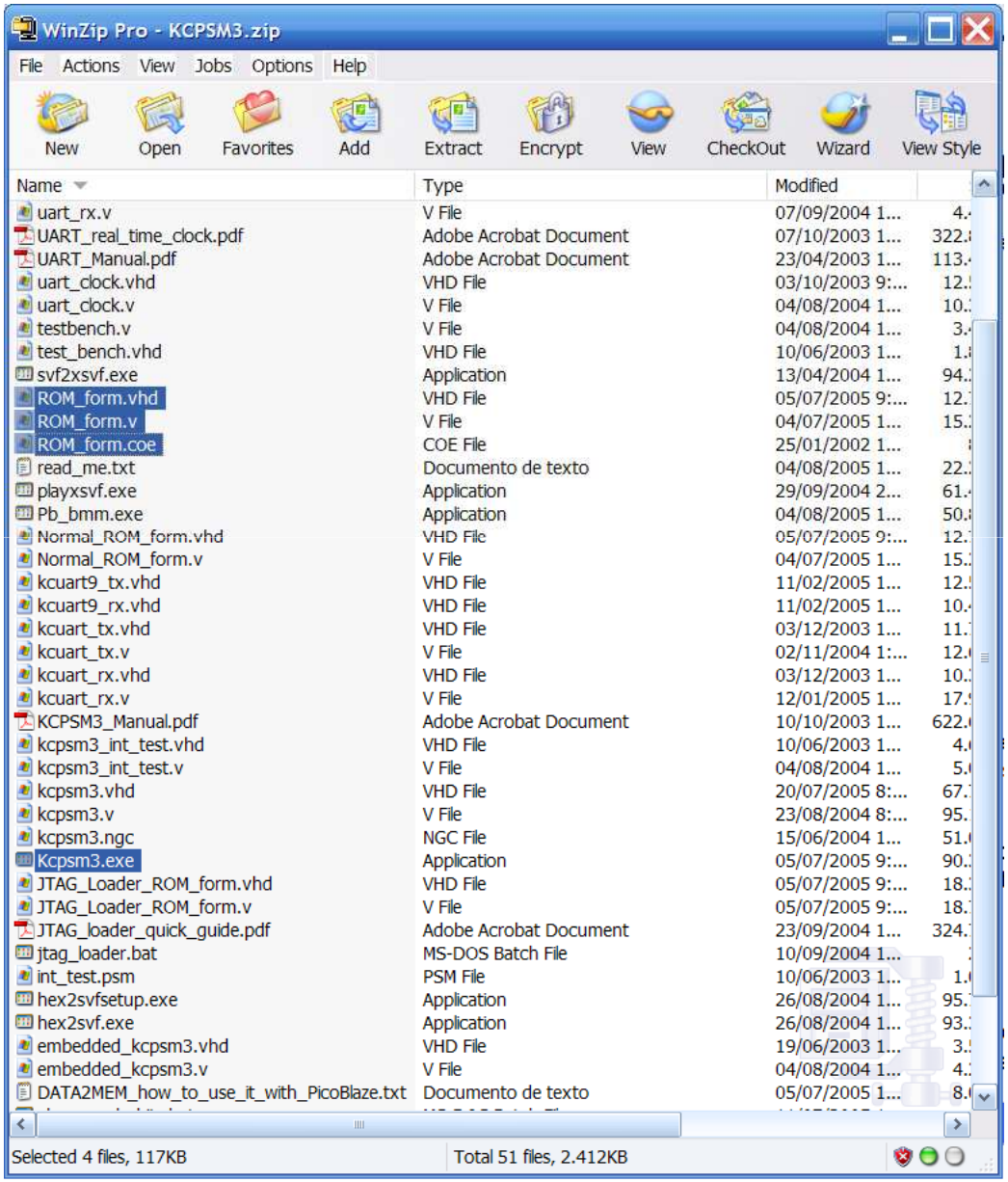
PicoBlaze Lounge

Welcome to the PicoBlaze™ Lounge. This site provides you with access to the latest PicoBlaze reference design files. Please remember that the content of this site is covered by the Xilinx Reference Design License agreement and should be treated as such. You may now browse and download the latest PicoBlaze reference design files.

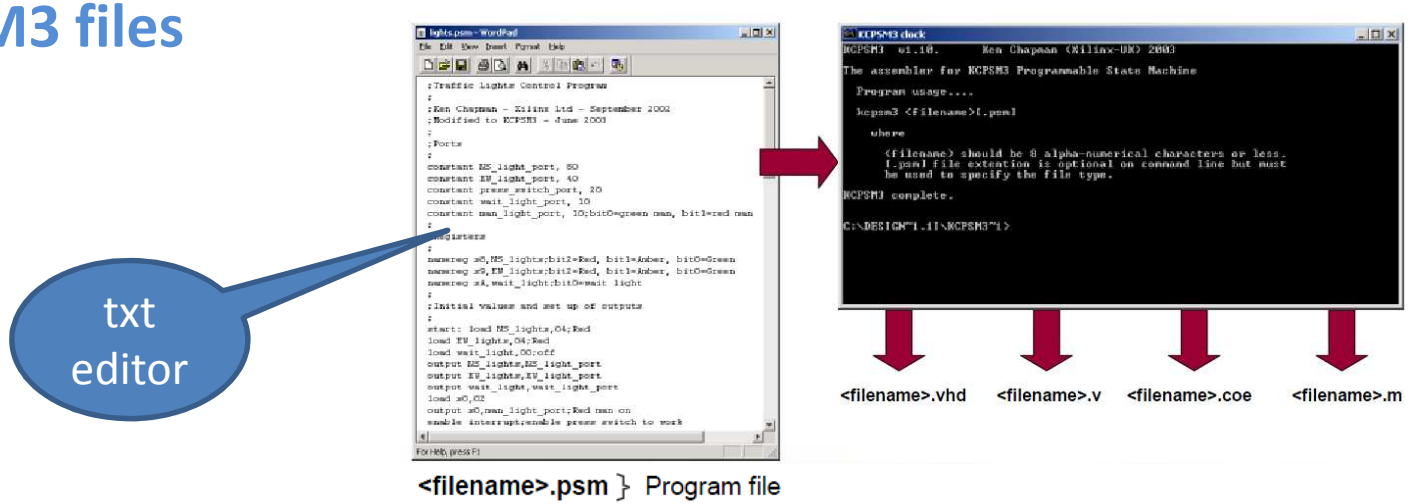
PicoBlaze for Virtex™-6 FPGAs >> New – KCPSM6 optimized for LUT6 architectures	Download design files
PicoBlaze for Spartan™-6 FPGAs >> New – KCPSM6 optimized for LUT6 architectures	Download design files
PicoBlaze for Virtex-5 FPGAs	Download design files
PicoBlaze for Spartan-3, Virtex-4, Virtex-II and Virtex-II Pro FPGAs	Download design files
PicoBlaze for Virtex, Virtex-E, Spartan-II and Spartan-IIE FPGAs	Download design files
PicoBlaze for Virtex-II, Virtex-II Pro FPGAs	Download design files
PicoBlaze for CoolRunner™-II CPLDs	Download design files

KCPSM3.zip

KCPSM3.zip


WinZip Pro - KCPSM3.zip
File Actions View Jobs Options Help
New Open Favorites Add Extract Encrypt View CheckOut Wizard View Style
Name Type Modified
uart_rx.v V File 07/09/2004 1...
UART_real_time_clock.pdf Adobe Acrobat Document 07/10/2003 1...
UART_Manual.pdf Adobe Acrobat Document 23/04/2003 1...
uart_clock.vhd VHD File 03/10/2003 9:...
uart_clock.v V File 04/08/2004 1...
testbench.v V File 04/08/2004 1...
test_bench.vhd VHD File 10/06/2003 1...
svf2xsvf.exe Application 13/04/2004 1...
ROM_form.vhd VHD File 05/07/2005 9:...
ROM_form.v V File 04/07/2005 1...
ROM_form.coe COE File 25/01/2002 1...
read_me.txt Documento de texto 04/08/2005 1...
playxsvf.exe Application 29/09/2004 2...
Pb_bmm.exe Application 04/08/2005 1...
Normal_ROM_form.vhd VHD File 05/07/2005 9:...
Normal_ROM_form.v V File 04/07/2005 1...
kcuart9_tx.vhd VHD File 11/02/2005 1...
kcuart9_rx.vhd VHD File 11/02/2005 1...
kcuart_tx.vhd VHD File 03/12/2003 1...
kcuart_tx.v V File 02/11/2004 1:...
kcuart_rx.vhd VHD File 03/12/2003 1...
kcuart_rx.v V File 12/01/2005 1...
KCPSM3_Manual.pdf Adobe Acrobat Document 10/10/2003 1...
kcpsm3_int_test.vhd VHD File 10/06/2003 1...
kcpsm3_int_test.v V File 04/08/2004 1...
kcpsm3.vhd VHD File 20/07/2005 8:...
kcpsm3.v V File 23/08/2004 8:...
kcpsm3.ngc NGC File 15/06/2004 1...
Kcpsm3.exe Application 05/07/2005 9:...
JTAG_Loader_ROM_form.vhd VHD File 05/07/2005 9:...
JTAG_Loader_ROM_form.v V File 05/07/2005 9:...
JTAG_loader_quick_guide.pdf Adobe Acrobat Document 23/09/2004 1...
jtag_loader.bat MS-DOS Batch File 10/09/2004 1...
int_test.psm PSM File 10/06/2003 1...
hex2svfsetup.exe Application 26/08/2004 1...
hex2svf.exe Application 26/08/2004 1...
embedded_kcpsm3.vhd VHD File 19/06/2003 1...
embedded_kcpsm3.v V File 04/08/2004 1...
DATA2MEM_how_to_use_it_with_PicoBlaze.txt Documento de texto 05/07/2005 1...
Selected 4 files, 117KB
Total 51 files, 2.412KB

KCPSM3 files



Consultar el fichero **KCPSM3_Manual.pdf**

PicoBlaze IDE

	Xilinx KCPSM3	Mediatronix pBlazIDE	Xilinx System Generator
Platform Support	Windows	Windows 98, Windows 2000, Windows NT, Windows ME, Windows XP	Windows 2000, Windows XP
Assembler	Command-line in DOS window	Graphical	Command-line within System Generator
Instruction Syntax	KCPSM3	PBlazIDE	KCPSM3
Instruction Set Simulator	Facilities provided for VHDL simulation	Graphical/Interactive	Graphical/Interactive
Simulator Breakpoints	N/A	Yes	Yes
Register Viewer	N/A	Yes	Yes
Memory Viewer	N/A	Yes	Yes

Directivas del Ensamblador

Function	KCPSM3 Directive	PBlazIDE Directive
Locating Code	`ADDRESS 3FF`	`ORG $3FF`
Aliasing Register Names	`NAMEREG s5, myregname`	`myregname EQU s5`
Declaring Constants	`CONSTANT myconstant, 80`	`myconstant EQU $80`
Naming the program ROM file	Named using the same base filename as the assembler source file	`VHDL "template.vhd", "target.vhd", "entity_name"`
Equating symbolic name for an I/O port ID.	N/A	keyboard **DSIN** $0E switch **DSIN** $0F LED **DSOUT** $15

Diferencias en los Nemónicos Ensamblador

KCPSM3 Instruction	pBlazIDE Instruction
RETURN	RET
RETURN C	RET C
RETURN NC	RET NC
RETURN Z	RET Z
RETURN NZ	RET NZ
RETURNI ENABLE	RETI ENABLE
RETURNI DISABLE	RETI DISABLE
ADDCY	ADDC
SUBCY	SUBC
INPUT sX, (sY)	IN sX, sY (no parentheses)
INPUT sX, kk	IN sX, kk
OUTPUT sX, (sY)	OUT sX, sY (no parentheses)
OUTPUT sX, kk	OUT sX, kk
ENABLE INTERRUPT	EINT
DISABLE INTERRUPT	DINT
COMPARE	COMP
STORE sX, (sY)	STORE sX, sY (no parentheses)
FETCH sX, (sY)	FETCH sX, sY (no parentheses)

Diferencias en los Nemónicos Ensamblador (III)

KCMPSM Source Code

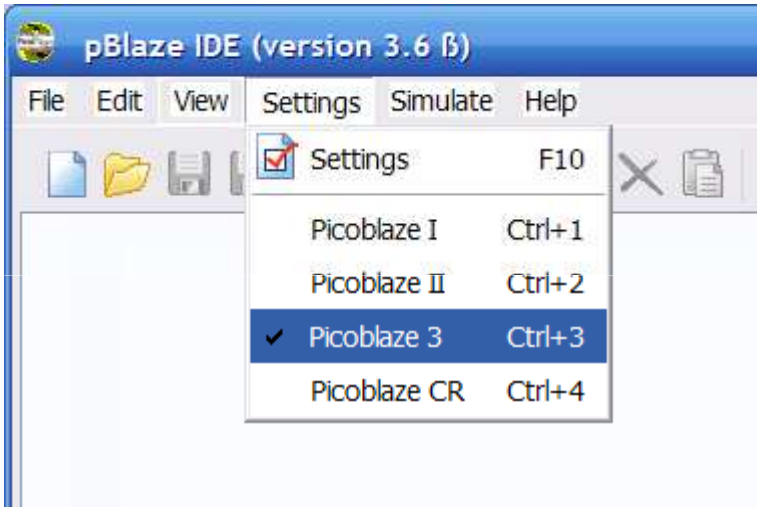
```
                CONSTANT myconstant, A5

                NAMEREG s0, count16_lsb
                NAMEREG s1, count16_msb

                ADDRESS 000

           main:
; initialize 16-bit counter, enable interrupts
                LOAD count16_lsb, myconstant
                ENABLE INTERRUPT

           loop:
; continuously increment 16-bit counter
                CALL increment_count
                JUMP loop
      end_main:

increment_count:
; add 1 to LSB of 16-bit counter
                ADD count16_lsb, 01
; only add one to MSB if carry generated by LSB
                ADDCY count16_msb, 00
                RETURN

            isr:
; decrement 16-bit counter by one on interrupt
; subtract 1 from LSB of 16-bit counter
                SUB count16_lsb, 01
; only subtract one from MSB if borrow
;   generated by LSB
                SUBCY count16_msb, 00
                RETURNI ENABLE

; interrupt vector is always in last memory location
                ADDRESS 3FF
; jump to interrupt service routing (ISR)
                JUMP isr
```

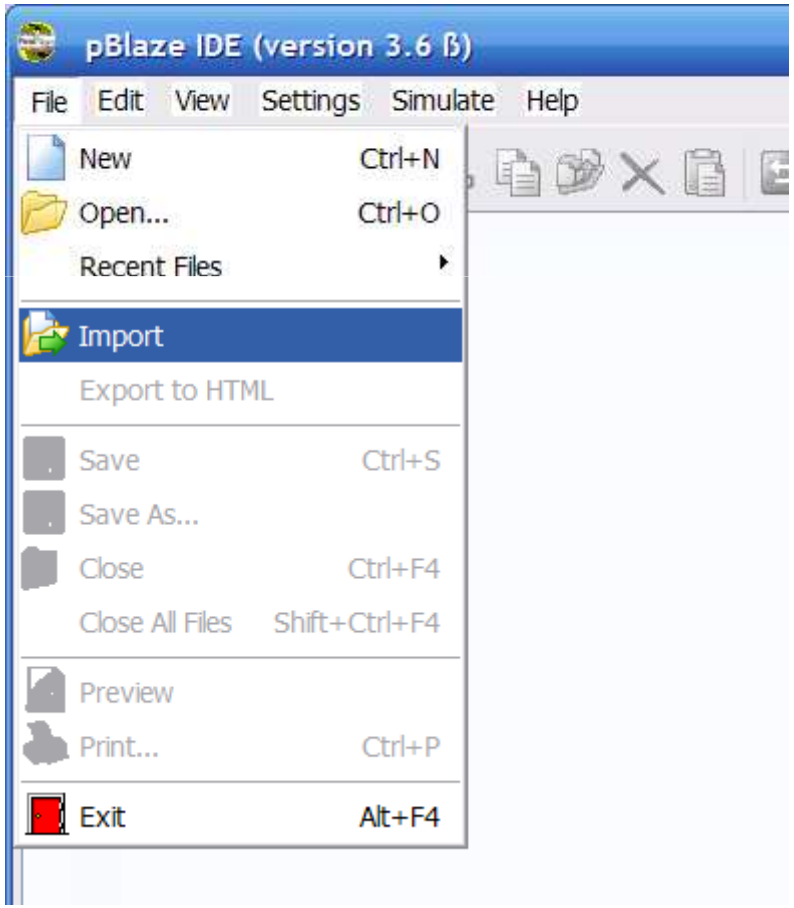
Code Imported/Converted into pBlaze IDE

```
myconstant      EQU     $A5

count16_lsb     EQU     s0
count16_msb     EQU     s1

                ORG     0

main:
; initialize 16-bit counter, enable interrupts
                LOAD    count16_lsb, myconstant
                EINT

loop:
; continuously increment 16-bit counter
                CALL    increment_count
                JUMP    loop
end_main:

increment_count:
; add 1 to LSB of 16-bit counter
                ADD     count16_lsb, 1
; only add one to MSB if carry generated by LSB
                ADDC    count16_msb, 0
                RET

isr:
; decrement 16-bit counter by one on interrupt
; subtract 1 from LSB of 16-bit counter
                SUB     count16_lsb, 1
; only subtract one from MSB if borrow
; generated by LSB
                SUBC    count16_msb, 0
                RETI    ENABLE

; interrupt vector is always in last memory location
                ORG     $3FF
; jump to interrupt service routing (ISR)
                JUMP    isr
```

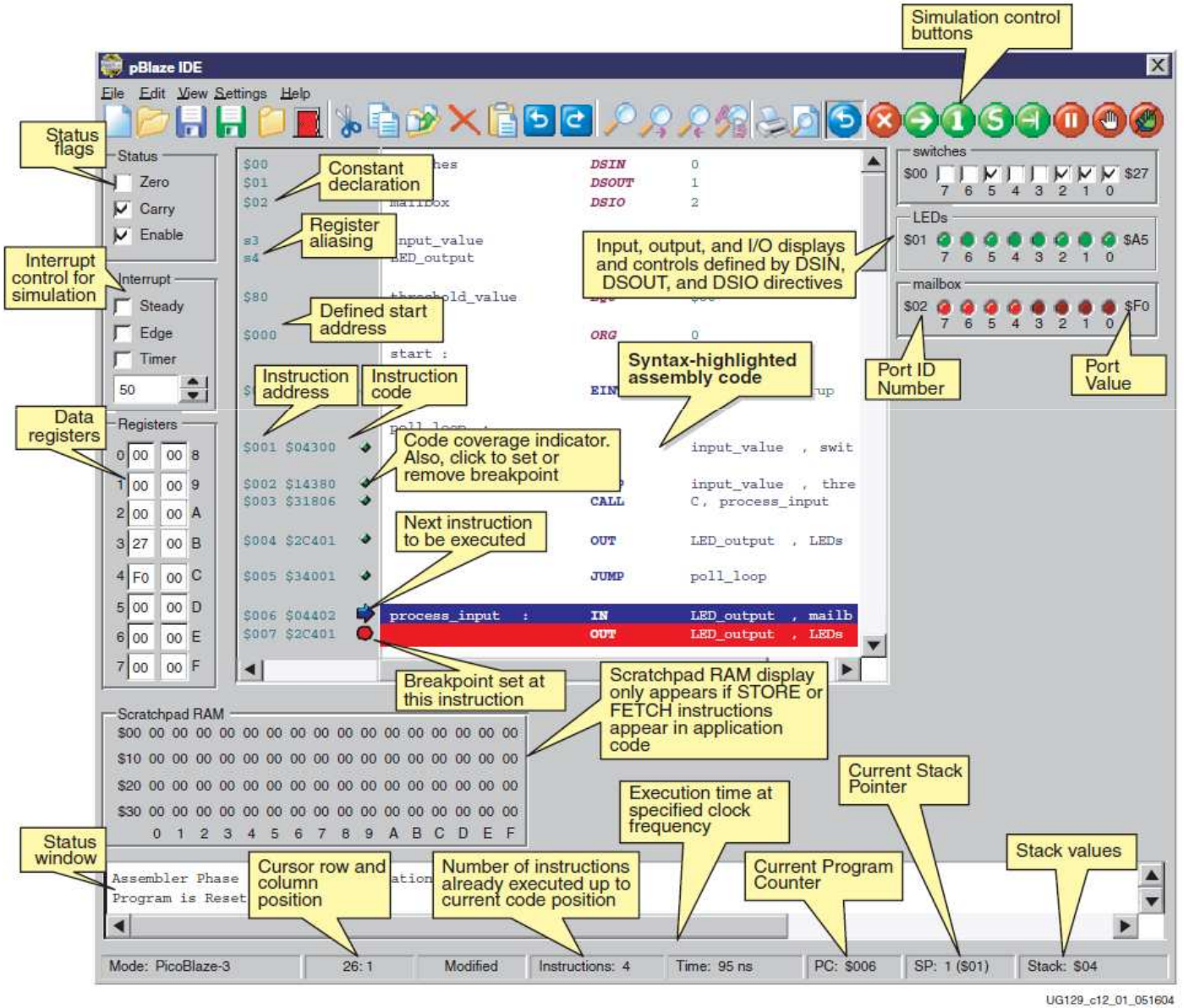
UG129_c10_04_052004

Utilidades en pBlaze IDE

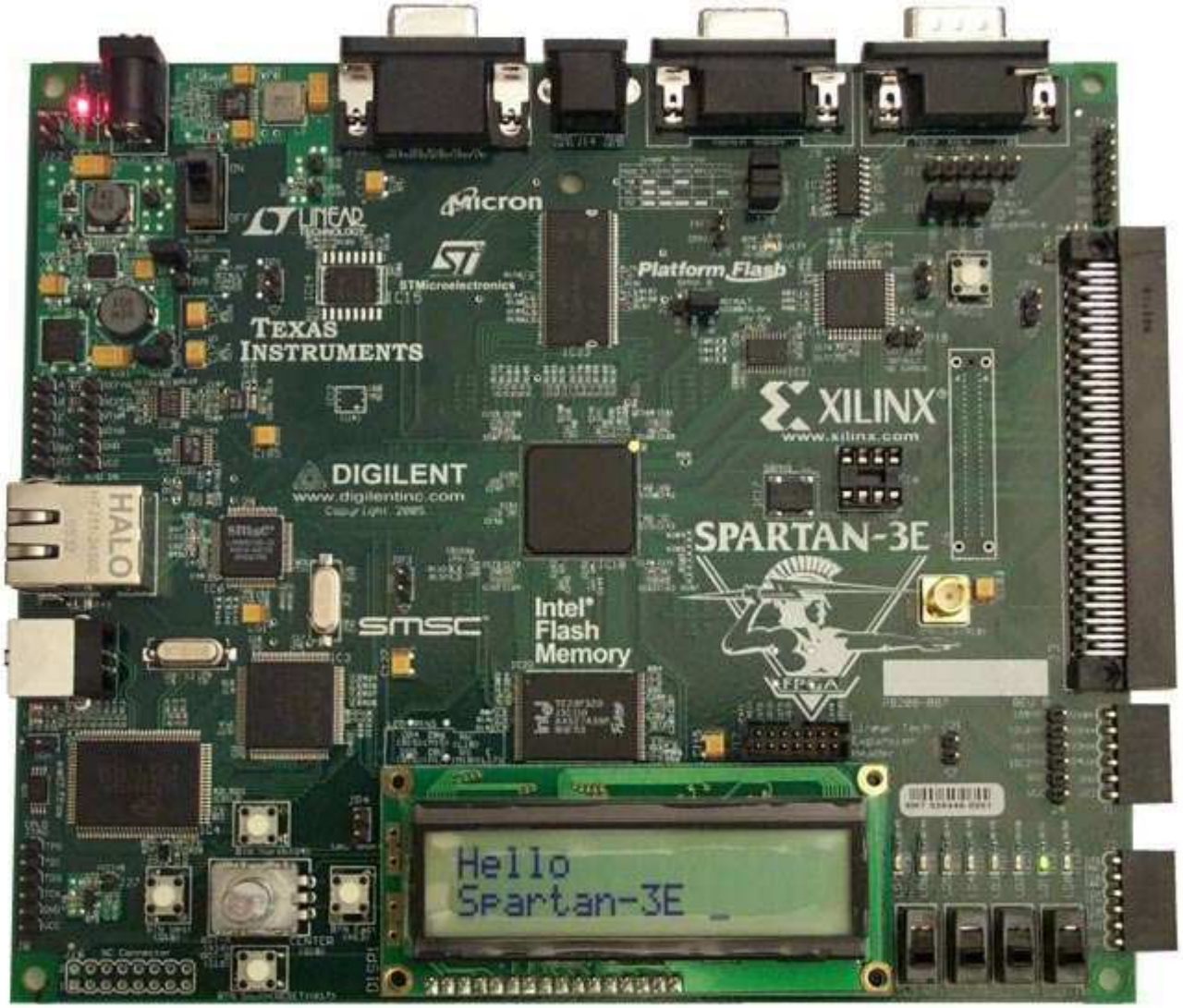




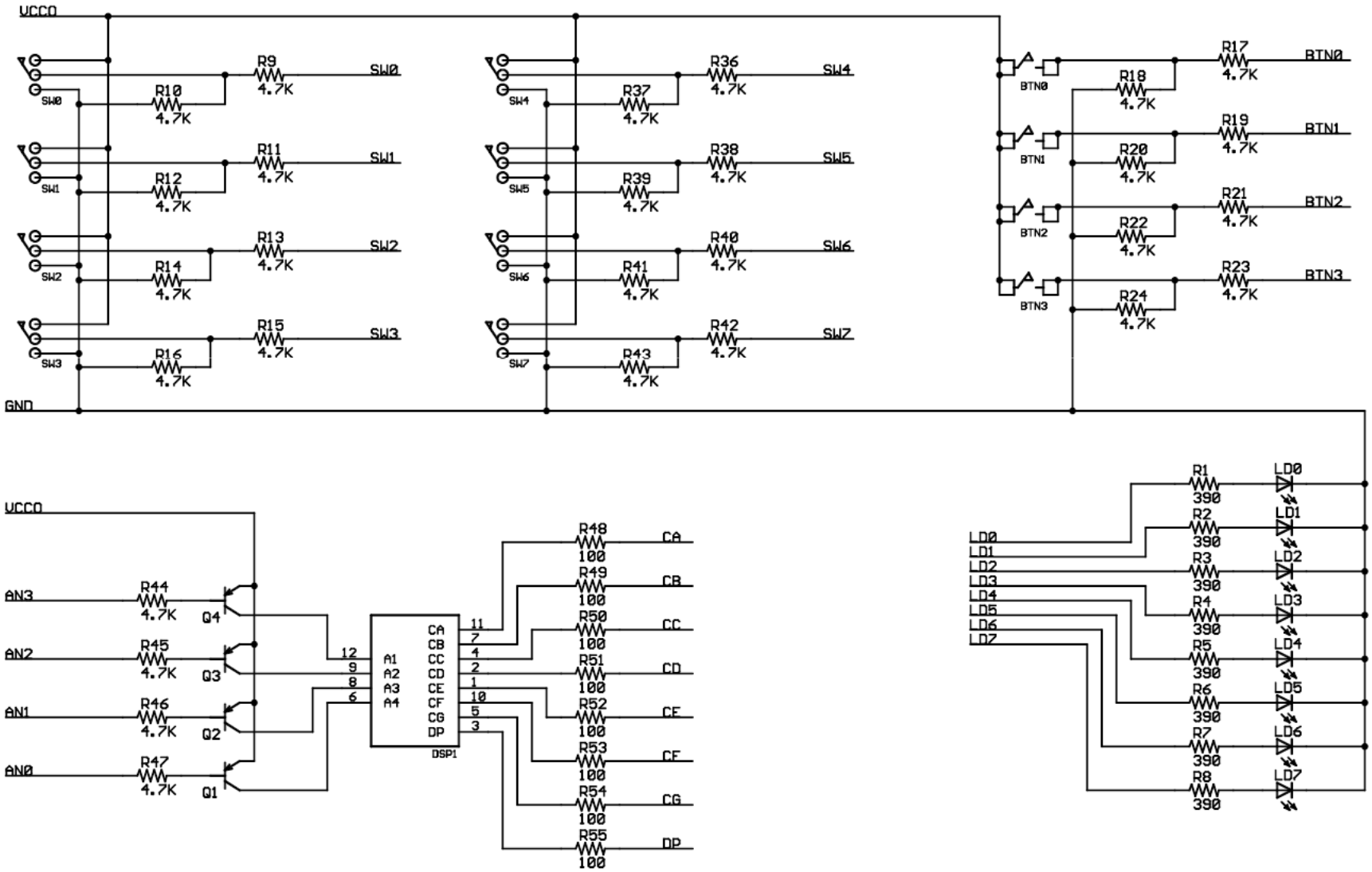
http://www.mediatronix.com/pBlazeIDE.htm

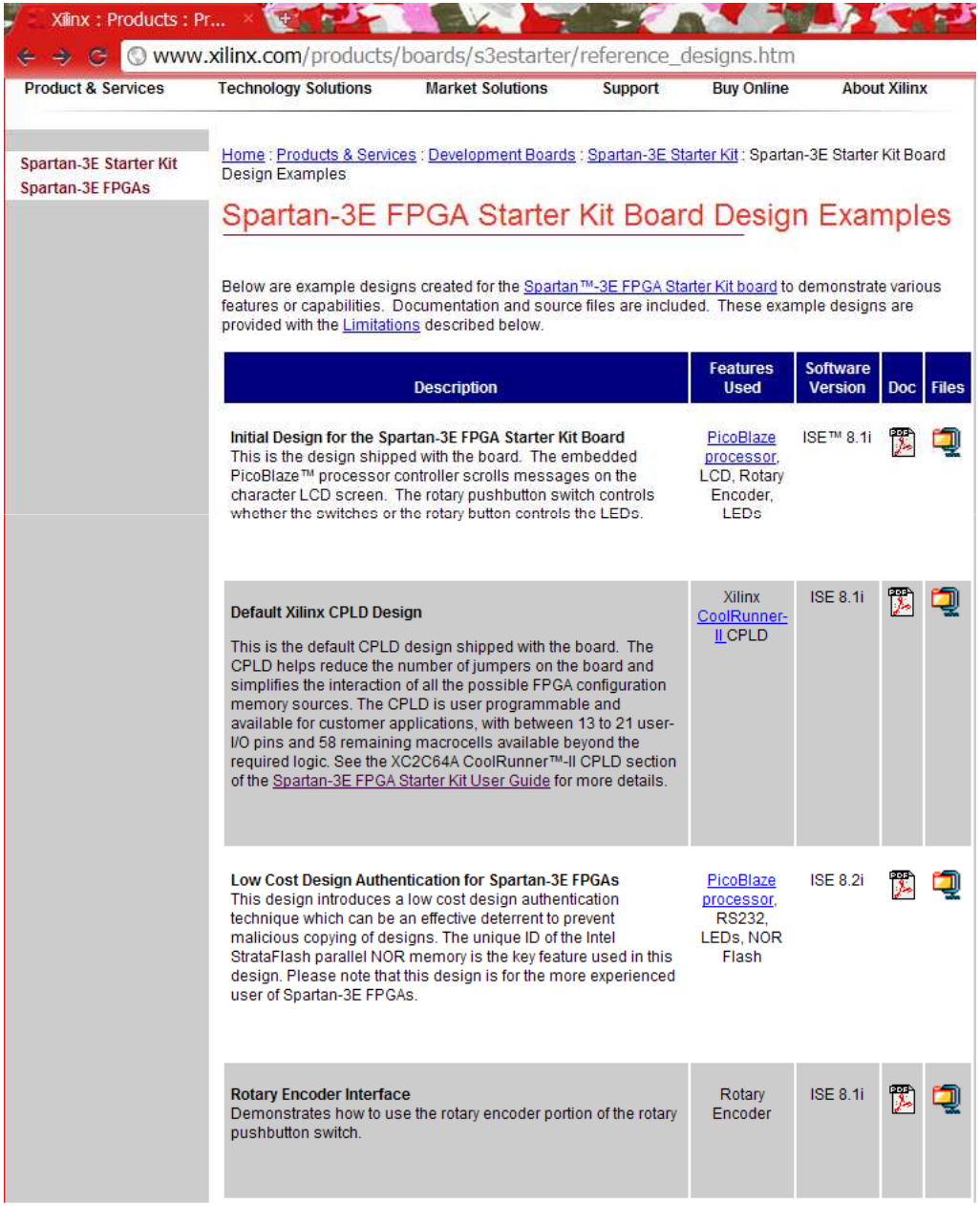


El objetivo final de todo esto . . . . . El Hardware!!



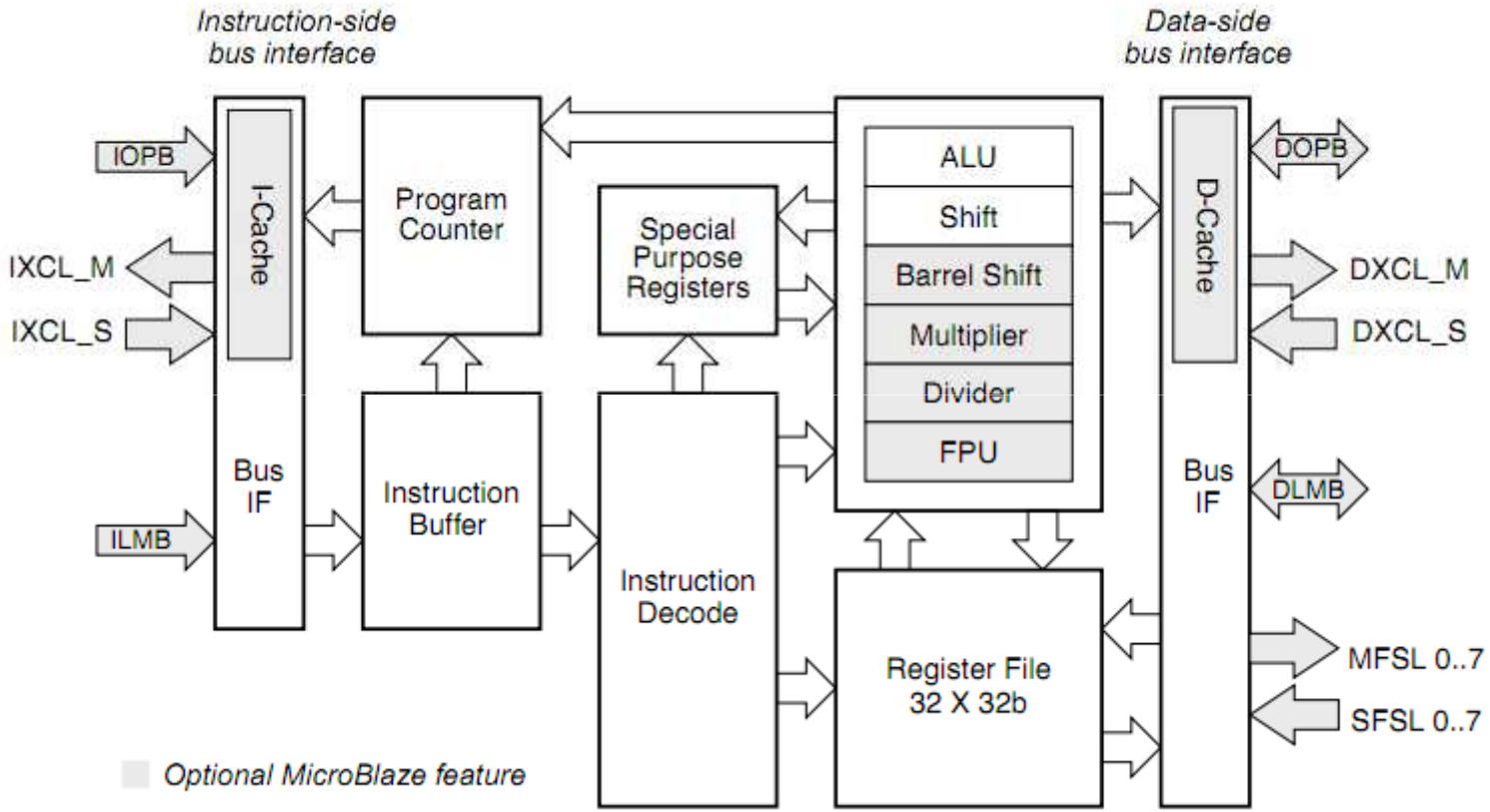
Entradas y Salidas, mapeadas




Xilinx : Products : Pr...
www.xilinx.com/products/boards/s3estarter/reference_designs.htm
Product & Services
Technology Solutions
Market Solutions
Support
Buy Online
About Xilinx
Spartan-3E Starter Kit
Spartan-3E FPGAs
Home : Products & Services : Development Boards : Spartan-3E Starter Kit : Spartan-3E Starter Kit Board Design Examples
Spartan-3E FPGA Starter Kit Board Design Examples
Below are example designs created for the Spartan™-3E FPGA Starter Kit board to demonstrate various features or capabilities. Documentation and source files are included. These example designs are provided with the Limitations described below.
Description
Features Used
Software Version
Doc
Files
Initial Design for the Spartan-3E FPGA Starter Kit Board
This is the design shipped with the board. The embedded PicoBlaze™ processor controller scrolls messages on the character LCD screen. The rotary pushbutton switch controls whether the switches or the rotary button controls the LEDs.
PicoBlaze processor, LCD, Rotary Encoder, LEDs
ISE™ 8.1i
Default Xilinx CPLD Design
This is the default CPLD design shipped with the board. The CPLD helps reduce the number of jumpers on the board and simplifies the interaction of all the possible FPGA configuration memory sources. The CPLD is user programmable and available for customer applications, with between 13 to 21 user-I/O pins and 58 remaining macrocells available beyond the required logic. See the XC2C64A CoolRunner™-II CPLD section of the Spartan-3E FPGA Starter Kit User Guide for more details.
Xilinx CoolRunner-II CPLD
ISE 8.1i
Low Cost Design Authentication for Spartan-3E FPGAs
This design introduces a low cost design authentication technique which can be an effective deterrent to prevent malicious copying of designs. The unique ID of the Intel StrataFlash parallel NOR memory is the key feature used in this design. Please note that this design is for the more experienced user of Spartan-3E FPGAs.
PicoBlaze processor, RS232, LEDs, NOR Flash
ISE 8.2i
Rotary Encoder Interface
Demonstrates how to use the rotary encoder portion of the rotary pushbutton switch.
Rotary Encoder
ISE 8.1i

MicroBlaze™

MicroBlaze


Instruction-side bus interface
Data-side bus interface
IOPB
I-Cache
IXCL_M
IXCL_S
Bus IF
ILMB
Program Counter
Instruction Buffer
Special Purpose Registers
Instruction Decode
ALU
Shift
Barrel Shift
Multiplier
Divider
FPU
Register File 32 X 32b
D-Cache
Bus IF
DOPB
DXCL_M
DXCL_S
DLMB
MFSL 0..7
SFSL 0..7
Optional MicroBlaze feature