CPUGEN TUTORIAL V1.00

gferrante@opencores.org

Getting start

1) Decompress cpugen.zip inside a working directory (ex. C:\cpugen)

2) Add the working directory into the windows PATH

3) Altera users: from Altera site (www.altera.com)

3.1) Download and install Quartus II [ver. >= 3.0] (Altera FPGA development tool); ask for a free/commercial license

3.2) If you want to do VHDL simulations: download and install Modelsim AE (VHDL simulator); ask for a commercial license [ver. 5.7c]

4) Xilinx users: from Xilinx site (www.xilinx.com)

4.1) Download WebPack ISE [ver. >= 5.2i] (Xilinx FPGA development tool); ask for a free/commercial license

4.2) If you want to do VHDL simulations: download and install Modelsim XE (VHDL simulator); ask for a free/commercial license [ver. 5.6e]

Simulation notes: Quartus has a waveform simulator inside; ISE needs a VHDL simulator.

Applications

You find some test projects inside applications directory.
The VHDL cpu source code, assembler and rom/ram files are inside the relative design directory; others VHDL files used by the design are inside Components and Testbench directories too.

Each design has 3 implementations:

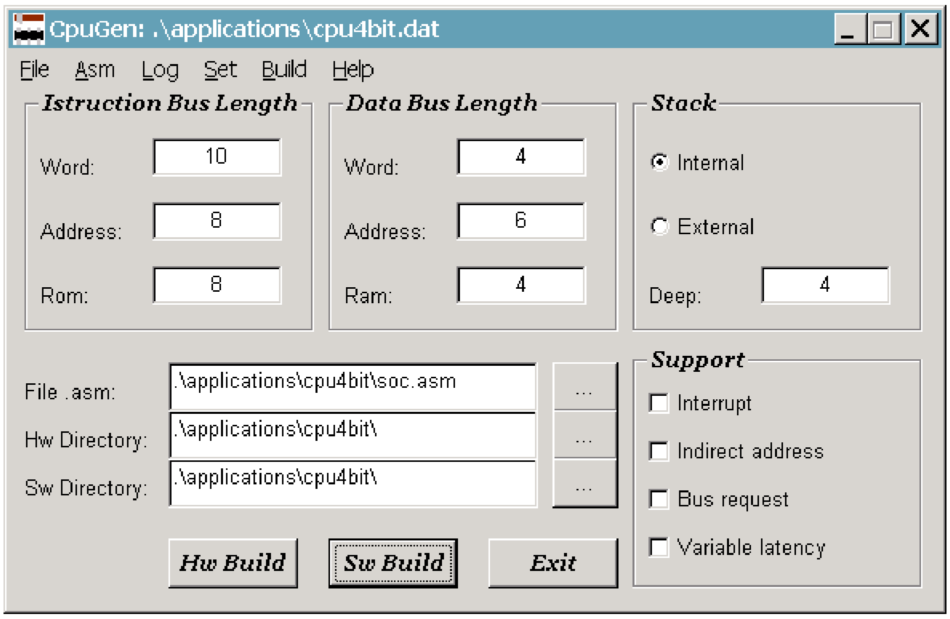
1) RTL design (vendor independent)	:Testbench\testbench_apre.mpf or testbench_xpre.mpf
2) Altera design	:Testbench\testbench_apost.mpf
3) Xilinx design	:Testbench\testbench_xpost.mpf

Cpu4Bit

Goal:

- compare 4 digital input with 4 fixed thresolds and move a PWM output consequently
- use a microcontroller architecture inside a low cost FPGA device (< 10K gates device)
- use few logic elements and memory resources

Run **cpugen** and load Applications\cpu4bit.dat cpu configuration:


CpuGen: .\applications\cpu4bit.dat

Istruction Bus Length
Word: 10
Address: 8
Rom: 8

Data Bus Length
Word: 4
Address: 6
Ram: 4

Stack
[✓ Internal]
External
Deep: 4

File .asm: .\applications\cpu4bit\soc.asm
Hw Directory: .\applications\cpu4bit\
Sw Directory: .\applications\cpu4bit\

Support
☐ Interrupt
☐ Indirect address
☐ Bus request
☐ Variable latency

RTL design:

VHDL sources:

1) CPU

- cpu_utils.vhd
- cpu_iu.vhd
- cpu_cu.vhd
- cpu_du.vhd
- cpu_oa.vhd
- cpu.vhd

2) COMPONENTS

- ctrl4cpu.vhd
- pwm.vhd
- utils_pkg.vhd
- ram.vhd (just for RTL testbench; use ram.vin)
- rom.vhd (just for RTL testbench; use rom.vin)

3) TESTBENCH (RTL)

- cpu4bit.vhd
- clock.vhd
- world.vhd
- testbench_a.vhd

CPU 4 bit => Data Bus Length Word = 4
If I use a 8 bit Istruction Words Length (the minimum length), I have a 16 word (8 – 4) istruction page size (too small), so I need a 10 bit Istruction Words Length => 64 word istruction page size. The source code (Asm) isn't too long so 256 rom words (8) are enough. Istruction Address bus = 8 => 4 pages of 64 words each.
I have 16 nibble of RAM, a control register and a PWM peripheral.

Address mapping (4 bit size):

• RAM	0 ... 0xF	
• CTRL	0x10	= CTRL_IN = digital input; CTRL_OUT[0] = PWM enable
• PWMPL	0x12	= PWM period (low significative nibble)
• PWMPH	0x13	= PWM period (high significative nibble)
• PWMLL	0x14	= PWM low level (low significative nibble)
• PWMLH	0x15	= PWM low level (high significative nibble)

I need 4 bit of RAM address and at least 5 of data addressing (6).
I decide to poll the 4 input, so I don't need Interrupt support.

Code structure:

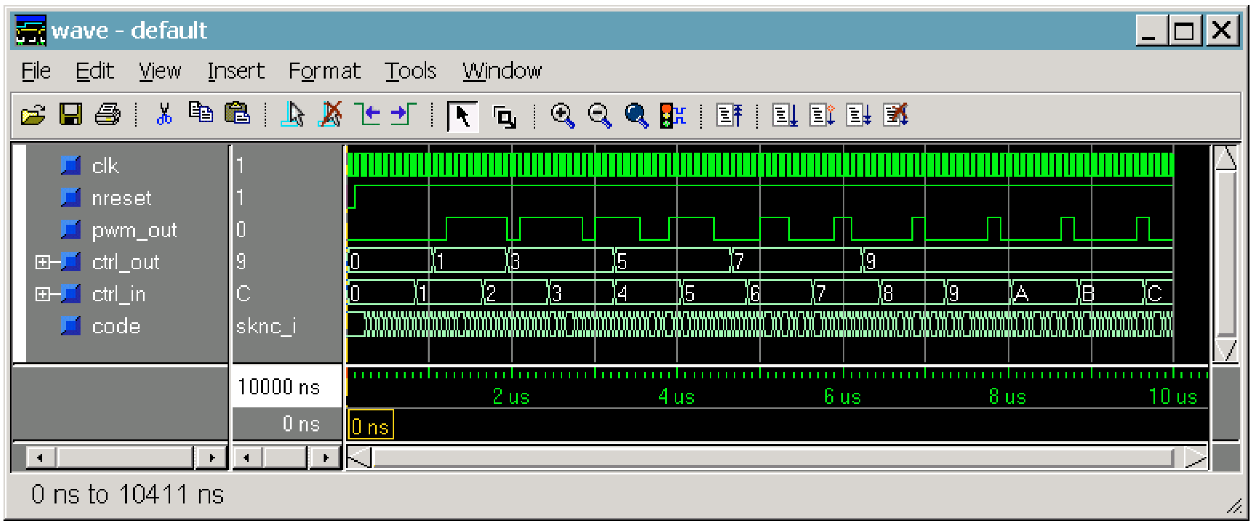
Istruction page 0 = polling loop
Istruction page 1 = registers initialization
Istruction page 2 = PWM setting

Now click on "ASM" to see the source code, "HW Build" to generate the VHDL CPU core and "SW Build" to generate Ram /Rom files. The "Log" menu shows the software compilation report.

If the result is correct we can start the VHDL simulation. The current testbench shows the PWM behaviour with a digital input ramp.

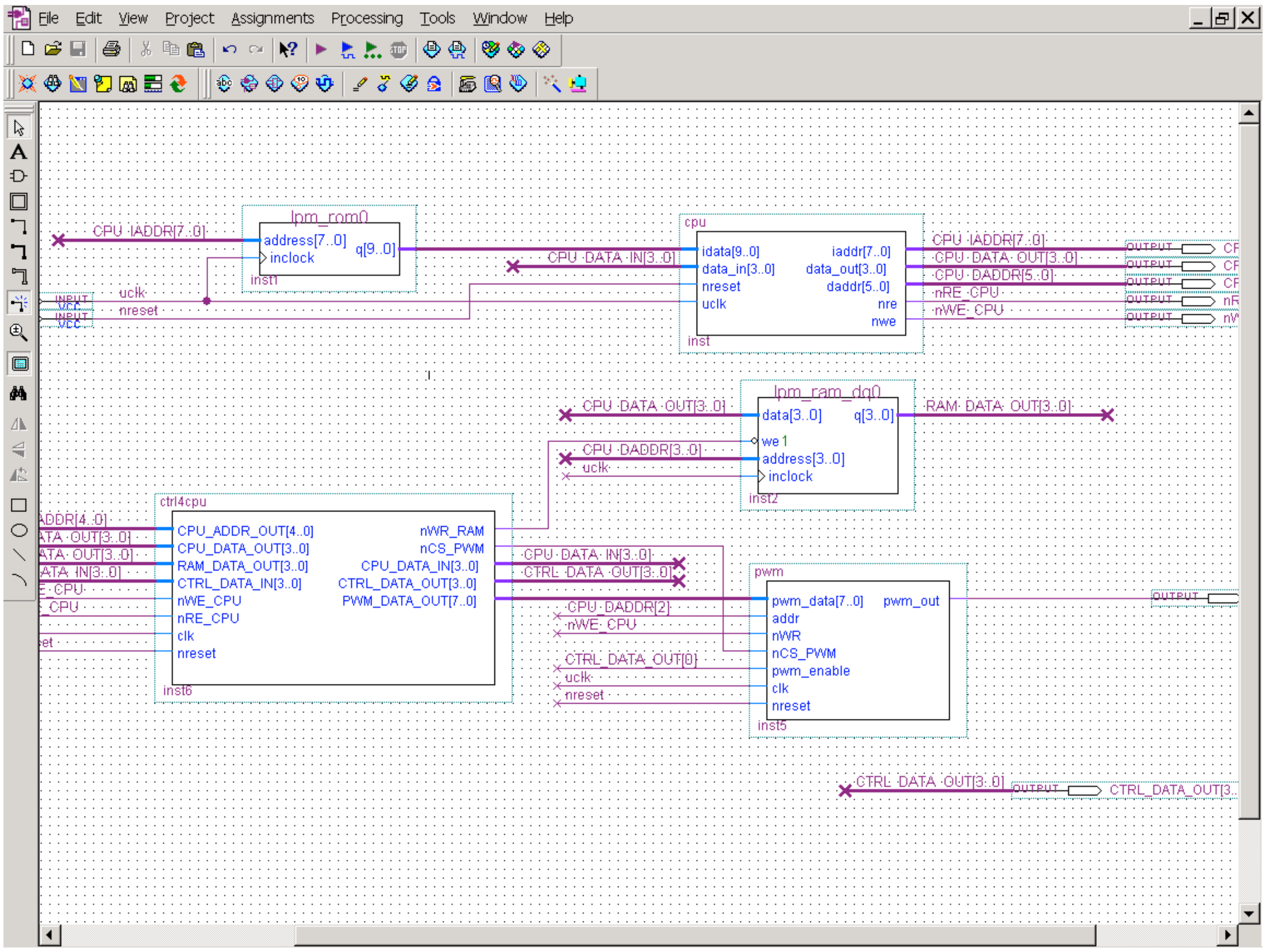
Run ModelSim and:

1) File\Open\Project select the path for cpu4bit\Testbench\testbench_xpre.mpf (or testbench_xpre.mpf)
2) Compile\Compile All
3) Simulate\Simulate: Load work\testbench\struct; Simulator resolution: ns
4) Simulate\Simulate Options: Default Run: 10000 ns
5) View\All windows
6) In the wave window select: File\Load Format\Wave_pre.do
7) Simulate\Run: Run 10 us



Altera Design:

Start cpu4bit.quartus in cpu4bit\Altera directory.

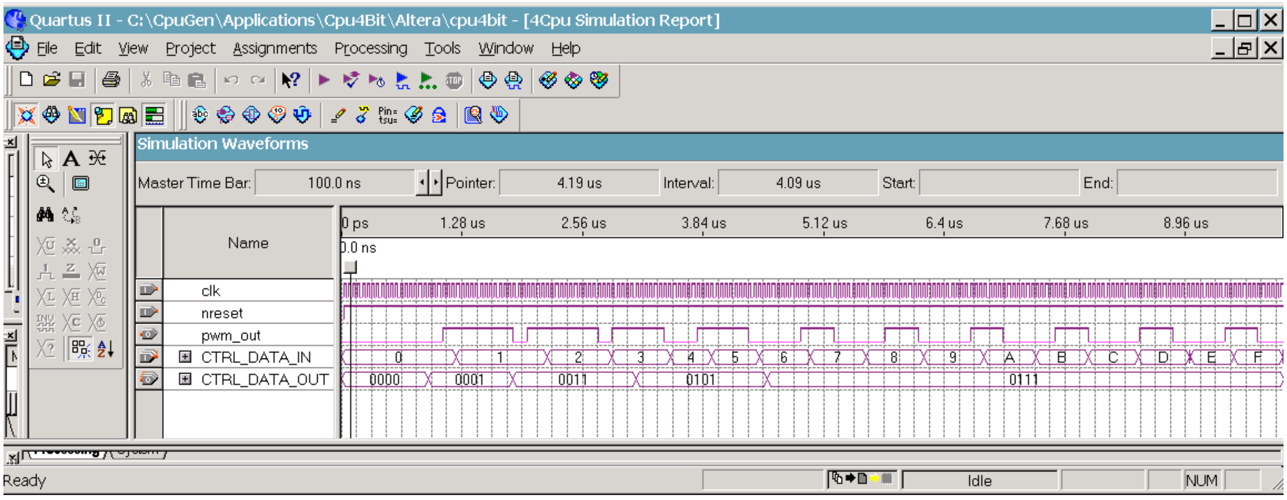


EAB (Embedded Array Blocks) are used for ram (ram.mif) and rom (rom.mif).
In Assignments\Device I choice the ACEX1K device and Auto Selected.
Then with Processing\Start Compilation Quartus synthetize the design.

Output:

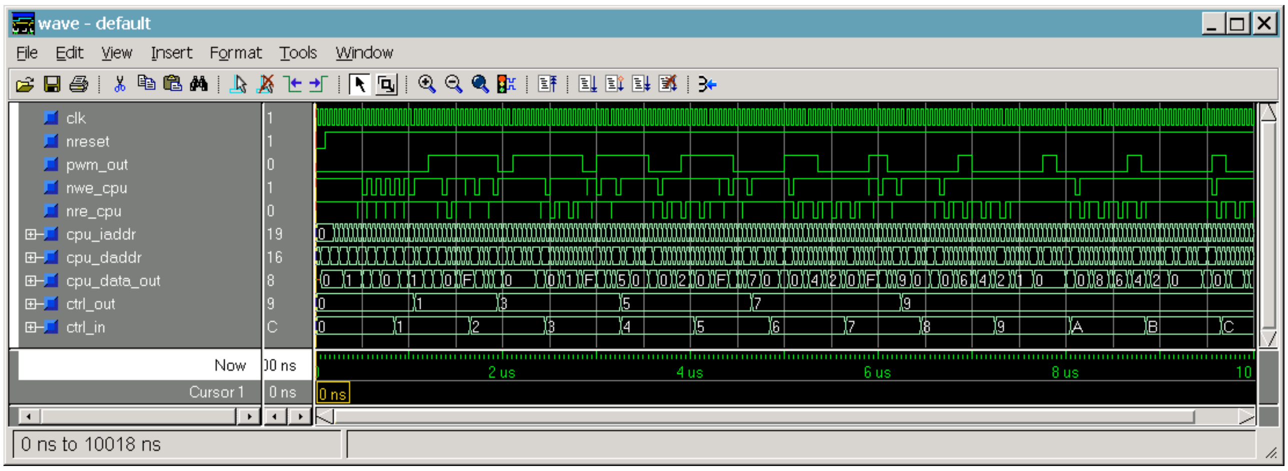
- Device EP1K10TC100-1 = 10K gates (auto select => fastest device)
- 413 / 576 (71%) = Logic Elements
- 2624 / 12288 (21%) = Memory bits
- 45.87 MHz = Max frequency clock

1) Simulate it with Quartus Waveform simulator:


Quartus II - C:\CpuGen\Applications\Cpu4Bit\Altera\cpu4bit - [4Cpu Simulation Report]
File Edit View Project Assignments Processing Tools Window Help
Simulation Waveforms
Master Time Bar: 100.0 ns
Pointer: 4.19 us
Interval: 4.09 us
Start:
End:
Name
0 ps
1.28 us
2.56 us
3.84 us
5.12 us
6.4 us
7.68 us
8.96 us
0.0 ns
clk
nreset
pwm_out
CTRL_DATA_IN
CTRL_DATA_OUT
0 1 2 3 4 5 6 7 8 9 A B C D E F
0000 0001 0011 0101 0111
Ready
Idle
NUM

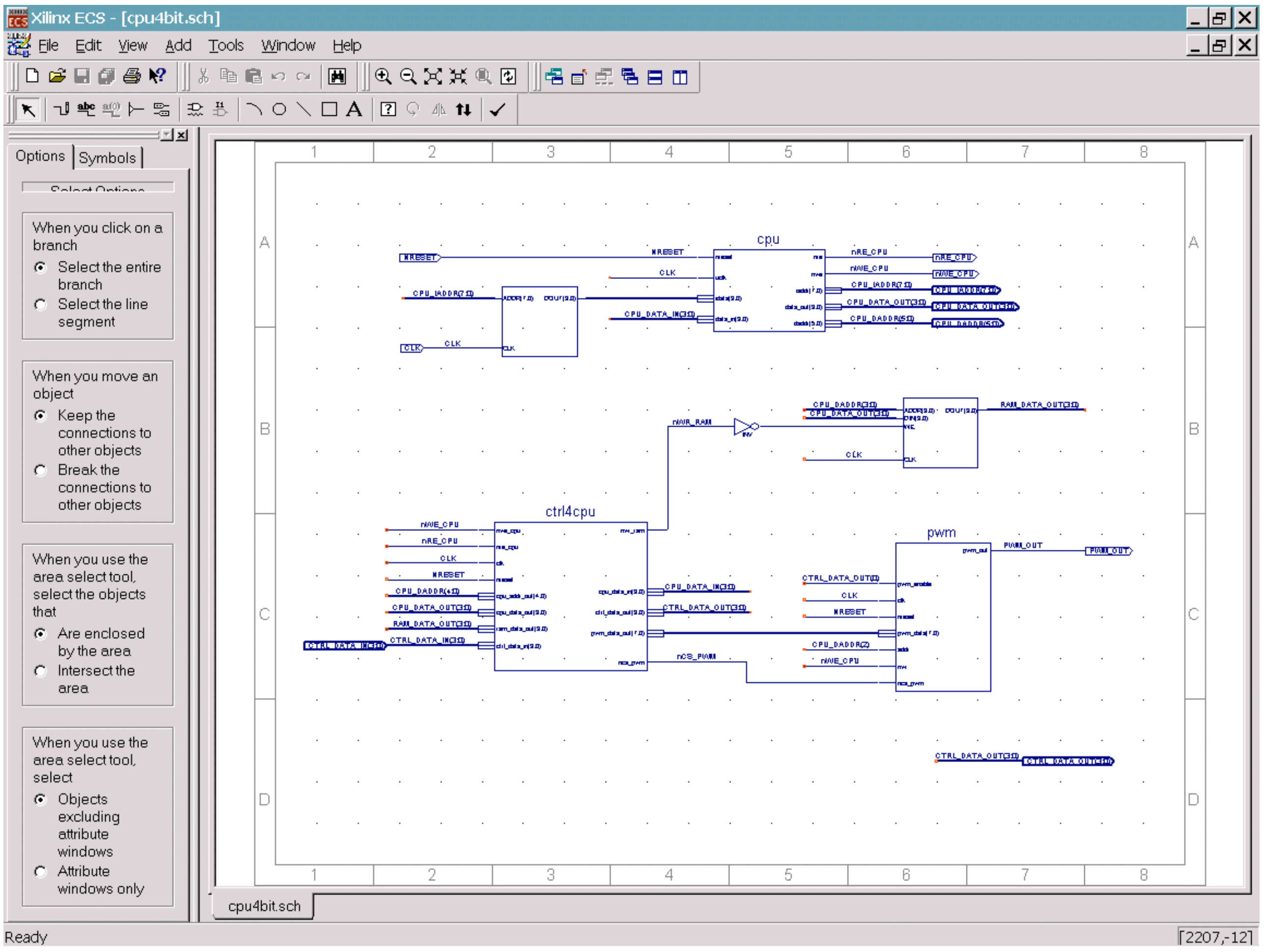
2) Simulate it with Modelsim AE: (cpu4bit.vho = Quartus output)

1) File\Open\Project select the path for cpu4bit\Testbench\testbench_apost.mpf.
2) Compile\Compile All
3) Simulate\Simulate: Load work\testbench\struct; Simulator resolution: ns
4) Simulate\Simulate Options: Default Run: 10000 ns
5) View\All windows
6) In the wave window select: File\Load Format\Wave_post.do
7) Simulate\Run: Run 10 us



Xilinx Design:

Start cpu4bit.npl in cpu4bit\Xilinx directory.


Xilinx ECS - [cpu4bit.sch]

Options | Symbols

Select Options

When you click on a branch
[✓ Select the entire branch]
Select the line segment

When you move an object
[✓ Keep the connections to other objects]
Break the connections to other objects

When you use the area select tool, select the objects that
[✓ Are enclosed by the area]
Intersect the area

When you use the area select tool, select
[✓ Objects excluding attribute windows]
Attribute windows only

cpu4bit.sch

Ready

[2207,-12]

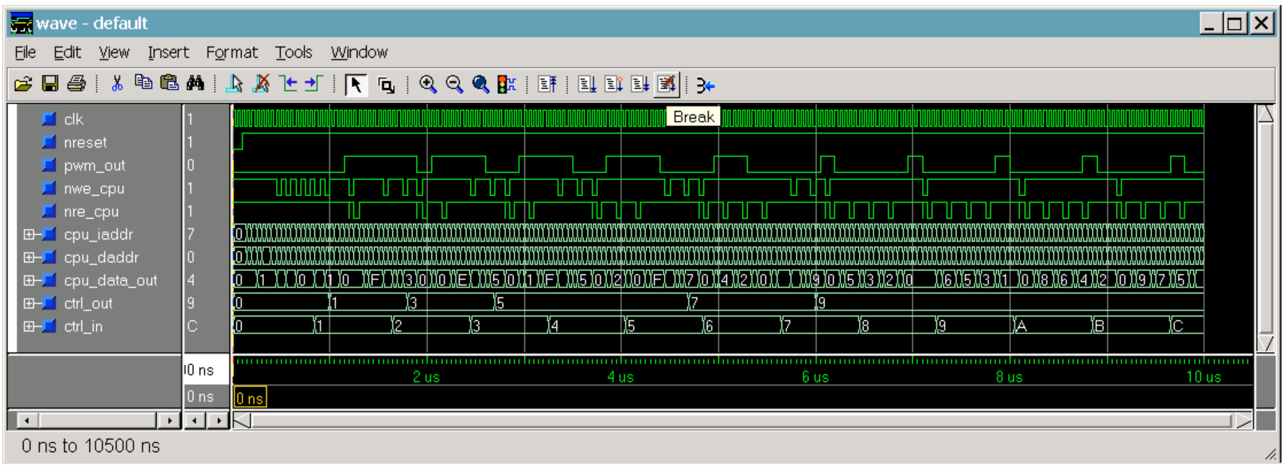
Memory Blocks are used for ram (ram.coe/ram.cfg) and rom (rom.coe/rom.cfg).
NOTE: It’s necessary to update initialization files if you use a different project path directory (rom/ram.xco)
Remeber to Regenerate Core and Update schematic when change hw/sw cpu.
I selected the Spartan2 device: XC2S30-5cs144
Then select cpu4bit and Generate Programming file.

Output:

-	Total Number Slice Registers:	119 out of	864	13%
-	Number of 4 input LUTs:	489 out of	864	56%
-	Number of occupied Slices:	271 out of	432	62%
-	Clock Period 22.069ns:	Max Frequency = 45.312 MHz		

1) Simulate it with Modelsim XE: (cpu4bit_translate.vhd = Xilinx output)

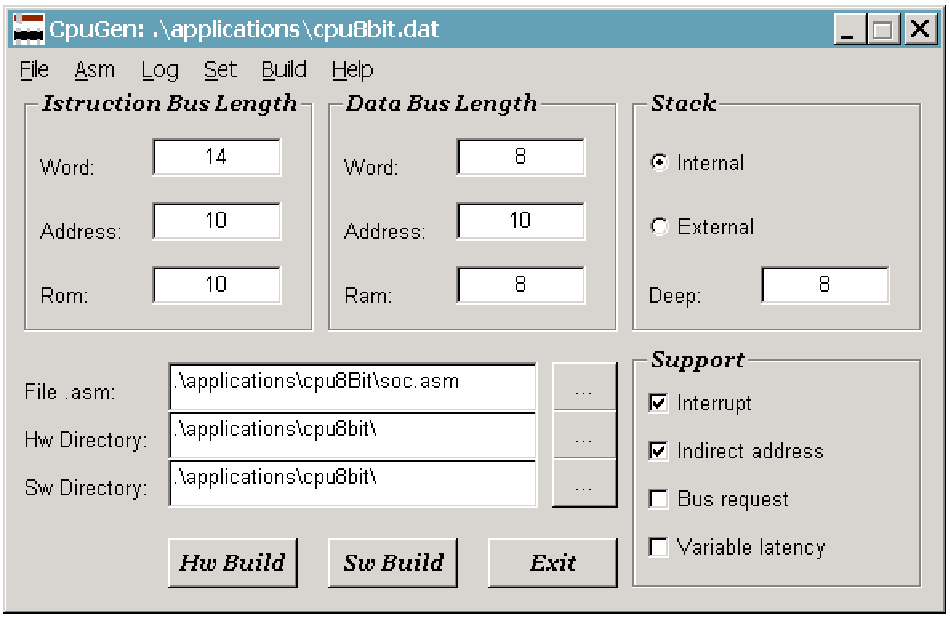
1) File\Open\Project select the path for cpu4bit\Testbench\testbench_xpost.mpf.
2) Compile\Compile All
3) Simulate\Simulate: Load work\testbench\struct; Simulator resolution: ns
4) Simulate\Simulate Options: Default Run: 10000 ns
5) View\All windows
6) In the wave window select: File\Load Format\Wave_post.do
7) Simulate\Run: Run 10 us



Cpu8Bit

Goal:
- check 4 digital input as fast as possible (4 interrupt sources)
- manage 8 bit ports (4 In/Out ports configurable => 32 I/O)
- manage a Timer (1 interrupt source)
- send/receive data via UART (3 interrupt source)

Run **cpugen** and load Applications\cpu8bit.dat cpu configuration:



Istruction Bus Length
Word: 14
Address: 10
Rom: 10

Data Bus Length
Word: 8
Address: 10
Ram: 8

Stack
(● Internal) ○ External
Deep: 8

File .asm: .\applications\cpu8Bit\soc.asm
Hw Directory: .\applications\cpu8bit\
Sw Directory: .\applications\cpu8bit\

Support
☑ Interrupt
☑ Indirect address
☐ Bus request
☐ Variable latency

RTL design:

VHDL sources:

1) CPU

- cpu_utils.vhd
- cpu_iu.vhd
- cpu_cu.vhd
- cpu_du.vhd
- cpu_oa.vhd
- cpu.vhd

2) COMPONENTS

- Ctrl8cpu.vhd
- Inout4reg.vhd
- Interrupt.vhd
- Timer.vhd
- Tx_uart.vhd
- Rx_uart.vhd
- Utils_pkg.vhd
- ram.vhd (just for RTL testbench; use ram.vin)
- rom.vhd (just for RTL testbench; use rom.vin)

3) TESTBENCH (RTL)

- cpu8bit.vhd
- clock.vhd
- world.vhd
- testbench_a.vhd

CPU 8 bit => Data Bus Length Word = 8
I use a 14 bit Istruction Words Length, so I have a 10 bit (14 – 4) istruction/data addressing capability => 1K words of code => Istruction address and ROM = 8 => no istruction/data pages needed.

Resources:

- 256 byte RAM (8)
- control register
- interrupt controller
- timer
- tx Uart
- rx Uart
- 4 ports I/O registers

Address mapping (8 bit size):

1.	RAM	0 ... 0xFF
2.	CTRL_ADDR	0x100
3.	INT_ADDR	0x104
4.	REG_ADDR	0x108
5.	TIMER_ADDR	0x10C
6.	UART_ADDR	0x110

2) CTRL_ADDR outputs bits:

- [3..0] = I/O ports direction (DIR = ‘1’ => output port)
- [4] = timer enable (‘1’ => enable)
- [5] = timer reset (‘1’ => reset)
- [6] = timer clear (‘1’ => clear count)

3) INT_ADDR:

- MASK/SOURCE [addr = 0x104]
- CLEAR[addr = 0x105]

Bits:

- [3..0] = External interrupt input
- [4] = OVERRUN UART interrupt
- [5] = RX UART interrupt
- [6] = TX UART interrupt
- [7] = TIMER interrupt

4)REG_ADDR:

- PORTA [addr = 0x108]
- PORTB [addr = 0x109]
- PORTC [addr = 0x10A]
- PORTD [addr = 0x10B]

4) TIMER_ADDR (input = current count, output = most significative byte counter):

- 8 bit prescaler

5) UART_ADDR

- TX/RX DATA REGISTER [addr = 0x110]
- BAUD RATE REGISTER [addr = 0x111]

I need interrupt and indirect addressing support.
I set a 8 deep stack (see Help/CpuCore: How to set stack deep)

Code structure:

1) Initialize parameters
2) Main Loop: indirect addressing operations.
3) ISR: check all interrupt sources (with priority order) and clear them.

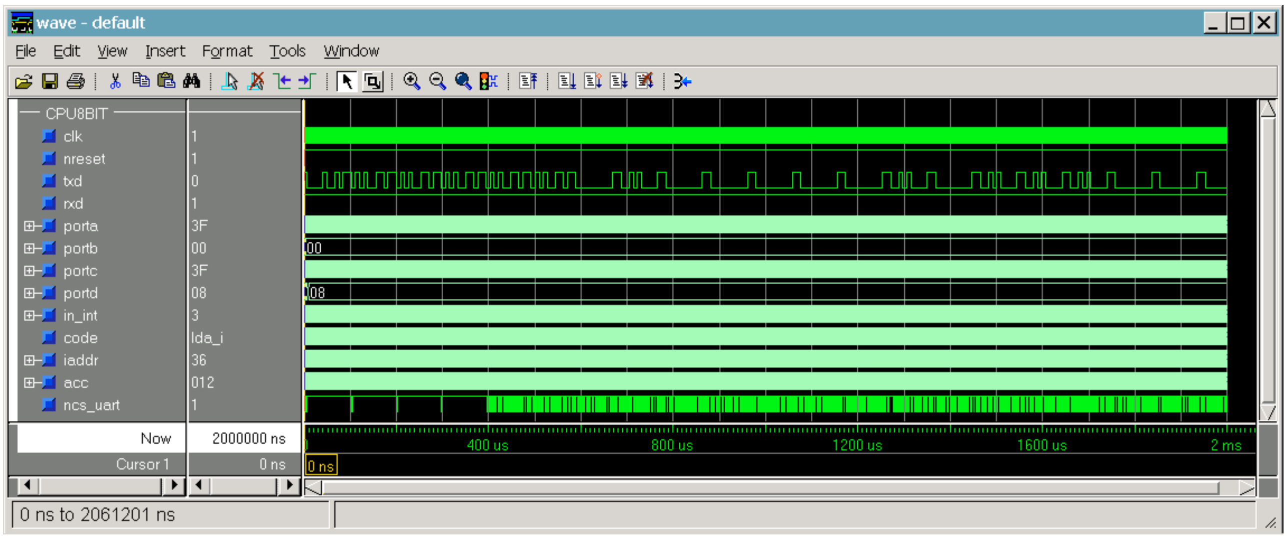
NOTE: The ISR address = num. of istruction address – 8 => (10 bit istruction address) = > 0x400h – 8 = 0x3f8h

Now click on on “ASM” to see the source code, “HW Build” to generate the VHDL CPU core and “SW Build” to generate Ram /Rom files. The “Log” menu shows the software compilation report.

If the result is correct we can start the VHDL simulation. The current testbench shows the signals behaviour with a test program.

Run ModelSim and:

1) File\Open\Project select the path for cpu8bit\Testbench\testbench_apre.mpf (or testbench_xpre.mpf)
2) Compile\Compile All (MACRO: compile_apre.do/compile_xpre.do)
3) Simulate\Simulate: Load work\testbench\struct; Simulator resolution: ns
4) Simulate\Simulate Options: Default Run: 2 ms
5) View\All windows
6) In the wave window select: File\Load Format\Wave_pre.do
7) Simulate\Run: Run 2 ms (MACRO: sim_pre.do)

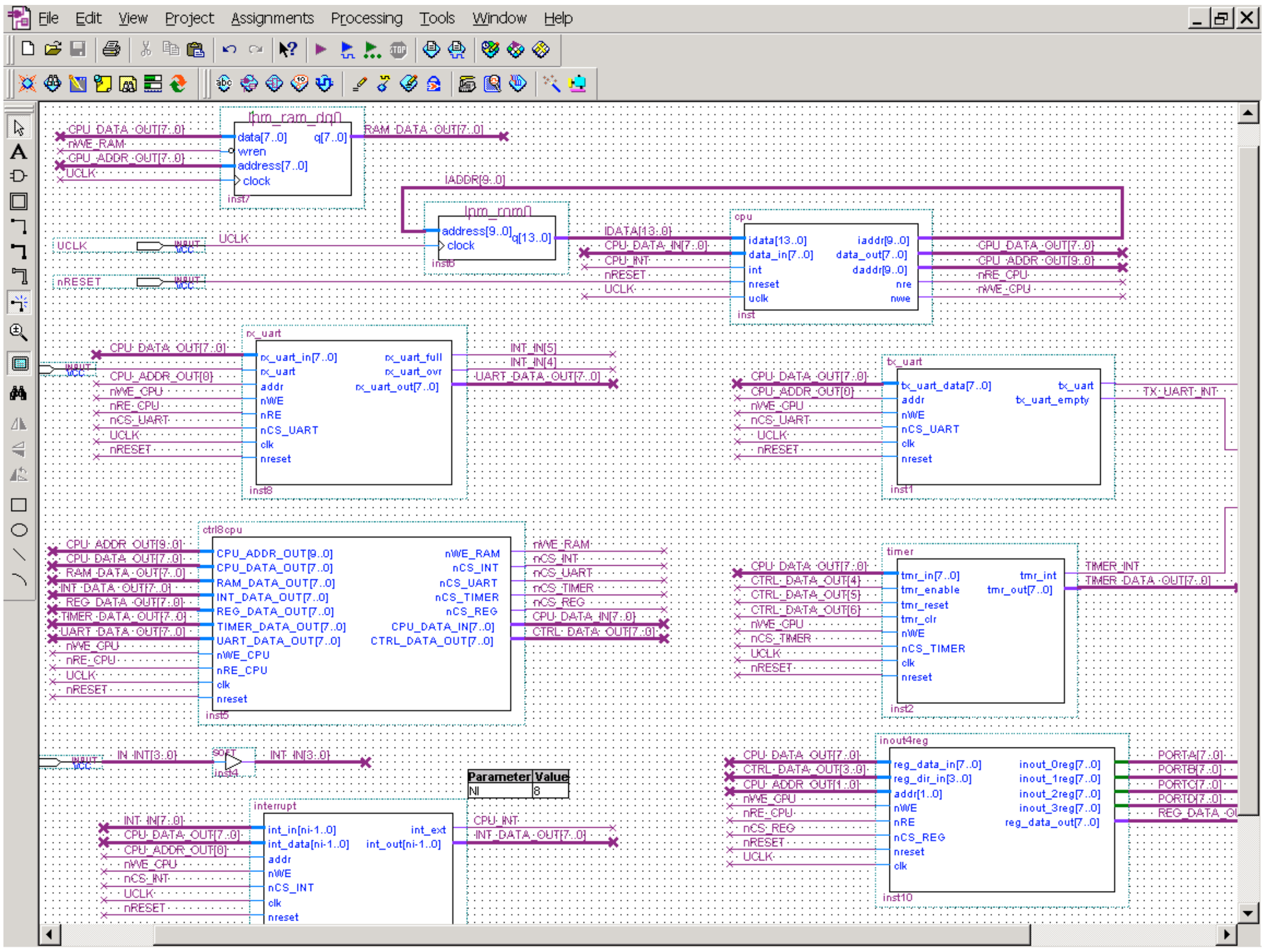


clk	1
nreset	1
txd	0
rxd	1
porta	3F
portb	00
portc	3F
portd	08
in_int	3
code	lda_i
iaddr	36
acc	012
ncs_uart	1
Now	2000000 ns
Cursor 1	0 ns

0 ns to 2061201 ns

Altera Design:

Start cpu8bit.quartus in cpu8bit\Altera directory.

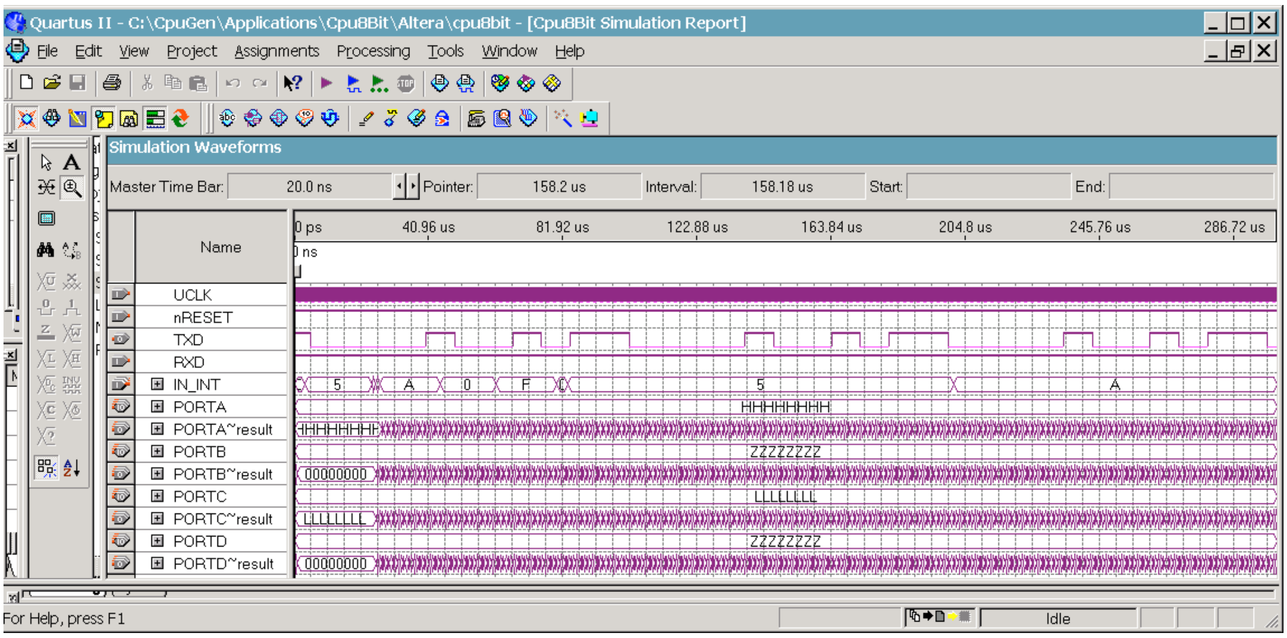


EAB (Embedded Array Blocks) are used for ram (ram.mif) and rom (rom.mif).
NOTE: both ram e rom EAB have to be created with single clock and q output port not registered.
In Assignments\Device I choice the CYCLONE device EP1C3T100C8.
Then with Processing\Start Compilation Quartus synthetize the design.

Output:

- 1611 / 2910 (55%) = Logic Elements
- 16384 / 59904 (27%) = Memory bits
- 33 MHz = Max frequency clock

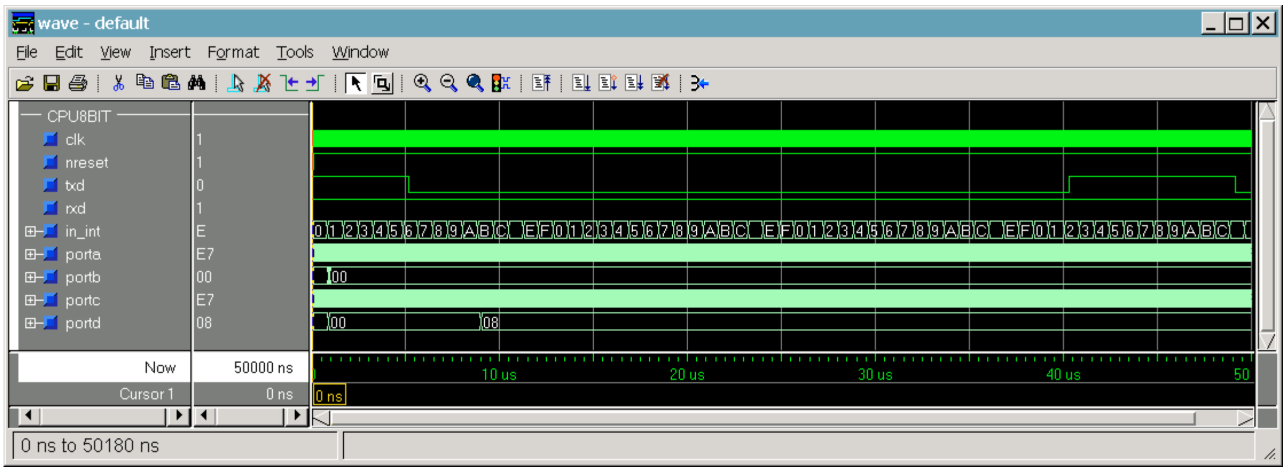
1) Simulate it with Quartus Waveform simulator:



2) Simulate it with Modelsim AE: (cpu8bit.vho = Quartus output)

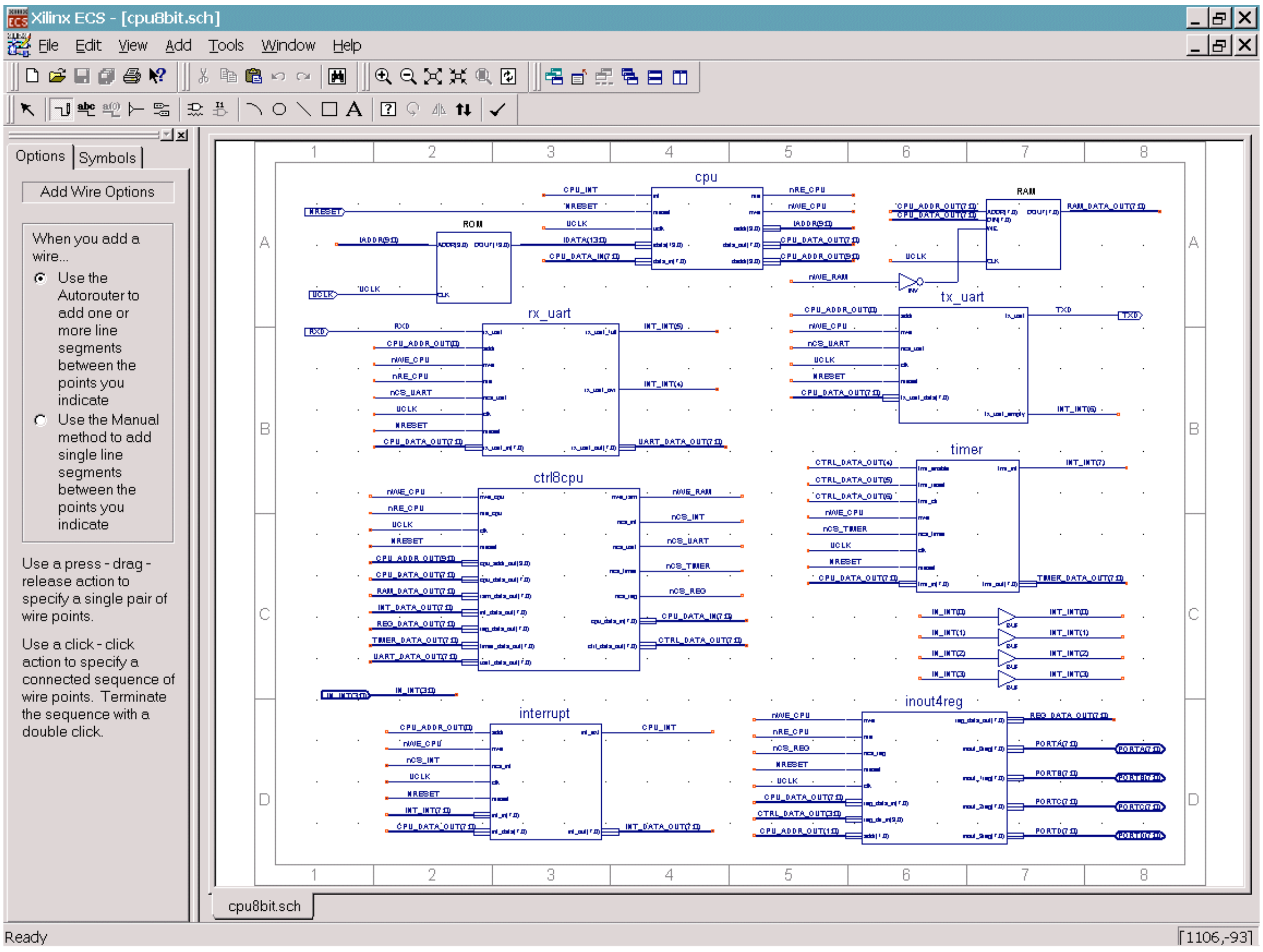
1) File\Open\Project select the path for cpu8bit\Testbench\testbench_apost.mpf.
2) Compile\Compile All (MACRO: compile_apost.do)
3) Simulate\Simulate: Load work\testbench\struct; Simulator resolution: ns
4) Simulate\Simulate Options: Default Run: 10000 ns
5) View\All windows
6) In the wave window select: File\Load Format\Wave_post.do

Simulate\Run: Run 100 us (MACRO: sim_post.do)



Xilinx Design:

Start cpu8bit.npl in cpu8bit\Xilinx directory.



When you add a wire...

[✓ Use the Autorouter to add one or more line segments between the points you indicate]

Use the Manual method to add single line segments between the points you indicate

Use a press - drag - release action to specify a single pair of wire points.

Use a click - click action to specify a connected sequence of wire points. Terminate the sequence with a double click.

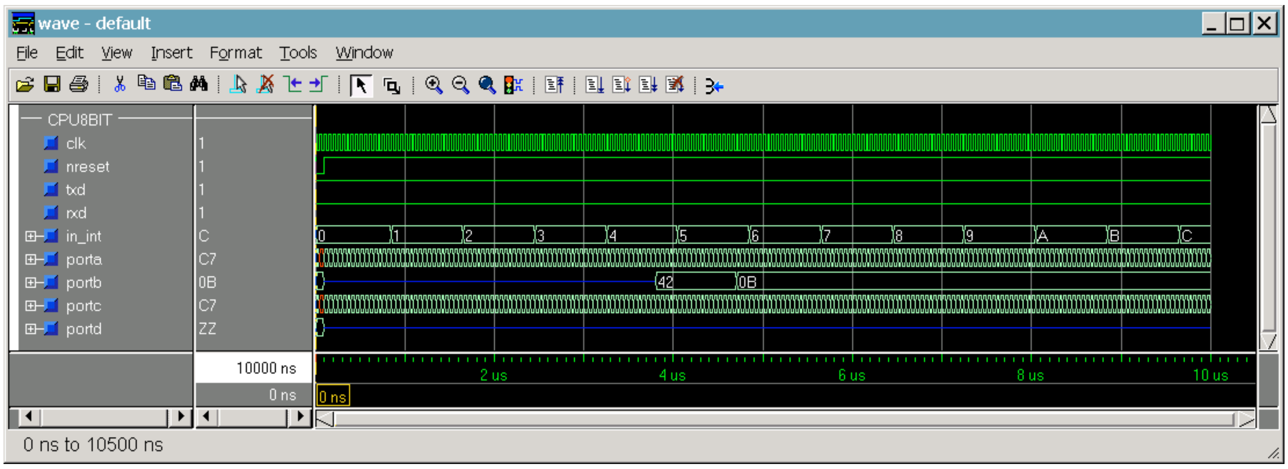
Memory Blocks are used for ram (ram.coe/ram.cfg) and rom (rom.coe/rom.cfg).
NOTE: It’s necessary to update initialization files if you use a different project path directory (rom/ram.xco)
Remeber to Regenerate Core and Update schematic when change hw/sw cpu.
I selected the Spartan2 device: XC2S50e-7tq144
Then select cpu8bit and Generate Programming file.

Output:

- Total Number Slice Registers:	537 out of 1,536 34%
- Total Number 4 input LUTs:	1,180 out of 1,536 76%
- Number of occupied Slices:	732 out of 768 95%
- Clock Period 30.705 ns:	Max Frequency = 32.568 MHz

1) Simulate it with Modelsim XE: (cpu8bit_timesim.vhd = Xilinx output)

1) File\Open\Project select the path for cpu8bit\Testbench\testbench_xpost.mpf.
2) Compile\Compile All
3) Simulate\Simulate: Load work\testbench\struct; Simulator resolution: ns
4) Simulate\Simulate Options: Default Run: 10000 ns
5) View\All windows
6) In the wave window select: File\Load Format\Wave_post.do
7) Simulate\Run: Run 40 us



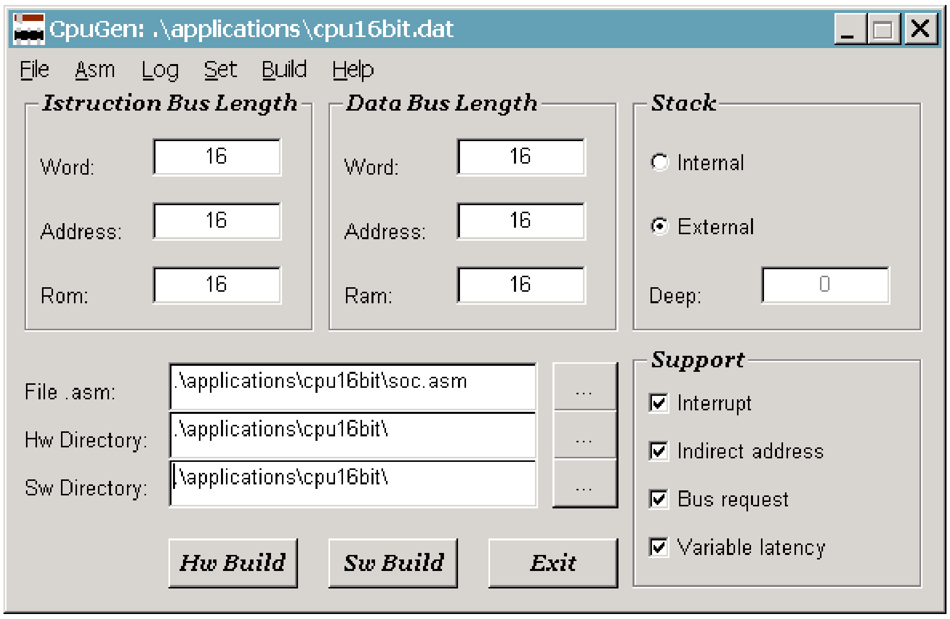
Cpu16Bit

Goal:

- use a 16 bit data/istruction bus
- use just one external asynchronous memory for code and data (Hardware to Von Neuman Architecture)
- manage interrupts, variable latency data access, multi-master bus access
- implement the stack with vendor specific memory blocks

NOTE: bus request and variable latency support are still in alpha stage.

Run **cpugen** and load Applications\cpu16bit.dat cpu configuration:



CpuGen: .\applications\cpu16bit.dat

Istruction Bus Length
Word: 16
Address: 16
Rom: 16

Data Bus Length
Word: 16
Address: 16
Ram: 16

Stack
Internal [✓ External]
Deep: 0

File .asm: .\applications\cpu16bit\soc.asm
Hw Directory: .\applications\cpu16bit\
Sw Directory: .\applications\cpu16bit\

Support
☑ Interrupt
☑ Indirect address
☑ Bus request
☑ Variable latency

RTL design:

VHDL sources:

1) CPU

- cpu_utils.vhd
- cpu_iu.vhd
- cpu_cu.vhd
- cpu_du.vhd
- cpu_oa.vhd
- cpu.vhd

2) COMPONENTS

- Ctrl16cpu.vhd
- fifo.vhd
- utils_pkg.vhd
- aram.vhd (just for RTL testbench; use ram.vin)
- arom.vhd (just for RTL testbench; use rom.vin)

3) TESTBENCH (RTL)

- cpu16bit.vhd
- clock.vhd
- world.vhd
- testbench_a.vhd

CPU 16 bit => Data Bus Length Word = 16
Now if I use a 16 bit Istruction Words, I have a 12 bit (16 – 4) istruction page size => 16 data/istruction pages of 4K words each and a data expansion register (4 bit needed).

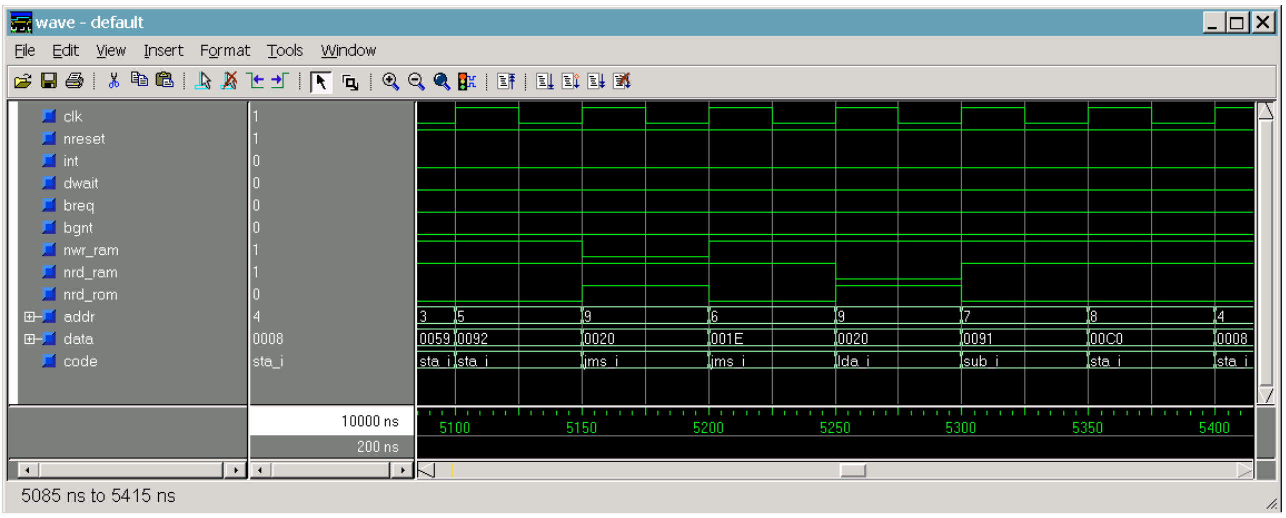
I have 16 word of RAM/ROM and decide to use 32K word RAM and 32K word ROM.
I insert a 256 word fifo as stack.

Now click on on “ASM” to see the source code, “HW Build” to generate the VHDL CPU core and “SW Build” to generate Ram /Rom files. The “Log” menu shows the software compilation report.

If the result is correct we can start the VHDL simulation. The current testbench shows the signals behaviour with a test program.

Run ModelSim and:

1) File\Open\Project select the path for cpu16bit\Testbench\testbench_apre.mpf (or testbench_xpre.mpf)
2) Compile\Compile All
3) Simulate\Simulate: Load work\testbench\struct; Simulator resolution: ns
4) Simulate\Simulate Options: Default Run: 10000 ns
5) View\All windows
6) In the wave window select: File\Load Format\Wave_pre.do
7) Simulate\Run: Run 10 us

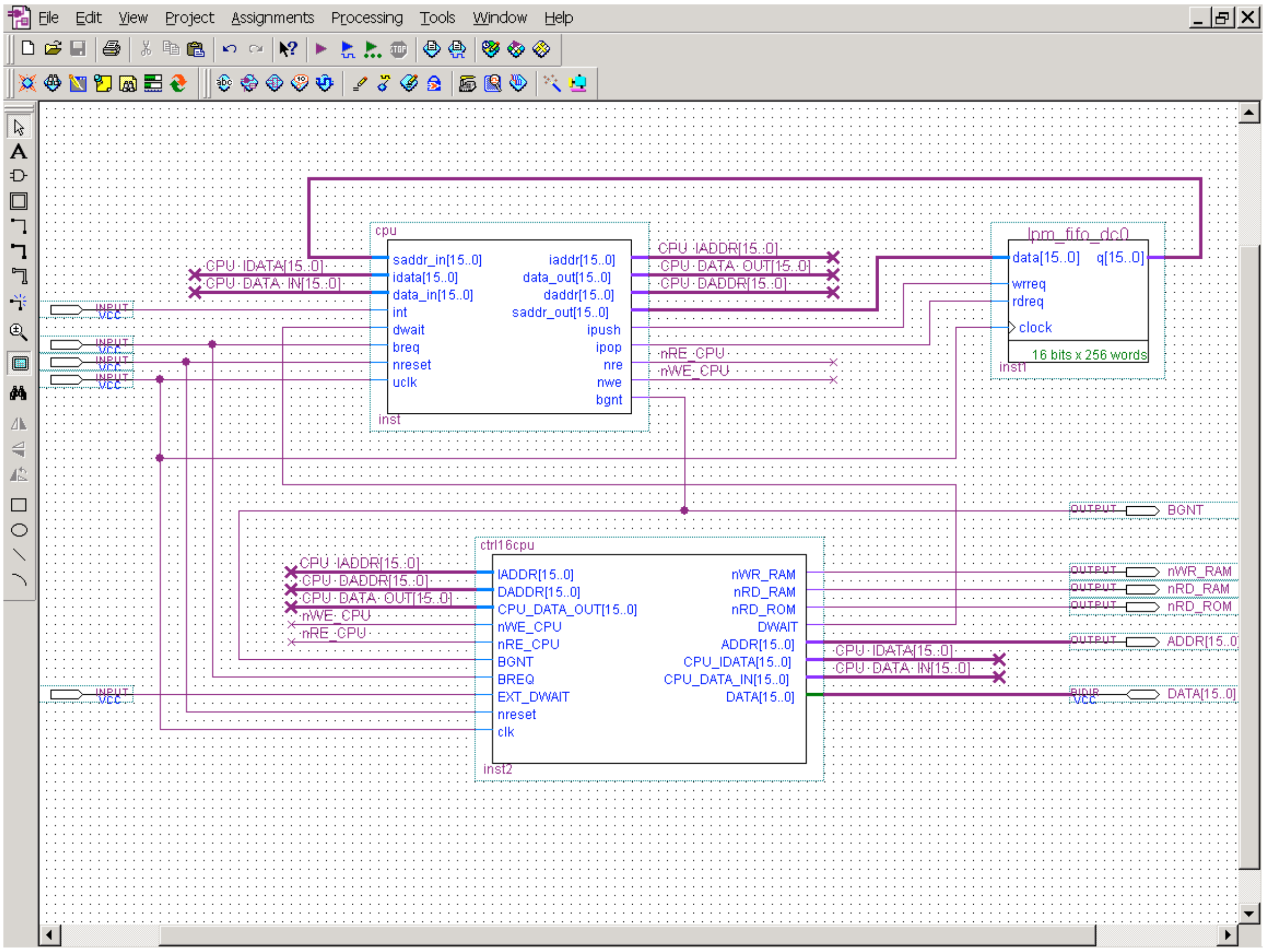

wave - default
File Edit View Insert Format Tools Window
5100
5150
5200
5250
5300
5350
5400
3 5 9 6 9 7 8 4
0059 0092 0020 001E 0020 0091 00C0 0008
sta_i sta_i jms_i jms_i lda_i sub_i sta_i sta_i

clk	1
nreset	1
int	0
dwait	0
breq	0
bgnt	0
nwr_ram	1
nrd_ram	1
nrd_rom	0
addr	4
data	0008
code	sta_i

10000 ns
200 ns
5085 ns to 5415 ns

Altera Design:

Start cpu16bit.quartus in cpu16bit\Altera directory.



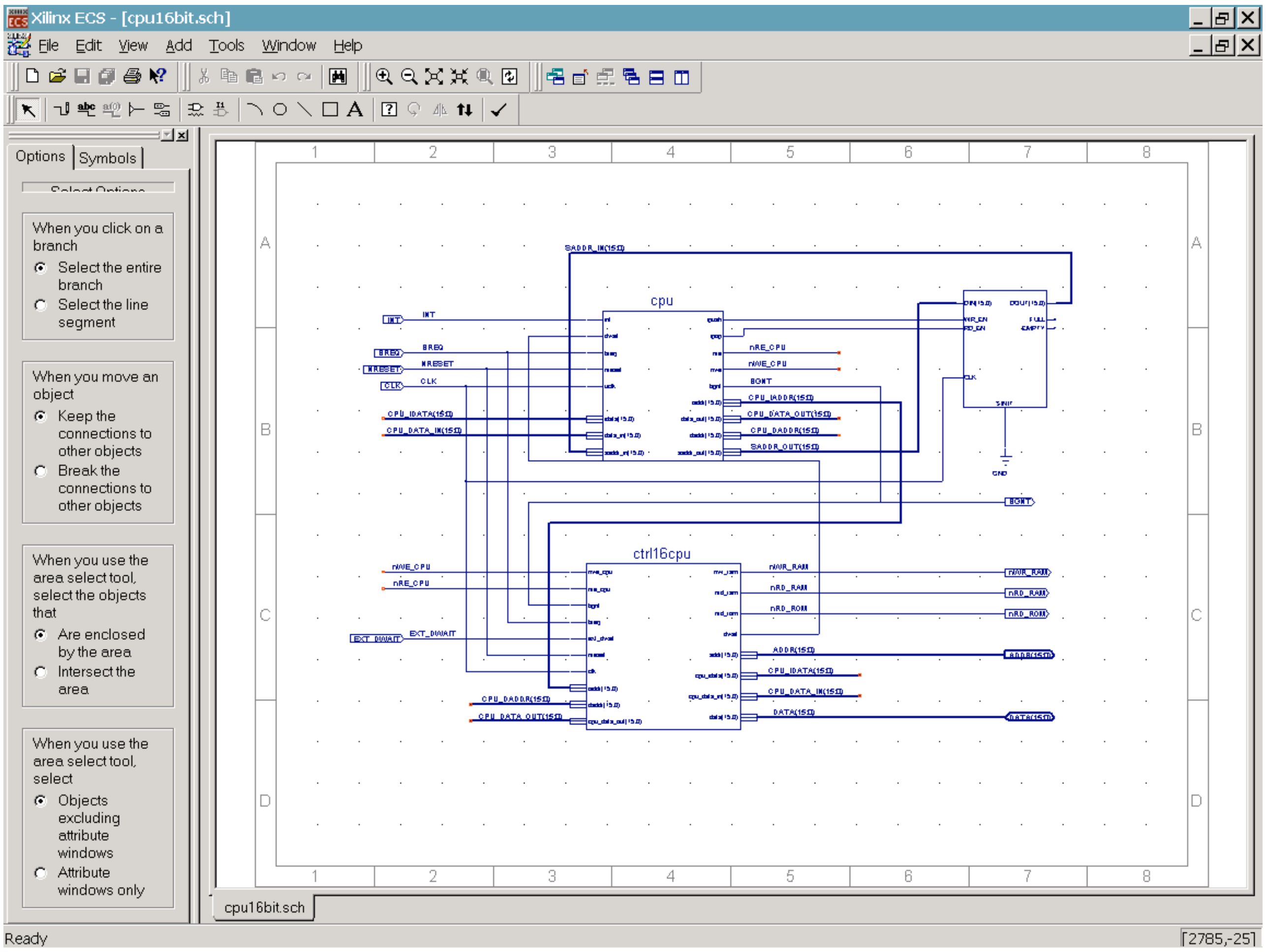
EAB (Embedded Array Blocks) are used for ram (ram.mif) and rom (rom.mif).
Remeber to Regenerate Core and Update schematic when change hw/sw cpu.
In Assignments\Device I choice the CYCLONE device EP1C3T100C8.
Then with Processing\Start Compilation Quartus synthetize the design.

Output:

- 1039 / 2910 (35%) = Logic Elements
- 4096 / 59904 (6%) = Memory bits
- 25.8 MHz = Max frequency clock

Xilinx Design:

Start cpu16bit.npl in cpu16bit\Xilinx directory.



Select Options

When you click on a branch
[✓ Select the entire branch]
Select the line segment

When you move an object
[✓ Keep the connections to other objects]
Break the connections to other objects

When you use the area select tool, select the objects that
[✓ Are enclosed by the area]
Intersect the area

When you use the area select tool, select
[✓ Objects excluding attribute windows]
Attribute windows only

Ready [2785,-25]

Memory Blocks are used for ram (ram.coe/ram.cfg) and rom (rom.coe/rom.cfg).
NOTE: It's necessary to update initialization files if you use a different project path directory (rom/ram.xco)
I selected the Spartan2 device: XC2S100e-7tq144
Then select cpu16bit and Generate Programming file.

Output:

- Total Number Slice Registers:	980 out of 1200 81%
- Total Number 4 input LUTs:	1,773 out of 2,400 73%
- Number of occupied Slices:	1,060 out of 1,200 88%
- Clock Period 45.036ns:	Max Frequency = 22.204 MHz

Design flow

Steps for a new design:

1) Define what processor have to do
2) Decide a first cpu architecture
3) Set the cpu parameters
4) Choice the kind of register/peripherals you need; fix its address.
5) Write the Assembler source code
6) Update the cpu parameters (busses size, memory needs, stack, support)
7) Generate the processor and ram/rom files
8) Design the complete circuit, with peripherals and registers
9) Choice the FPGA device you need (size, speed, cost)
10) Compile the design and try to fit it inside
11) Create the testbench
12) Make simulations
13) Load FPGA image inside the target board.

NOTE:

- in this flowchart design you can change the CPU architecture and code (assembler) whenever you want
- Multi task designs can be realized using more than one cpu (a cpu cluster), everyone fitted inside the same FPGA

Bibliography:

1) VHDL - CookBook: P.Ashenden
2) VHDL - Primer: J.Bhasker
3) Digital Design and Modeling with VHDL and Synthesis: K.C.Chang
4) Component Design By Example: Ben Cohen
5) VHDL and AHDL Digital System Implementation: Frank Scarpino
6) ARM system-on-chip architecture (second edition): Steve Furber
7) Computer Architecture, a quantitative approach (third edition): Hennessy & Patterson
8) PIC16/17 microcontroller DataBook
9) C Language: Kernighan & Ritchie

Legal Notice