Introducción al Picoblaze

Por que un micro empotrado y no una FSM maquina de estados?

Micropocessor (a “Programmable FSM”)

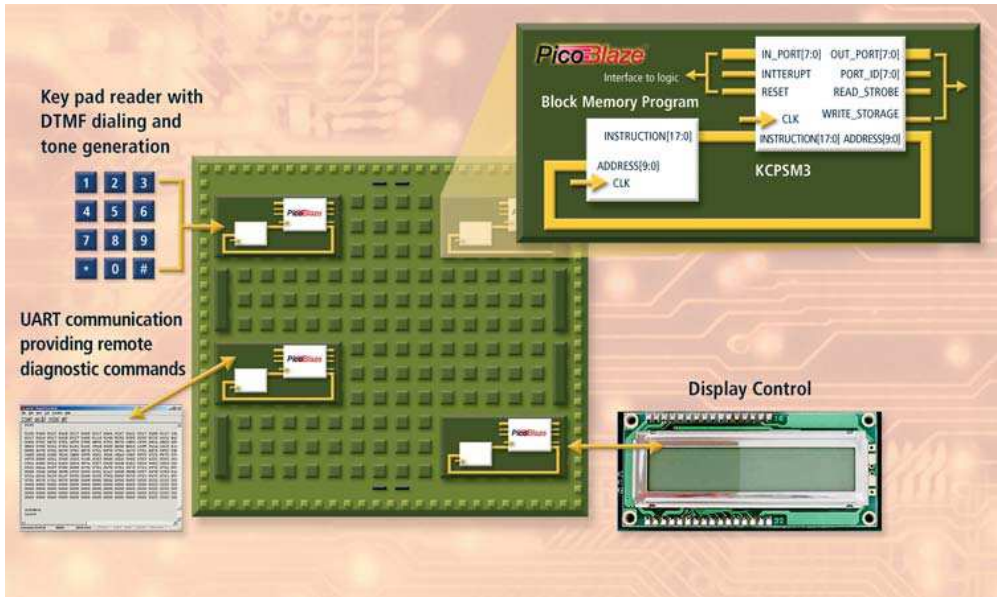
- Sequential execution
- Slower
- Better for complex flow diagram
- Better for Large FSM
- Easier to make changes in algorithm

“Regular” FSM

- Parallel execution
- Faster
- Better for simple flow diagram (but can be used for everything)
- More difficult to make changes in algorithm

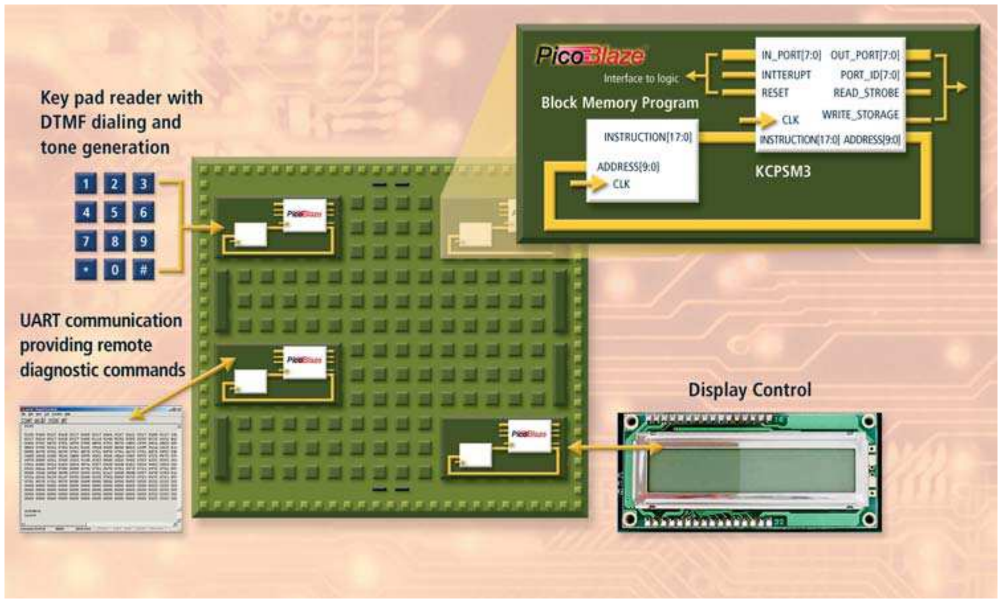
Introducción. Múltiples aplicaciones

- There are literally dozens of 8-bit microcontroller architectures and instruction sets. Modern FPGAs can efficiently implement practically any 8-bit microcontroller, and available FPGA soft cores support popular instruction sets such as the PIC, 8051, AVR, 6502, 8080, and Z80 microcontrollers.



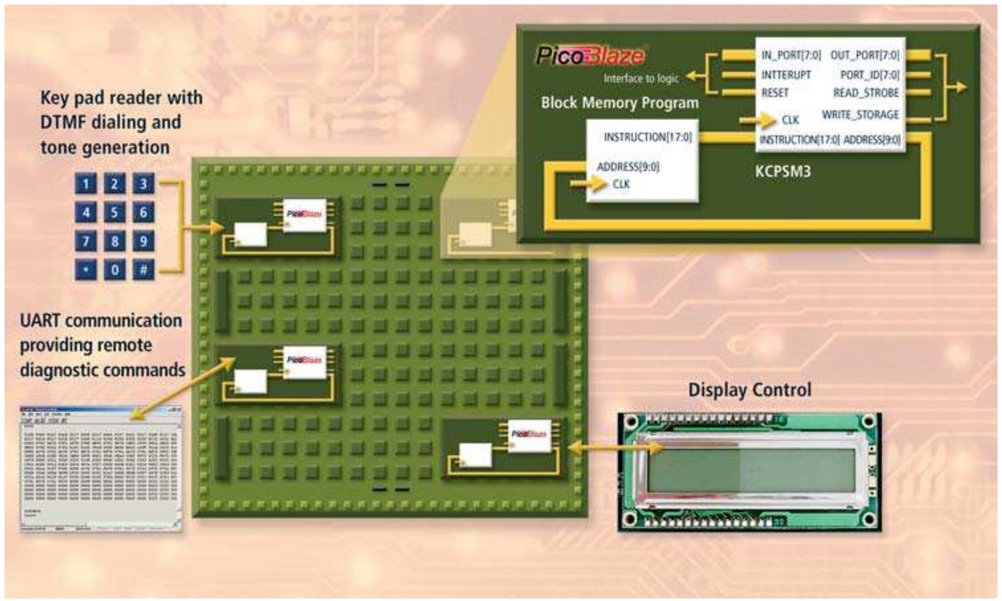
Introducción. Múltiples aplicaciones

- The Xilinx PicoBlaze microcontroller is specifically designed and optimized for the Virtex and Spartan series of FPGAs.



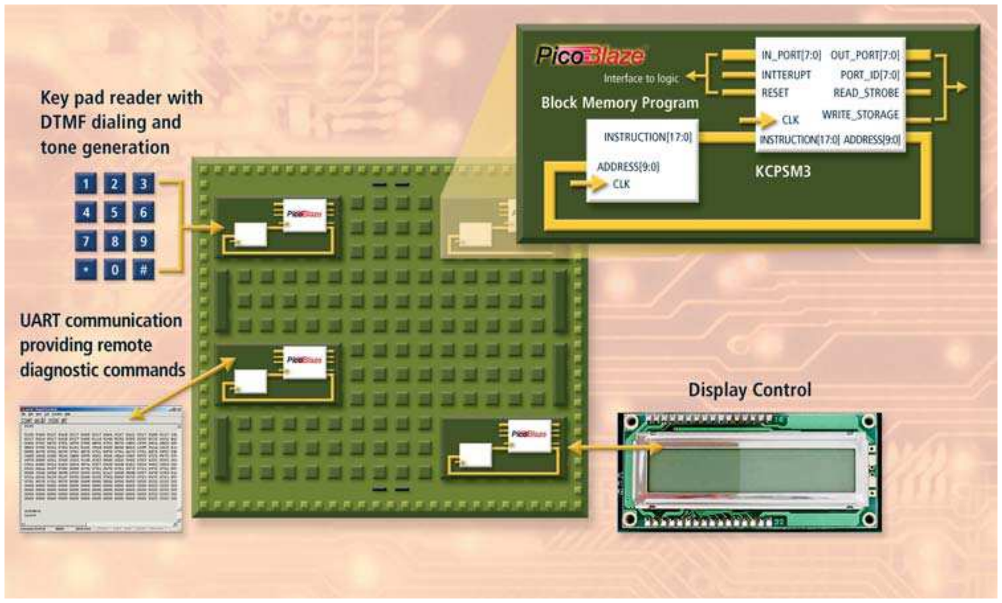
Introducción. Múltiples aplicaciones

- The PicoBlaze solution consumes considerably less resources than comparable 8-bit microcontroller architectures.



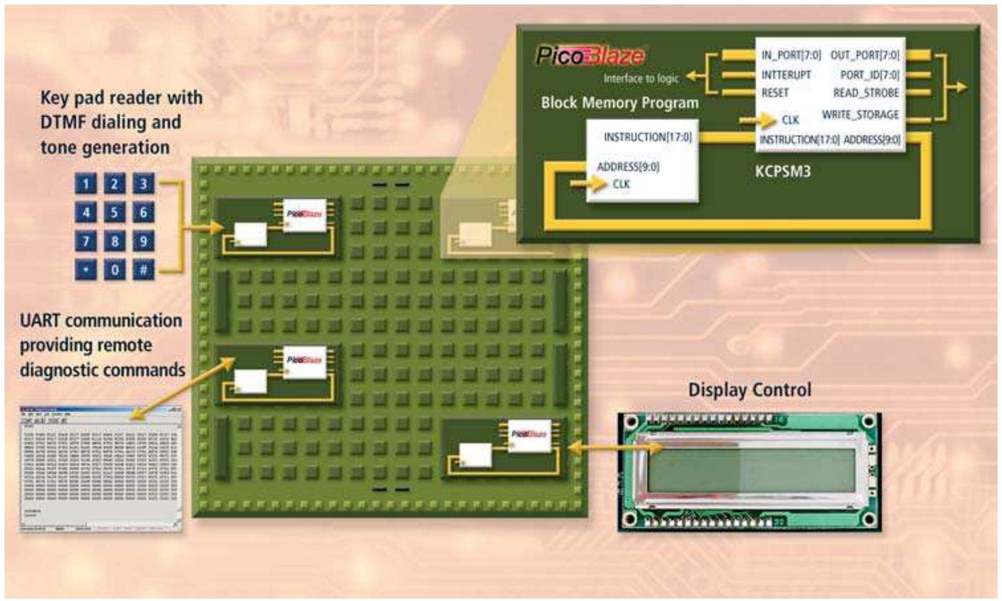
Introducción. Múltiples aplicaciones

- It is provided as a free, source-level VHDL file with royalty-free re-use within Xilinx FPGAs.



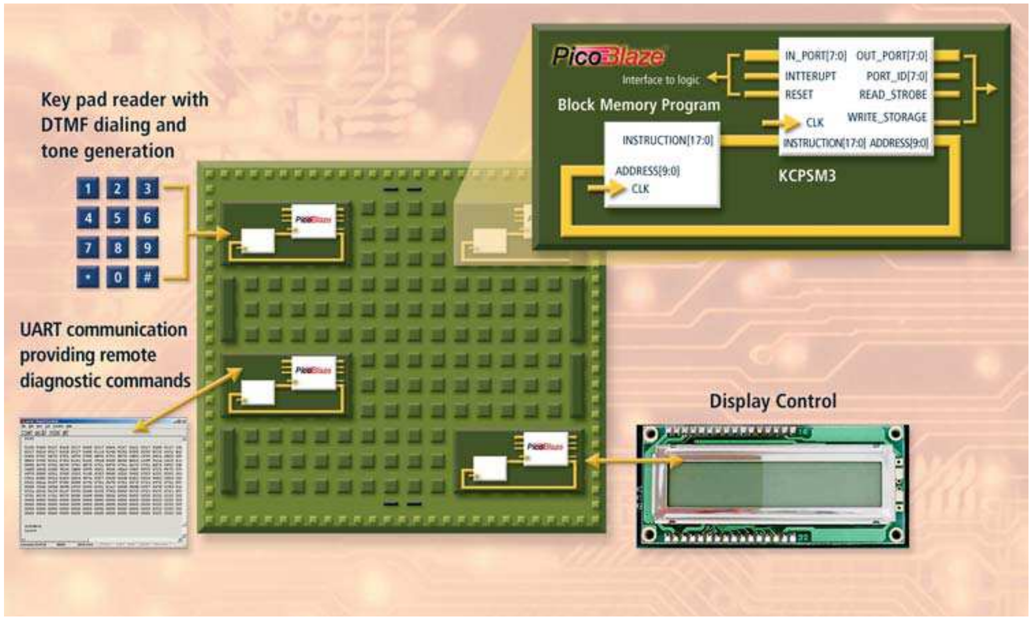
La solución para procesamiento sencillo

- PicoBlaze is a compact, capable, and cost-effective fully embedded 8-bit RISC microcontroller core optimized for the Spartan™-3, Virtex™-II, Virtex-II Pro™ and Virtex-4 FPGAs and CoolRunner™-II CPLDs.



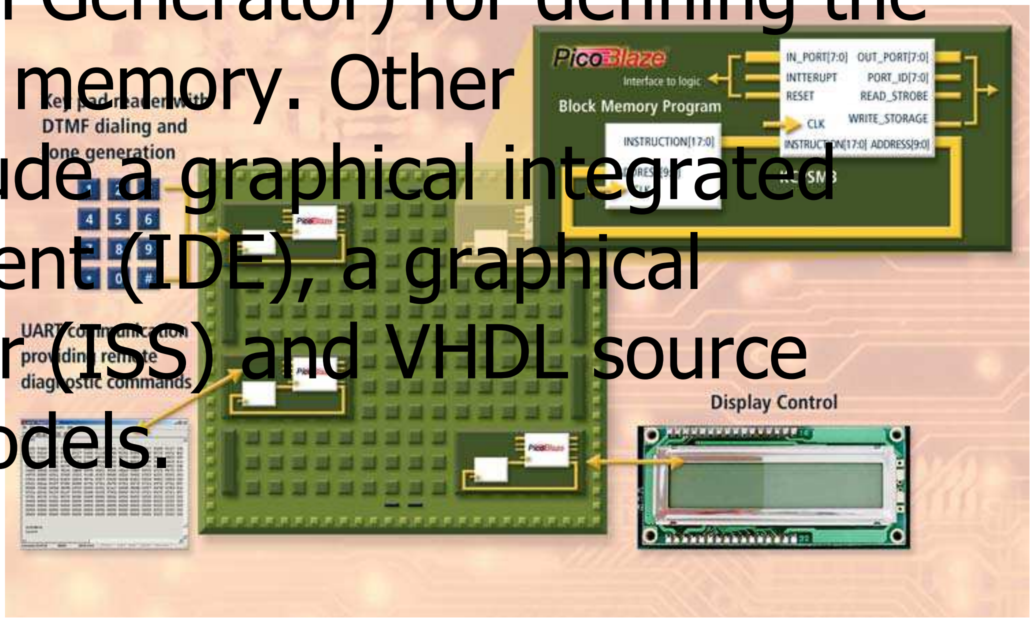
La solución para procesamiento sencillo: te DA

- **Free PicoBlaze Macro —** The PicoBlaze microcontroller is delivered as synthesizable VHDL source code. As a result, the core is future-proof and can be migrated to future FPGA and CPLD architectures.



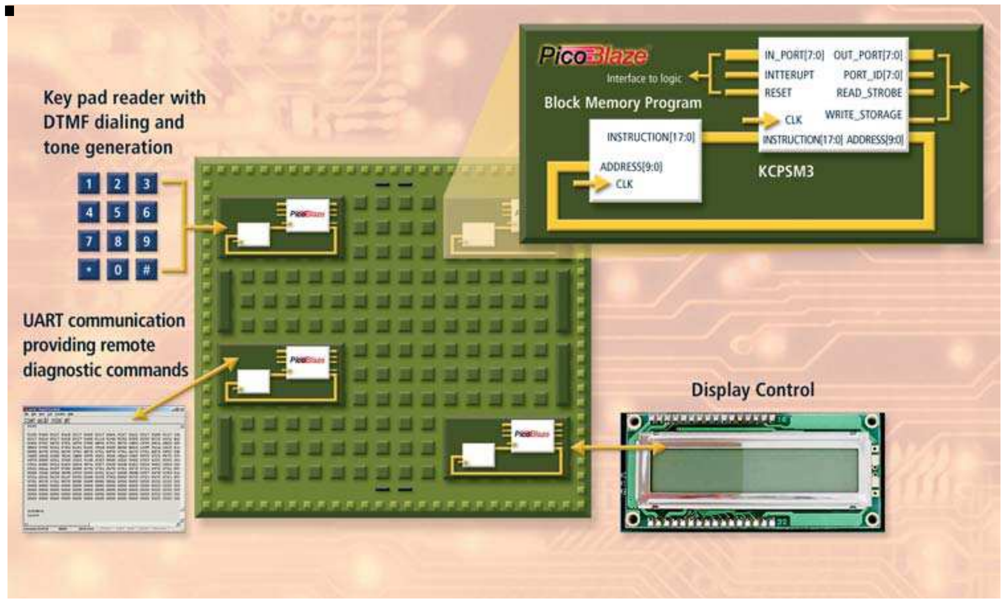
La solución para procesamiento sencillo

- **Ensamblador Facil de usar —** The PicoBlaze assembler is provided as a simple DOS executable. The assembler will compile your program in less than 3 seconds and generate VHDL, Verilog and an M-file (for Xilinx System Generator) for defining the program within a block memory. Other development tools include a graphical integrated development environment (IDE), a graphical instruction set simulator (ISS) and VHDL source code and simulation models.



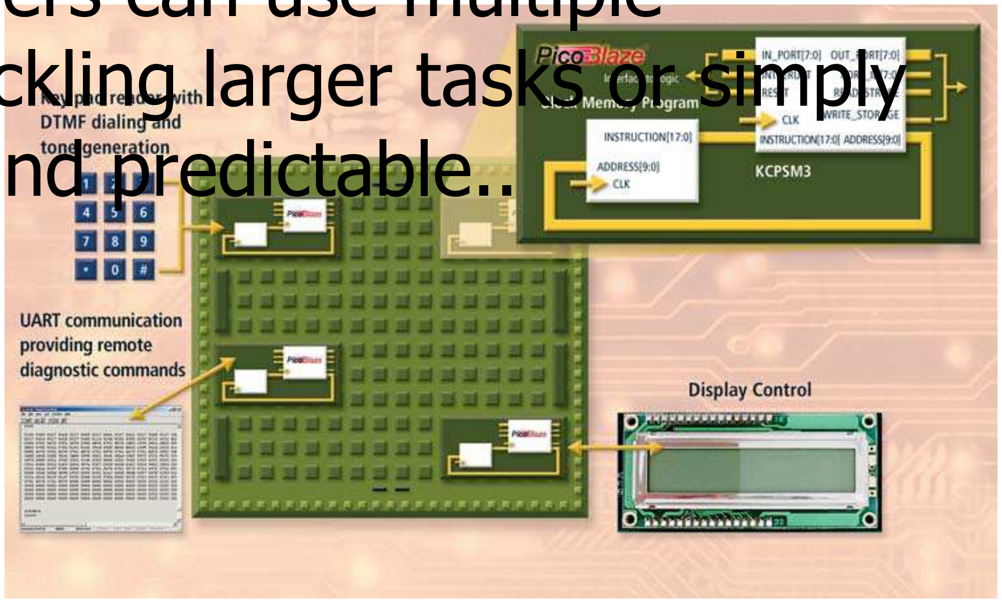
La solución para procesamiento sencillo

- **Alto rendimiento —** PicoBlaze delivers 44 to 100 million instructions per second (MIPS) depending on the target FPGA family and speed grade – many times faster than commercially available microcontroller devices..



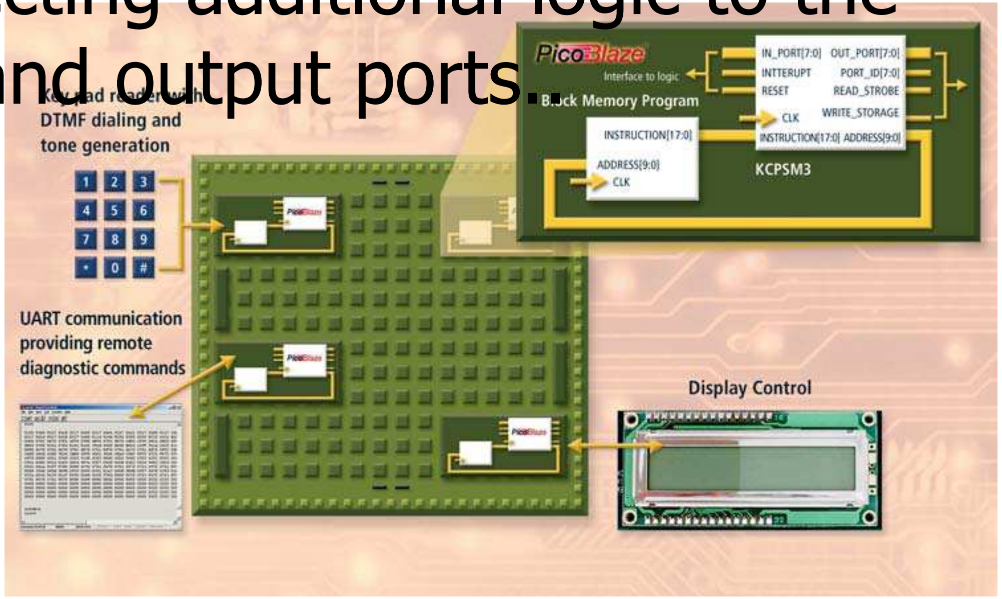
La solución para procesamiento sencillo

- **Tamaño Lógico Minimo —** PicoBlaze occupies 192 logic cells, which represents just 5% of a Spartan-3 XC3S200 device. Because the core only consumes a small fraction of the FPGA and CPLD resources, many engineers can use multiple PicoBlaze devices for tackling larger tasks or simply keeping tasks isolated and predictable..

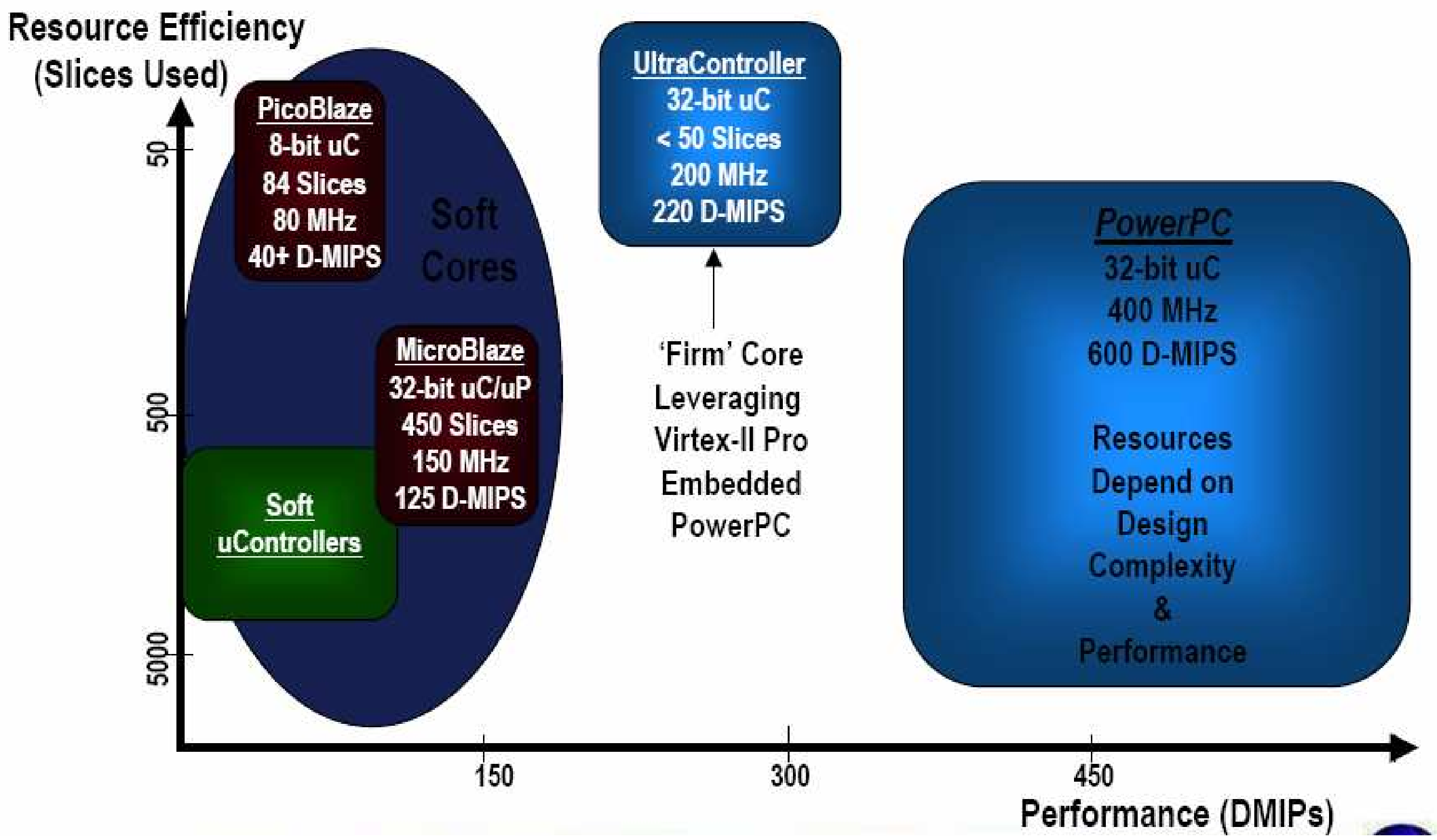


La solución para procesamiento sencillo

- **Empotrable al 100%—** The PicoBlaze microcontroller core is totally embedded within the target FPGA or CPLD and requires no external resources. Its basic functionality is easily extended and enhanced by connecting additional logic to the microcontroller's input and output ports..



Xilinx Processor Core Solutions



Características

- Es un 8-bit fully embedded microcontroller
- Tiene 16 8-bit general purpose registers,
- Tiene 8-bit ALU with the carry and zero flags
- 64-byte data memory, *scratch RAM*
- 10-bit instruction address, which supports a program up to 1024 instructions
- 256 input ports and 256 output ports,
- Automatic 31-location CALL/RETURN stack

Características (Cont.)

- Predictable performance, always two clock cycles per instruction, up to 200 MHz or 100 MIPS in a Virtex-4™ FPGA and 88 MHz or 44 MIPS in a Spartan-3 FPGA
- It also has facility for reset and interrupt
- Instruction size is 16-bits
- It also has facility for reset and interrupt
- Fast interrupt response; worst-case 5 clock cycles

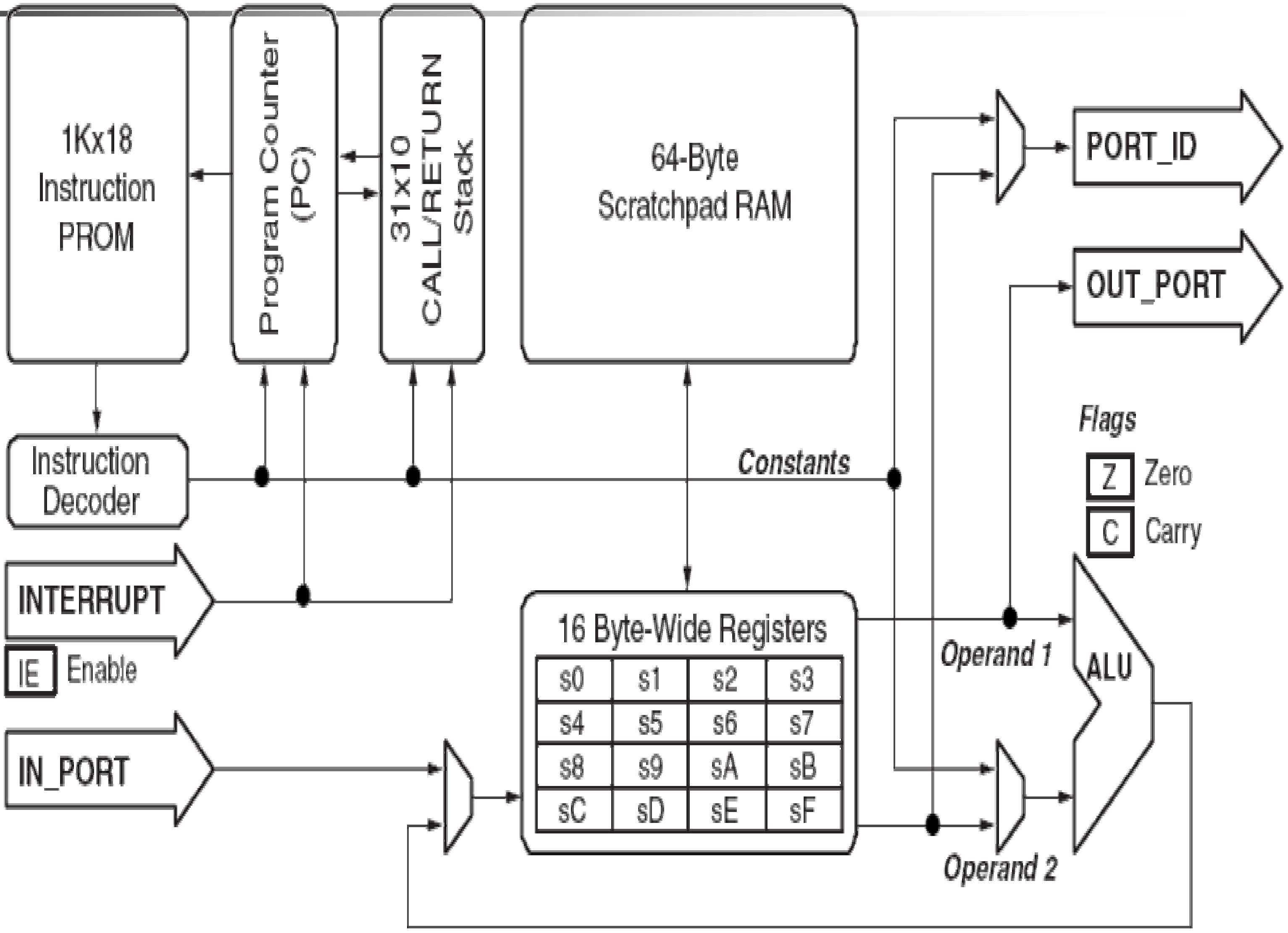
Constant(k) Coded machine

- KCPSM3 es en muchas formas una maquina basada en constantes
- Constant values are specified for use in the following aspects of a program:
  - Constant data value for use in an ALU operation
  - Constant port address to access a specific piece of information or control logic external to the PicoBlaze module
  - Constant address values for controlling the execution sequence of the program
- Constant Cycles
  - All instructions under all conditions execute over two clock cycles.
- Constant Program Length
  - The program length is 256 instructions

Estructura

3 parts:

- **ALU**
- **Reg file**
- **Control unit**



Estructura. Detalles

- Registers
  - 16 8-bit registers labelled from s0 to sF
- ALU
  - Takes two operands of 8-bits and returns the result to the first register
  - Add, Subtract with and without a carry are supported
  - LOAD, AND, OR and XOR
- Flow Control
  - Zero and Carry flags are set by the ALU operations
  - CALL/RETURN used for subroutine facilities

Estructura. Detalles (cont.)

- Reset
  - Processor goes back to initial state
  - Starts executing from address 00
  - Everything except the registers is reinitialized
- Input/Output
  - During an input the value on the indicated port is transferred to one of the registers
  - During an output the value from one of the registers is written on an output port
- Interrupt
  - A single interrupt is allowed

Porque uso un Microcontrolador dentro de una FPGA?

	PicoBlaze Microcontroller	**FPGA Logic**
Fortalezas	▪Fácil para programar aplicaciones de control y maquinas de estados ▪Los requerimientos de recursos se mantienen constante dentro aún cuando incrementa la complejidad ▪Re-uses recursos lógicos, excelente para funciones de baja rendimiento	▪Significativamente mucho mas rendimiento ▪Excelente para operaciones en paralelo ▪Secuential vs. parallelo La implementación secuencial frente a paralela sacrifica rendimiento optimo o coste ▪Respuesta rápida a entradas simultaneas
Debilidades	▪Ejecución secuencial ▪El rendimiento se degrada con el incremento de la complejidad ▪Los requerimientos de memoria incrementa con la complejidad ▪Respuesta lenta a estímulos simultáneos	▪Aplicaciones de control y maquinas de estado son mas difícil de programar ▪Recursos lógicos crece con el incremento de la complejidad

Usando el Microcontrolador PicoBlaze en una FPGA

- **Flujo de diseño en VHDL**
  - **El componente (macro o módulo) es suministrado en VHDL (**KCPSM3**.**vhd) y su declaración es la siguiente

```
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;);
```

Usando el Microcontrolador PicoBlaze en una FPGA

- **Flujo de diseño en VHDL**
  - **El componente (macro) es suministrado en VHDL (**KCPSM3.vhd) y su instanciación en un diseño es la siguiente

```
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 );
```

Usando el Microcontrolador PicoBlaze en una FPGA

- **Flujo de diseño en VHDL**
  - **La memoria de programa (ROM).** The PicoBlaze program ROM is used within a VHDL design flow.
  - The PicoBlaze assembler generates a VHDL file in which a block RAM and its initial contents are defined. This VHDL file can be used for both logic synthesis and simulation of the processor

Usando el Microcontrolador PicoBlaze en una FPGA

- **Flujo de diseño en VHDL**
  - **Declaración del componente (ROM).**

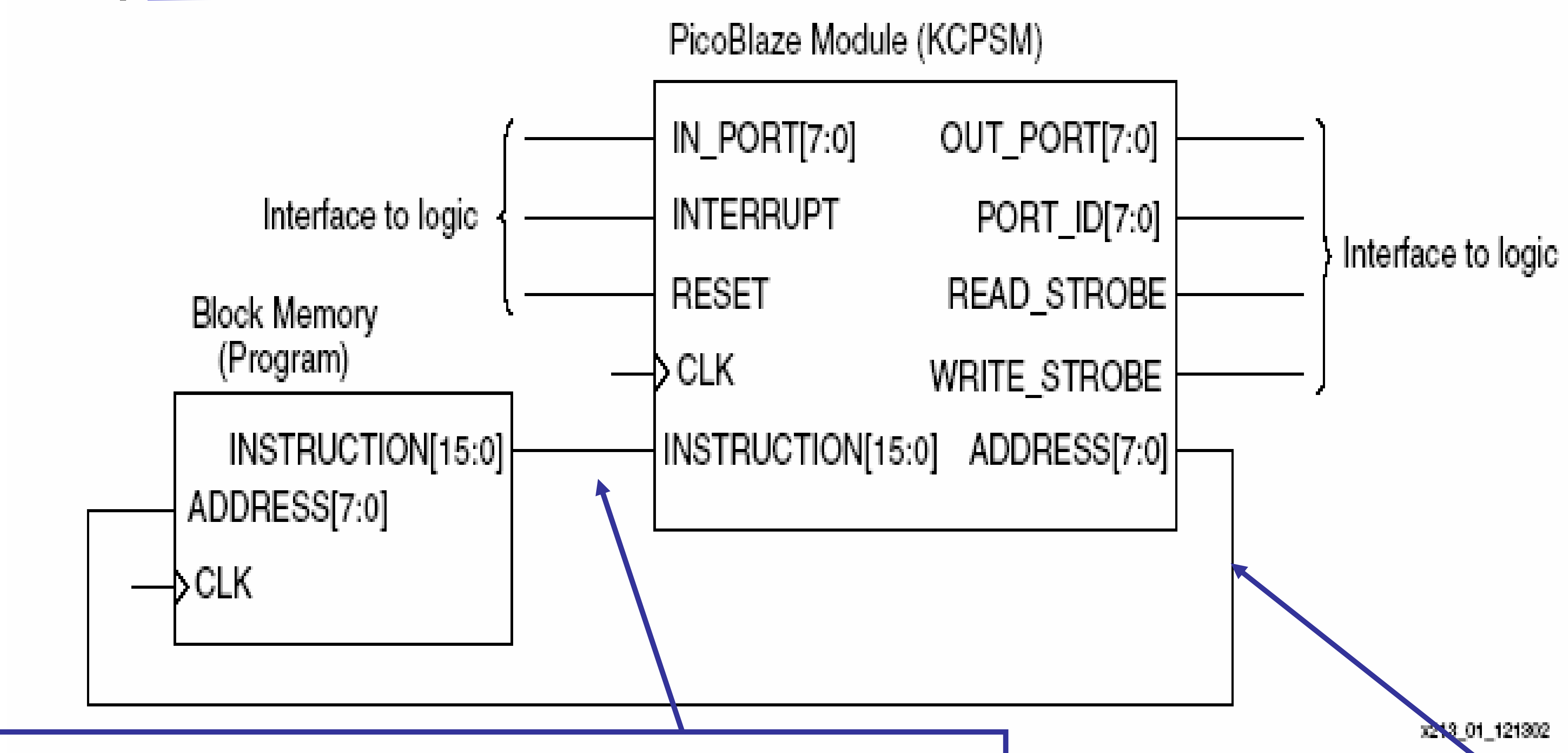
```
component prog_rom
port ( address : in   std_logic_vector( 9 downto 0);
       instruction : out std_logic_vector(17 downto 0);
       clk :             in std_logic);
end component;
```

  - El nombre del componente deriva de nombre del fichero con el programa en ensamblador. Por ejemplo si tu programa se llama *reloj.psm* la memoria (componente) en vhdl se llama *reloj.vhd*. Este proceso resulta de aplicar el compilador suministrado

Usando el Micro. PicoBlaze en una FPGA. Poniento todo Junto

- **Para acelerar el desarrollo se suministra un fichero** (embedded_kcpsm3.vhd)
  - **En este codigo VHDL (**plantilla**) se conectan el micro con la memoria de programa.**

Diagrama de bloques del Micro + ROM. Poniendo todo junto



signal instruction : std_logic_vector(17 downto 0);

signal address : std_logic_vector(9 downto 0);

PicoBlaze Herramientas de desarrollo

- Existen 3 herramientas de desarrollo
  - **Xilinx KCPSM3**
  - **Mediatronix pBlazIDE**
  - **Xilinx System Generator**

PicoBlaze Herramientas de desarrollo

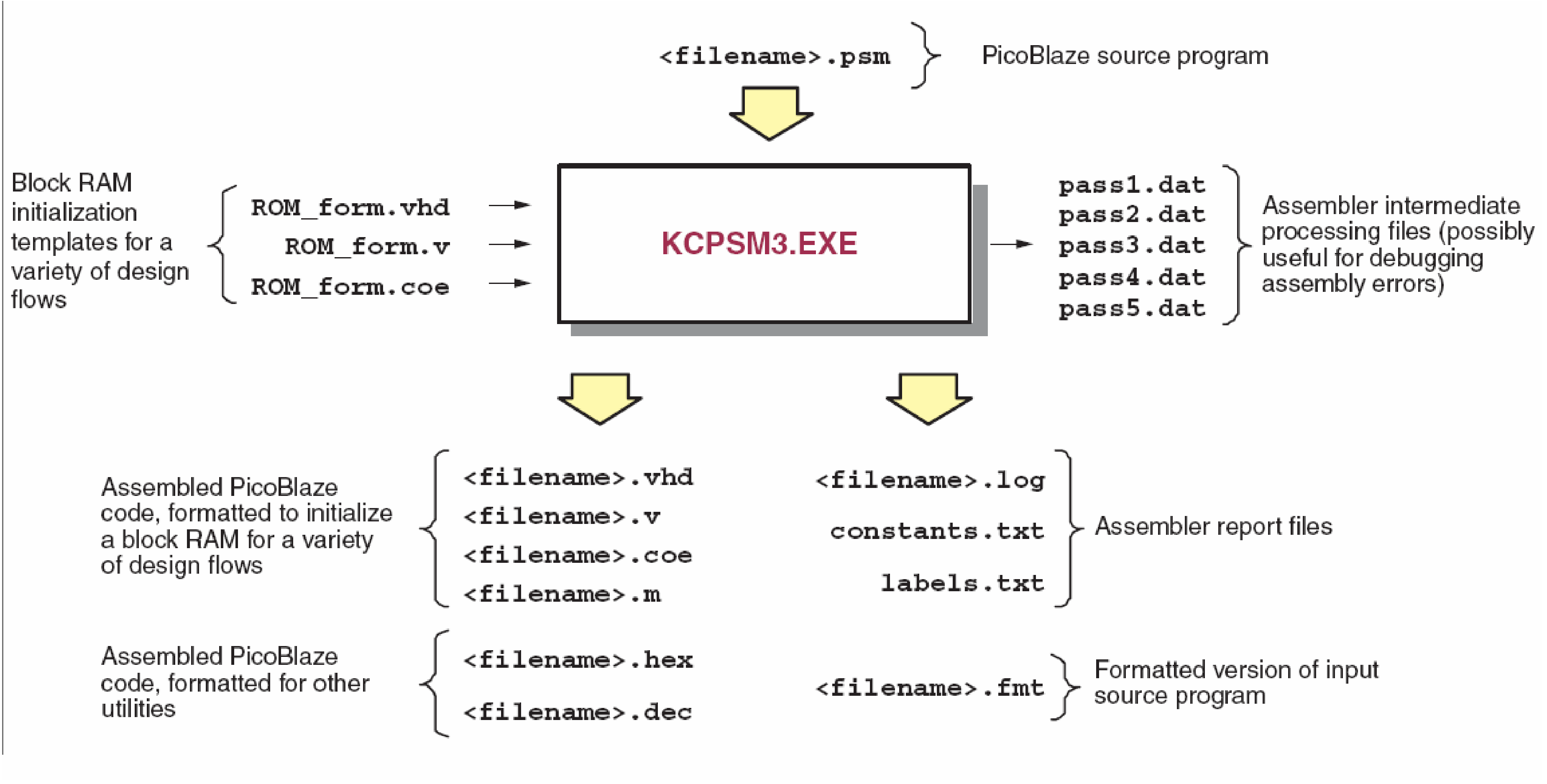
	Xilinx KCPSM3	**Mediatronix pBlazIDE**	**Xilinx System Generator**
Plataforma	Windows	Windows 98, 2000, NT, ME, XP	Windows 2000 y XP
Ensamblador	Línea de comando en Dos Wiindow	Grafico	Linea de comando dentro de System Generator
Sintaxis instruciones	KCPSM3	PBlazIDE	KCPSM3
Instruction Set Simulator	VHDL	Grafico/Interactivo	Grafico/Interactivo
Simulator Breakpoints	N/A	Si	Si
Register Viewer	N/A	Si	Si
Memory Viewer	N/A	Si	Si

PicoBlaze Herramientas de desarrollo

- **Xilinx KCPSM3**
  - The KCPSM3 Assembler is provided as a simple DOS executable file together with three template files.
  - **Importante** Crear un directorio de trabajo. Copy all the files *KCPSM3.EXE, ROM_form.vhd, ROM_form.v, and ROM_form.coe* dentro del directorio de trabajo.
  - Los programas pueden ser escritos con el ***wordpad***. Salvar con extensión **psm**
  - **Abre un cmd DOS.**
    - kcpsm3 <filename>[.psm]
    - El ensamblador da salida de errores por pantalla (peor muy rápida) usar un fichero de salida para depurar errores:
      - kcpsm3 <filename>[.psm] > screen_dump.txt

PicoBlaze Herramientas de desarrollo

- Compilación. Ficheros de entrada y salida.



<filename>.psm
PicoBlaze source program
Block RAM initialization templates for a variety of design flows
ROM_form.vhd
ROM_form.v
ROM_form.coe
KCPSM3.EXE
pass1.dat
pass2.dat
pass3.dat
pass4.dat
pass5.dat
Assembler intermediate processing files (possibly useful for debugging assembly errors)
Assembled PicoBlaze code, formatted to initialize a block RAM for a variety of design flows
<filename>.vhd
<filename>.v
<filename>.coe
<filename>.m
<filename>.log
constants.txt
labels.txt
Assembler report files
Assembled PicoBlaze code, formatted for other utilities
<filename>.hex
<filename>.dec
<filename>.fmt
Formatted version of input source program

Ejemplo de una Plantilla de un programa (KCPSM3)

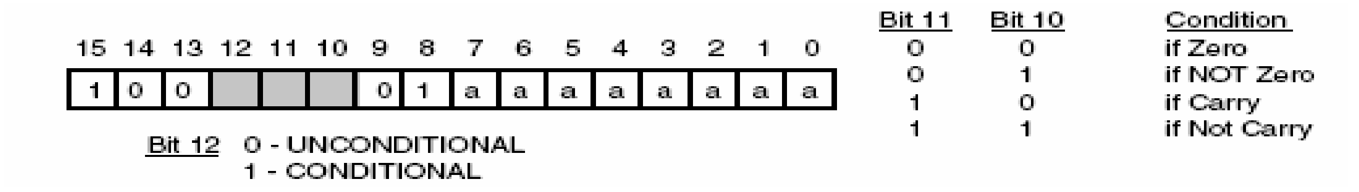
```
        NAMEREG sX, <name> ; Rename register sX with <name>
        CONSTANT <name>, 00 ; Define constant <name>, assign value

          ; ROM output file es siempre llamado
          ; <filename>.vhd
        ADDRESS 000 ; Programs always start at reset vector 0
        ENABLE INTERRUPT ; If using interrupts, be sure to enable
          ; the INTERRUPT input
BEGIN:
          ; <<< your code here >>>
        JUMP BEGIN ; Embedded applications never end
ISR: ; An Interrupt Service Routine (ISR) is
          ; required if using interrupts
          ; Interrupts are automatically disabled
          ; when an interrupt is recognized
          ; Never re-enable interrupts during the ISR
        RETURNI ENABLE ; Return from interrupt service routine
          ; Use RETURNI DISABLE to leave interrupts
          ; disabled
        ADDRESS 3FF ; Interrupt vector is located at highest
          ; instruction address
        JUMP ISR ; Jump to interrupt service routine, IS
```

Ejemplo de una Plantilla de un programa (pBlazIDE)

```
        <name> EQU sX ; Rename register sX with <name>
        <name> EQU $00 ; Define constant <name>, assign value
        ; name ROM output file generated by pBlazIDE assembler
        VHDL "template.vhd", "target.vhd", "entity_name"
        <name> DSIN <port_id> ; Create input port, assign port address
        <name> DSOUT <port_id>; Create output port, assign port address
        <name> DSIO <port_id> ; Create readable output port,
        ; assign port address
        ORG 0; Programs always start at reset vector 0
        EINT ; If using interrupts, be sure to enable the INTERRUPT input

BEGIN:
        ; <<< your code here >>>
JUMP BEGIN ; Embedded applications never end
ISR: ; An Interrupt Service Routine (ISR) is
        ; required if using interrupts
        ; Interrupts are automatically disabled
        ; when an interrupt is recognized
        ; Never re-enable interrupts during the ISR
        RETI ENABLE ; Return from interrupt service routine
        ; Use RETURNI DISABLE to leave interrupts
        ; disabled
        ORG $3FF ; Interrupt vector is located at highest
        ; instruction address
        JUMP ISR ; Jump to interrupt service routine, ISR
```

Conjunto de instrucciones

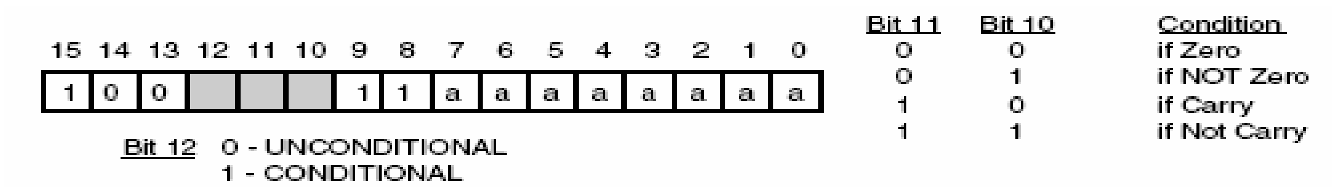
- Program Control Group (jump, call y return)
  - Jump Instruction – Both conditional and unconditional



Bit 11	Bit 10	Condition
0	0	if Zero
0	1	if NOT Zero
1	0	if Carry
1	1	if Not Carry

Conjunto de instrucciones

- Program Control Group (jump, call y return)
  - Call Instruction



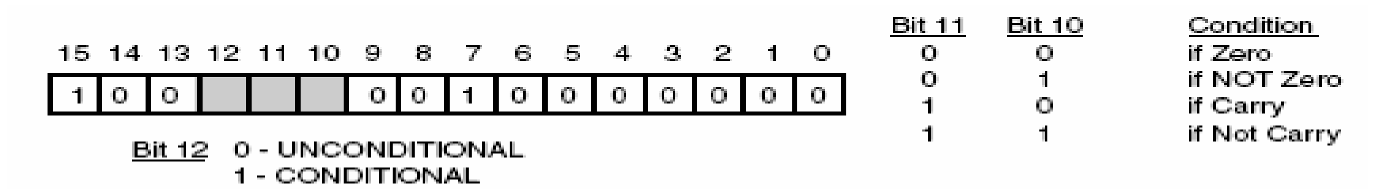
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	▨	▨	▨	1	1	a	a	a	a	a	a	a	a

Bit 12 0 - UNCONDITIONAL
1 - CONDITIONAL

Bit 11	Bit 10	Condition
0	0	if Zero
0	1	if NOT Zero
1	0	if Carry
1	1	if Not Carry

Conjunto de instrucciones

- Program Control Group (jump, call y return)
  - Return Instruction



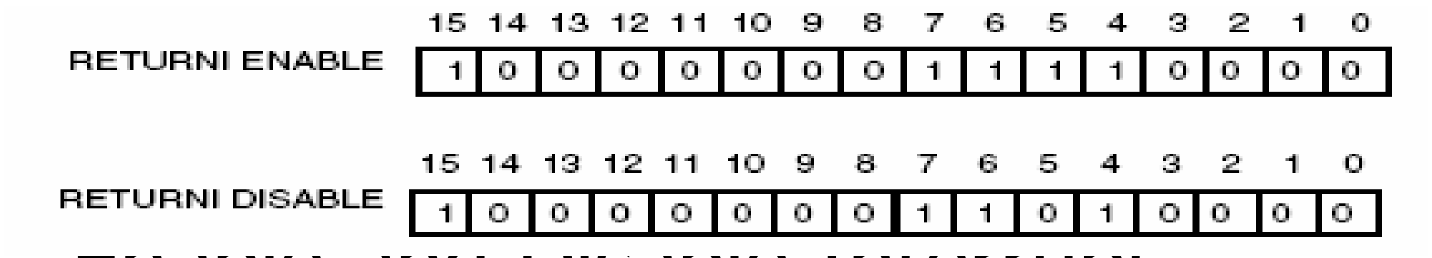
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	▨	▨	▨	0	0	1	0	0	0	0	0	0	0

Bit 12	
0	UNCONDITIONAL
1	CONDITIONAL

Bit 11	Bit 10	Condition
0	0	if Zero
0	1	if NOT Zero
1	0	if Carry
1	1	if Not Carry

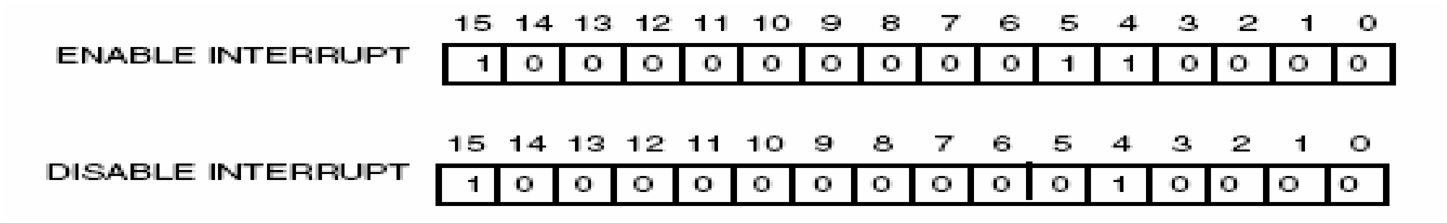
Interrupt Group

- ReturnI



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RETURNI ENABLE	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
RETURNI DISABLE	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0

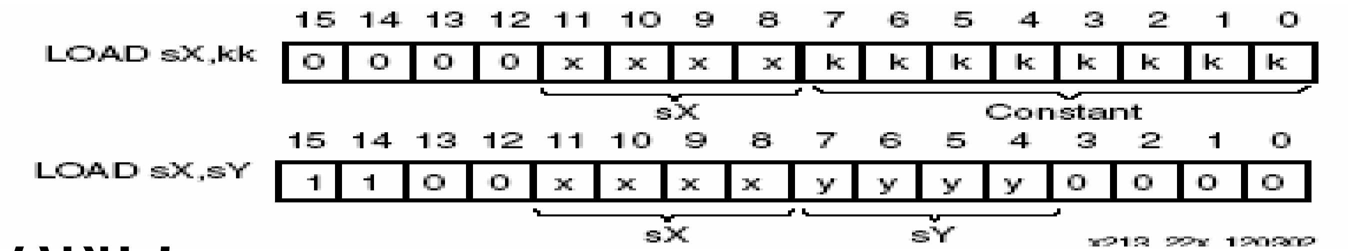
- Enable and Disable Interrupt



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENABLE INTERRUPT	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
DISABLE INTERRUPT	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Logical Group

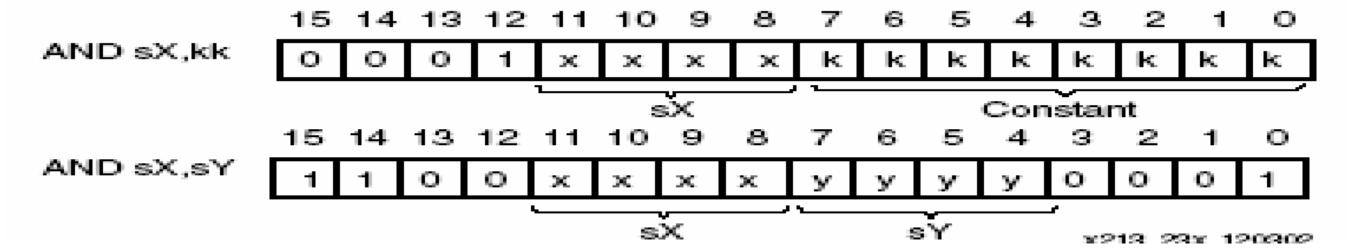
- LOAD



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD sX,kk	0	0	0	0	x	x	x	x	k	k	k	k	k	k	k	k
					sX				Constant							
LOAD sX,sY	1	1	0	0	x	x	x	x	y	y	y	y	0	0	0	0
					sX				sY							

x213_22x_120302

- AND

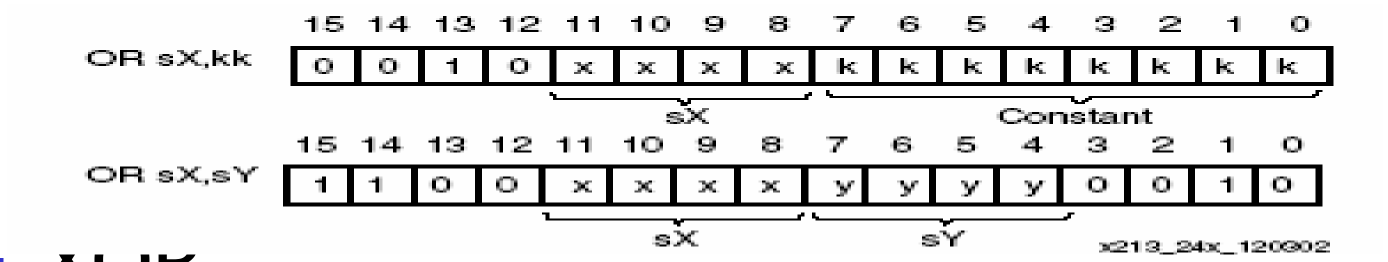


	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AND sX,kk	0	0	0	1	x	x	x	x	k	k	k	k	k	k	k	k
					sX				Constant							
AND sX,sY	1	1	0	0	x	x	x	x	y	y	y	y	0	0	0	1
					sX				sY							

x213_23x_120302

Logical Group (contd.)

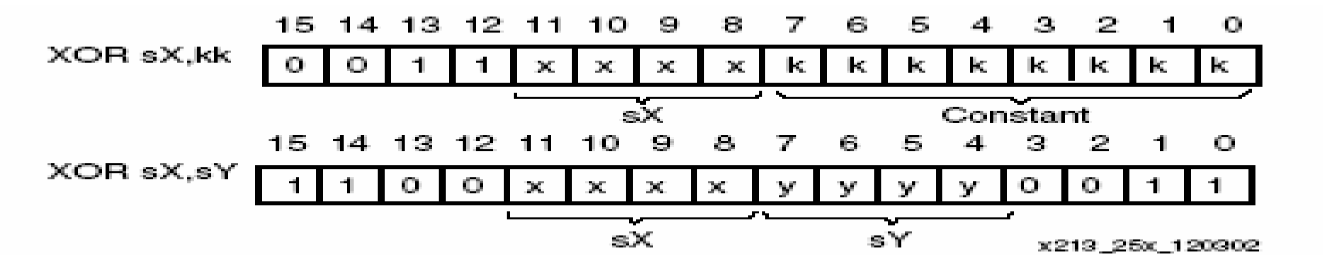
- OR



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OR sX,kk	0	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
					sX				Constant							
OR sX,sY	1	1	0	0	x	x	x	x	y	y	y	y	0	0	1	0
					sX				sY							

x213_24x_120302

- XOR

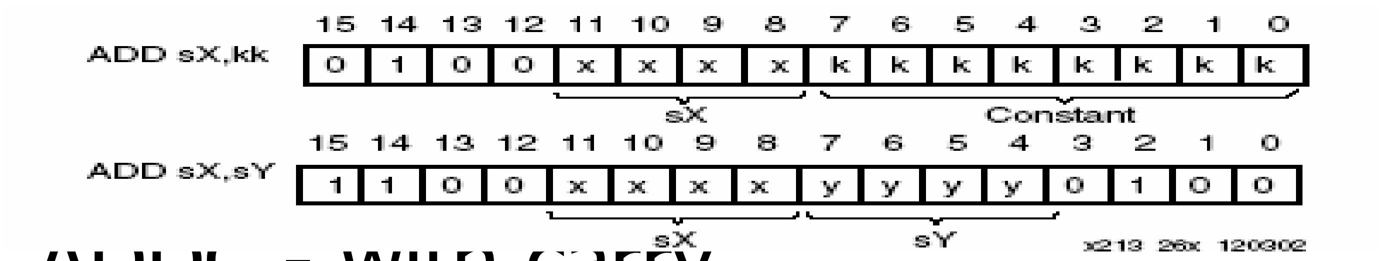


	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XOR sX,kk	0	0	1	1	x	x	x	x	k	k	k	k	k	k	k	k
					sX				Constant							
XOR sX,sY	1	1	0	0	x	x	x	x	y	y	y	y	0	0	1	1
					sX				sY							

x213_25x_120302

Grupo aritmético

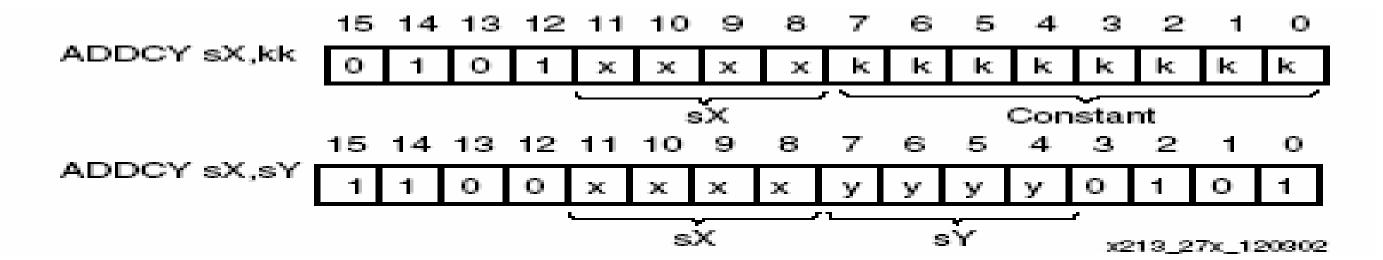
- ADD – without carry



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD sX,kk	0	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
					sX				Constant							
ADD sX,sY	1	1	0	0	x	x	x	x	y	y	y	y	0	1	0	0
					sX				sY							

x213_26x_120302

- ADDC - with carry

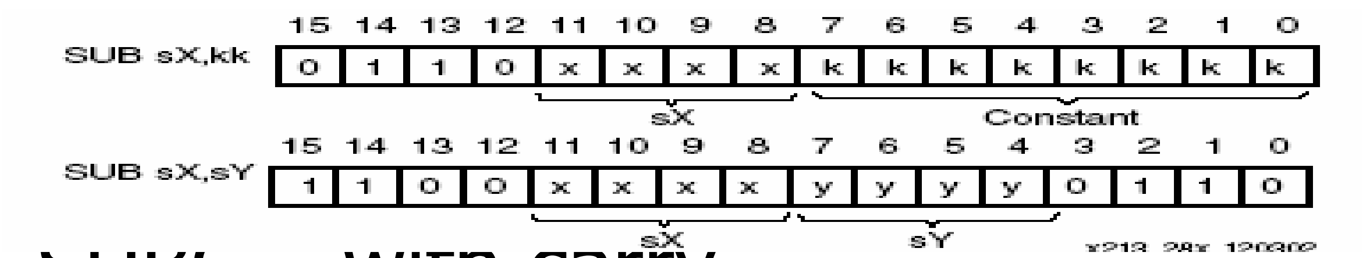


	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDCY sX,kk	0	1	0	1	x	x	x	x	k	k	k	k	k	k	k	k
					sX				Constant							
ADDCY sX,sY	1	1	0	0	x	x	x	x	y	y	y	y	0	1	0	1
					sX				sY							

x213_27x_120302

Arithmetic Group (contd.)

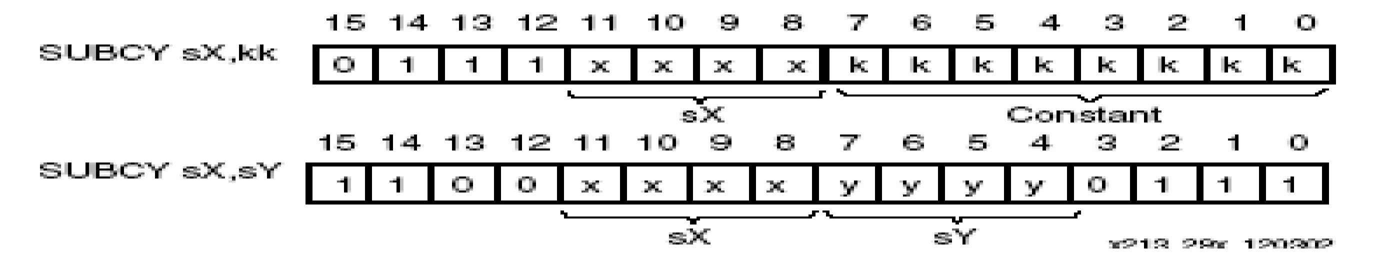
- SUB – without carry



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUB sX,kk	0	1	1	0	x	x	x	x	k	k	k	k	k	k	k	k
					sX				Constant							
SUB sX,sY	1	1	0	0	x	x	x	x	y	y	y	y	0	1	1	0
					sX				sY							

x213_28x_120302

- SUBC – with carry

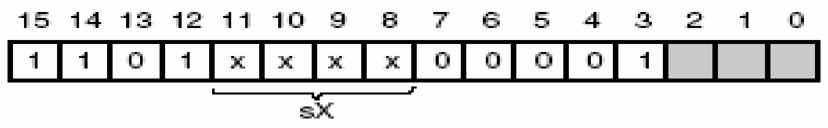


	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUBCY sX,kk	0	1	1	1	x	x	x	x	k	k	k	k	k	k	k	k
					sX				Constant							
SUBCY sX,sY	1	1	0	0	x	x	x	x	y	y	y	y	0	1	1	1
					sX				sY							

x213_28x_120302

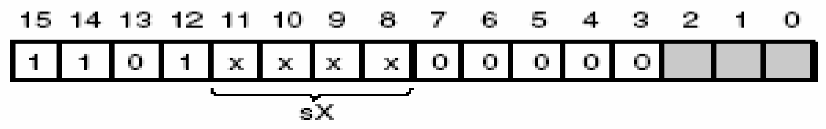
Shift and Rotate Group

- Shift right



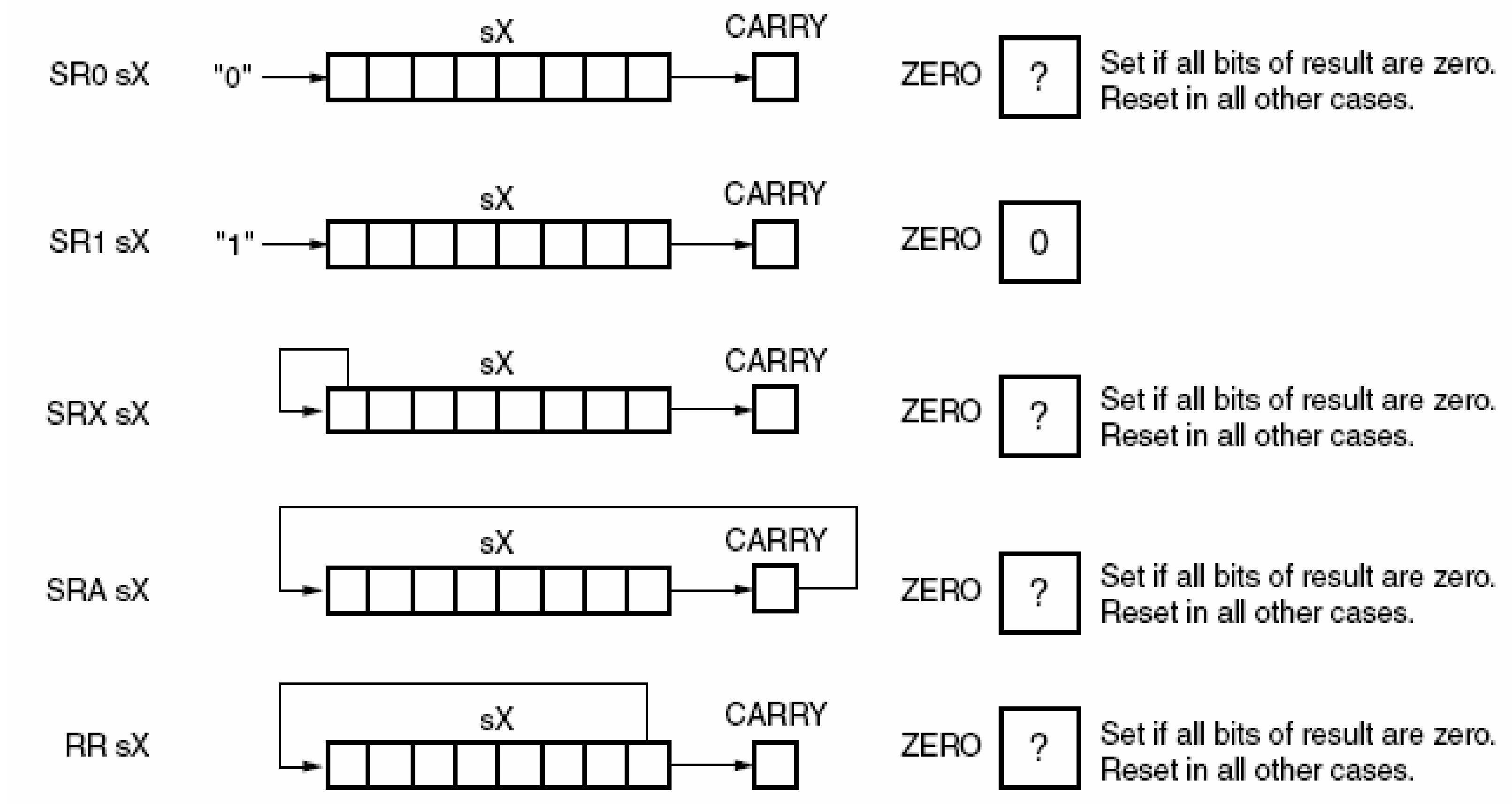
Bit 2	Bit 1	Bit0	Instruction1
1	1	0	SR0 sX
1	1	1	SR1 sX
0	1	0	SRX sX
0	0	0	SRA sX
1	0	0	RR sX

- Shift left



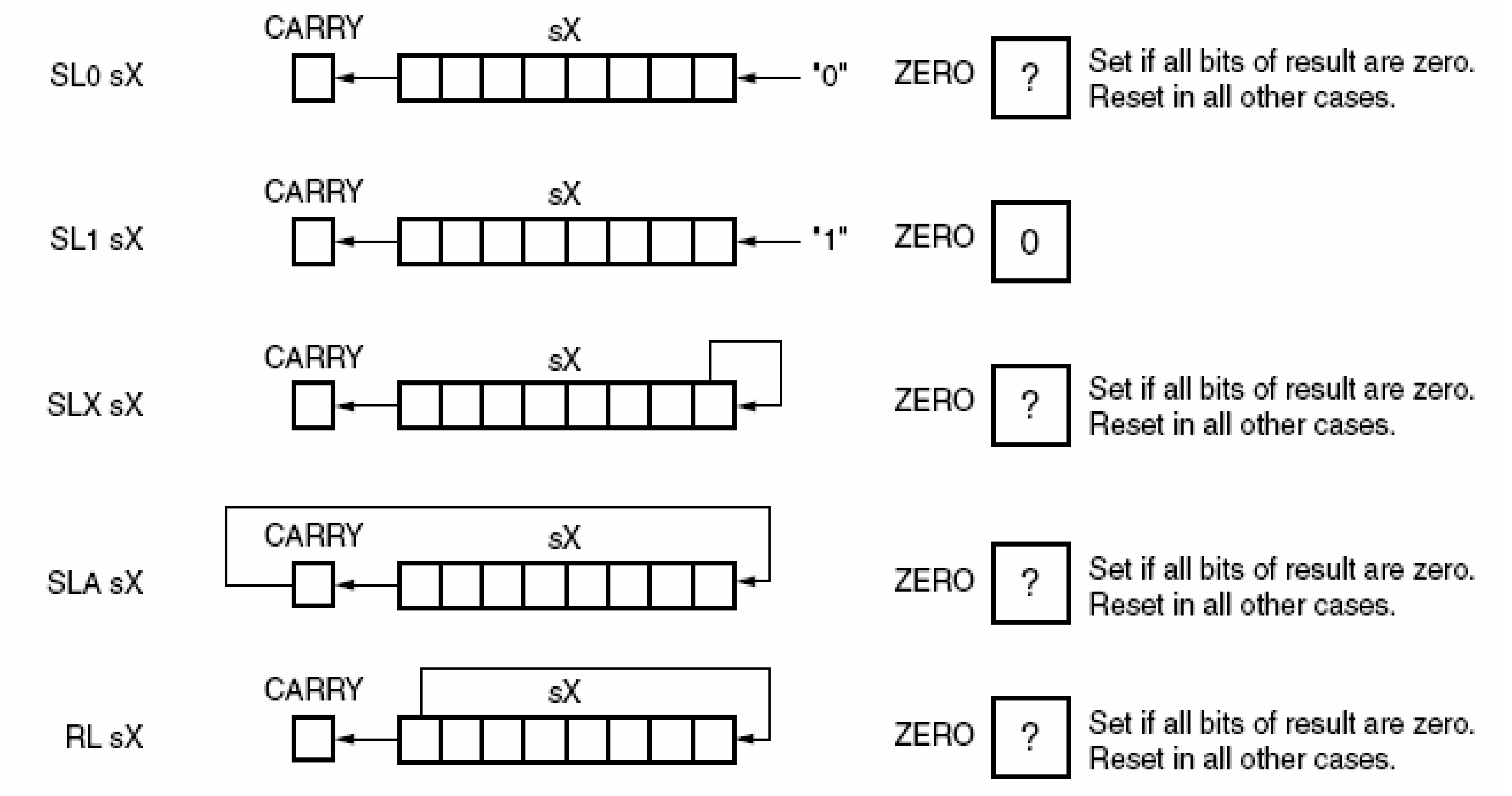
Bit 2	Bit 1	Bit0	Instruction1
1	1	0	SL0 sX
1	1	1	SL1 sX
1	0	0	SLX sX
0	0	0	SLA sX
0	1	0	RL sX

Shift right



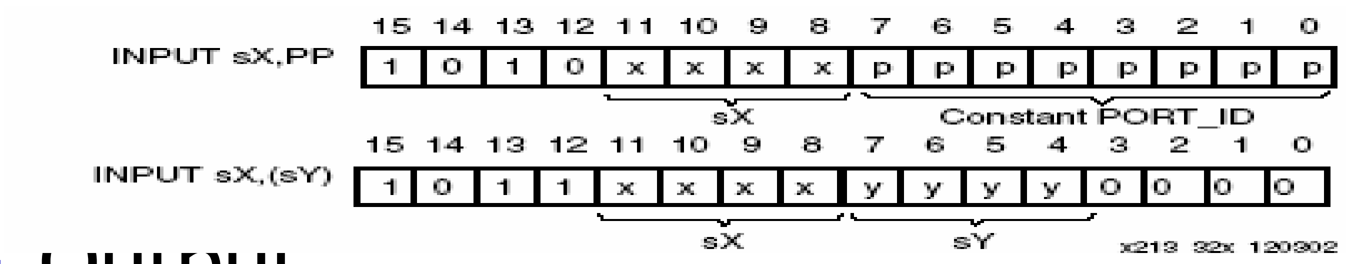
SR0 sX
"0"
sX
CARRY
ZERO
?
Set if all bits of result are zero.
Reset in all other cases.
SR1 sX
"1"
sX
CARRY
ZERO
0
SRX sX
sX
CARRY
ZERO
?
Set if all bits of result are zero.
Reset in all other cases.
SRA sX
sX
CARRY
ZERO
?
Set if all bits of result are zero.
Reset in all other cases.
RR sX
sX
CARRY
ZERO
?
Set if all bits of result are zero.
Reset in all other cases.

Shift left


SL0 sX
CARRY
sX
"0"
ZERO
?
Set if all bits of result are zero.
Reset in all other cases.
SL1 sX
CARRY
sX
"1"
ZERO
0
SLX sX
CARRY
sX
ZERO
?
Set if all bits of result are zero.
Reset in all other cases.
SLA sX
CARRY
sX
ZERO
?
Set if all bits of result are zero.
Reset in all other cases.
RL sX
CARRY
sX
ZERO
?
Set if all bits of result are zero.
Reset in all other cases.

Input and Output Group

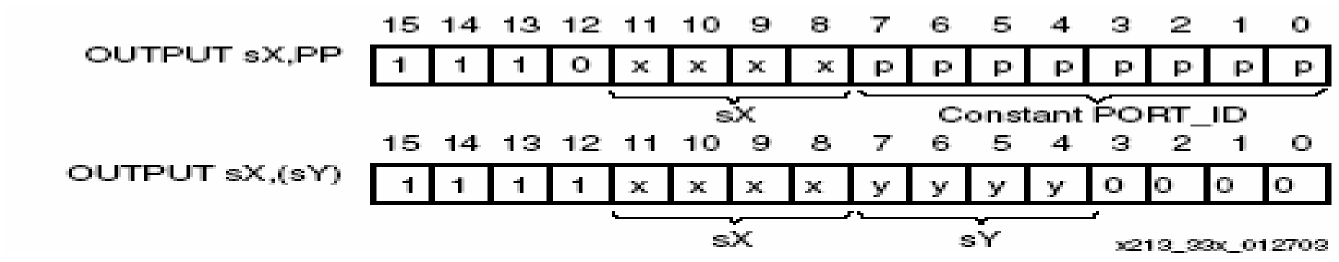
- Input



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INPUT sX,PP	1	0	1	0	x	x	x	x	p	p	p	p	p	p	p	p
					sX				Constant PORT_ID							
INPUT sX,(sY)	1	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
					sX				sY							

x213_32x_120302

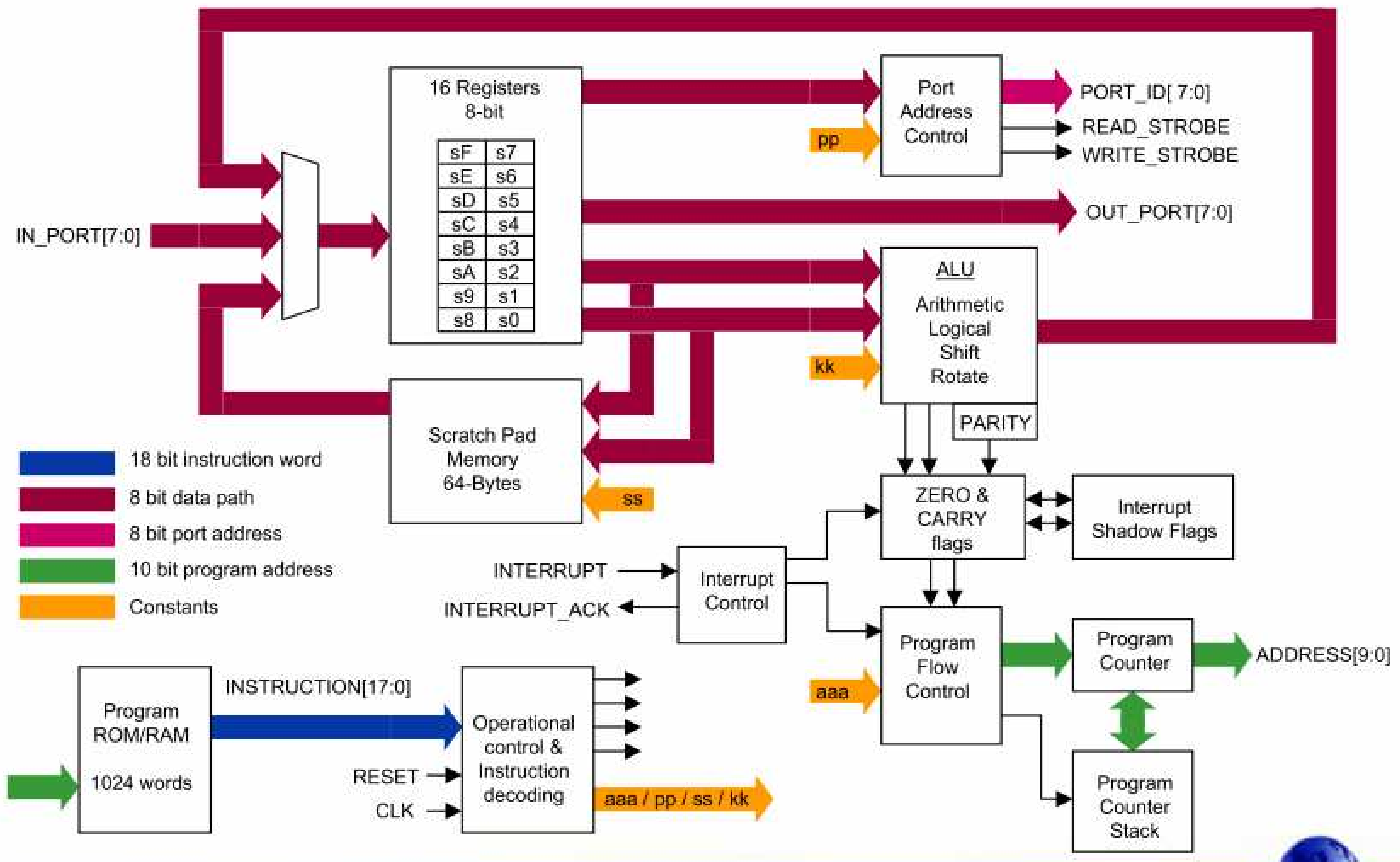
- Output



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTPUT sX,PP	1	1	1	0	x	x	x	x	p	p	p	p	p	p	p	p
					sX				Constant PORT_ID							
OUTPUT sX,(sY)	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
					sX				sY							

x213_33x_012703

KCPSM3 Architecture



Individual Modules

- Operand Select
- Arithmetic Group
- Logic Group
- Shift and Rotate Group
- ALU Multiplexer
- ALU Control
- Program Counter
- Stack Counter

KCPSM3 Instruction Set

'X' and 'Y' refer to the definition of the storage registers 's' in the range 0 to F.
'kk' represents a constant value in the range 00 to FF.
'aaa' represents an address in the range 000 to 3FF.
'pp' represents a port address in the range 00 to FF.
'ss' represents an internal storage address in the range 00 to 3F.

Program Control Group

```
JUMP aaa
JUMP Z,aaa
JUMP NZ,aaa
JUMP C,aaa
JUMP NC,aaa

CALL aaa
CALL Z,aaa
CALL NZ,aaa
CALL C,aaa
CALL NC,aaa

RETURN
RETURN Z
RETURN NZ
RETURN C
RETURN NC
```

Note that call and return supports up to a stack depth of 31.

Arithmetic Group

```
ADD sX,kk
ADDCY sX,kk
SUB sX,kk
SUBCY sX,kk
COMPARE sX,kk

ADD sX,sY
ADDCY sX,sY
SUB sX,sY
SUBCY sX,sY
COMPARE sX,sY
```

Logical Group

```
LOAD sX,kk
AND sX,kk
OR sX,kk
XOR sX,kk
TEST sX,kk

LOAD sX,sY
AND sX,sY
OR sX,sY
XOR sX,sY
TEST sX,sY
```

Shift and Rotate Group

```
SR0 sX
SR1 sX
SRX sX
SRA sX
RR sX

SL0 sX
SL1 sX
SLX sX
SLA sX
RL sX
```

Interrupt Group

```
RETURNI ENABLE
RETURNI DISABLE

ENABLE INTERRUPT
DISABLE INTERRUPT
```

Storage Group

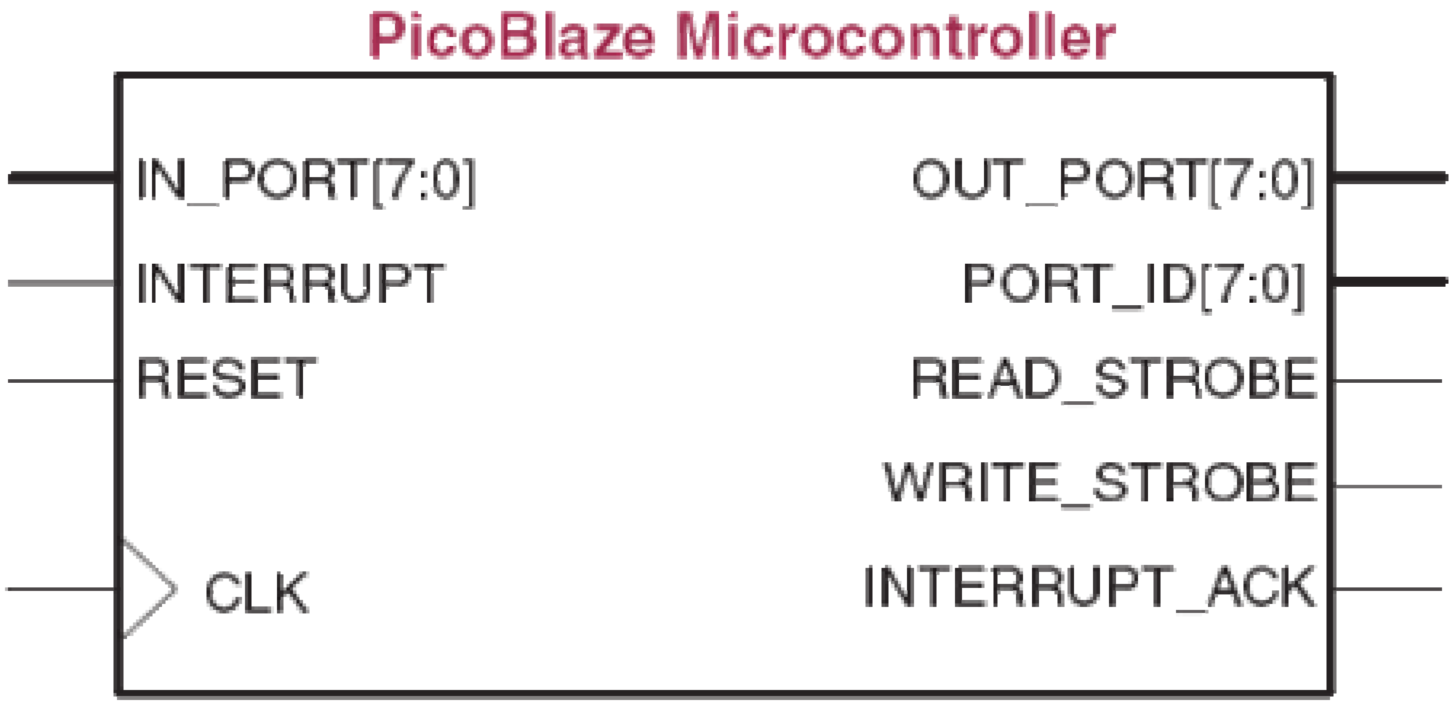
```
STORE sX,ss
STORE sX,(sY)
FETCH sX,ss
FETCH sX,(sY)
```

Input/Output Group

```
INPUT sX,pp
INPUT sX,(sY)
OUTPUT sX,pp
OUTPUT sX,(sY)
```

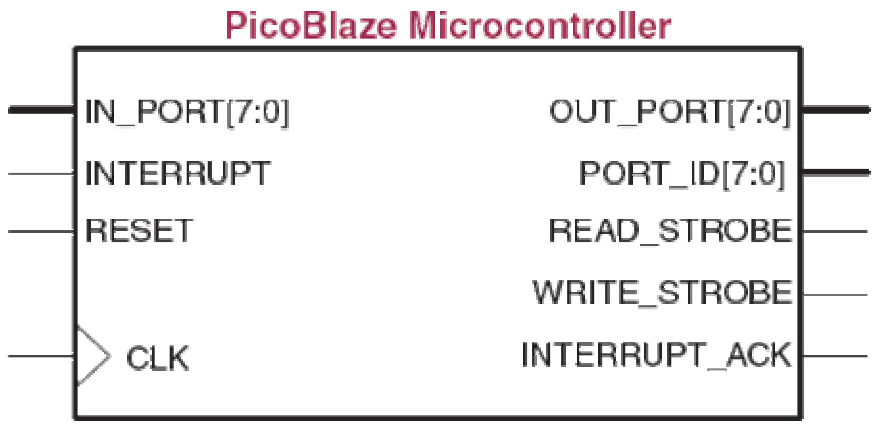
Individual Modules

CPU Interface

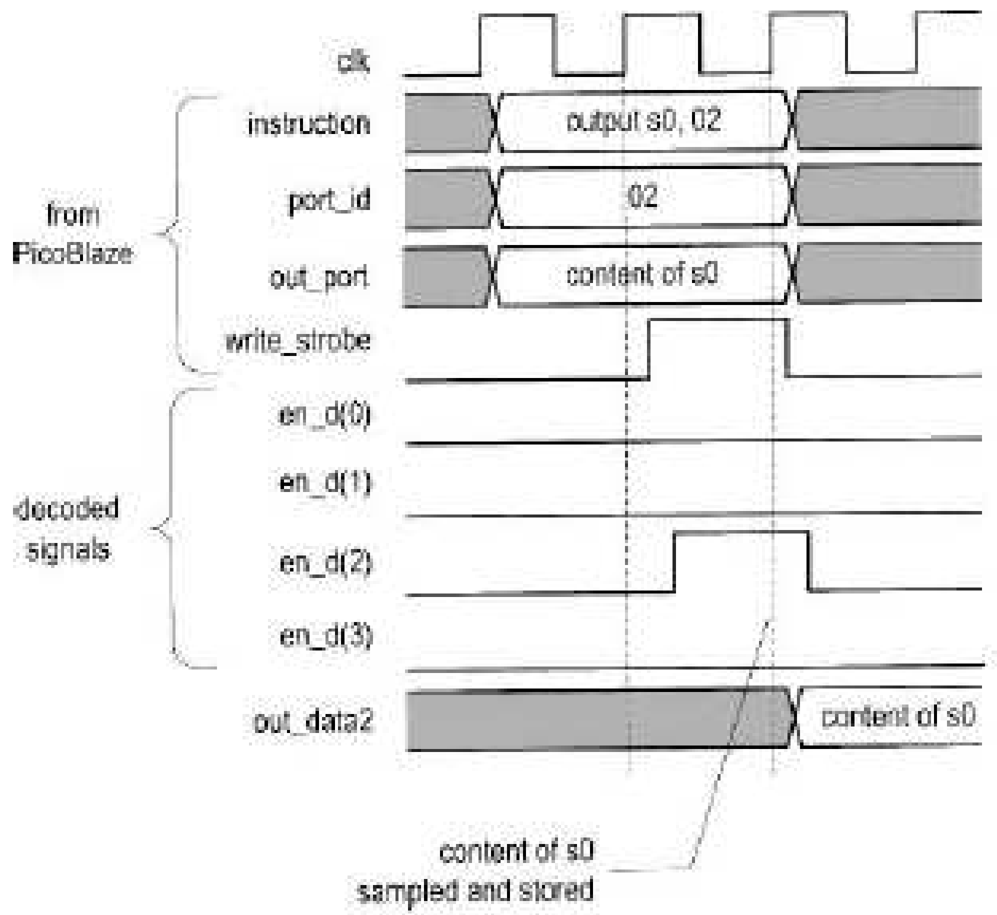


CPU Interface

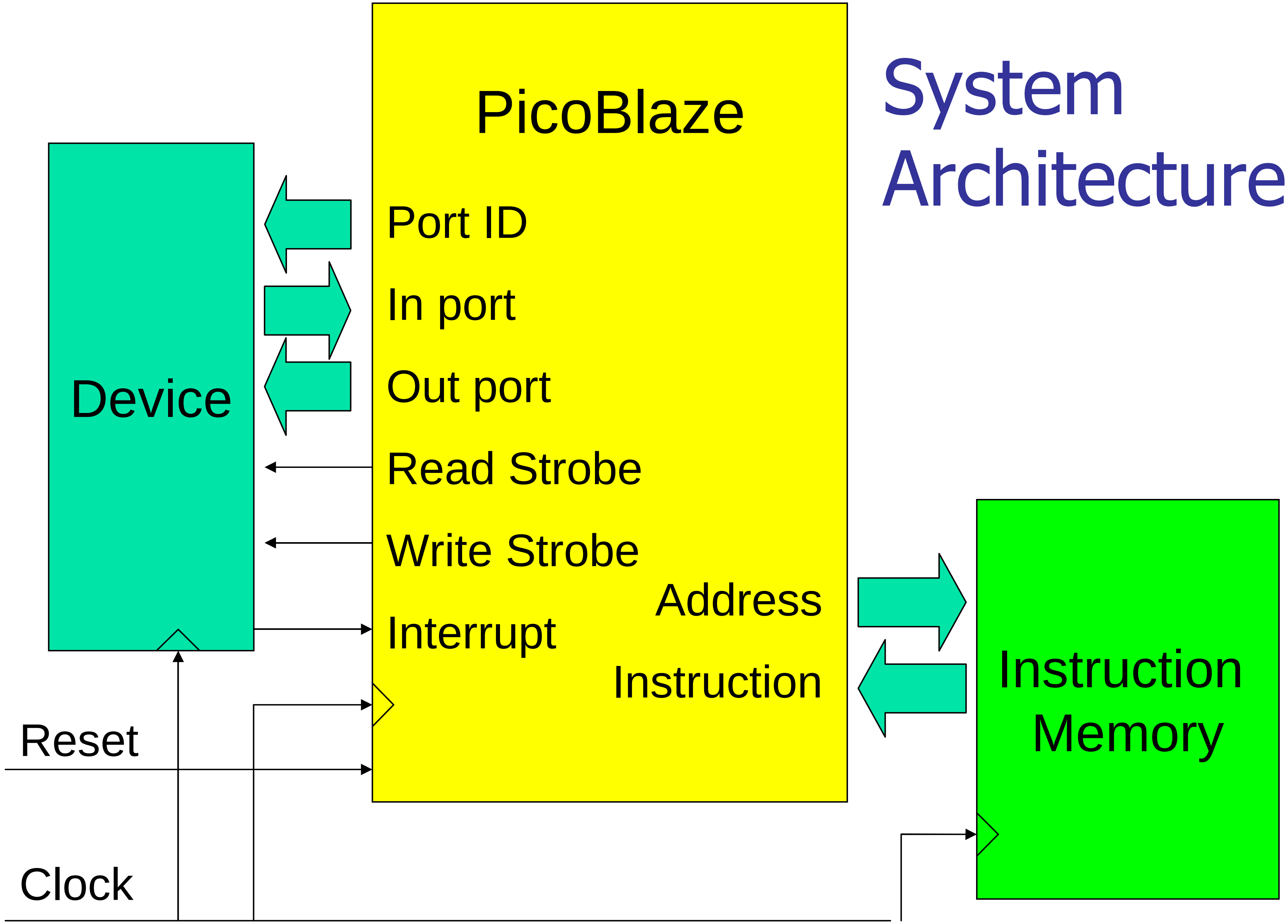
- Picoblaze no tiene Periféricos integrados estos se construyen si se necesitan …
- Se usan las instrucciones **input** y **ouput** para comunicase con ellos



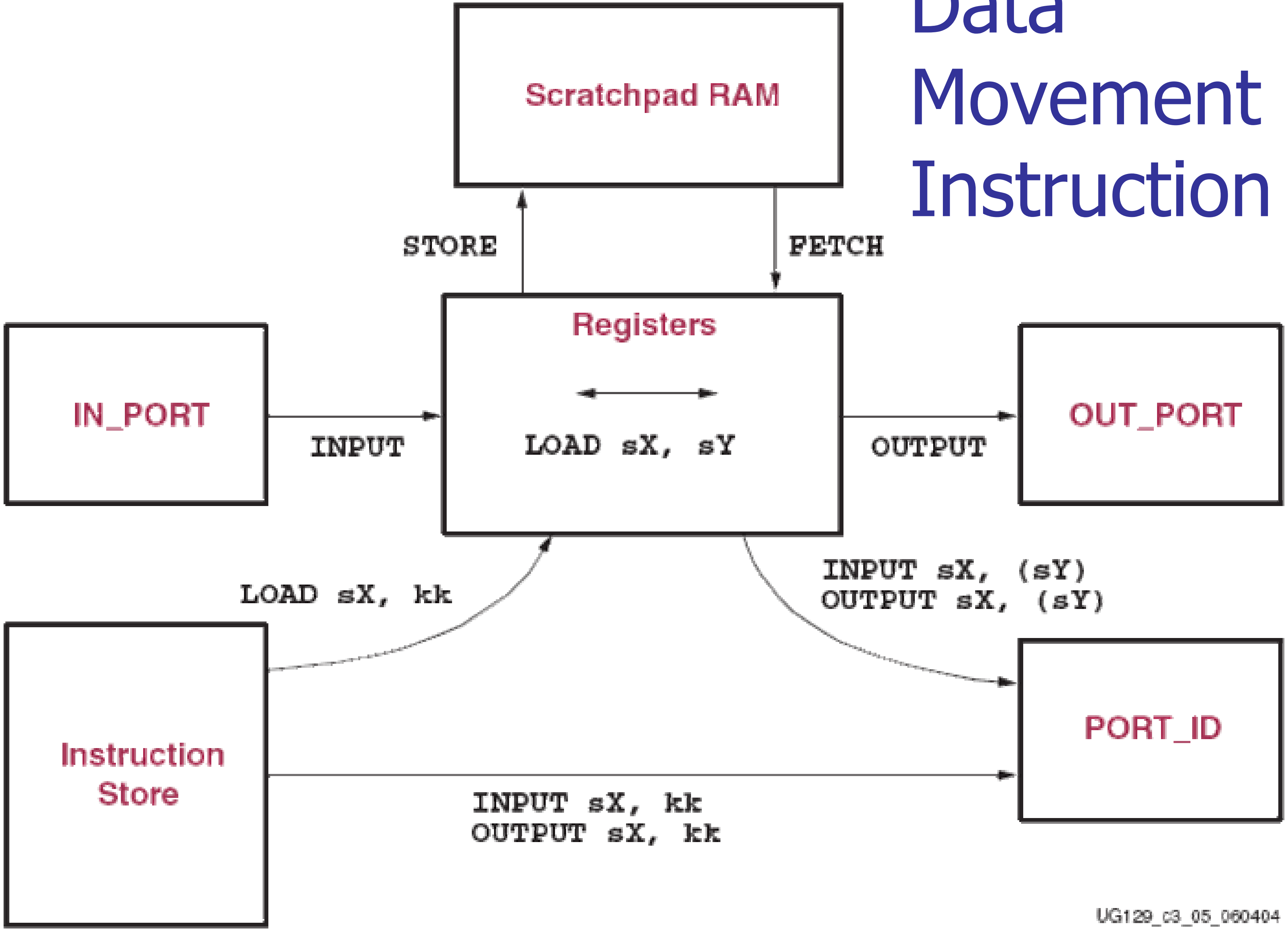
CPU Interface



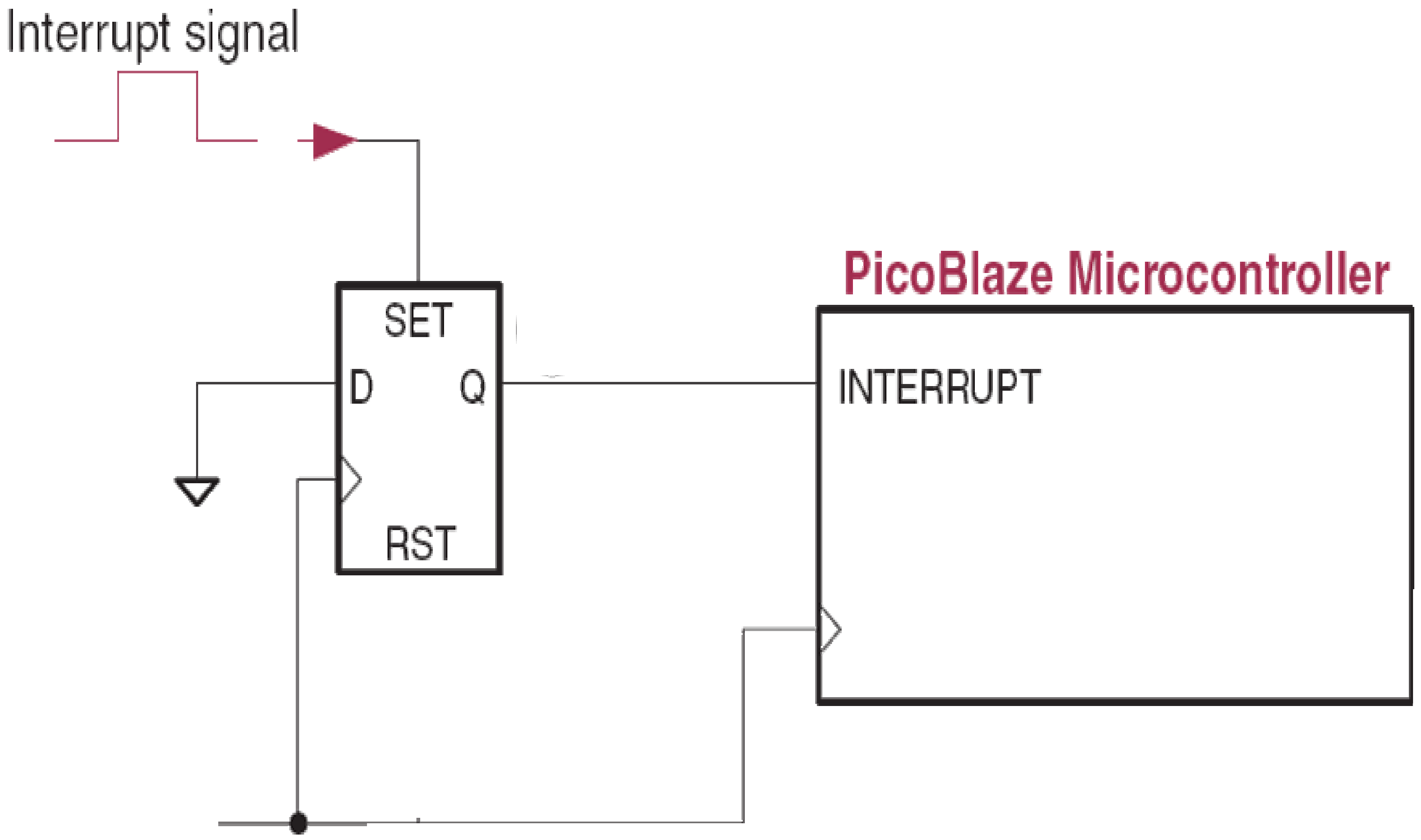
System Architecture



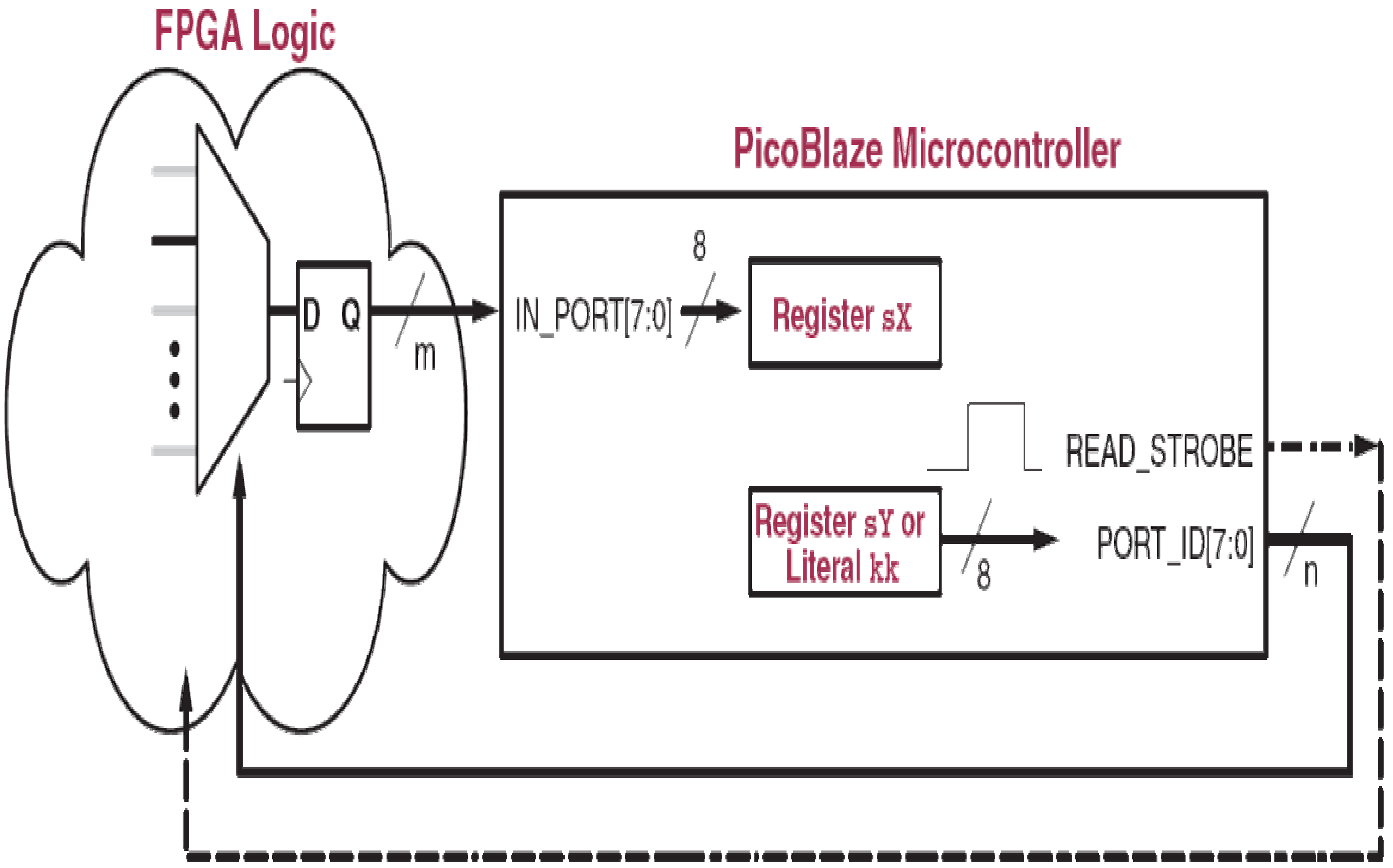
Data Movement Instruction



Interrupt PicoBlaze

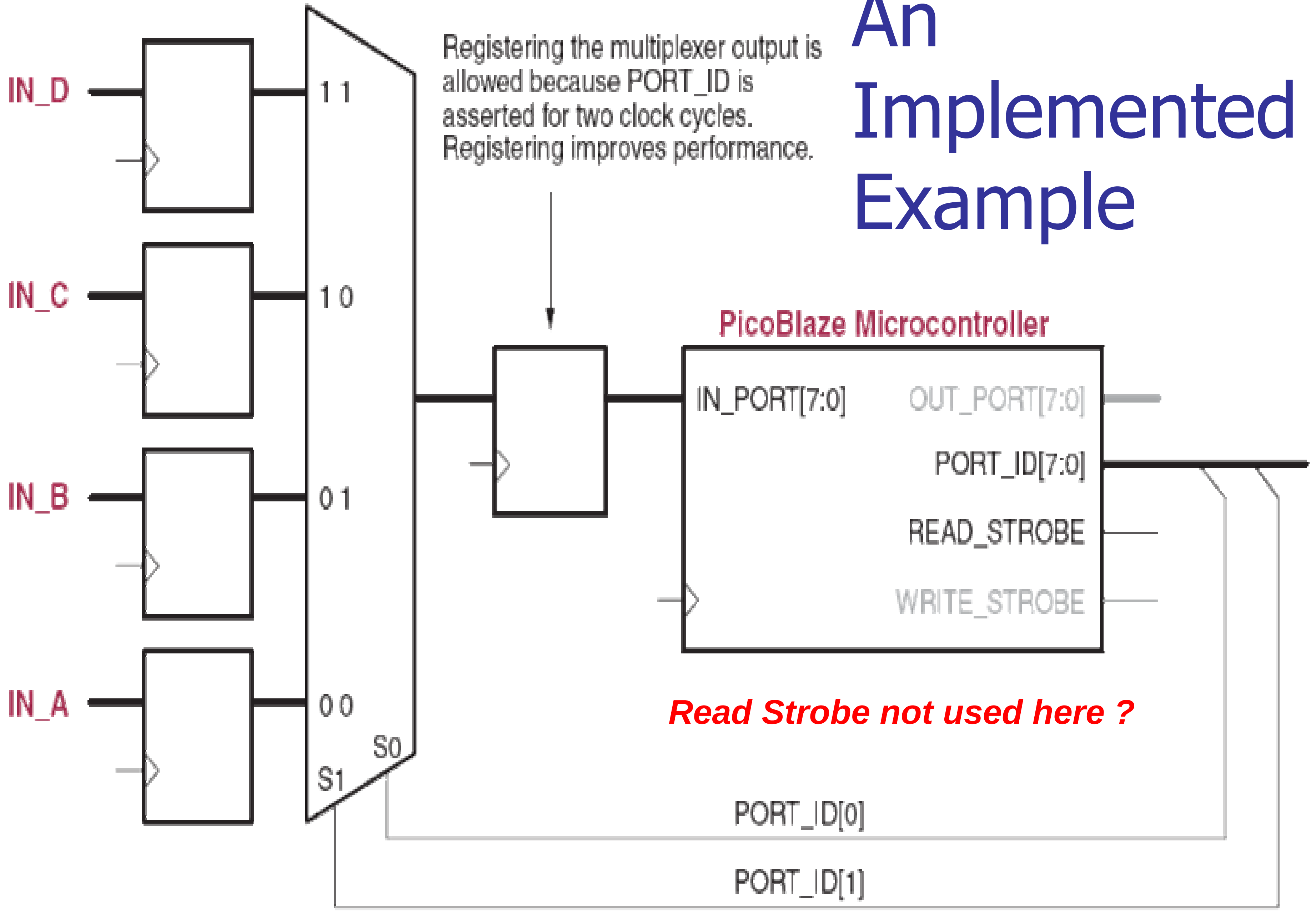


Port Input



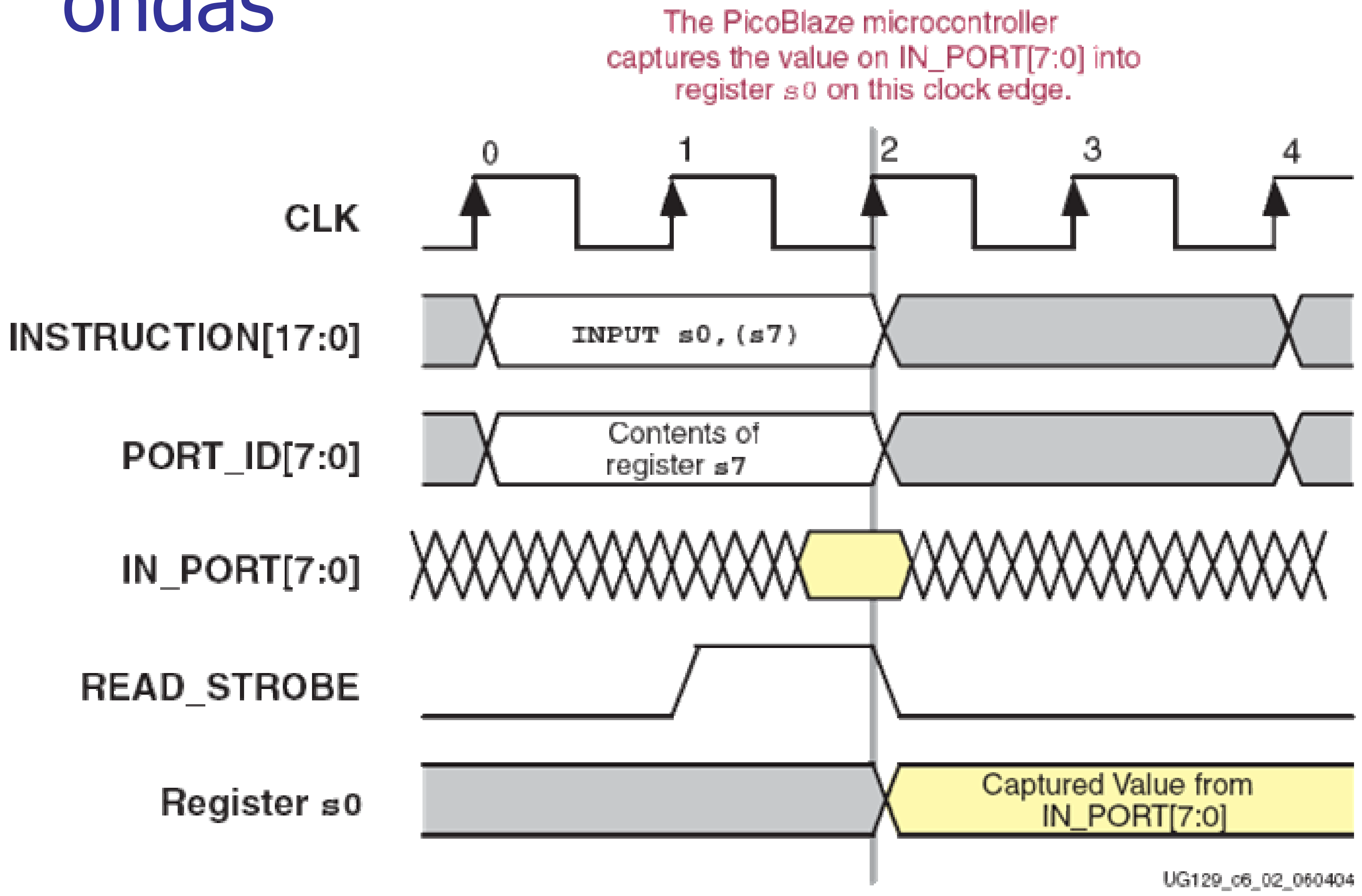
FPGA Logic
D
Q
m
PicoBlaze Microcontroller
IN_PORT[7:0]
8
Register sX
READ_STROBE
Register sY or Literal kk
8
PORT_ID[7:0]
n

An Implemented Example

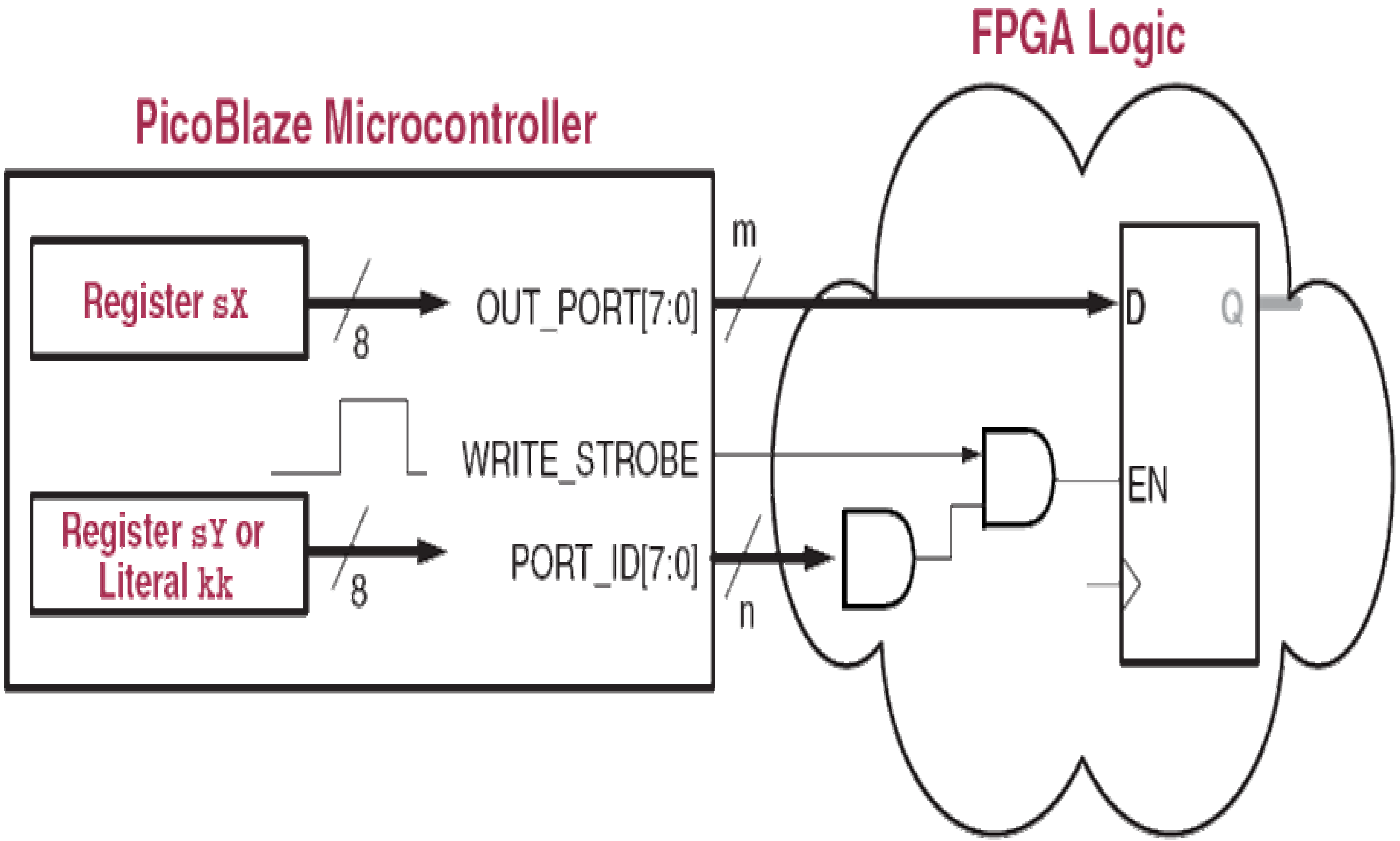

IN_D
IN_C
IN_B
IN_A
1 1
1 0
0 1
0 0
S0
S1
Registering the multiplexer output is allowed because PORT_ID is asserted for two clock cycles. Registering improves performance.
PicoBlaze Microcontroller
IN_PORT[7:0]
OUT_PORT[7:0]
PORT_ID[7:0]
READ_STROBE
WRITE_STROBE
PORT_ID[0]
PORT_ID[1]

Read Strobe not used here ?

Lectura de un puerto. Formas de ondas

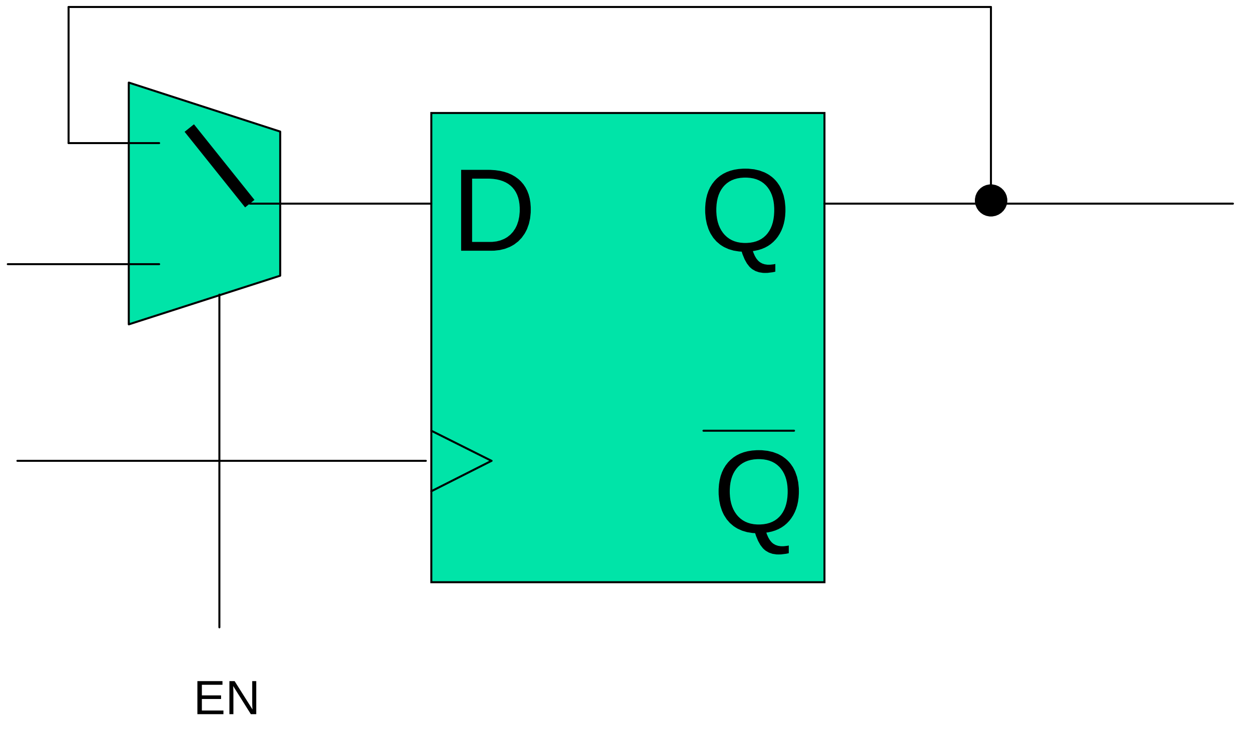

The PicoBlaze microcontroller captures the value on IN_PORT[7:0] into register s0 on this clock edge.
0
1
2
3
4
CLK
INSTRUCTION[17:0]
INPUT s0,(s7)
PORT_ID[7:0]
Contents of register s7
IN_PORT[7:0]
READ_STROBE
Register s0
Captured Value from IN_PORT[7:0]
UG129_c6_02_060404

Port Output

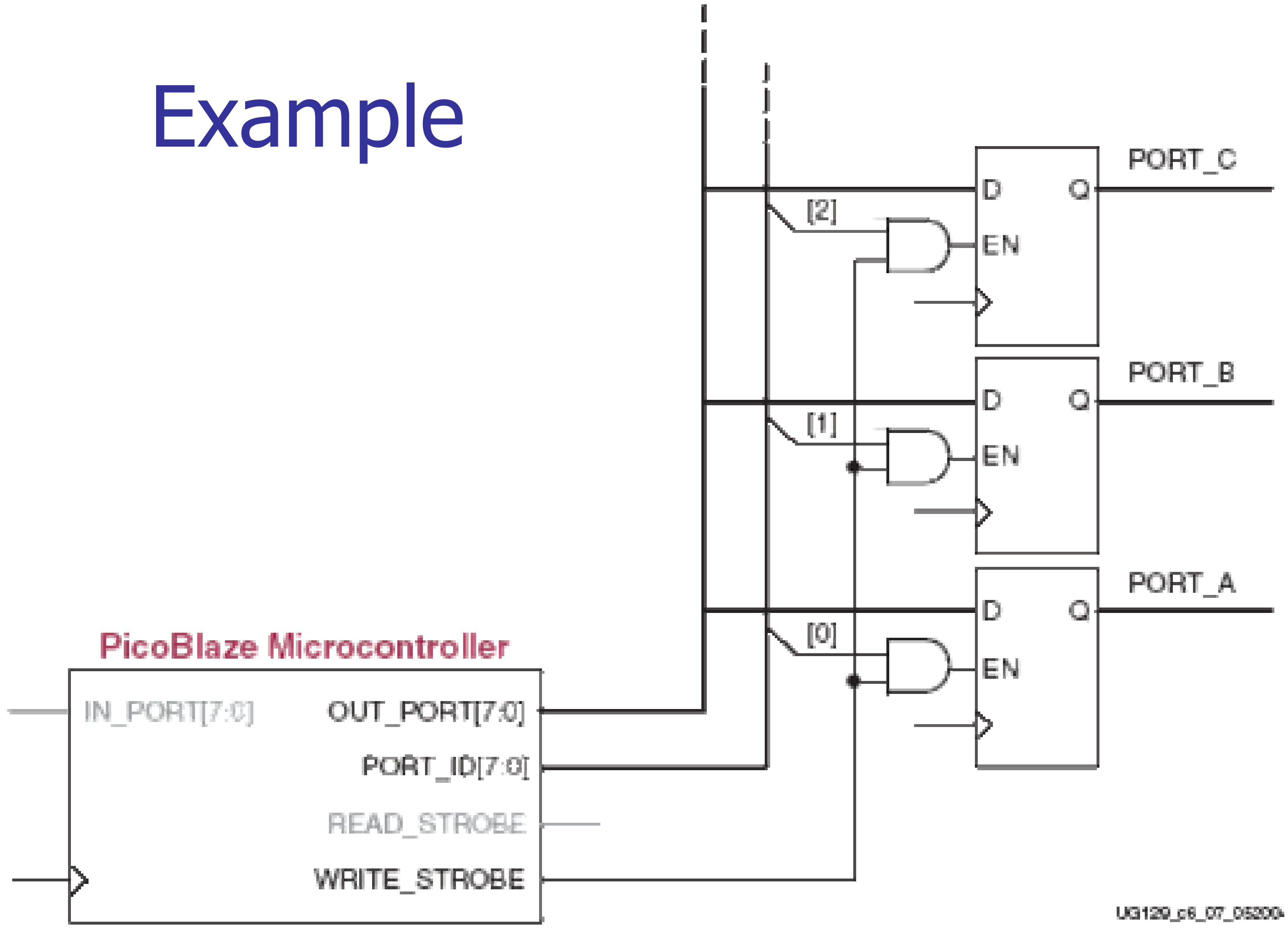


Clock of output Device is the same as CPU clock

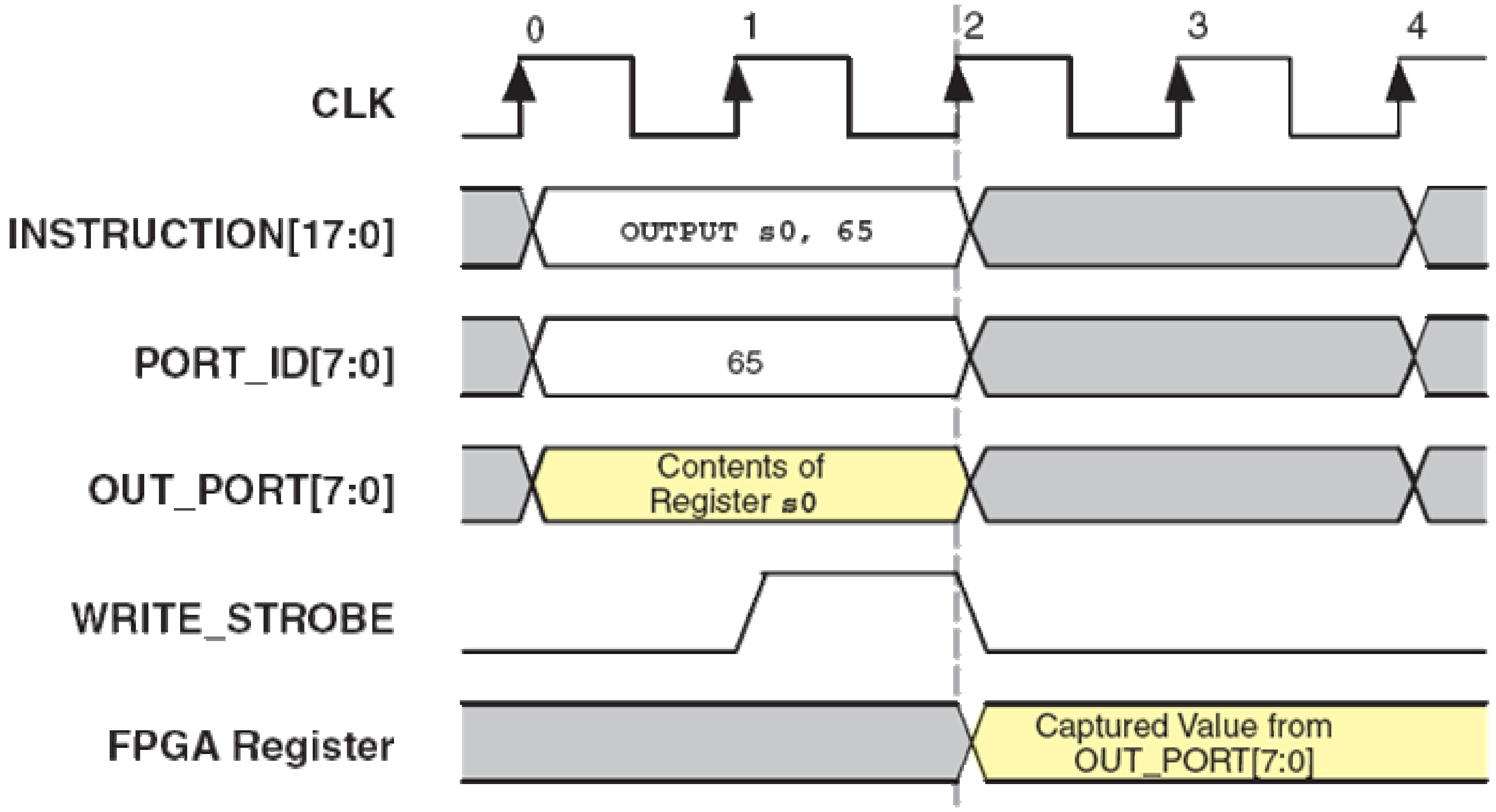
Write Enable



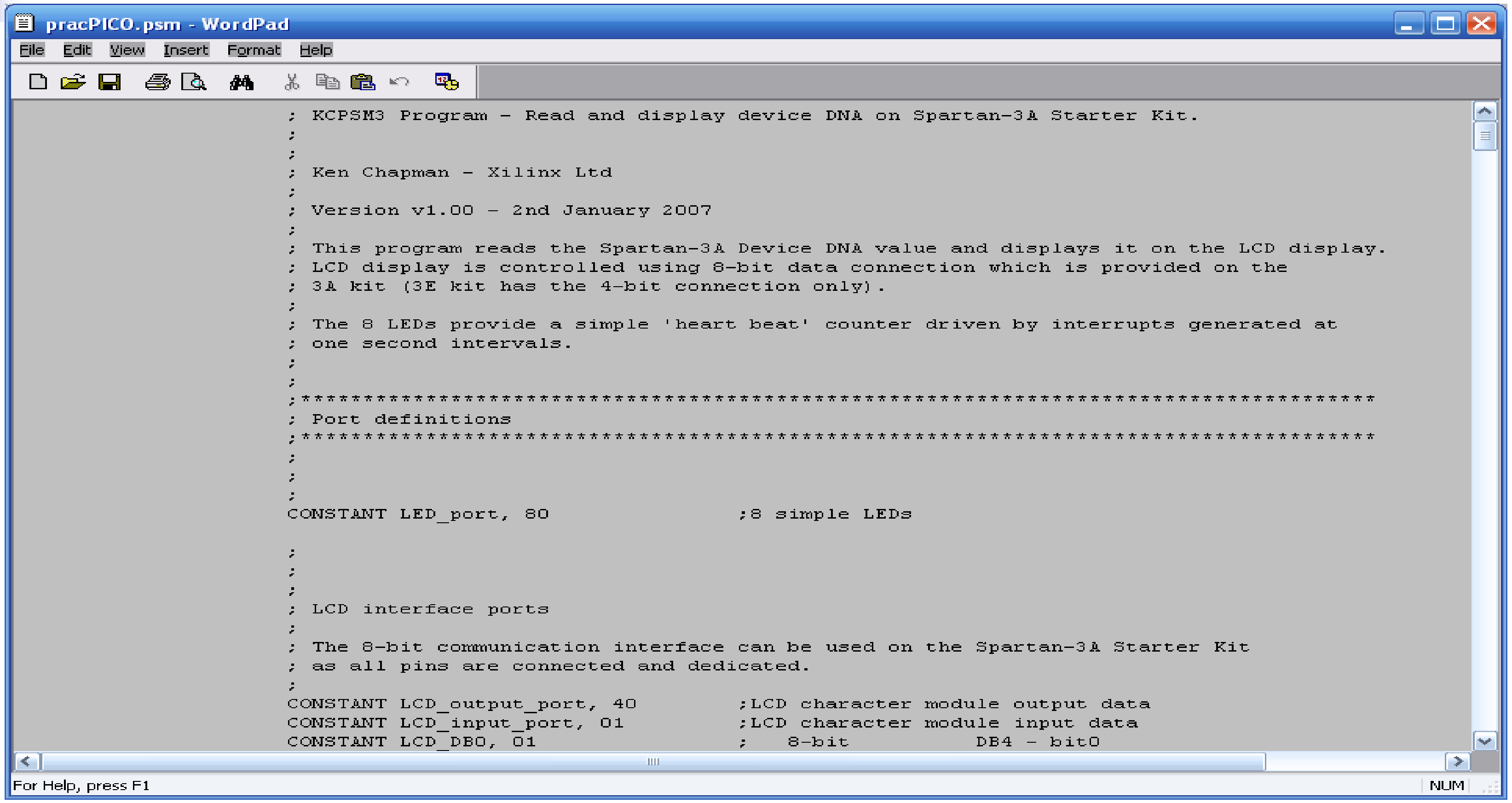
Example


PicoBlaze Microcontroller
IN_PORT[7:0]
OUT_PORT[7:0]
PORT_ID[7:0]
READ_STROBE
WRITE_STROBE
[2]
[1]
[0]
D
Q
EN
PORT_C
PORT_B
PORT_A
UG129_c6_07_052004

Port Write Waveform

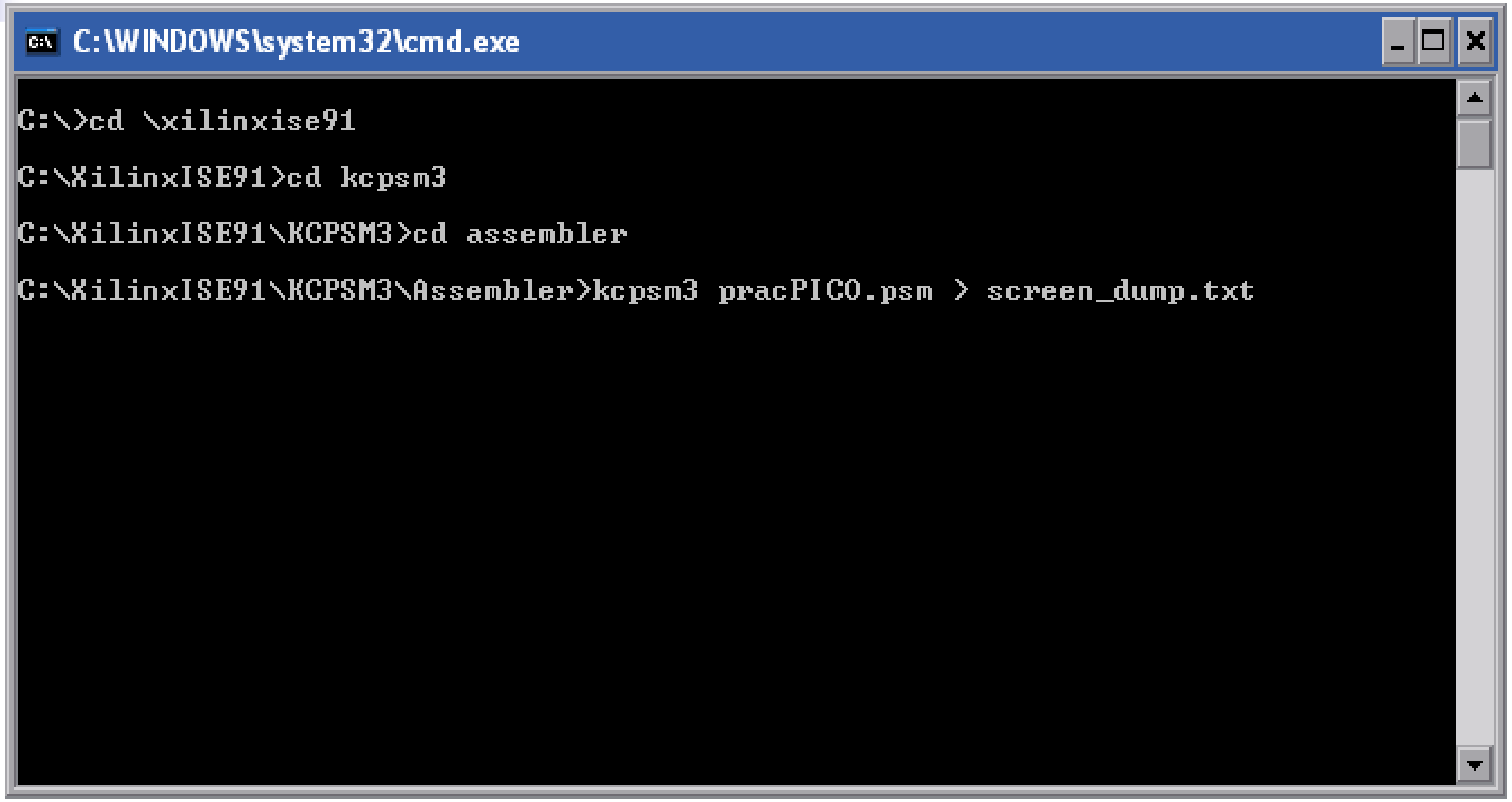


Editing PSM file



```
; KCPSM3 Program - Read and display device DNA on Spartan-3A Starter Kit.
;
;
; Ken Chapman - Xilinx Ltd
;
; Version v1.00 - 2nd January 2007
;
; This program reads the Spartan-3A Device DNA value and displays it on the LCD display.
; LCD display is controlled using 8-bit data connection which is provided on the
; 3A kit (3E kit has the 4-bit connection only).
;
; The 8 LEDs provide a simple 'heart beat' counter driven by interrupts generated at
; one second intervals.
;
;
;**************************************************************************************
; Port definitions
;**************************************************************************************
;
;
;
CONSTANT LED_port, 80                ;8 simple LEDs

;
;
;
; LCD interface ports
;
; The 8-bit communication interface can be used on the Spartan-3A Starter Kit
; as all pins are connected and dedicated.
;
CONSTANT LCD_output_port, 40         ;LCD character module output data
CONSTANT LCD_input_port, 01          ;LCD character module input data
CONSTANT LCD_DB0, 01                 ;   8-bit           DB4 - bit0
```

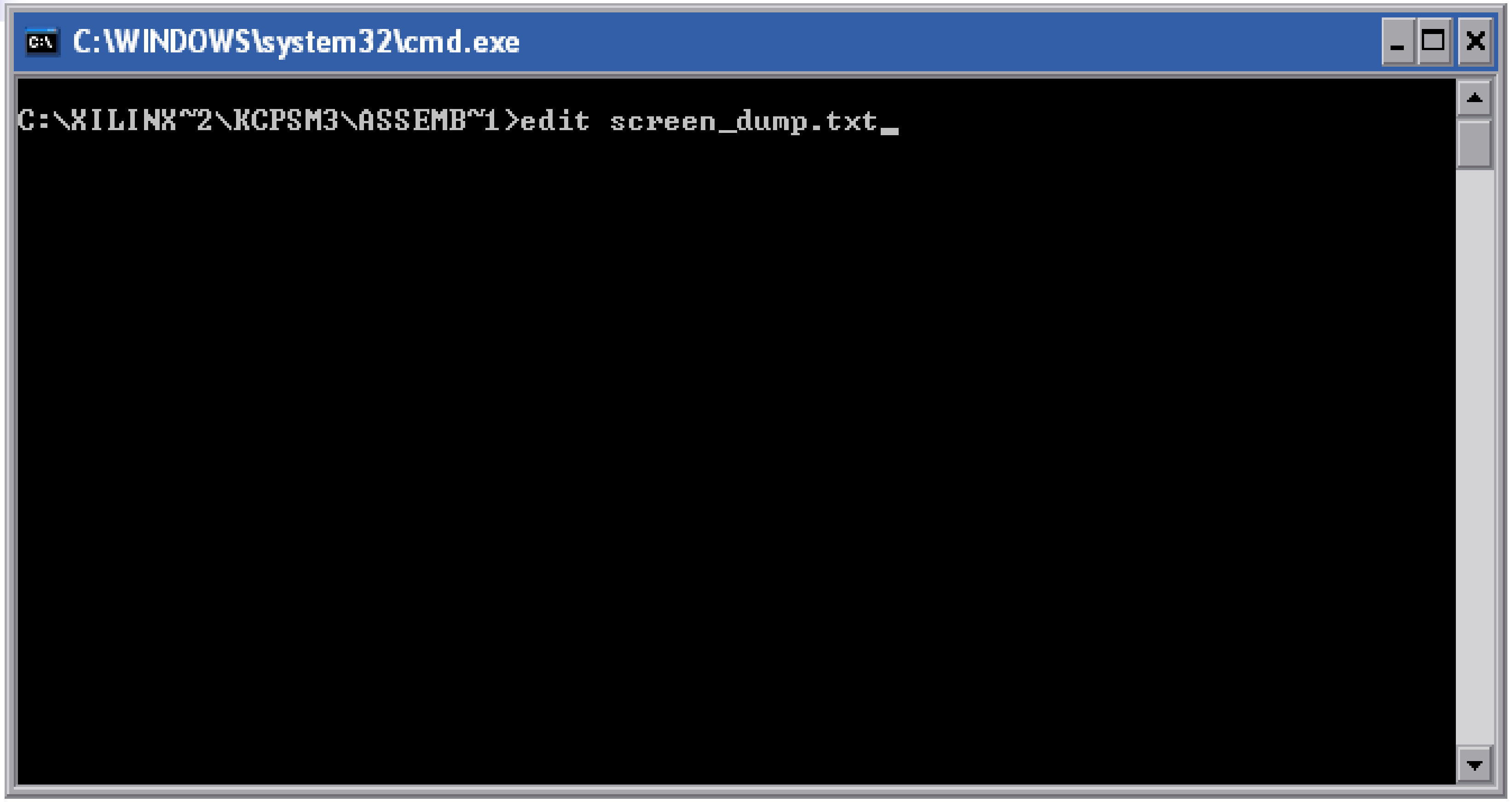
Compiling



C:\WINDOWS\system32\cmd.exe

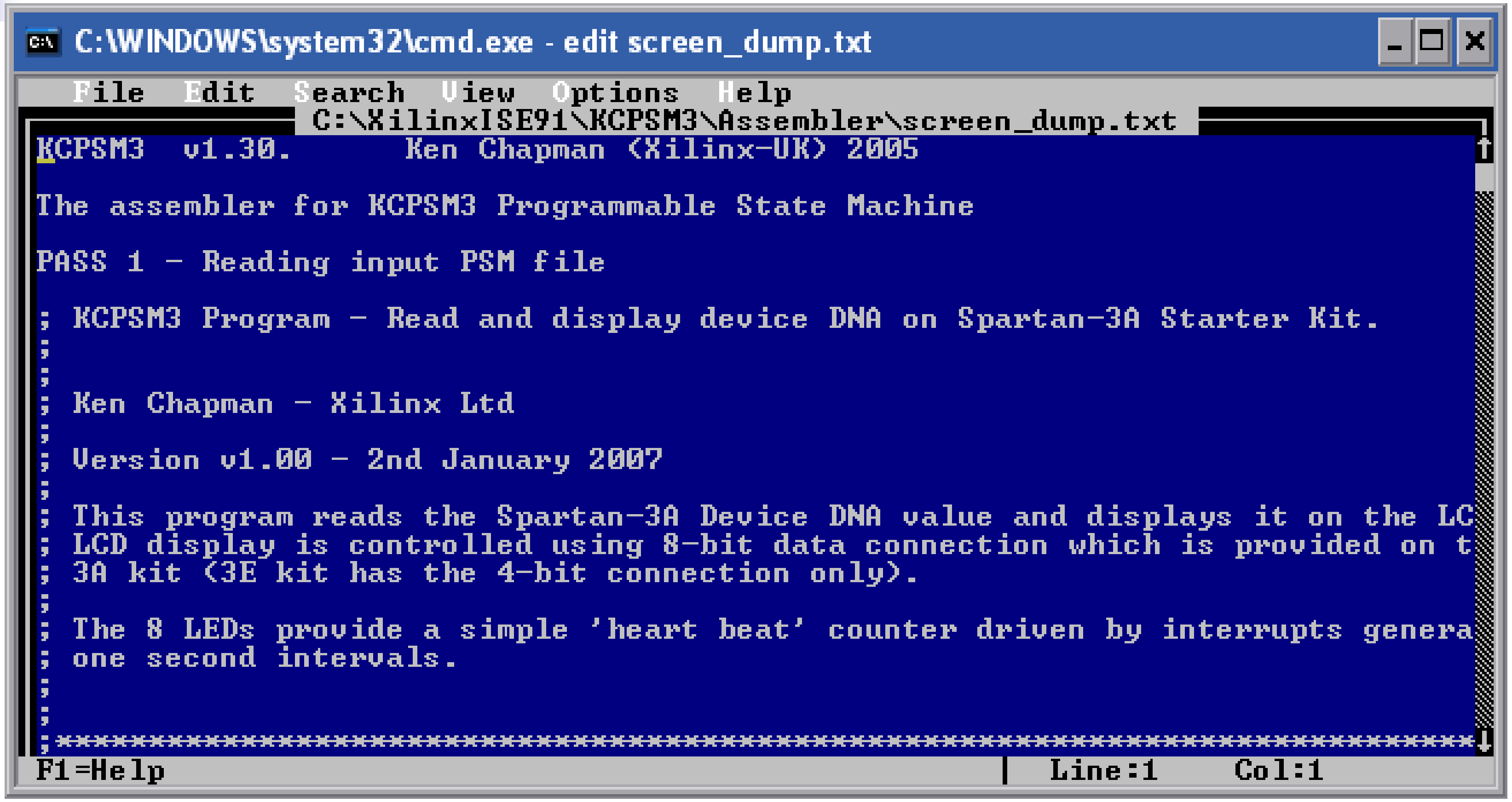
```
C:\>cd \xilinxise91

C:\XilinxISE91>cd kcpsm3

C:\XilinxISE91\KCPSM3>cd assembler

C:\XilinxISE91\KCPSM3\Assembler>kcpsm3 pracPICO.psm > screen_dump.txt
```

Verifying Compilation



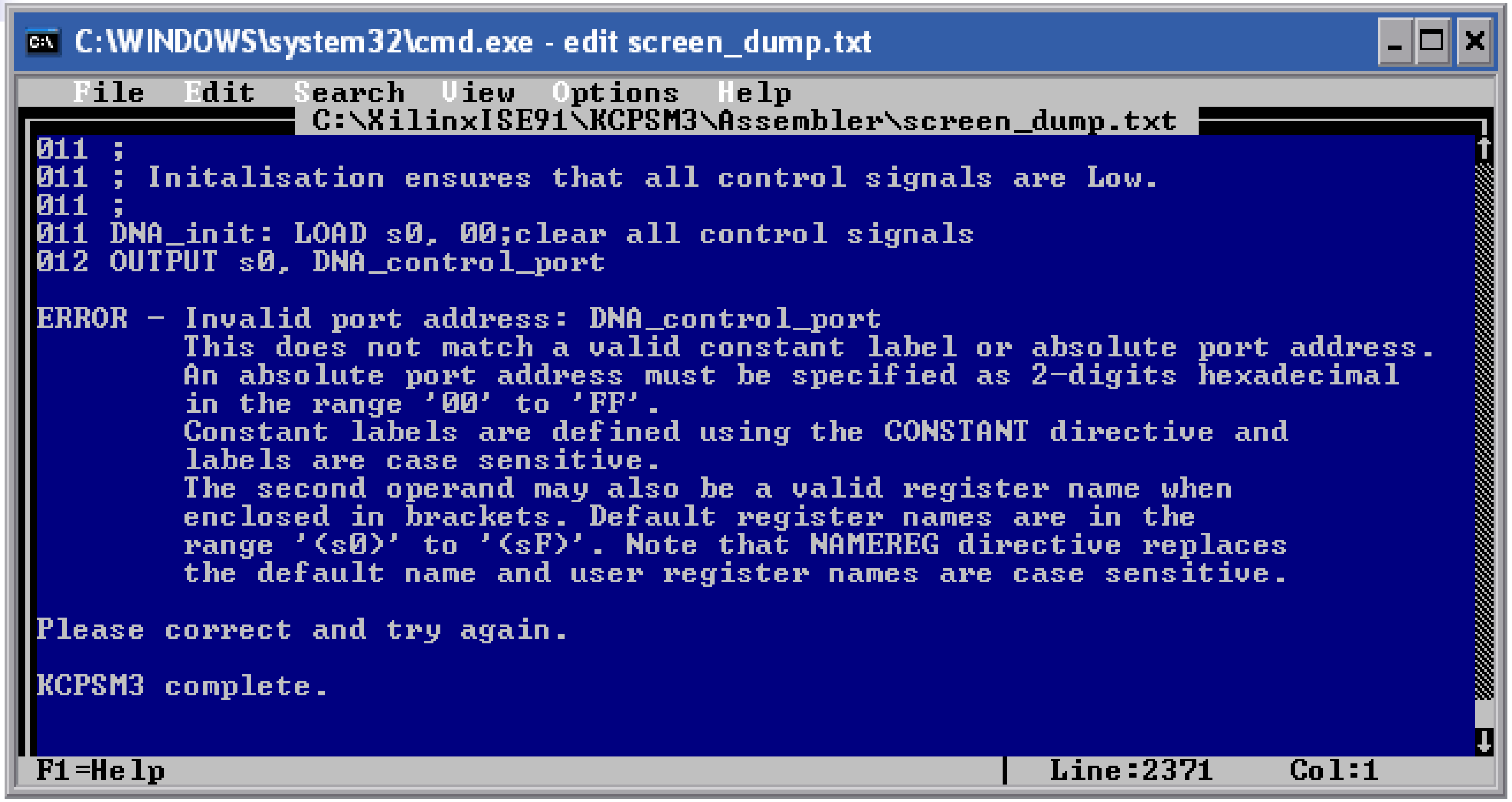
```
C:\XILINX~2\KCPSM3\ASSEMB~1>edit screen_dump.txt_
```

Verifying Compilation


C:\WINDOWS\system32\cmd.exe - edit screen_dump.txt
File Edit Search View Options Help
C:\XilinxISE91\KCPSM3\Assembler\screen_dump.txt
F1=Help
Line:1 Col:1

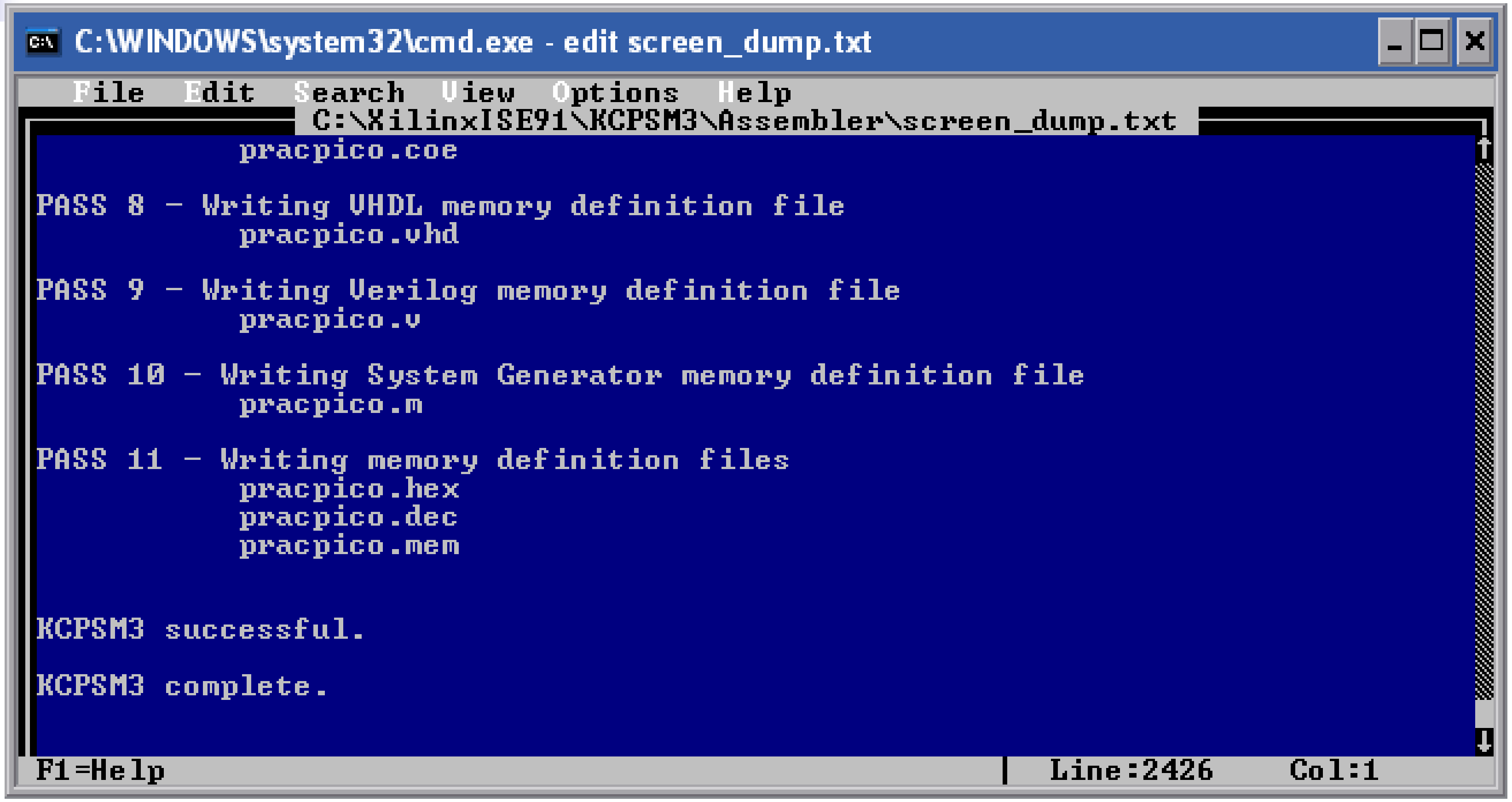
```
KCPSM3  v1.30.      Ken Chapman (Xilinx-UK) 2005

The assembler for KCPSM3 Programmable State Machine

PASS 1 - Reading input PSM file

; KCPSM3 Program - Read and display device DNA on Spartan-3A Starter Kit.
;
;
; Ken Chapman - Xilinx Ltd
;
; Version v1.00 - 2nd January 2007
;
; This program reads the Spartan-3A Device DNA value and displays it on the LC
; LCD display is controlled using 8-bit data connection which is provided on t
; 3A kit (3E kit has the 4-bit connection only).
;
; The 8 LEDs provide a simple 'heart beat' counter driven by interrupts genera
; one second intervals.
;
;
;******************************************************************************
```

Identifying an error


C:\WINDOWS\system32\cmd.exe - edit screen_dump.txt
File Edit Search View Options Help
C:\XilinxISE91\KCPSM3\Assembler\screen_dump.txt
F1=Help
Line:2371 Col:1

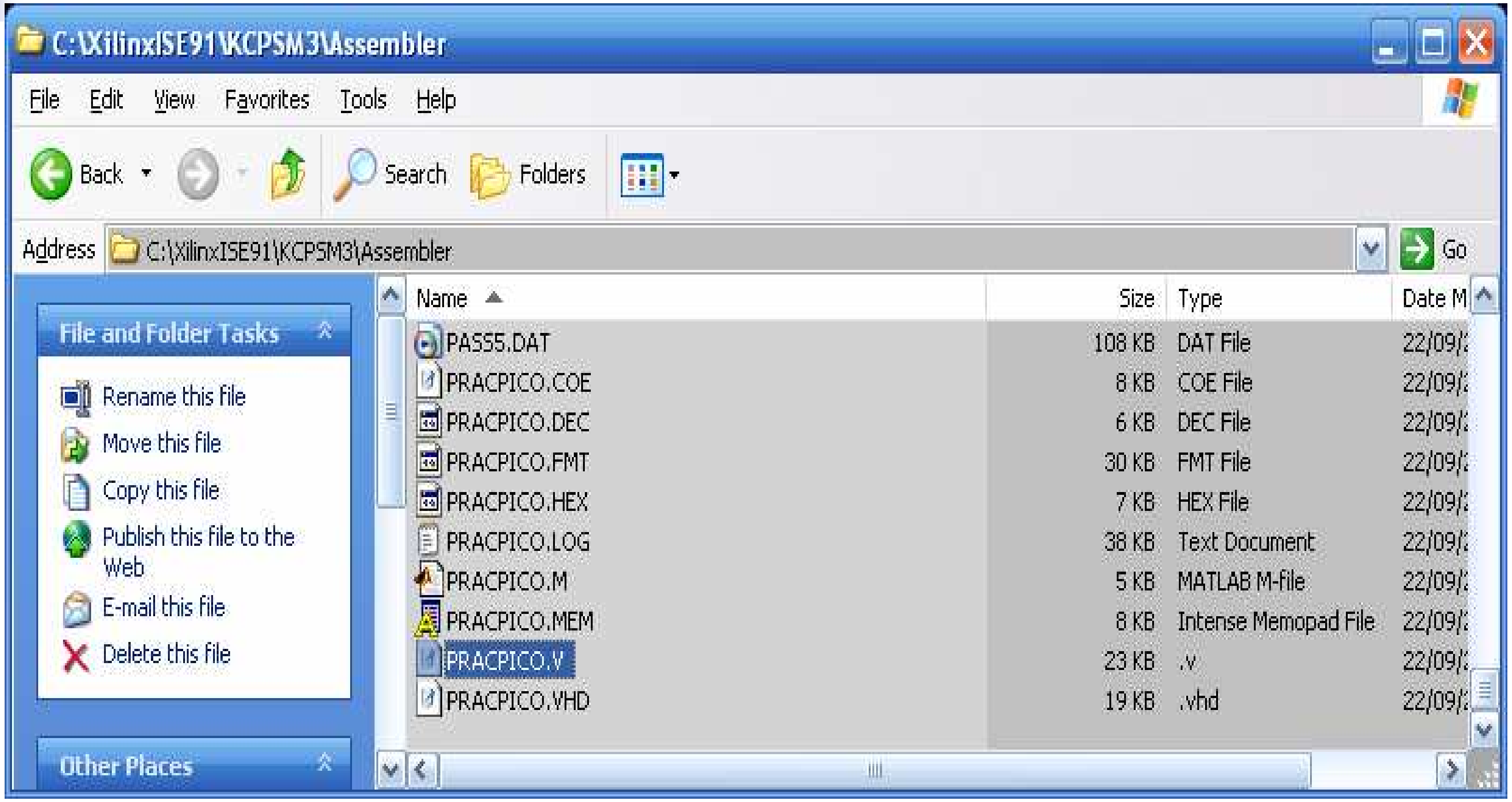
```
011 ;
011 ; Initalisation ensures that all control signals are Low.
011 ;
011 DNA_init: LOAD s0, 00;clear all control signals
012 OUTPUT s0, DNA_control_port

ERROR - Invalid port address: DNA_control_port
        This does not match a valid constant label or absolute port address.
        An absolute port address must be specified as 2-digits hexadecimal
        in the range '00' to 'FF'.
        Constant labels are defined using the CONSTANT directive and
        labels are case sensitive.
        The second operand may also be a valid register name when
        enclosed in brackets. Default register names are in the
        range '(s0)' to '(sF)'. Note that NAMEREG directive replaces
        the default name and user register names are case sensitive.

Please correct and try again.

KCPSM3 complete.
```

Verifying Everything is OK


C:\WINDOWS\system32\cmd.exe - edit screen_dump.txt
File Edit Search View Options Help
C:\XilinxISE91\KCPSM3\Assembler\screen_dump.txt
F1=Help
Line:2426 Col:1

```
           pracpico.coe

PASS 8 - Writing VHDL memory definition file
           pracpico.vhd

PASS 9 - Writing Verilog memory definition file
           pracpico.v

PASS 10 - Writing System Generator memory definition file
           pracpico.m

PASS 11 - Writing memory definition files
           pracpico.hex
           pracpico.dec
           pracpico.mem


KCPSM3 successful.

KCPSM3 complete.
```

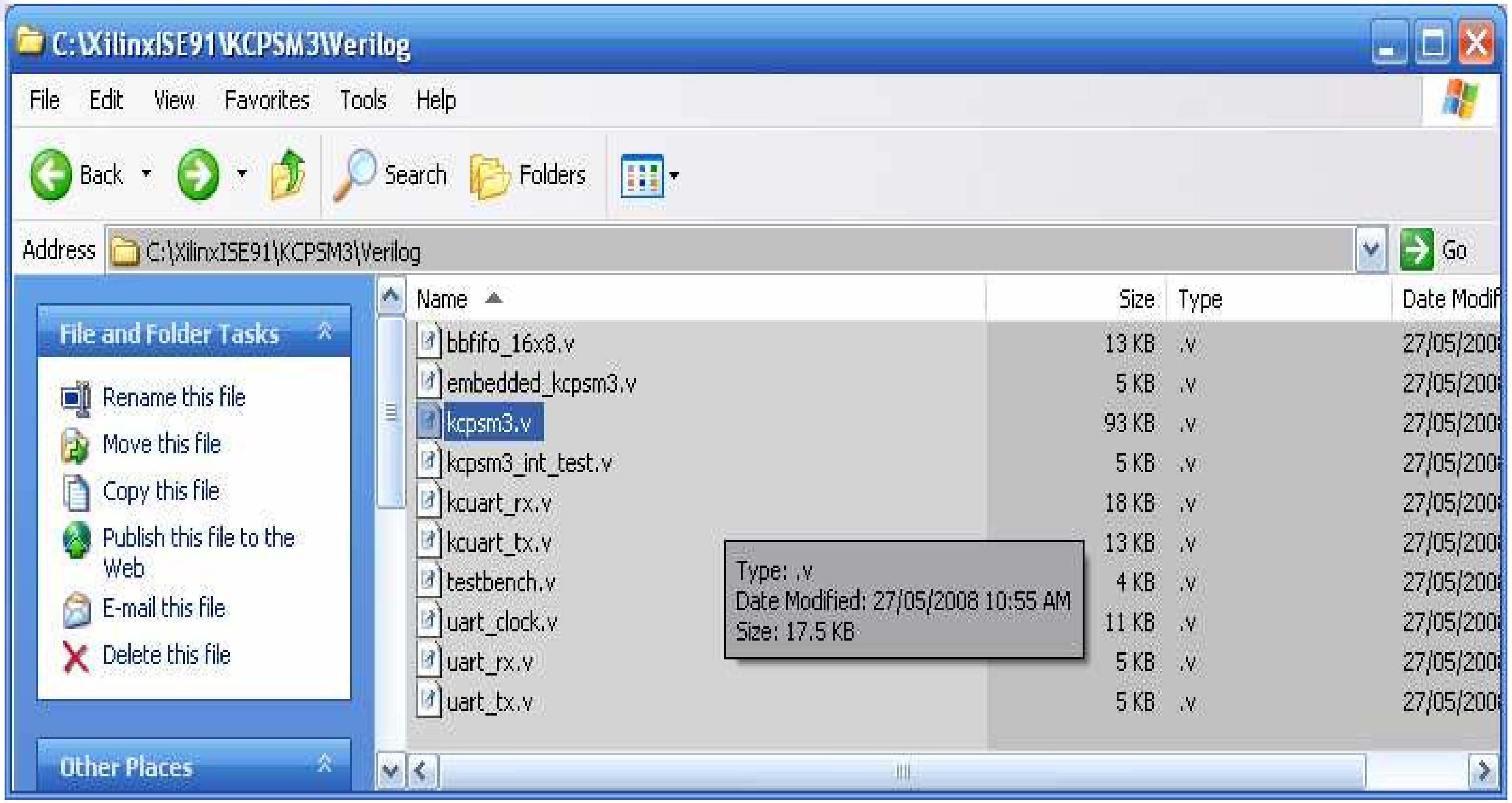
Copy generated verilog file



C:\XilinxISE91\KCPSM3\Assembler
File Edit View Favorites Tools Help
Back
Search
Folders
Address C:\XilinxISE91\KCPSM3\Assembler
Go
File and Folder Tasks
Rename this file
Move this file
Copy this file
Publish this file to the Web
E-mail this file
Delete this file
Other Places

Name	Size	Type	Date M
PASS5.DAT	108 KB	DAT File	22/09/
PRACPICO.COE	8 KB	COE File	22/09/
PRACPICO.DEC	6 KB	DEC File	22/09/
PRACPICO.FMT	30 KB	FMT File	22/09/
PRACPICO.HEX	7 KB	HEX File	22/09/
PRACPICO.LOG	38 KB	Text Document	22/09/
PRACPICO.M	5 KB	MATLAB M-file	22/09/
PRACPICO.MEM	8 KB	Intense Memopad File	22/09/
PRACPICO.V	23 KB	.v	22/09/
PRACPICO.VHD	19 KB	.vhd	22/09/

Copy kcpsm3.v to project directory



Name	Size	Type	Date Modif
bbfifo_16x8.v	13 KB	.v	27/05/200
embedded_kcpsm3.v	5 KB	.v	27/05/200
kcpsm3.v	93 KB	.v	27/05/200
kcpsm3_int_test.v	5 KB	.v	27/05/200
kcuart_rx.v	18 KB	.v	27/05/200
kcuart_tx.v	13 KB	.v	27/05/200
testbench.v	4 KB	.v	27/05/200
uart_clock.v	11 KB	.v	27/05/200
uart_rx.v	5 KB	.v	27/05/200
uart_tx.v	5 KB	.v	27/05/200

Type: .v
Date Modified: 27/05/2008 10:55 AM
Size: 17.5 KB

Integration

- We need to integrate the Verilog code generated by the assembler into the project
- We'll see how to do this together in class (it's not hard!)

Conclusion

- The assembler gives the verilog file for a program written in the instruction set
- A module of the Program ROM is used together with the processor’s main module
- Together they define the complete microcontroller
- Just like the PicoCtrl (Except that with the PicoCtrl we did the assembly process manually!)