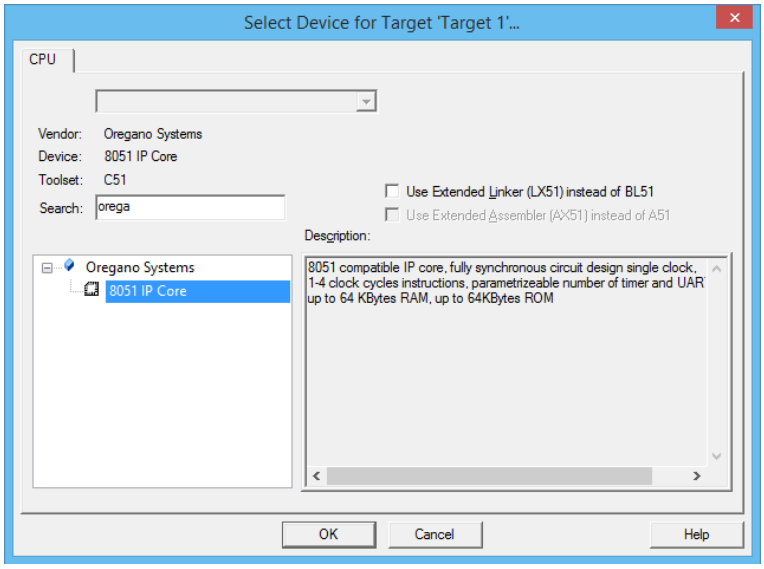
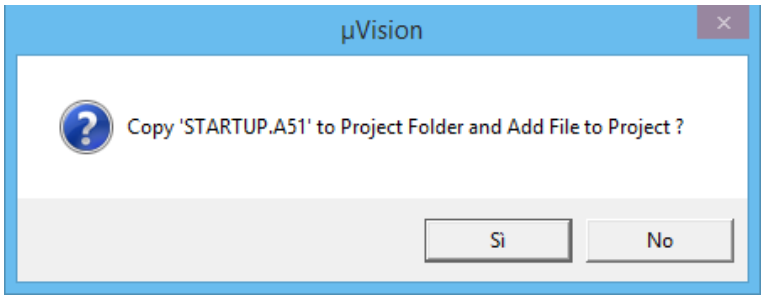
Step 1 - Generate a firmware

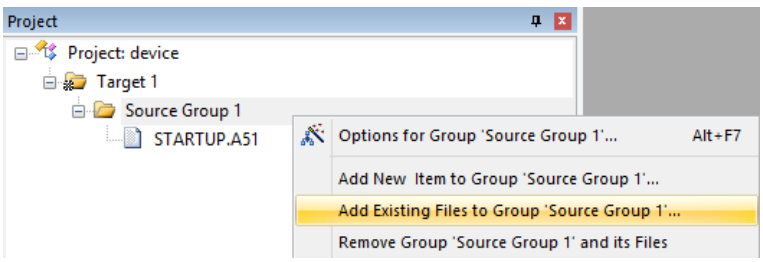
- Download Keil uVision 5 (http://www.keil.com/dd/chip/3509.htm → CA51 Compiler Kit and *REG51.H*)
- Add *REG51.H* to *C:\Keil_v5\C51\INC*
- Keil uVision 5 → New uVision Project… → Oregano Systems (8051 IP Core)

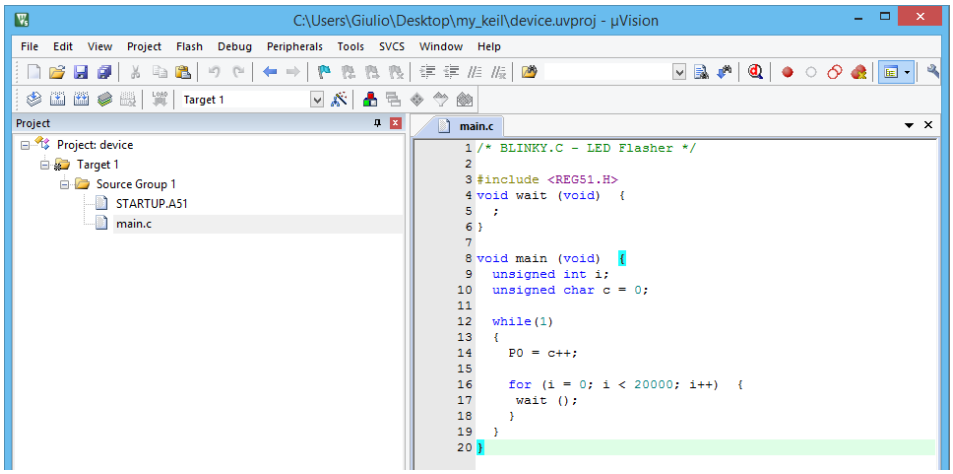


- Answer *Yes*

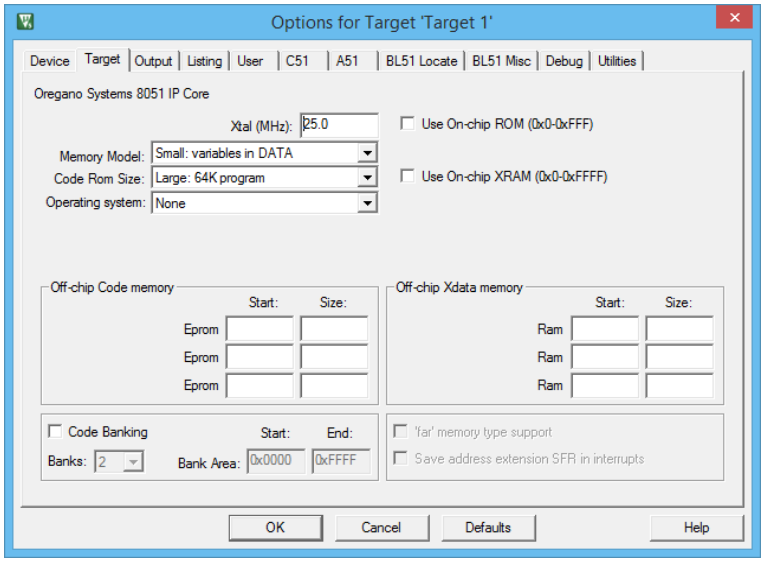


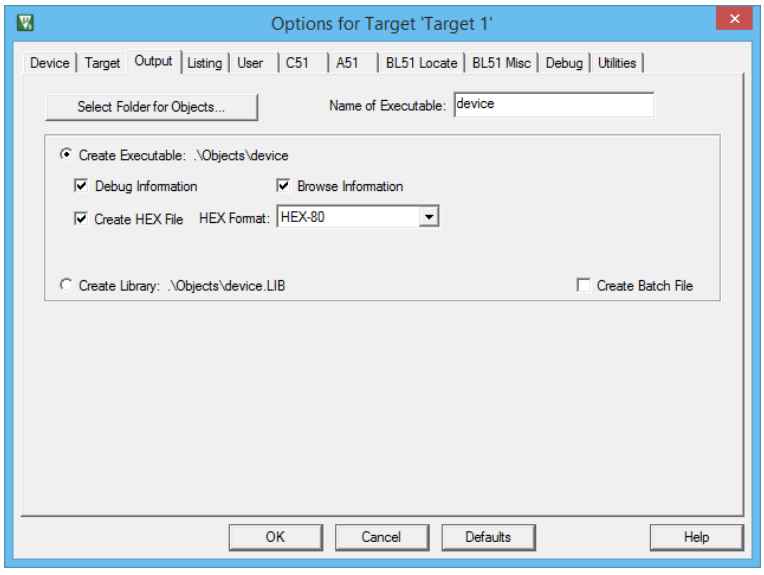
- Create a new file, paste source code, save as *main.c*
- Add *main.c* to project





- Project → Options for Target

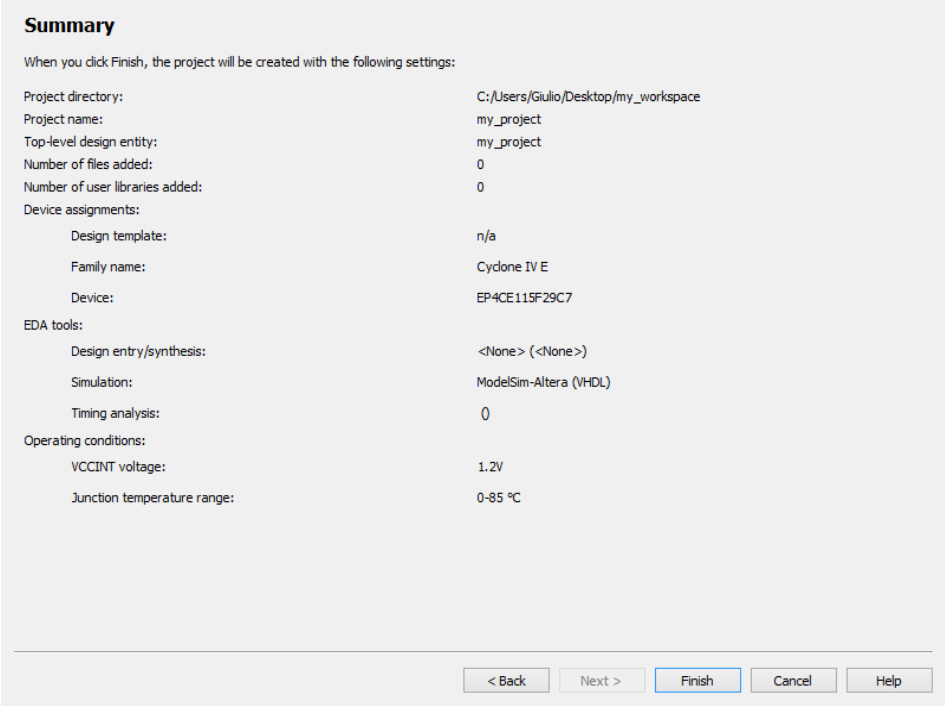




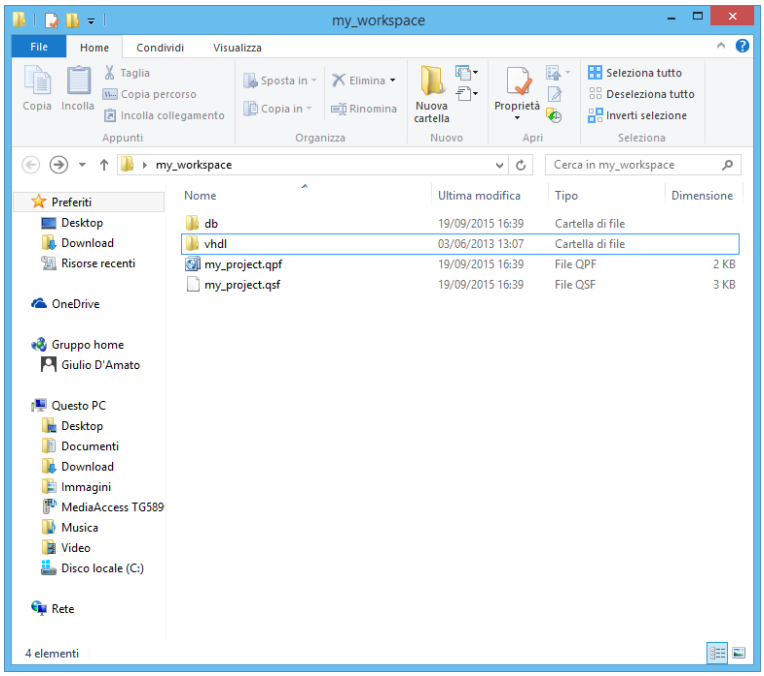
- Build → In *project_directory/Objects* a *project_name.hex* file is created.

Step 2 – Import 8051 IP Core

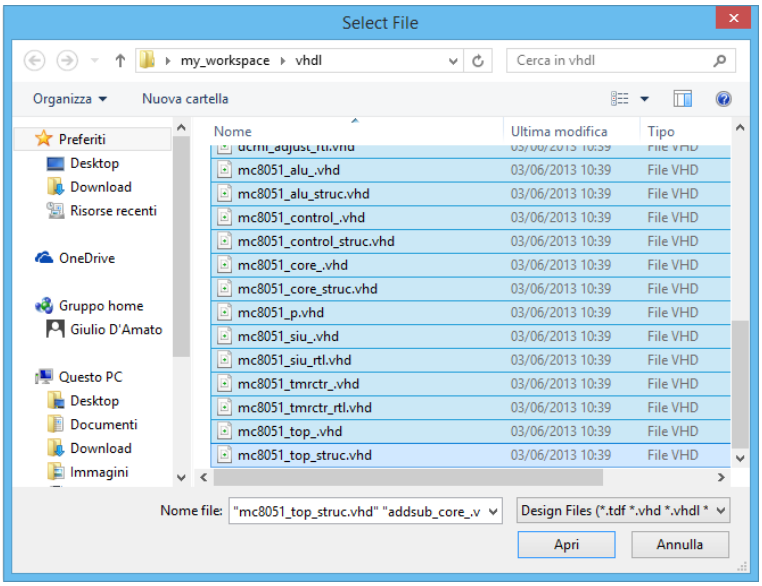
- Download Quartus II 14.1
- Download 8051 IP Core (http://www.oreganosystems.at/ → mc8051_cyclone_nios.zip)
- Quartus II 14.1 → New Project Wizard…

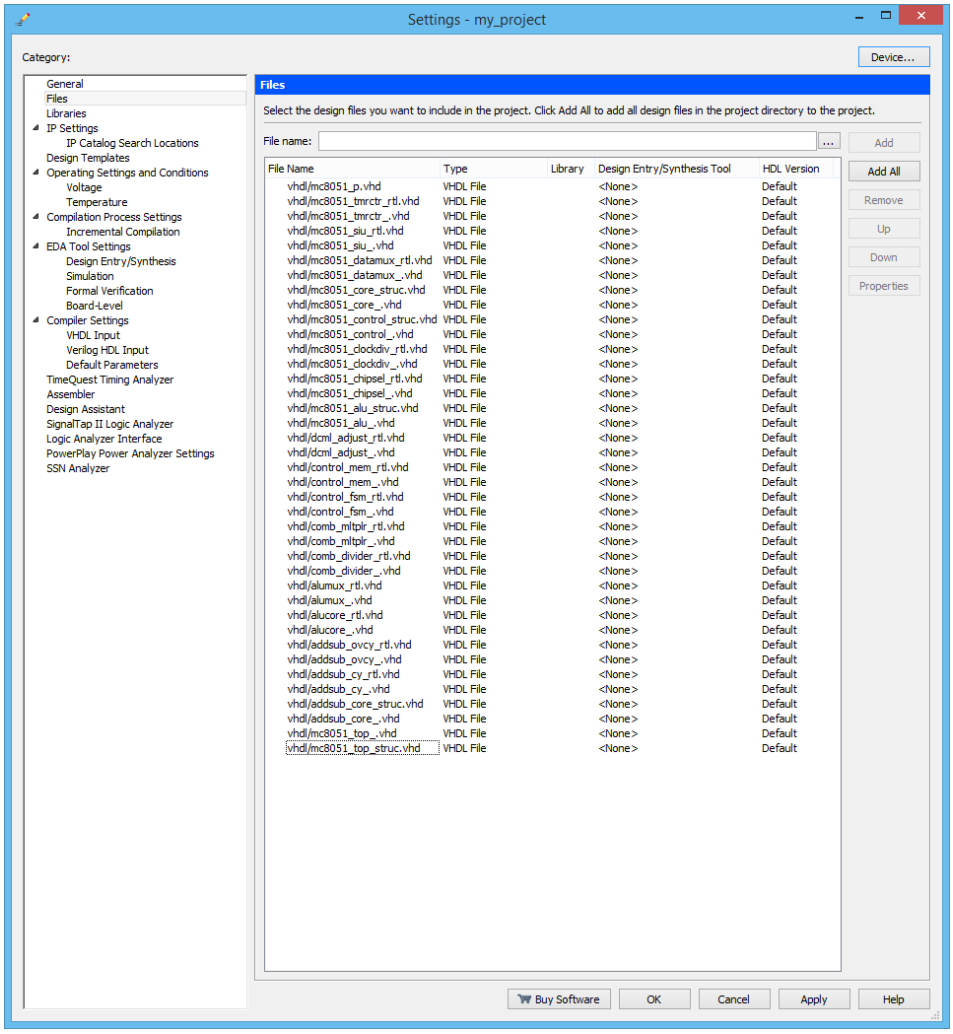


- Copy *vhdl* from mc8051_cyclone_nios.zip to project directory and remove all **_cfg.vhd* files



- Add/Remove Files in Project…





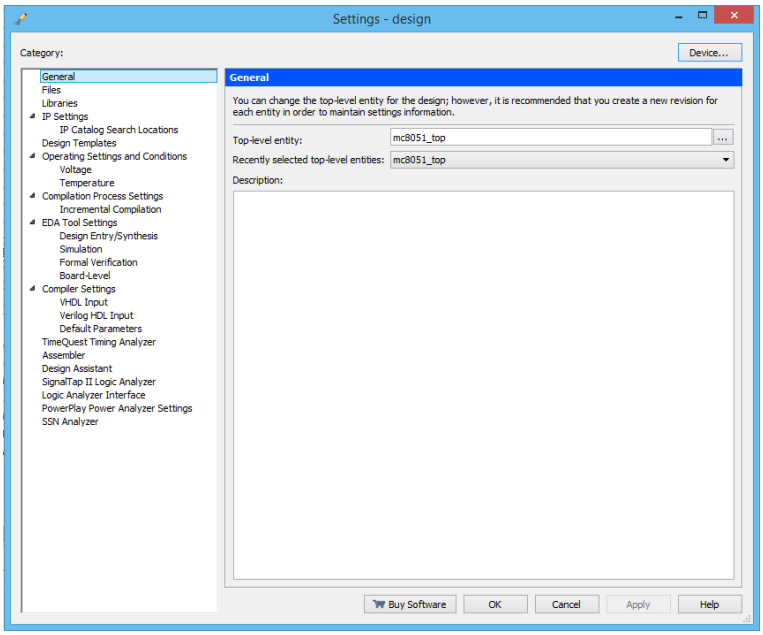
Order is important!

Move up → *vhdl/mc8051_p.vhd*

Move down → *vhdl/mc8051_top_.vhd*

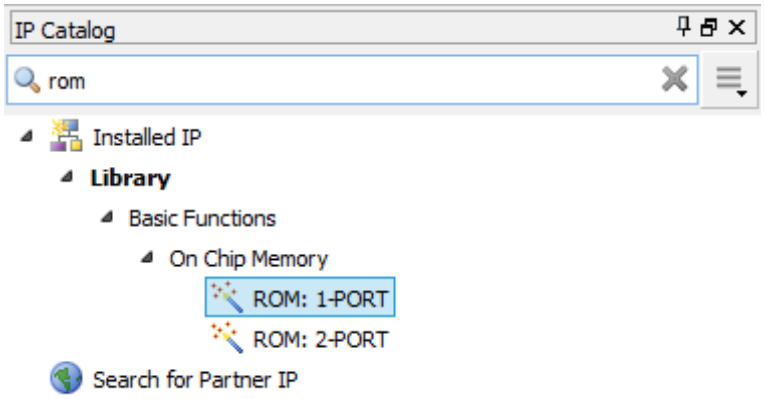
Move down → *vhdl/mc8051_struc.vhd*

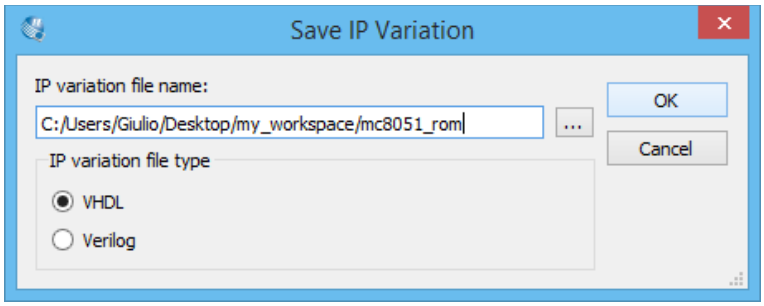
- General → Top-Level Entity → *mc8051_top*

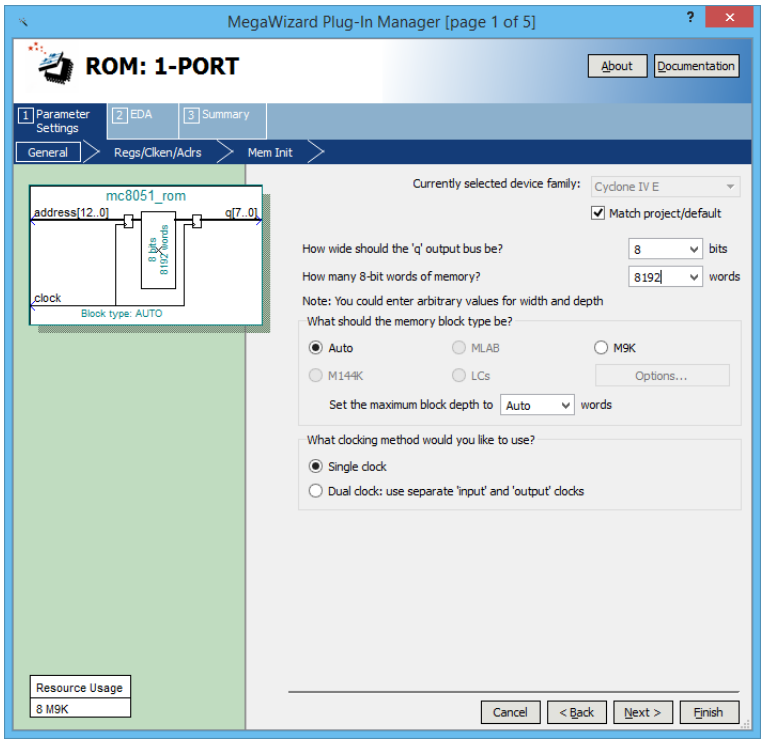


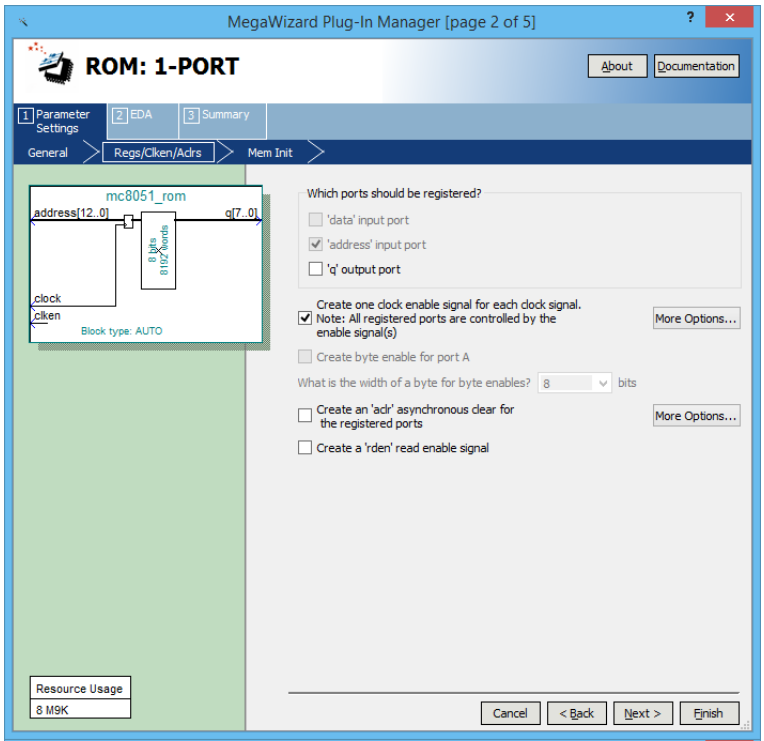
Step 3 – Finalizing architecture

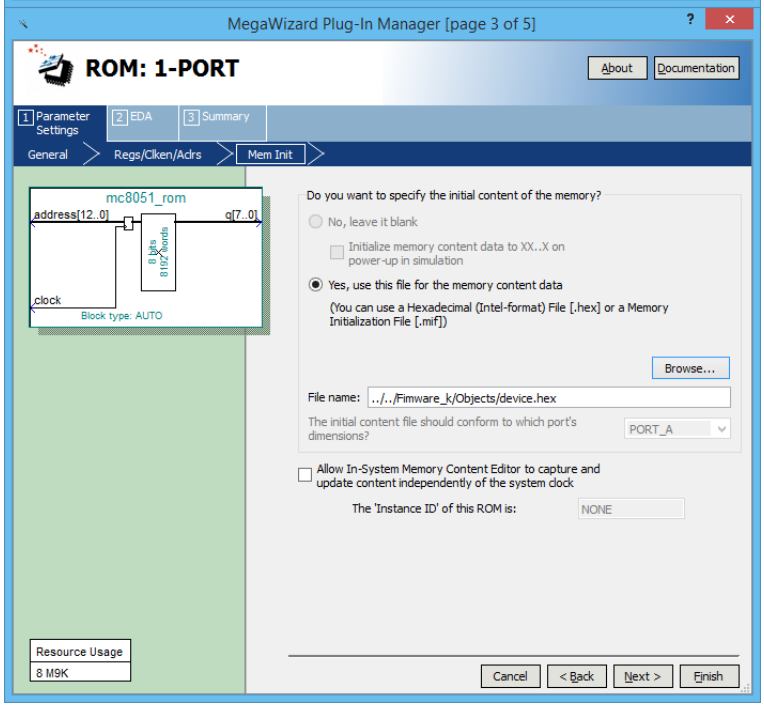
- Create instance of mc8051_rom (ROM: 1-PORT)



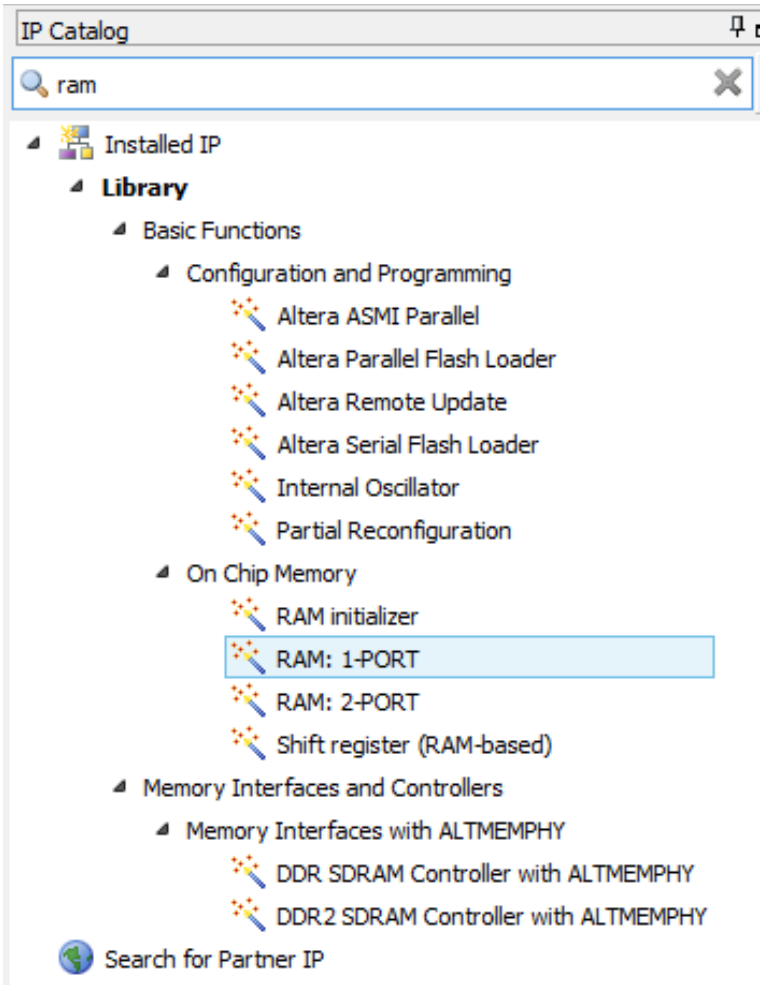


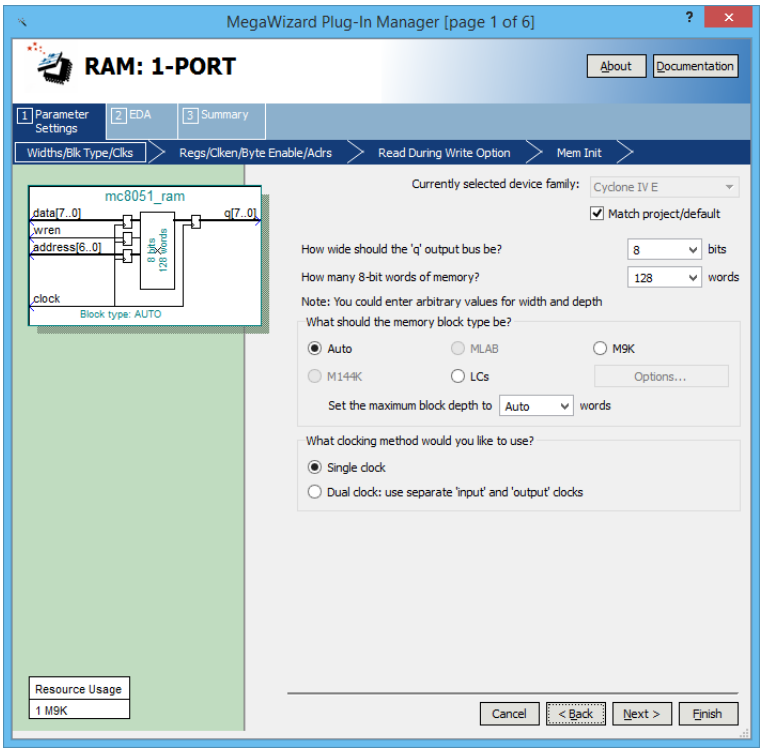


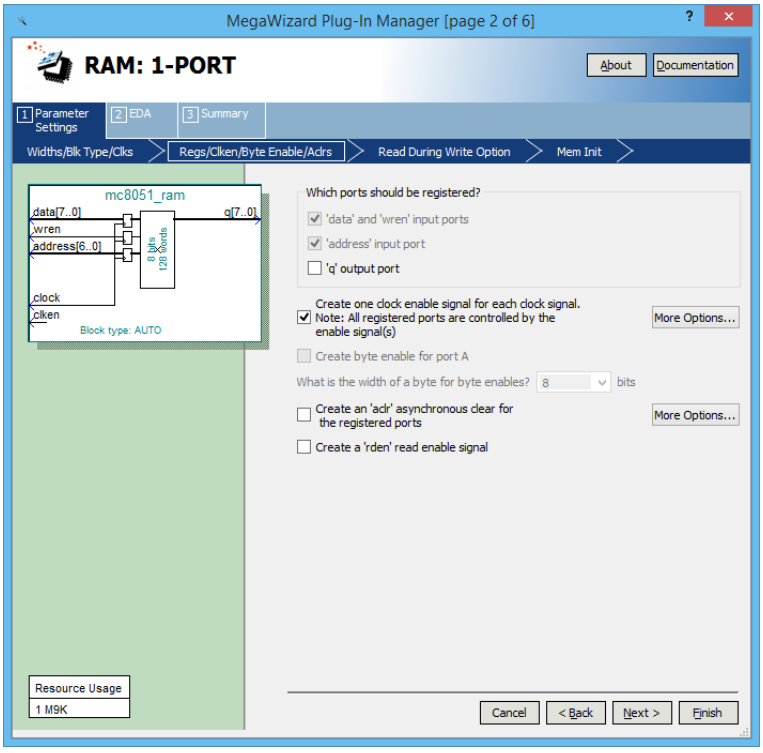

MegaWizard Plug-In Manager [page 2 of 5]
ROM: 1-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
Regs/Clken/Aclrs
Mem Init
mc8051_rom
address[12..0]
q[7..0]
8 bits
8192 words
clock
clken
Block type: AUTO
Which ports should be registered?
'data' input port
'address' input port
'q' output port
Create one clock enable signal for each clock signal.
Note: All registered ports are controlled by the enable signal(s)
More Options...
Create byte enable for port A
What is the width of a byte for byte enables?
8
bits
Create an 'aclr' asynchronous clear for the registered ports
More Options...
Create a 'rden' read enable signal
Resource Usage
8 M9K
Cancel
< Back
Next >
Finish


MegaWizard Plug-In Manager [page 3 of 5]
ROM: 1-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
General
Regs/Clken/Aclrs
Mem Init
mc8051_rom
address[12..0]
q[7..0]
8 bits
8192 words
clock
Block type: AUTO
Do you want to specify the initial content of the memory?
No, leave it blank
Initialize memory content data to XX..X on power-up in simulation
Yes, use this file for the memory content data
(You can use a Hexadecimal (Intel-format) File [.hex] or a Memory Initialization File [.mif])
Browse...
File name:
../../Fimware_k/Objects/device.hex
The initial content file should conform to which port's dimensions?
PORT_A
Allow In-System Memory Content Editor to capture and update content independently of the system clock
The 'Instance ID' of this ROM is:
NONE
Resource Usage
8 M9K
Cancel
< Back
Next >
Finish

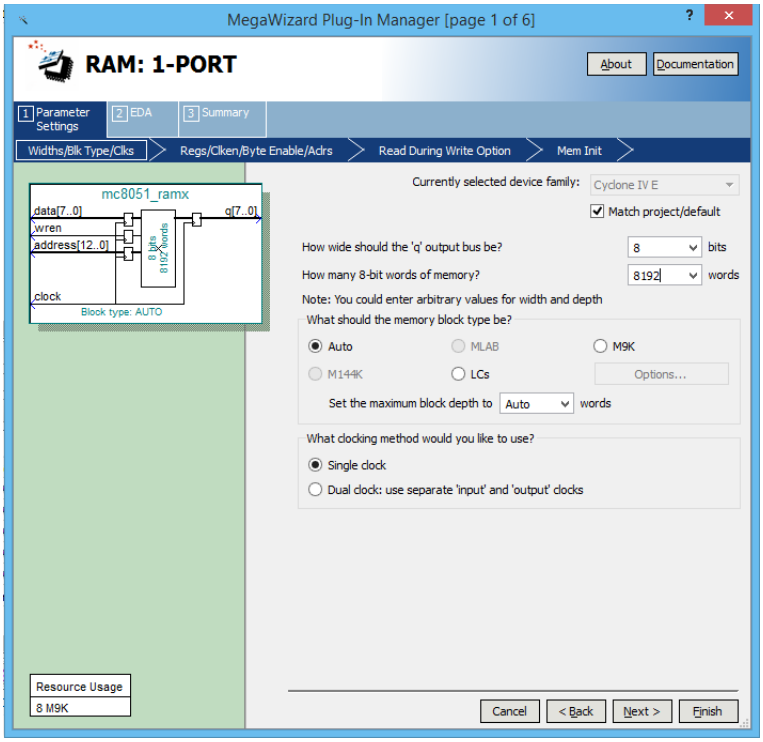
- Create instance of mc8051_ram (RAM: 1-PORT)

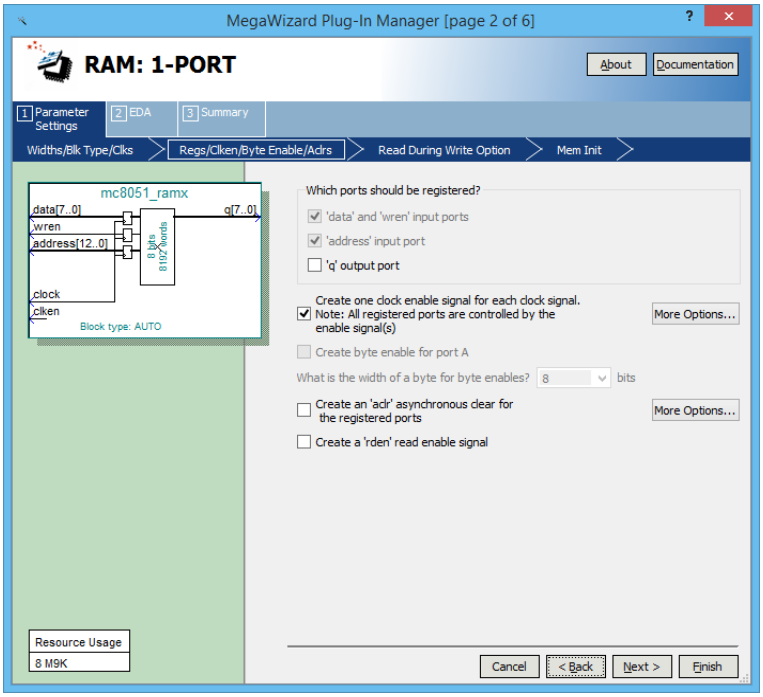



MegaWizard Plug-In Manager [page 1 of 6]
RAM: 1-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
Widths/Blk Type/Clks
Regs/Clken/Byte Enable/Adrs
Read During Write Option
Mem Init
mc8051_ram
data[7..0]
wren
address[6..0]
clock
q[7..0]
8 bits
128 words
Block type: AUTO
Currently selected device family:
Cyclone IV E
Match project/default
How wide should the 'q' output bus be?
8
bits
How many 8-bit words of memory?
128
words
Note: You could enter arbitrary values for width and depth
What should the memory block type be?
Auto
MLAB
M9K
M144K
LCs
Options...
Set the maximum block depth to
Auto
words
What clocking method would you like to use?
Single clock
Dual clock: use separate 'input' and 'output' clocks
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish

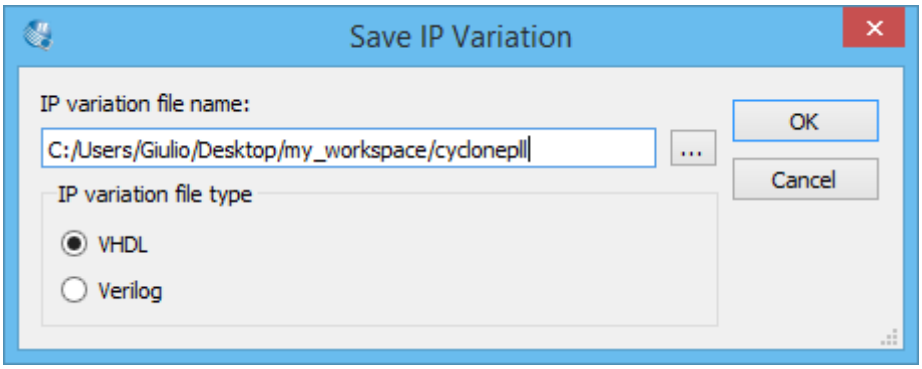

MegaWizard Plug-In Manager [page 2 of 6]
RAM: 1-PORT
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
Widths/Blk Type/Clks
Regs/Clken/Byte Enable/Adrs
Read During Write Option
Mem Init
mc8051_ram
data[7..0]
wren
address[6..0]
clock
clken
q[7..0]
8 bits
128 words
Block type: AUTO
Which ports should be registered?
'data' and 'wren' input ports
'address' input port
'q' output port
Create one clock enable signal for each clock signal. Note: All registered ports are controlled by the enable signal(s)
More Options...
Create byte enable for port A
What is the width of a byte for byte enables?
8
bits
Create an 'aclr' asynchronous clear for the registered ports
More Options...
Create a 'rden' read enable signal
Resource Usage
1 M9K
Cancel
< Back
Next >
Finish

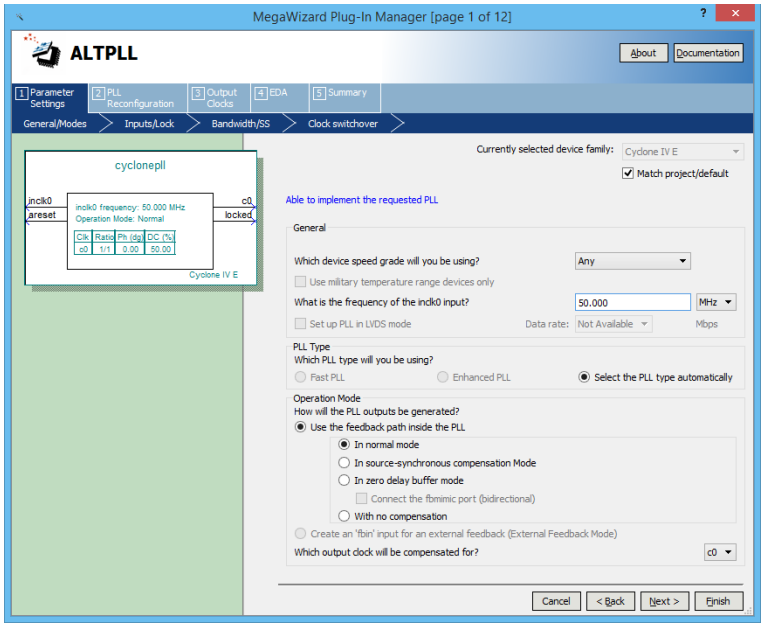
- Create instance of mc8051_ramx (RAM: 1-PORT)

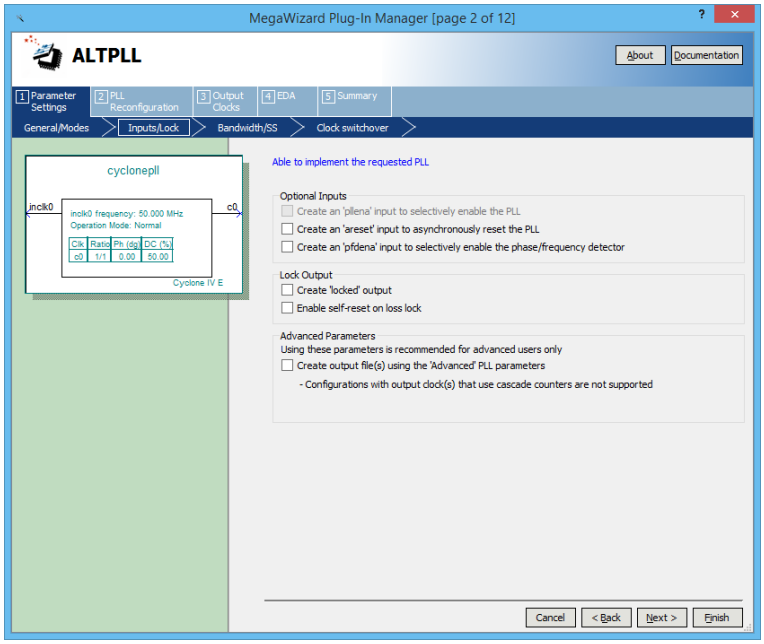


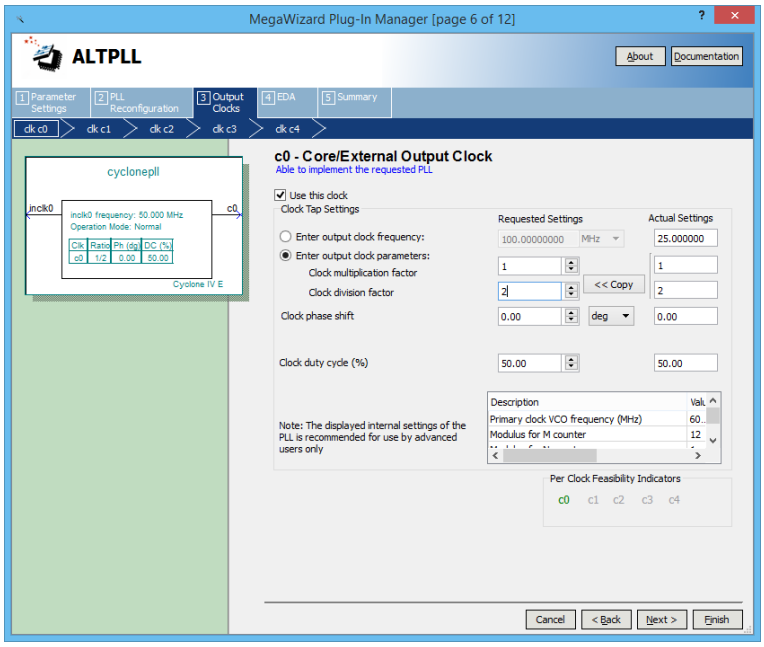


- Create instance of cyclonepll (ALTPLL)

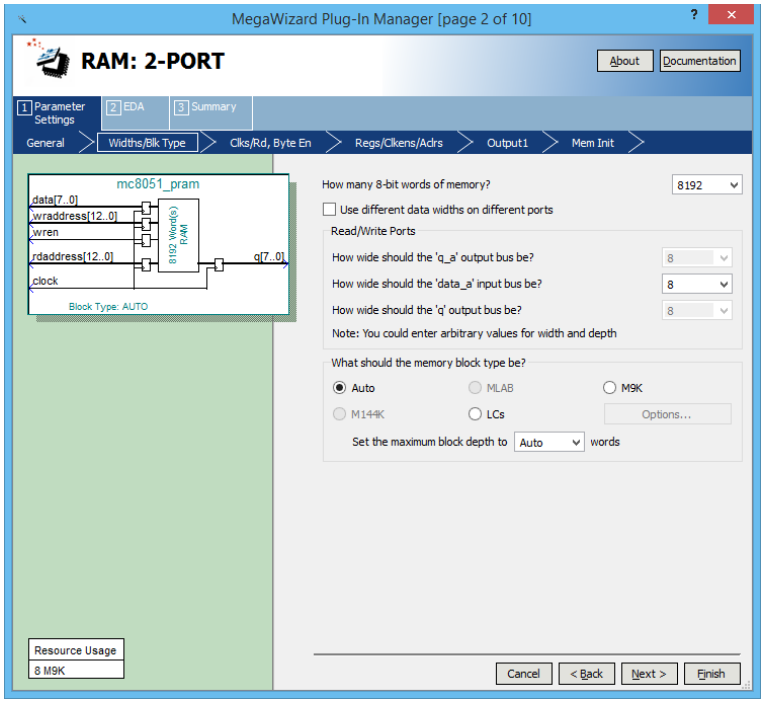


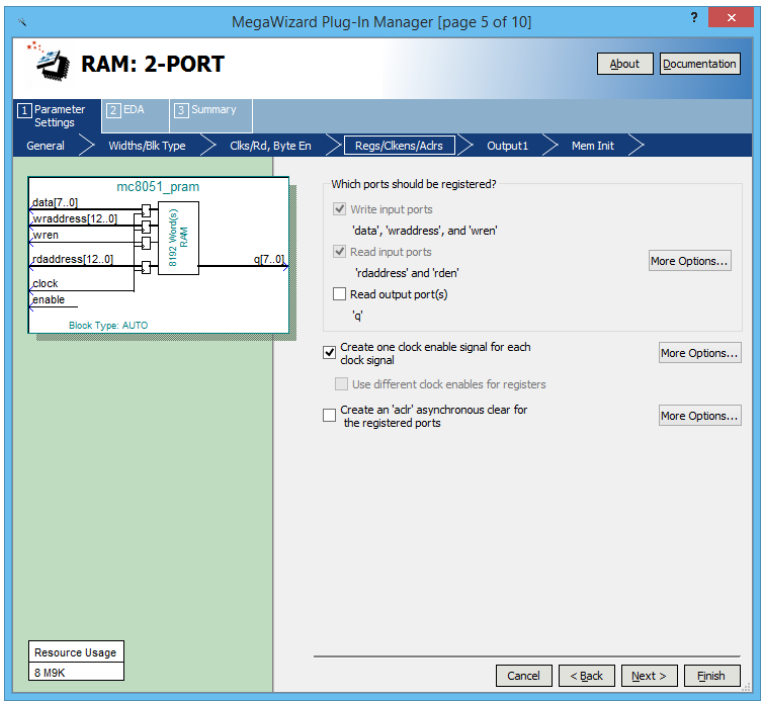





MegaWizard Plug-In Manager [page 6 of 12]
ALTPLL
About
Documentation
1 Parameter Settings
2 PLL Reconfiguration
3 Output Clocks
4 EDA
5 Summary
clk c0
clk c1
clk c2
clk c3
clk c4
cyclonepll
inclk0
c0
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk
Ratio
Ph (dg)
DC (%)
c0
1/2
0.00
50.00
Cyclone IV E
c0 - Core/External Output Clock
Able to implement the requested PLL
Use this clock
Clock Tap Settings
Requested Settings
Actual Settings
Enter output clock frequency:
100.00000000
MHz
25.000000
Enter output clock parameters:
Clock multiplication factor
1
1
Clock division factor
2
<< Copy
2
Clock phase shift
0.00
deg
0.00
Clock duty cycle (%)
50.00
50.00
Description
Primary clock VCO frequency (MHz)
60
Modulus for M counter
12
Note: The displayed internal settings of the PLL is recommended for use by advanced users only
Per Clock Feasibility Indicators
c0
c1
c2
c3
c4
Cancel
< Back
Next >
Finish

- Create instance of mc8051_pram (RAM: 2-PORT)



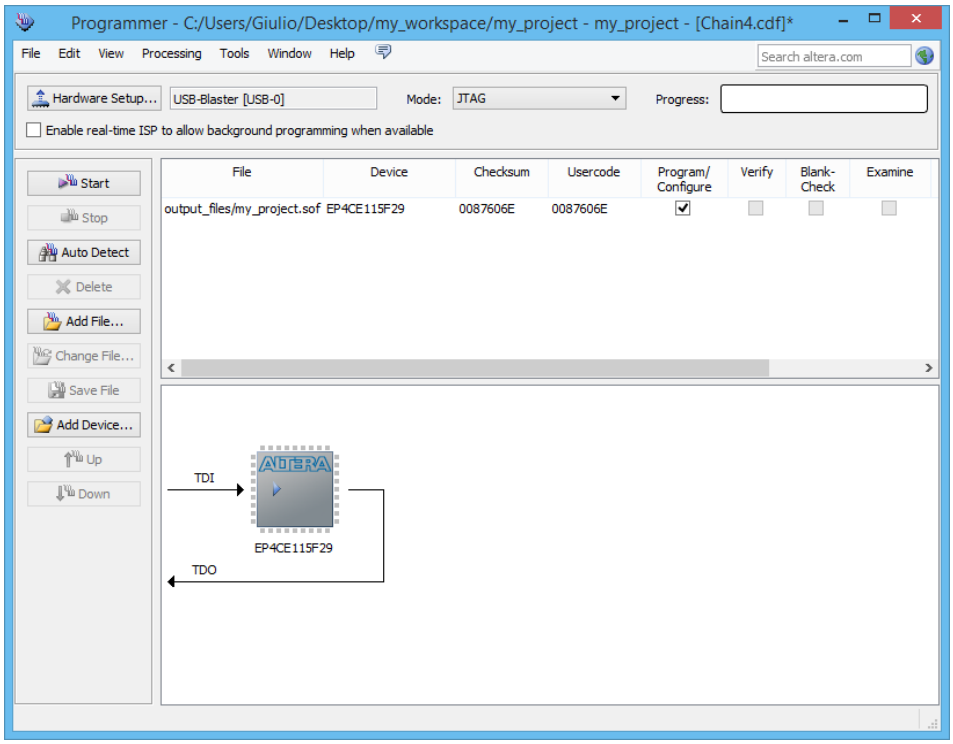


Step 4 – Pin Assignment

Node Name	Direction	Location
all_rxd_i[0]	Input	
all_rxd_o[0]	Output	
all_rxdwr_o[0]	Output	
all_t0_i[0]	Input	
all_t1_i[0]	Input	
all_txd_o[0]	Output	
clk	Input	PIN_Y2
int0_i[0]	Input	
int1_i[0]	Input	
p0_i[7]	Input	
p0_i[6]	Input	
p0_i[5]	Input	
p0_i[4]	Input	
p0_i[3]	Input	
p0_i[2]	Input	
p0_i[1]	Input	
p0_i[0]	Input	
p0_o[7]	Output	PIN_G21
p0_o[6]	Output	PIN_G22
p0_o[5]	Output	PIN_G20
p0_o[4]	Output	PIN_H21
p0_o[3]	Output	PIN_E24
p0_o[2]	Output	PIN_E25
p0_o[1]	Output	PIN_E22
p0_o[0]	Output	PIN_E21
p1_i[7]	Input	
p1_i[6]	Input	
p1_i[5]	Input	
p1_i[4]	Input	
p1_i[3]	Input	
p1_i[2]	Input	
p1_i[1]	Input	
p1_i[0]	Input	
p1_o[7]	Output	
p1_o[6]	Output	
p1_o[5]	Output	
p1_o[4]	Output	
p1_o[3]	Output	
p1_o[2]	Output	
p1_o[1]	Output	
p1_o[0]	Output	
p2_i[7]	Input	
p2_i[6]	Input	
p2_i[5]	Input	
p2_i[4]	Input	
p2_i[3]	Input	
p2_i[2]	Input	
p2_i[1]	Input	
p2_i[0]	Input	
p2_o[7]	Output	
p2_o[6]	Output	
p2_o[5]	Output	
p2_o[4]	Output	
p2_o[3]	Output	
p2_o[2]	Output	
p2_o[1]	Output	
p2_o[0]	Output	
p3_i[7]	Input	
p3_i[6]	Input	
p3_i[5]	Input	
p3_i[4]	Input	
p3_i[3]	Input	
p3_i[2]	Input	
p3_i[1]	Input	
p3_i[0]	Input	
p3_o[7]	Output	
p3_o[6]	Output	
p3_o[5]	Output	
p3_o[4]	Output	
p3_o[3]	Output	
p3_o[2]	Output	
p3_o[1]	Output	
p3_o[0]	Output	
reset_n	Input	PIN_AB28
<<new node>>		

Step 5 – Compile Design

- Compile design
- Program Device



- Enhance compilation speed during firmware revisions by enabling *smart compilation*