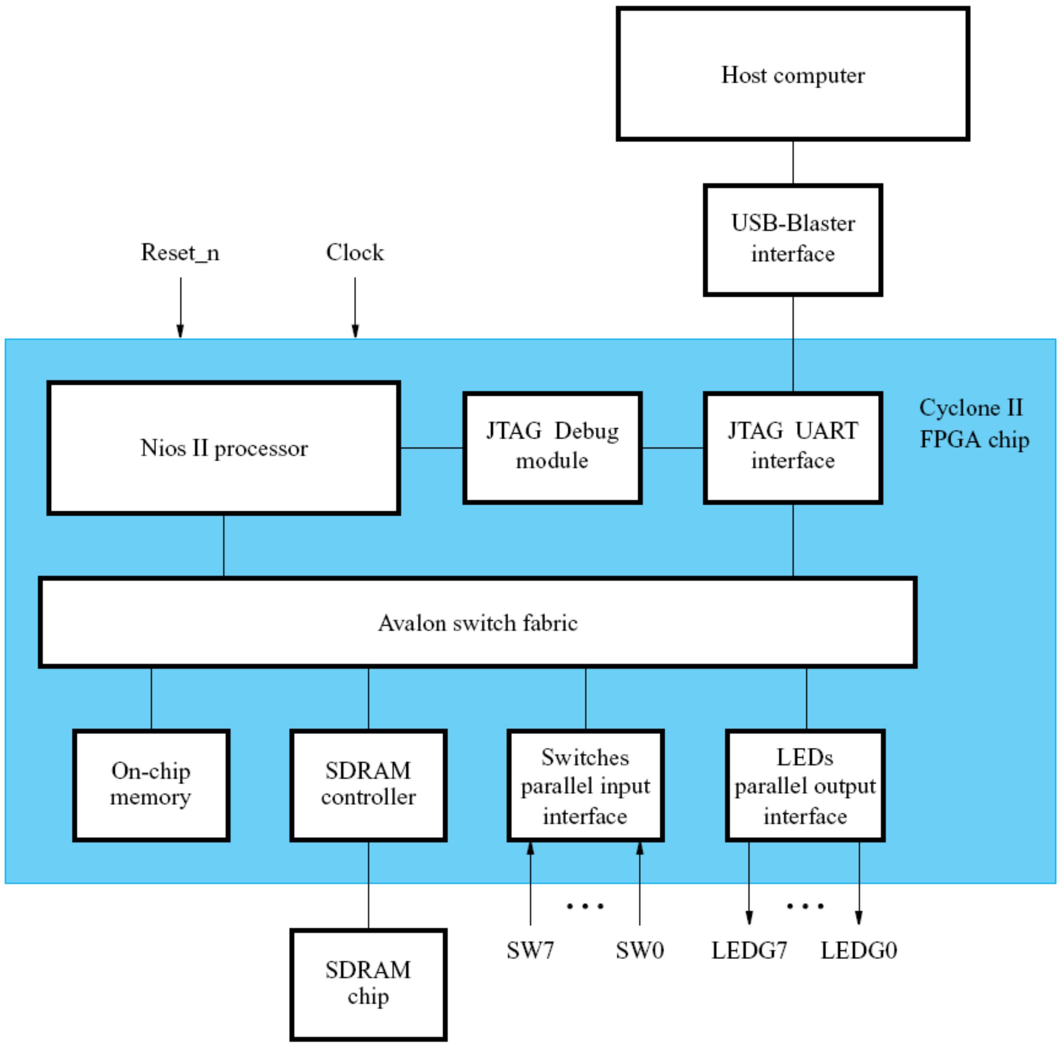
Lab 3　SDRAM 控制

銘傳大學電腦與通訊工程學系

陳慶逸

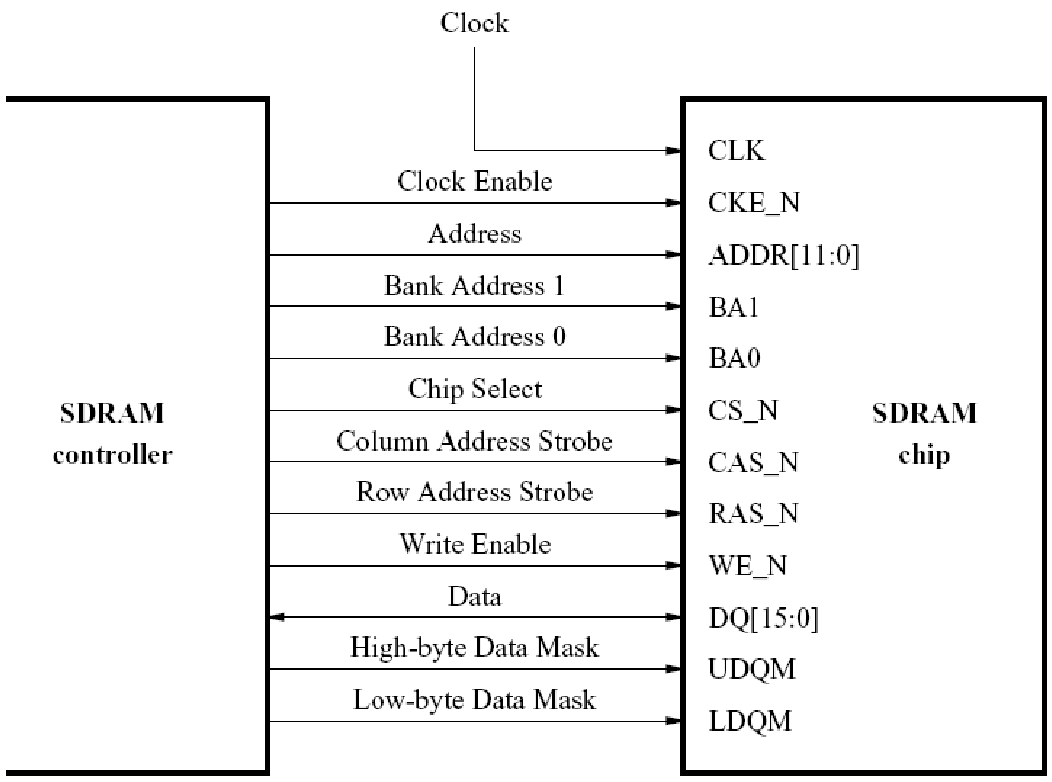
一、背景知識

1. 如下例所示，本次實驗中我們將在SOPC Builder中加入SDRAM 介面到Nios II system之中。



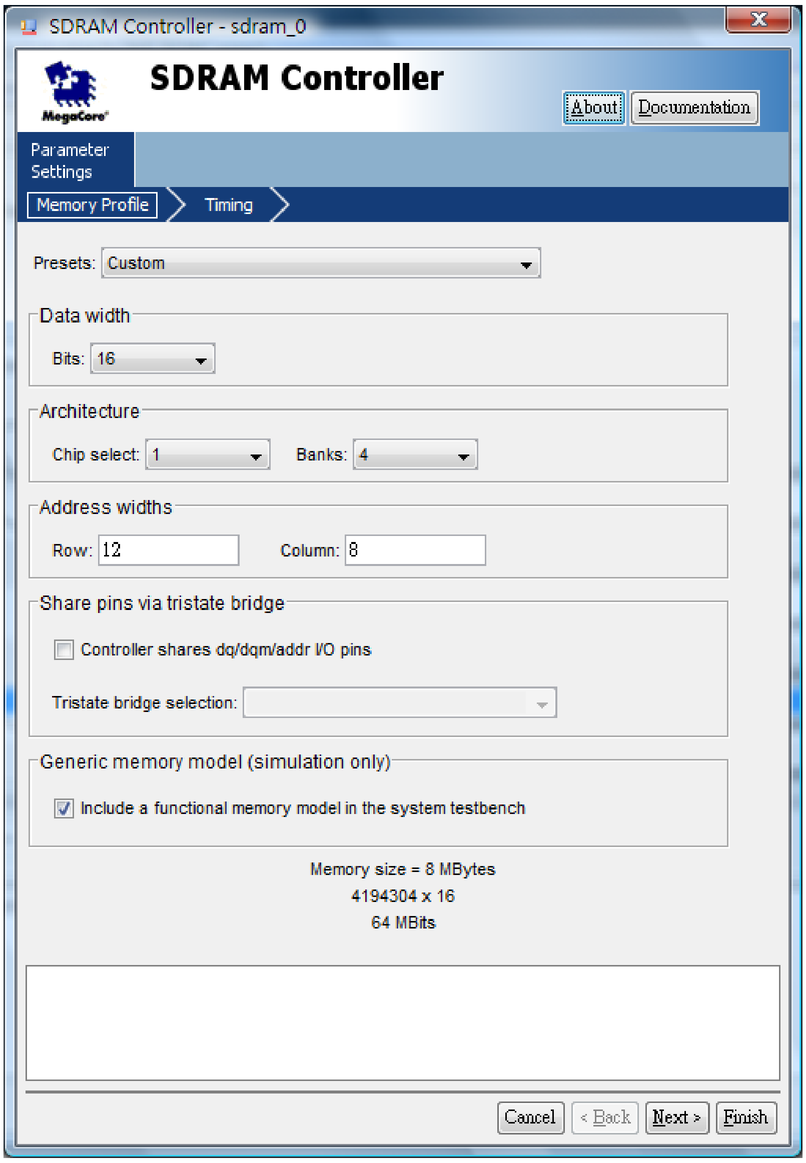
2. The SDRAM Interface

由圖可知，除了 SDRAM 所需要的 clock，SDRAM controller 可提供所有的訊號，而 SOPC Builder 已經提供了 SDRAM controller，所以我們唯一要操心的只剩下 clock 問題。



3. Using the SOPC Builder to Generate the Nios II System

1. SDRAM Controller



Presets: Custom

Data width

Bits: 16

Architecture

Chip select: 1 Banks: 4

Address widths

Row: 12 Column: 8

Share pins via tristate bridge

☐ Controller shares dq/dqm/addr I/O pins

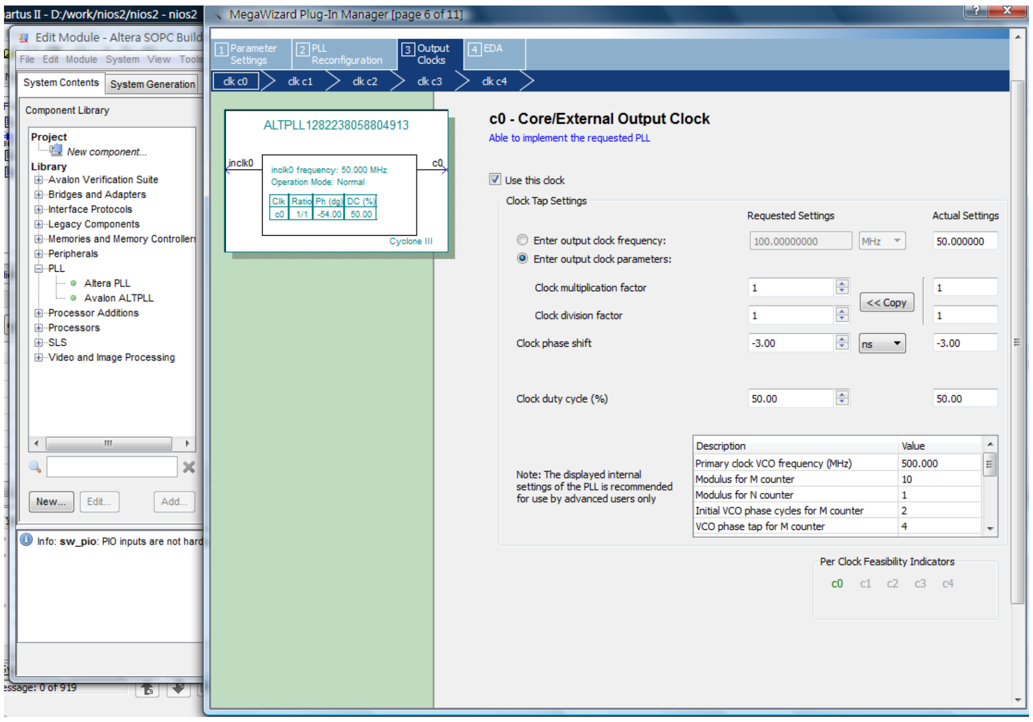
Tristate bridge selection:

Generic memory model (simulation only)

☑ Include a functional memory model in the system testbench

Memory size = 8 MBytes
4194304 x 16
64 MBits

2. PLL



Info: **sw_pio**: PIO inputs are not hard

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	-54.00	50.00

c0 - Core/External Output Clock

Able to implement the requested PLL

[✓] Use this clock

Clock Tap Settings

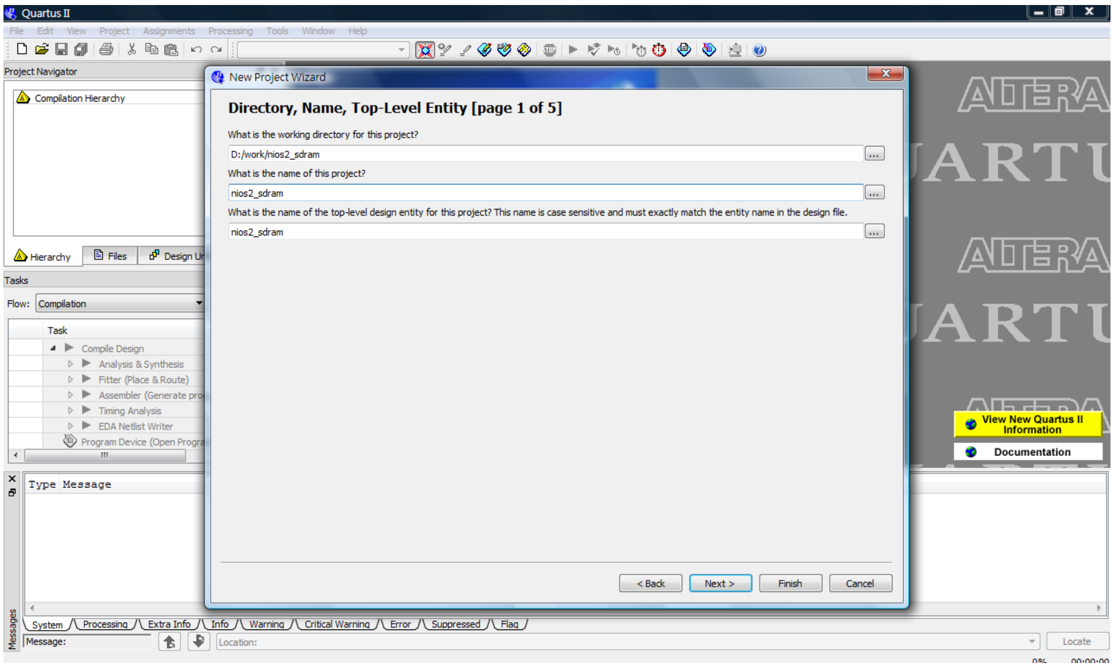
	Requested Settings		Actual Settings
○ Enter output clock frequency:	100.00000000	MHz	50.000000
● Enter output clock parameters:			
Clock multiplication factor	1		1
Clock division factor	1		1
Clock phase shift	-3.00	ns	-3.00
Clock duty cycle (%)	50.00		50.00

Note: The displayed internal settings of the PLL is recommended for use by advanced users only

Description	Value
Primary clock VCO frequency (MHz)	500.000
Modulus for M counter	10
Modulus for N counter	1
Initial VCO phase cycles for M counter	2
VCO phase tap for M counter	4

二、硬體設計(Nios II Hardware Development)

1. Create New Project (nios2_sdram)

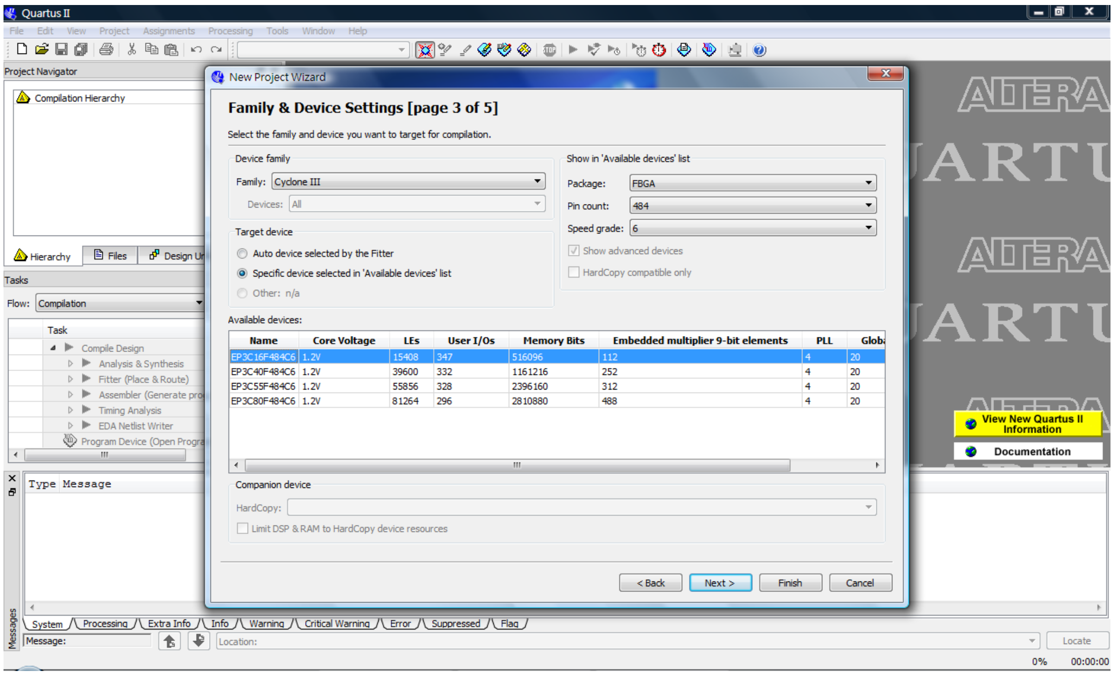


Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?
D:/work/nios2_sdram

What is the name of this project?
nios2_sdram

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.
nios2_sdram



Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.

Device family
Family: Cyclone III
Devices: All

Show in 'Available devices' list
Package: FBGA
Pin count: 484
Speed grade: 6
[✓] Show advanced devices
[ ] HardCopy compatible only

Target device
( ) Auto device selected by the Fitter
(●) Specific device selected in 'Available devices' list
( ) Other: n/a

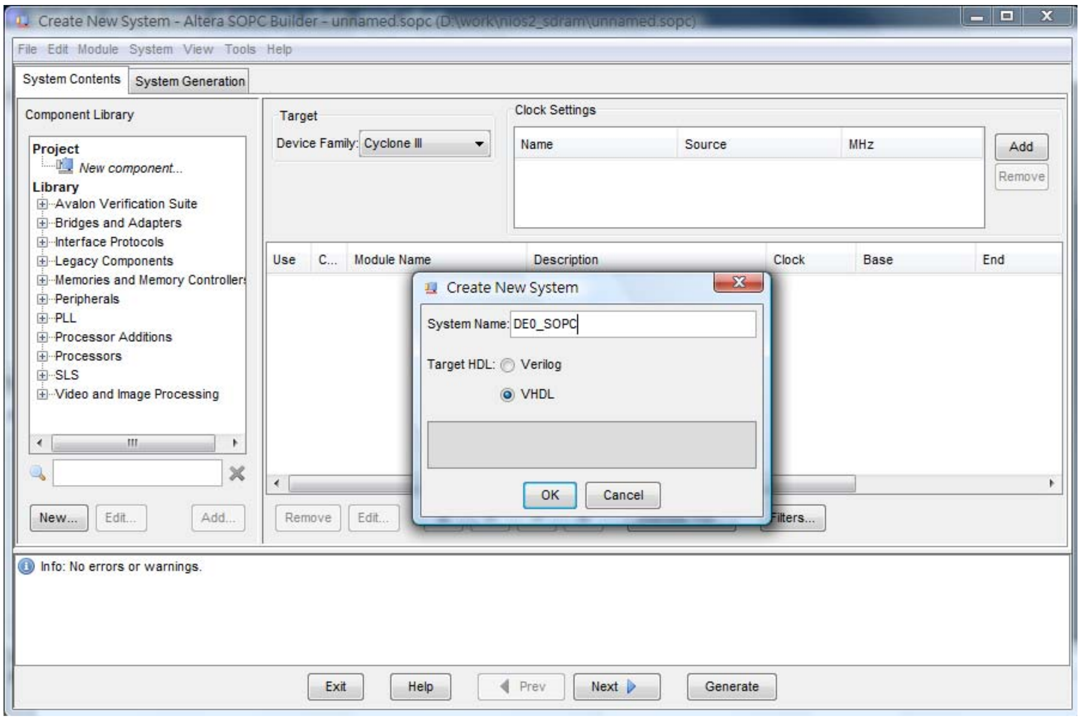
Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit elements	PLL	Globa
EP3C16F484C6	1.2V	15408	347	516096	112	4	20
EP3C40F484C6	1.2V	39600	332	1161216	252	4	20
EP3C55F484C6	1.2V	55856	328	2396160	312	4	20
EP3C80F484C6	1.2V	81264	296	2810880	488	4	20

Companion device
HardCopy:
[ ] Limit DSP & RAM to HardCopy device resources

2. **Using the SOPC Builder to Generate the Nios II System**..

2-1 Choose SOPC Builder (Tools menu), SOPC Builder displays the Create New System dialog box.



Device Family: Cyclone III

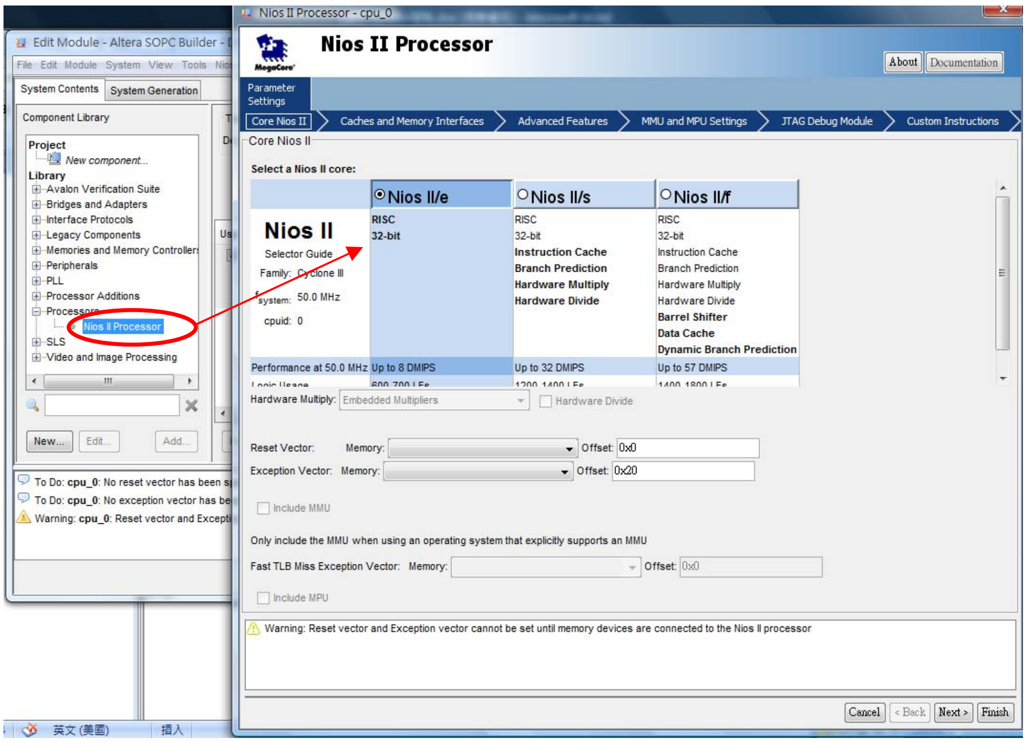
System Name: DE0_SOPC

Target HDL: Verilog [✓ VHDL]

Info: No errors or warnings.

Fig.3 Type **DE0_SOPC** in the System Name field.

2-2 Under Altera SOPC Builder Components category, select **Nios II Processor**.



Select a Nios II core: [✓ Nios II/e] Nios II/s Nios II/f

Nios II Selector Guide Family: Cyclone III fsystem: 50.0 MHz cpuid: 0	Nios II/e	Nios II/s	Nios II/f
	RISC 32-bit	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Prediction
Performance at 50.0 MHz	Up to 8 DMIPS	Up to 32 DMIPS	Up to 57 DMIPS

Hardware Multiply: Embedded Multipliers; Hardware Divide

Reset Vector: Memory: Offset: 0x0

Exception Vector: Memory: Offset: 0x20

Include MMU

Only include the MMU when using an operating system that explicitly supports an MMU

Fast TLB Miss Exception Vector: Memory: Offset: 0x0

Include MPU

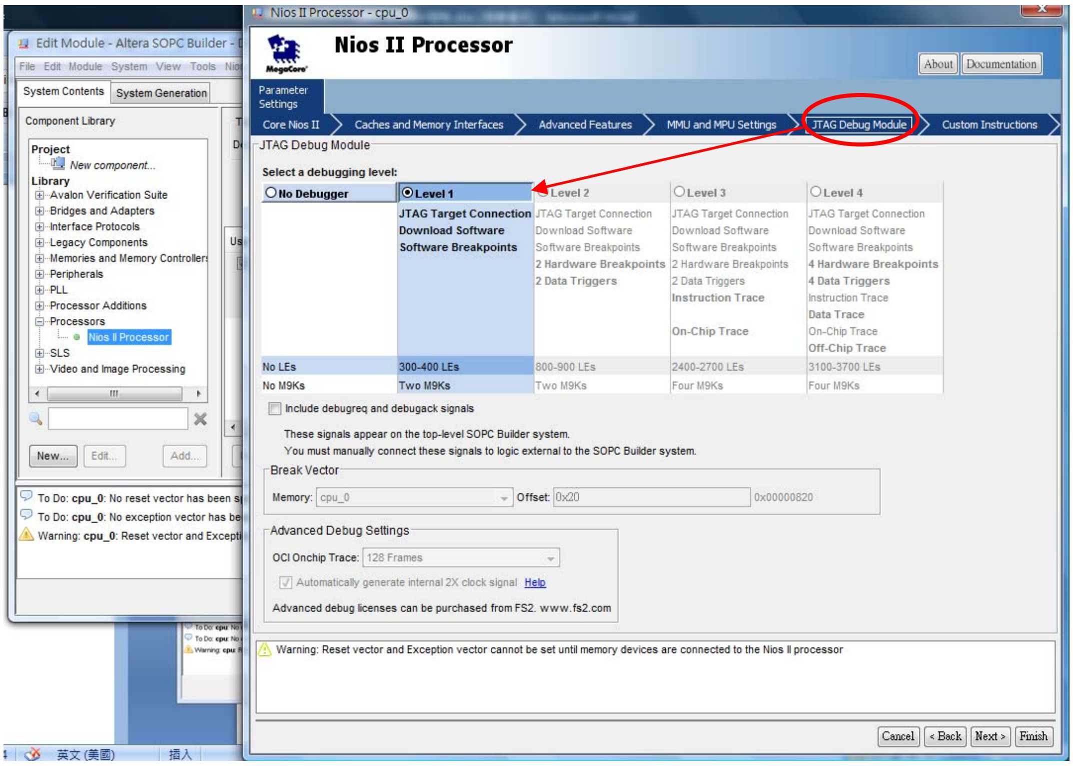
Warning: Reset vector and Exception vector cannot be set until memory devices are connected to the Nios II processor

To Do: cpu_0: No reset vector has been s

To Do: cpu_0: No exception vector has be

Warning: cpu_0: Reset vector and Excepti

Fig. 4 Nios II Processor



Nios II Processor - cpu_0

Nios II Processor

Parameter Settings

Core Nios II > Caches and Memory Interfaces > Advanced Features > MMU and MPU Settings > JTAG Debug Module > Custom Instructions

JTAG Debug Module

Select a debugging level:

No Debugger	Level 1	Level 2	Level 3	Level 4
	JTAG Target Connection Download Software Software Breakpoints	JTAG Target Connection Download Software Software Breakpoints 2 Hardware Breakpoints 2 Data Triggers	JTAG Target Connection Download Software Software Breakpoints 2 Hardware Breakpoints 2 Data Triggers Instruction Trace On-Chip Trace	JTAG Target Connection Download Software Software Breakpoints 4 Hardware Breakpoints 4 Data Triggers Instruction Trace Data Trace On-Chip Trace Off-Chip Trace
No LEs	300-400 LEs	800-900 LEs	2400-2700 LEs	3100-3700 LEs
No M9Ks	Two M9Ks	Two M9Ks	Four M9Ks	Four M9Ks

Include debugreq and debugack signals

These signals appear on the top-level SOPC Builder system.
You must manually connect these signals to logic external to the SOPC Builder system.

Break Vector

Memory: cpu_0 Offset: 0x20 0x00000820

Advanced Debug Settings

OCI Onchip Trace: 128 Frames

Automatically generate internal 2X clock signal Help

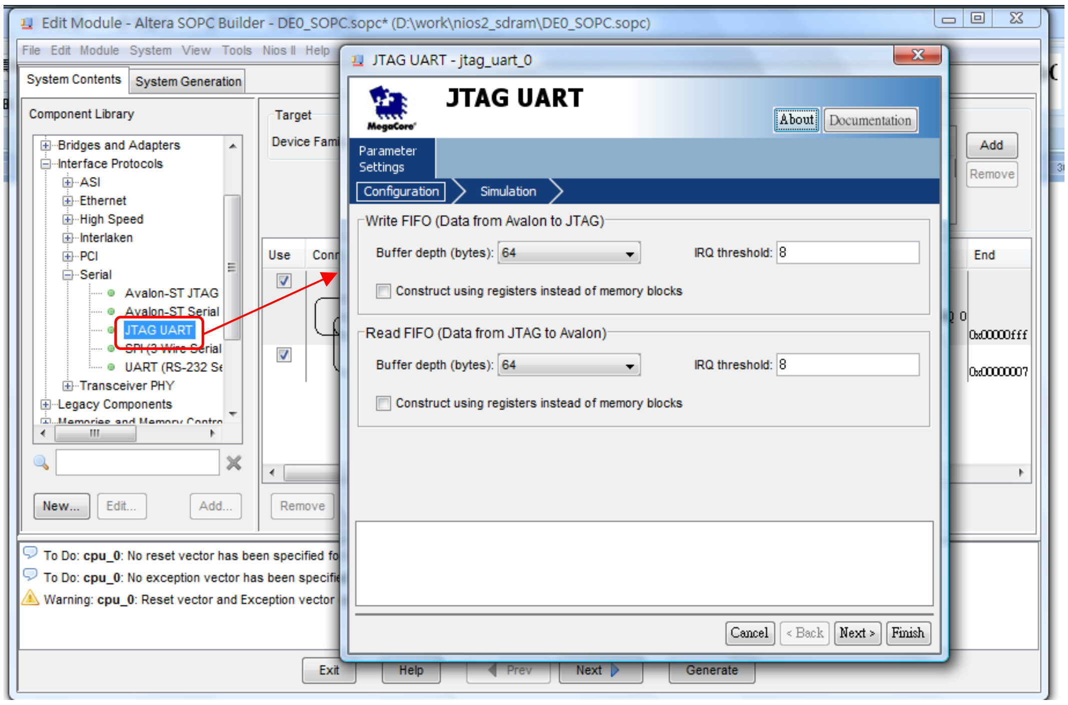
Advanced debug licenses can be purchased from FS2. www.fs2.com

Warning: Reset vector and Exception vector cannot be set until memory devices are connected to the Nios II processor

To Do: cpu_0: No reset vector has been s
To Do: cpu_0: No exception vector has be
Warning: cpu_0: Reset vector and Excepti

Fig. 5 Click the **JTAG Debug Module** tab and choose the “Level 1” degugging level.

2-3 Select **JTAG UART** under the **Altera SOPC Builder** > Interface Protocols > **Serial** category. Add JTAG_UART.



Edit Module - Altera SOPC Builder - DE0_SOPC.sopc* (D:\work\nios2_sdram\DE0_SOPC.sopc)

JTAG UART - jtag_uart_0

JTAG UART

Parameter Settings

Configuration > Simulation

Write FIFO (Data from Avalon to JTAG)

Buffer depth (bytes): 64 IRQ threshold: 8

Construct using registers instead of memory blocks

Read FIFO (Data from JTAG to Avalon)

Buffer depth (bytes): 64 IRQ threshold: 8

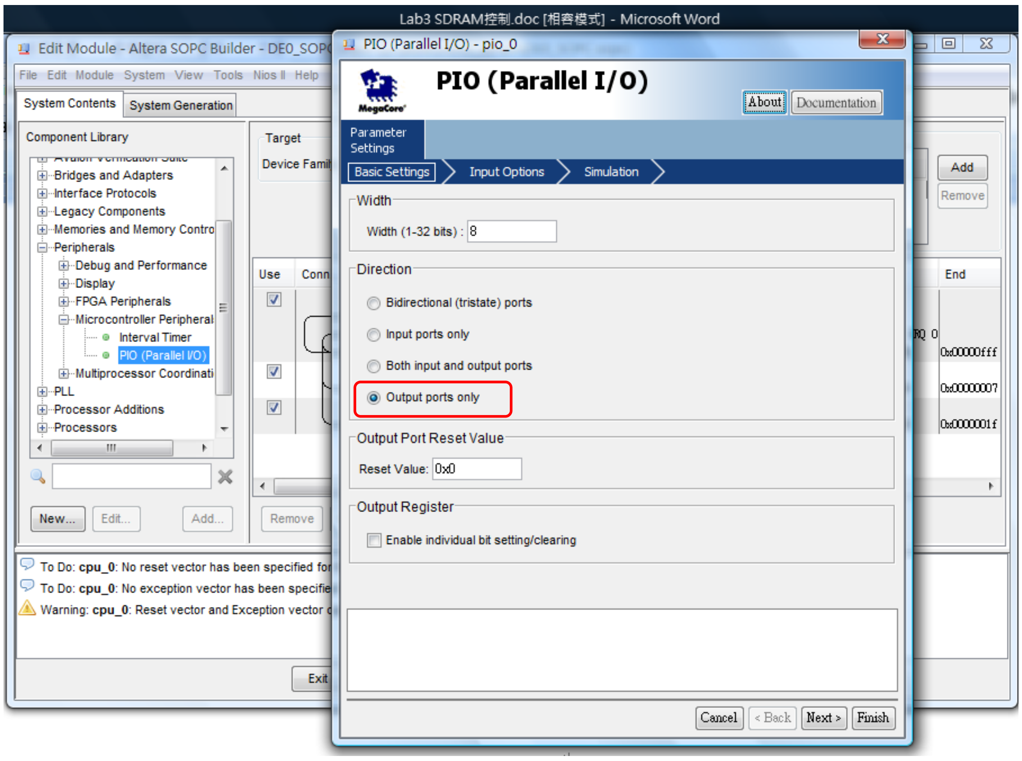
Construct using registers instead of memory blocks

To Do: cpu_0: No reset vector has been specified fo
To Do: cpu_0: No exception vector has been specifie
Warning: cpu_0: Reset vector and Exception vector

Add JTAG UART

2-4 Select **PIO (Parallel I/O)** under the **Altera SOPC Builder**> **Peripherals >**

Microcontroller Peripherals category. Add Output PIO. Right-click **pio** and select Rename. Type **led_pio** and press Enter.



PIO (Parallel I/O) - pio_0

PIO (Parallel I/O)

Basic Settings > Input Options > Simulation

Width

Width (1-32 bits) : 8

Direction

Bidirectional (tristate) ports

Input ports only

Both input and output ports

[✓ Output ports only]

Output Port Reset Value

Reset Value: 0x0

Output Register

Enable individual bit setting/clearing

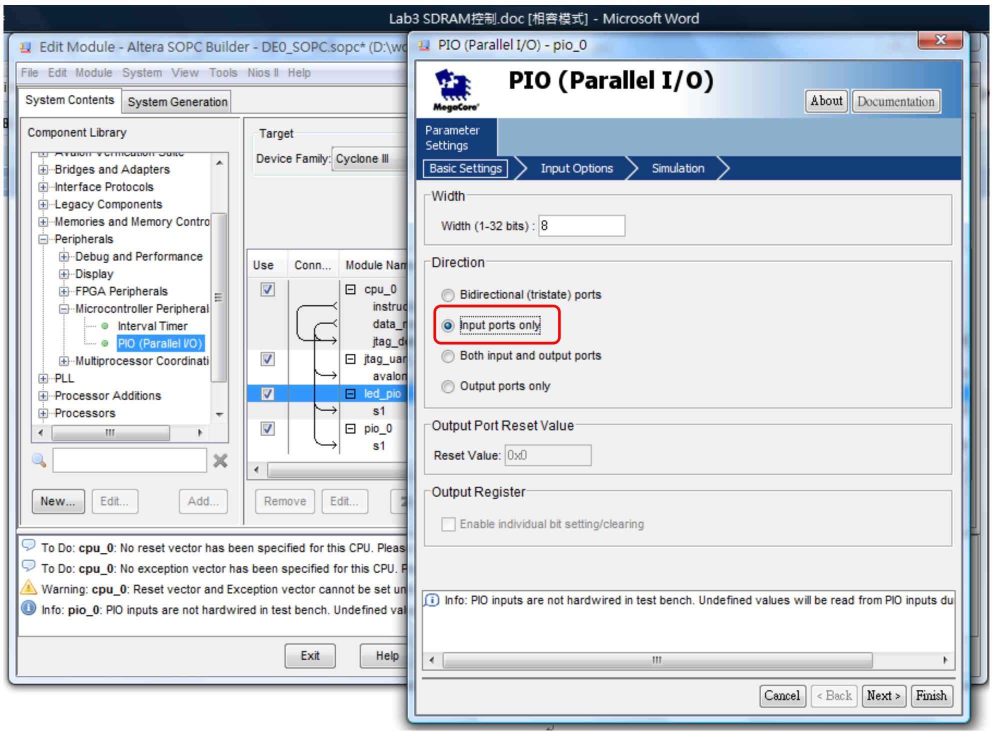
To Do: cpu_0: No reset vector has been specified fo

To Do: cpu_0: No exception vector has been specifie

Warning: cpu_0: Reset vector and Exception vector c

Add Output PIO (Rename “led_pio”)

Add Intput PIO. Right-click **pio** and select Rename. Type **sw_pio** and press Enter.



PIO (Parallel I/O) - pio_0

PIO (Parallel I/O)

Basic Settings > Input Options > Simulation

Width

Width (1-32 bits) : 8

Direction

Bidirectional (tristate) ports

[✓ Input ports only]

Both input and output ports

Output ports only

Output Port Reset Value

Reset Value: 0x0

Output Register

Enable individual bit setting/clearing

Info: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs du

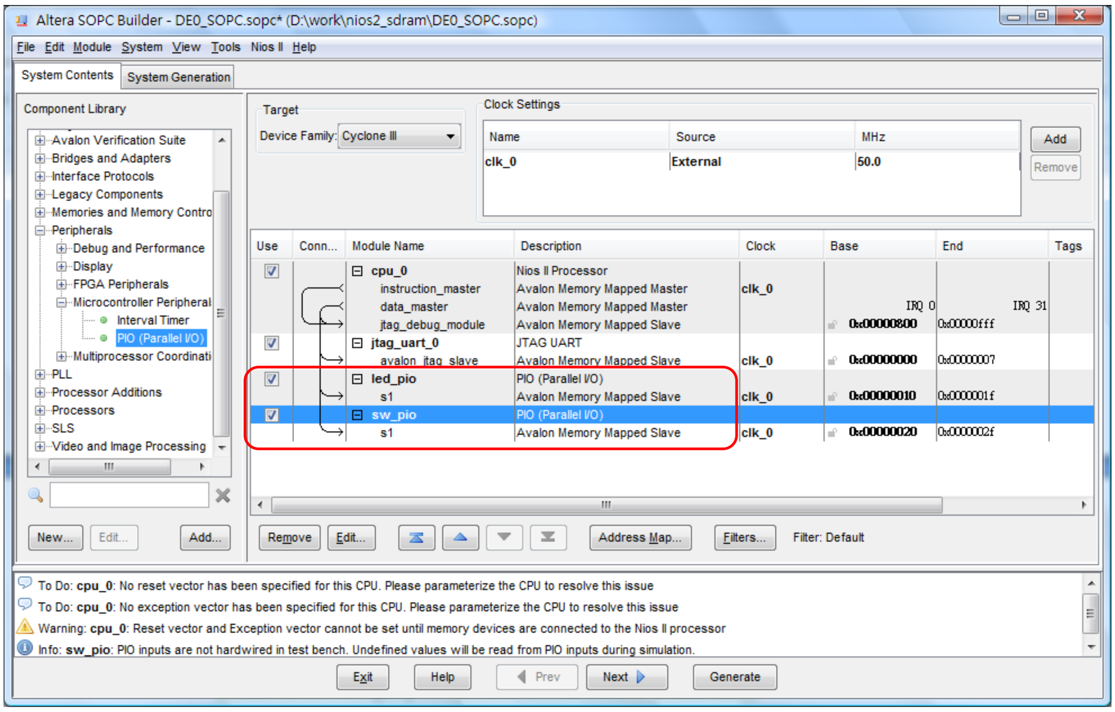
To Do: cpu_0: No reset vector has been specified for this CPU. Pleas

To Do: cpu_0: No exception vector has been specified for this CPU. P

Warning: cpu_0: Reset vector and Exception vector cannot be set un

Info: pio_0: PIO inputs are not hardwired in test bench. Undefined val

Add Input PIO (Rename “sw_pio”)


Altera SOPC Builder - DE0_SOPC.sopc* (D:\work\nios2_sdram\DE0_SOPC.sopc)
File Edit Module System View Tools Nios II Help
System Contents
System Generation
Component Library
Avalon Verification Suite
Bridges and Adapters
Interface Protocols
Legacy Components
Memories and Memory Contro
Peripherals
Debug and Performance
Display
FPGA Peripherals
Microcontroller Peripheral
Interval Timer
PIO (Parallel I/O)
Multiprocessor Coordinati
PLL
Processor Additions
Processors
SLS
Video and Image Processing
New... Edit... Add...
Remove Edit... Address Map... Filters...
Exit Help Prev Next Generate

Target
Device Family: Cyclone III

Clock Settings

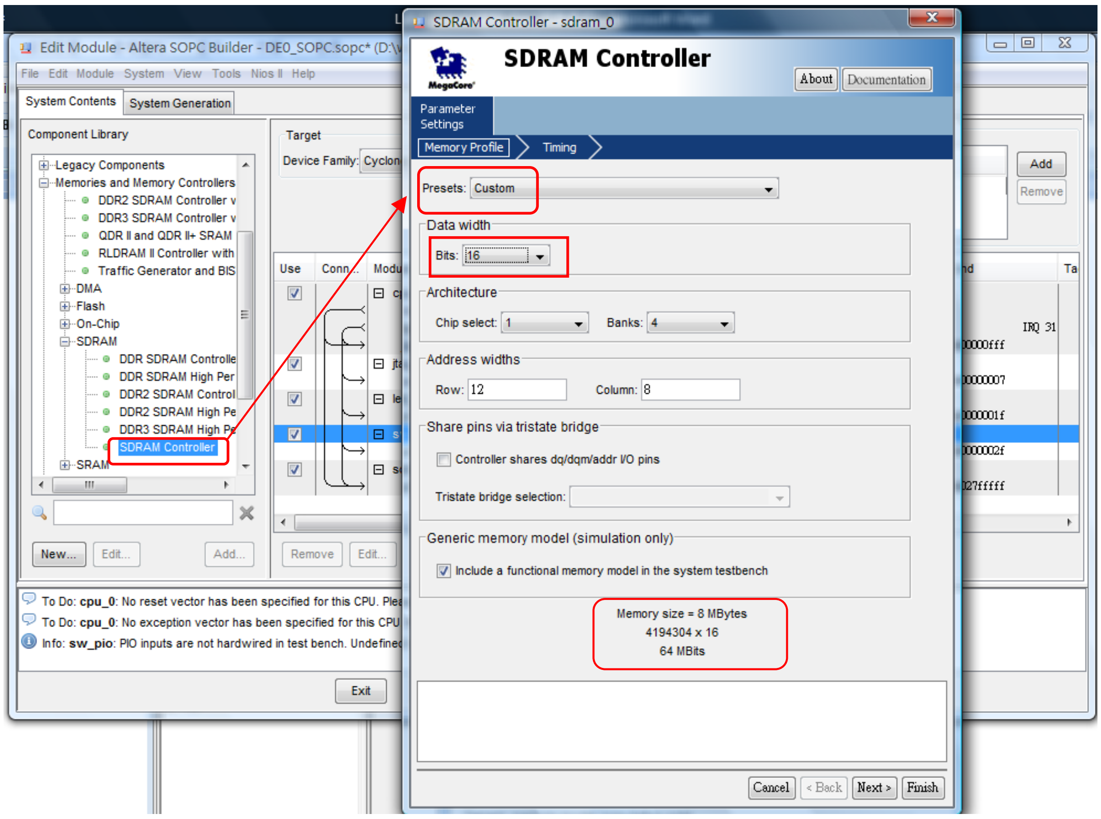
Name	Source	MHz
clk_0	External	50.0

Use	Conn...	Module Name	Description	Clock	Base	End	Tags
✓		cpu_0	Nios II Processor				
		instruction_master	Avalon Memory Mapped Master	clk_0			
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31	
		jtag_debug_module	Avalon Memory Mapped Slave		0x00000800	0x00000fff	
✓		jtag_uart_0	JTAG UART				
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00000000	0x00000007	
✓		led_pio	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	clk_0	0x00000010	0x0000001f	
✓		sw_pio	PIO (Parallel I/O)				
		s1	Avalon Memory Mapped Slave	clk_0	0x00000020	0x0000002f	

Filter: Default

To Do: **cpu_0**: No reset vector has been specified for this CPU. Please parameterize the CPU to resolve this issue
To Do: **cpu_0**: No exception vector has been specified for this CPU. Please parameterize the CPU to resolve this issue
Warning: **cpu_0**: Reset vector and Exception vector cannot be set until memory devices are connected to the Nios II processor
Info: **sw_pio**: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

2-5 Select **SDRAM** under the **Altera SOPC Builder** > Memory and Memory Controllers category > SDRAM> SDRAM Controller.



SDRAM Controller

Presets: Custom

Data width
Bits: 16

Architecture
Chip select: 1   Banks: 4

Address widths
Row: 12   Column: 8

Share pins via tristate bridge
☐ Controller shares dq/dqm/addr I/O pins
Tristate bridge selection:

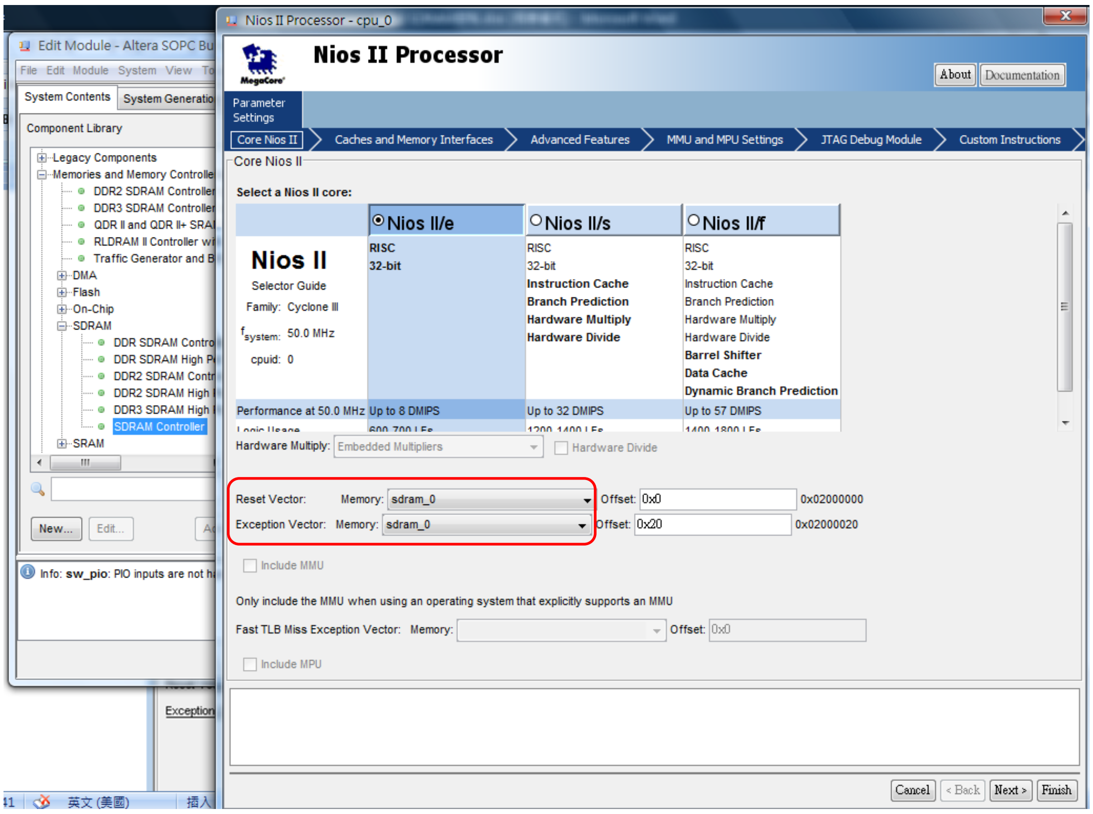
Generic memory model (simulation only)
☑ Include a functional memory model in the system testbench

Memory size = 8 MBytes
4194304 x 16
64 MBits

To Do: **cpu_0**: No reset vector has been specified for this CPU. Plea
To Do: **cpu_0**: No exception vector has been specified for this CPU
Info: **sw_pio**: PIO inputs are not hardwired in test bench. Undefine

Fig.6 Add SDRAM Controller

2-6 Modify the Reset Vector & Exception Vector of the NIOS II Processor.



Nios II Processor - cpu_0

Core Nios II

Select a Nios II core: ● Nios II/e ○ Nios II/s ○ Nios II/f

Nios II	Nios II/e	Nios II/s	Nios II/f
Selector Guide Family: Cyclone III f_{system}: 50.0 MHz cpuid: 0	RISC 32-bit	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide	RISC 32-bit Instruction Cache Branch Prediction Hardware Multiply Hardware Divide Barrel Shifter Data Cache Dynamic Branch Prediction
Performance at 50.0 MHz	Up to 8 DMIPS	Up to 32 DMIPS	Up to 57 DMIPS
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs

Hardware Multiply: Embedded Multipliers   Hardware Divide

Reset Vector: Memory: sdram_0   Offset: 0x0   0x02000000

Exception Vector: Memory: sdram_0   Offset: 0x20   0x02000020

Include MMU

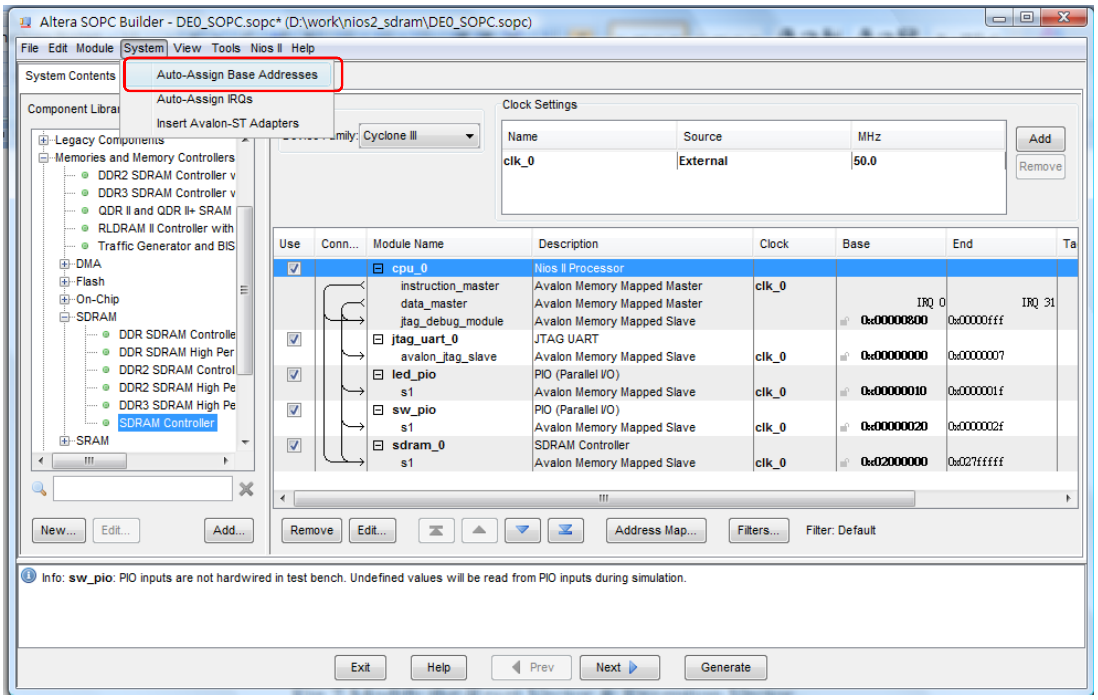
Only include the MMU when using an operating system that explicitly supports an MMU

Fast TLB Miss Exception Vector: Memory: Offset: 0x0

Include MPU

Fig.7 Modify the Reset Vector & Exception Vector

註：若發生位址衝突的錯誤訊息，可執行 System > Auto-Assign Base Address



Altera SOPC Builder - DE0_SOPC.sopc* (D:\work\nios2_sdram\DE0_SOPC.sopc)

System > Auto-Assign Base Addresses

Device Family: Cyclone III

Clock Settings

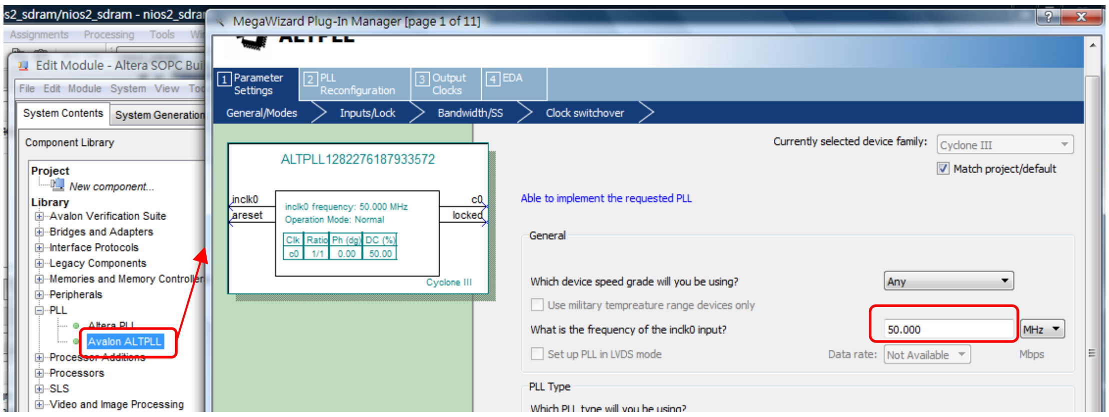
Name	Source	MHz
clk_0	External	50.0

Use	Conn...	Module Name	Description	Clock	Base	End
✓		cpu_0	Nios II Processor			
		instruction_master	Avalon Memory Mapped Master	clk_0		
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31
		jtag_debug_module	Avalon Memory Mapped Slave		0x00000800	0x00000fff
✓		jtag_uart_0	JTAG UART			
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00000000	0x00000007
✓		led_pio	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	clk_0	0x00000010	0x0000001f
✓		sw_pio	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	clk_0	0x00000020	0x0000002f
✓		sdram_0	SDRAM Controller			
		s1	Avalon Memory Mapped Slave	clk_0	0x02000000	0x027fffff

Info: sw_pio: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

2-7 Select **ALTPLL** under the **Altera SOPC Builder** > PLL > Avalon ALTPLL.

2-7-1 Select the frequency of the inclock0 input.


s2_sdram/nios2_sdram - nios2_sdra

Edit Module - Altera SOPC Bu

System Contents | System Generation

Component Library

Project
New component...
Library
- Avalon Verification Suite
- Bridges and Adapters
- Interface Protocols
- Legacy Components
- Memories and Memory Controller
- Peripherals
- PLL
  - Altera PLL
  - Avalon ALTPLL
- Processor Additions
- Processors
- SLS
- Video and Image Processing

MegaWizard Plug-In Manager [page 1 of 11]

ALTPLL

1 Parameter Settings | 2 PLL Reconfiguration | 3 Output Clocks | 4 EDA

General/Modes > Inputs/Lock > Bandwidth/SS > Clock switchover >

ALTPLL1282276187933572
inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

inclk0, areset, c0, locked — Cyclone III

Currently selected device family: Cyclone III
☑ Match project/default

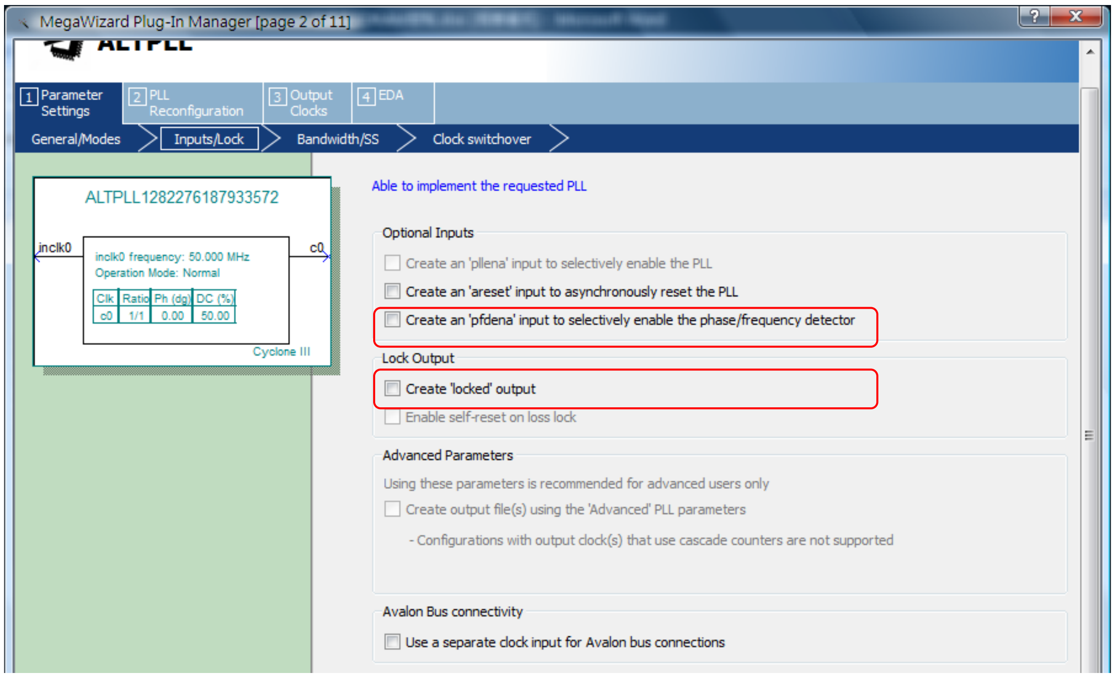
Able to implement the requested PLL

General

Which device speed grade will you be using? Any
☐ Use military temprature range devices only
What is the frequency of the inclk0 input? 50.000 MHz
☐ Set up PLL in LVDS mode Data rate: Not Available Mbps

PLL Type
Which PLL type will you be using?

2-7-2 Optional inputs


MegaWizard Plug-In Manager [page 2 of 11]

ALTPLL

1 Parameter Settings | 2 PLL Reconfiguration | 3 Output Clocks | 4 EDA

General/Modes > Inputs/Lock > Bandwidth/SS > Clock switchover >

ALTPLL1282276187933572
inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

inclk0, c0 — Cyclone III

Able to implement the requested PLL

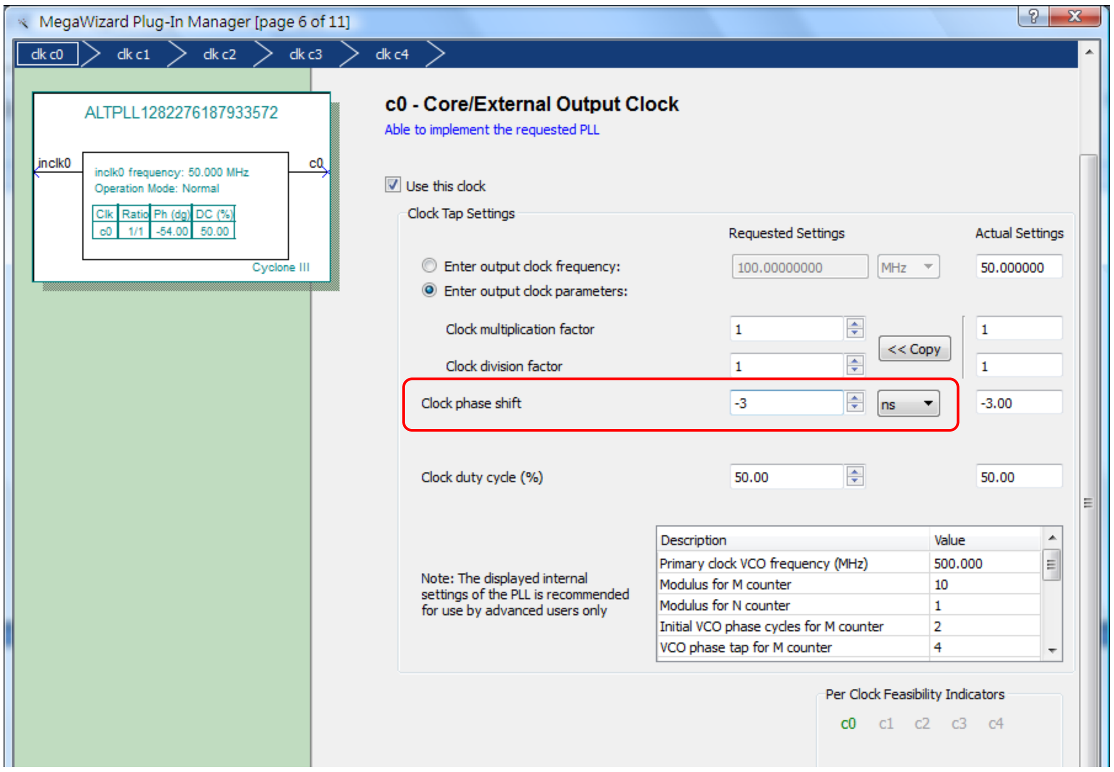
Optional Inputs
☐ Create an 'pllena' input to selectively enable the PLL
☐ Create an 'areset' input to asynchronously reset the PLL
☐ Create an 'pfdena' input to selectively enable the phase/frequency detector

Lock Output
☐ Create 'locked' output
☐ Enable self-reset on loss lock

Advanced Parameters
Using these parameters is recommended for advanced users only
☐ Create output file(s) using the 'Advanced' PLL parameters
- Configurations with output clock(s) that use cascade counters are not supported

Avalon Bus connectivity
☐ Use a separate clock input for Avalon bus connections

2-7-3 Clock Tap Settings (Clock phase shift: -3ns)


MegaWizard Plug-In Manager [page 6 of 11]

c0 - Core/External Output Clock

Able to implement the requested PLL

ALTPLL1282276187933572

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	-54.00	50.00

☑ Use this clock

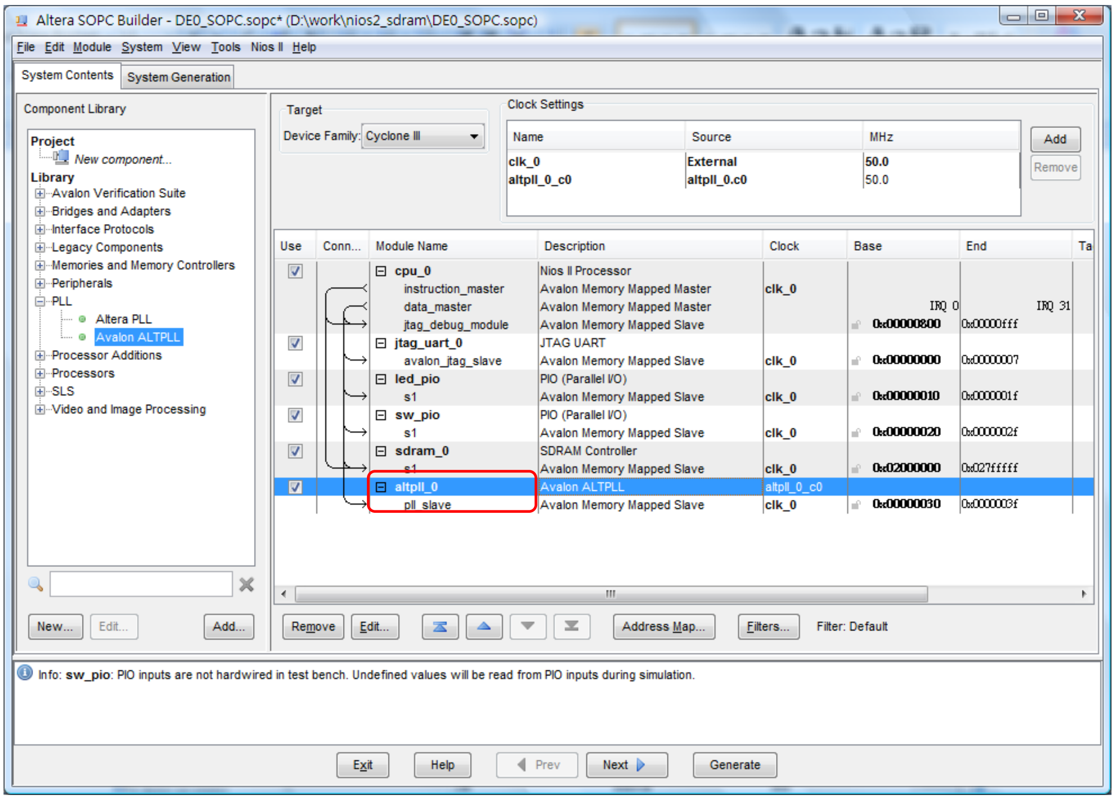
Clock Tap Settings

	Requested Settings		Actual Settings
○ Enter output clock frequency:	100.00000000	MHz	50.000000
◉ Enter output clock parameters:			
Clock multiplication factor	1	<< Copy	1
Clock division factor	1		1
Clock phase shift	-3	ns	-3.00
Clock duty cycle (%)	50.00		50.00

Note: The displayed internal settings of the PLL is recommended for use by advanced users only

Description	Value
Primary clock VCO frequency (MHz)	500.000
Modulus for M counter	10
Modulus for N counter	1
Initial VCO phase cycles for M counter	2
VCO phase tap for M counter	4

Per Clock Feasibility Indicators


Altera SOPC Builder - DE0_SOPC.sopc* (D:\work\nios2_sdram\DE0_SOPC.sopc)

Target — Device Family: Cyclone III

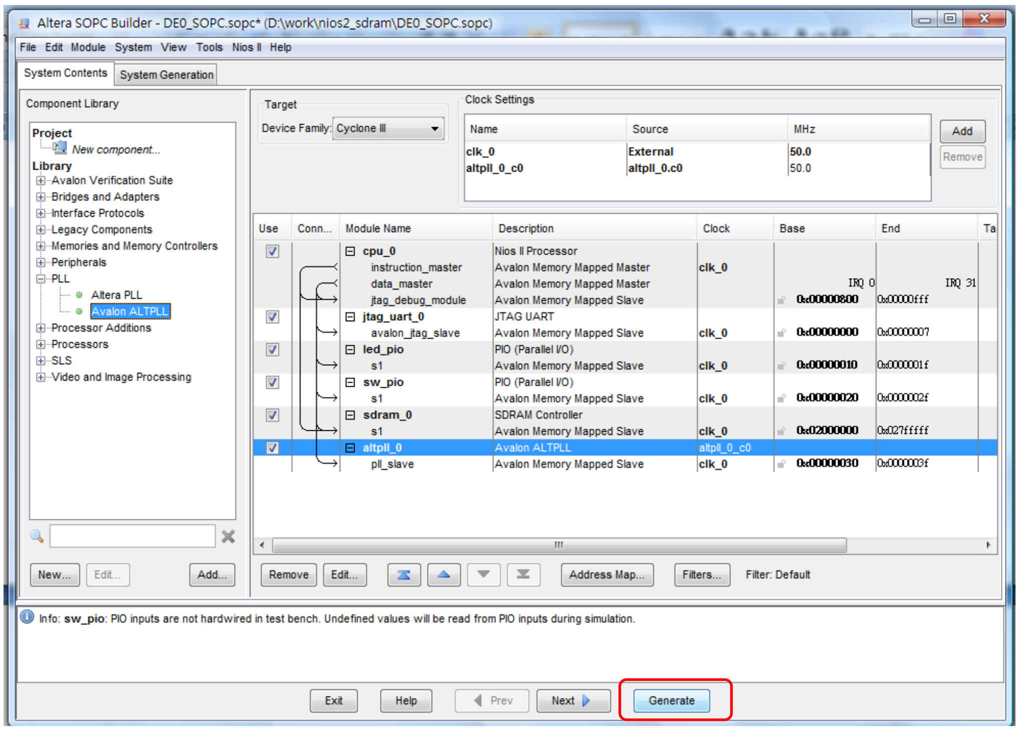
Clock Settings

Name	Source	MHz
clk_0	External	50.0
altpll_0_c0	altpll_0.c0	50.0

Use	Module Name	Description	Clock	Base	End
☑	cpu_0	Nios II Processor			
	instruction_master	Avalon Memory Mapped Master	clk_0		
	data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31
	jtag_debug_module	Avalon Memory Mapped Slave		0x00000800	0x00000fff
☑	jtag_uart_0	JTAG UART			
	avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00000000	0x00000007
☑	led_pio	PIO (Parallel I/O)			
	s1	Avalon Memory Mapped Slave	clk_0	0x00000010	0x0000001f
☑	sw_pio	PIO (Parallel I/O)			
	s1	Avalon Memory Mapped Slave	clk_0	0x00000020	0x0000002f
☑	sdram_0	SDRAM Controller			
	s1	Avalon Memory Mapped Slave	clk_0	0x02000000	0x027fffff
☑	altpll_0	Avalon ALTPLL	altpll_0_c0		
	pll_slave	Avalon Memory Mapped Slave	clk_0	0x00000030	0x0000003f

Info: sw_pio: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

2-8 Click the **Generate** tab. The system generation process begins. The generation process can take several minutes. When it completes, the System Generation tab displays a message "**SUCCESS: SYSTEM GENERATION COMPLETED**".



Device Family: Cyclone III

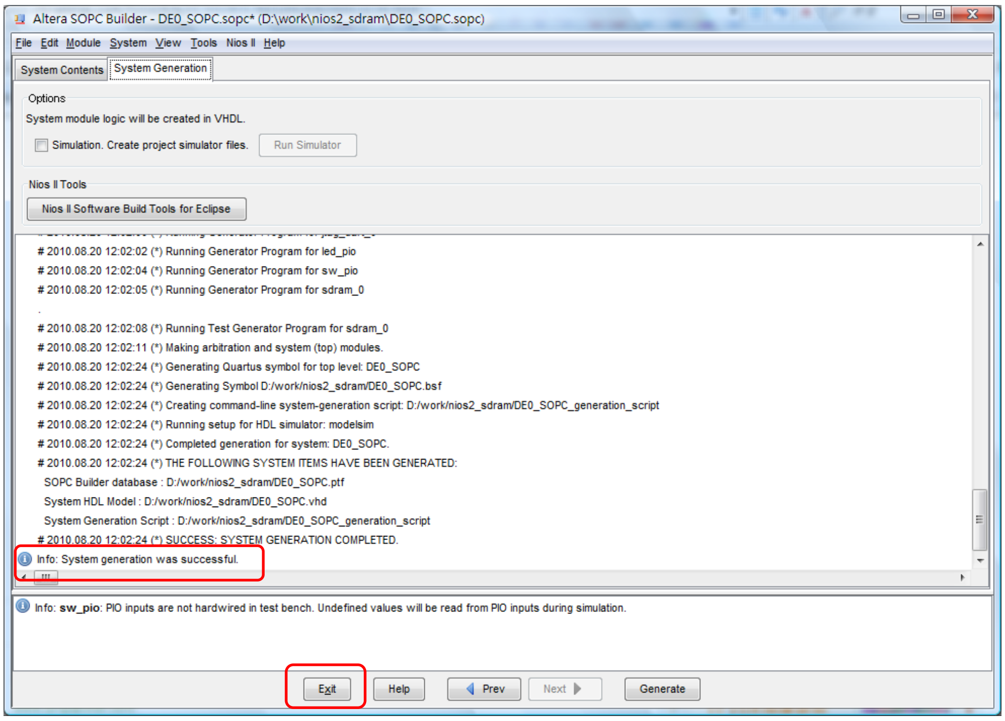
Clock Settings

Name	Source	MHz
clk_0	External	50.0
altpll_0_c0	altpll_0.c0	50.0

Use	Conn...	Module Name	Description	Clock	Base	End
✓		cpu_0	Nios II Processor			
		instruction_master	Avalon Memory Mapped Master	clk_0		
		data_master	Avalon Memory Mapped Master		IRQ 0	IRQ 31
		jtag_debug_module	Avalon Memory Mapped Slave		0x00000800	0x00000fff
✓		jtag_uart_0	JTAG UART			
		avalon_jtag_slave	Avalon Memory Mapped Slave	clk_0	0x00000000	0x00000007
✓		led_pio	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	clk_0	0x00000010	0x0000001f
✓		sw_pio	PIO (Parallel I/O)			
		s1	Avalon Memory Mapped Slave	clk_0	0x00000020	0x0000002f
✓		sdram_0	SDRAM Controller			
		s1	Avalon Memory Mapped Slave	clk_0	0x02000000	0x027fffff
✓		altpll_0	Avalon ALTPLL	altpll_0_c0		
		pll_slave	Avalon Memory Mapped Slave	clk_0	0x00000030	0x0000003f

Info: **sw_pio**: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

Fig. 13 Click **Generate**


Altera SOPC Builder - DE0_SOPC.sopc* (D:\work\nios2_sdram\DE0_SOPC.sopc)
File Edit Module System View Tools Nios II Help
System Contents
System Generation
Options
Run Simulator
Nios II Tools
Nios II Software Build Tools for Eclipse
Exit
Help
Prev
Next
Generate

System module logic will be created in VHDL.

☐ Simulation. Create project simulator files.

```
# 2010.08.20 12:02:02 (*) Running Generator Program for led_pio
# 2010.08.20 12:02:04 (*) Running Generator Program for sw_pio
# 2010.08.20 12:02:05 (*) Running Generator Program for sdram_0
.
# 2010.08.20 12:02:08 (*) Running Test Generator Program for sdram_0
# 2010.08.20 12:02:11 (*) Making arbitration and system (top) modules.
# 2010.08.20 12:02:24 (*) Generating Quartus symbol for top level: DE0_SOPC
# 2010.08.20 12:02:24 (*) Generating Symbol D:/work/nios2_sdram/DE0_SOPC.bsf
# 2010.08.20 12:02:24 (*) Creating command-line system-generation script: D:/work/nios2_sdram/DE0_SOPC_generation_script
# 2010.08.20 12:02:24 (*) Running setup for HDL simulator: modelsim
# 2010.08.20 12:02:24 (*) Completed generation for system: DE0_SOPC.
# 2010.08.20 12:02:24 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED:
  SOPC Builder database : D:/work/nios2_sdram/DE0_SOPC.ptf
  System HDL Model : D:/work/nios2_sdram/DE0_SOPC.vhd
  System Generation Script : D:/work/nios2_sdram/DE0_SOPC_generation_script
# 2010.08.20 12:02:24 (*) SUCCESS: SYSTEM GENERATION COMPLETED.
Info: System generation was successful.
```

Info: **sw_pio**: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

3. Integration of the Nios II System into the Quartus II Project

3-1 Adding the Quartus II Symbol to the BDF:

3-1-1 File>New>Block Diagram/Schematic File

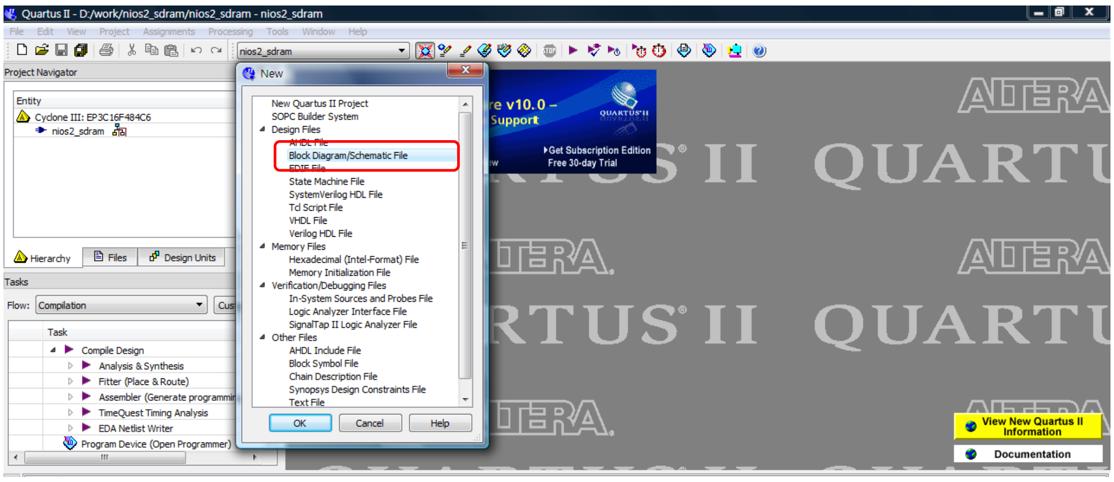


Fig.14 File>New>Block Diagram/Schematic File

3-1-2 Select Project >DE2_SOPC. The Symbol dialog box displays the DE0_SOPC symbol. Click OK.

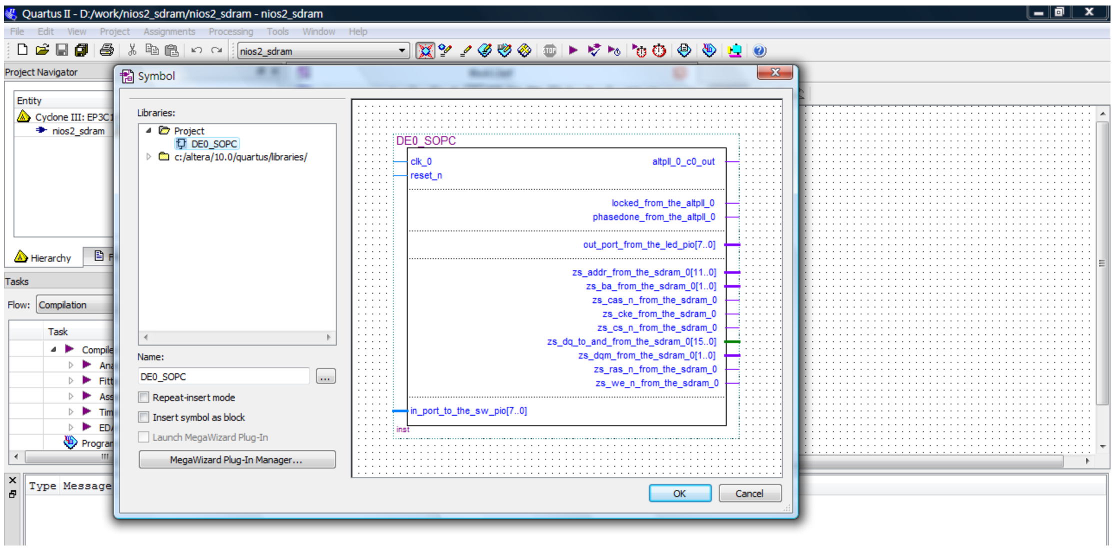
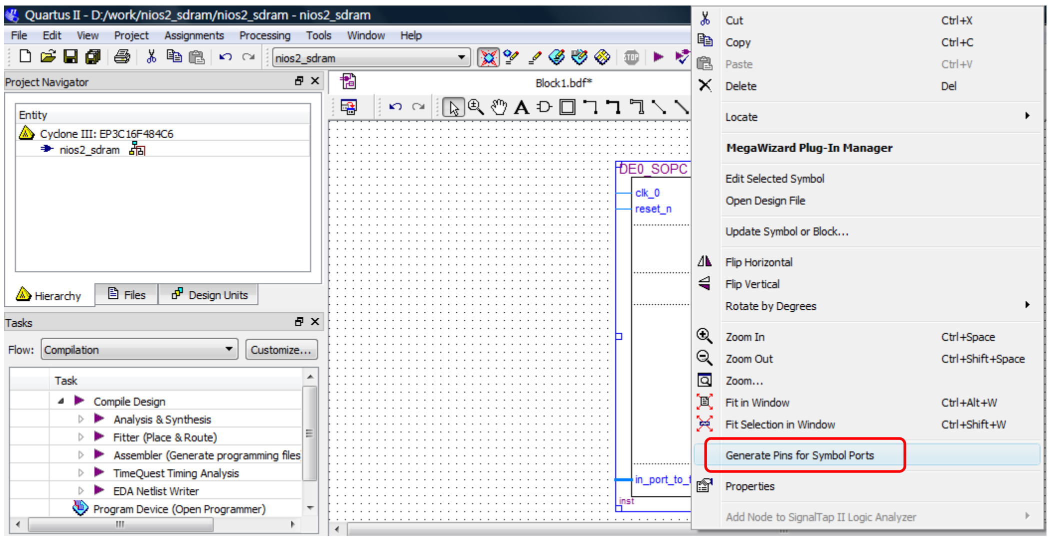
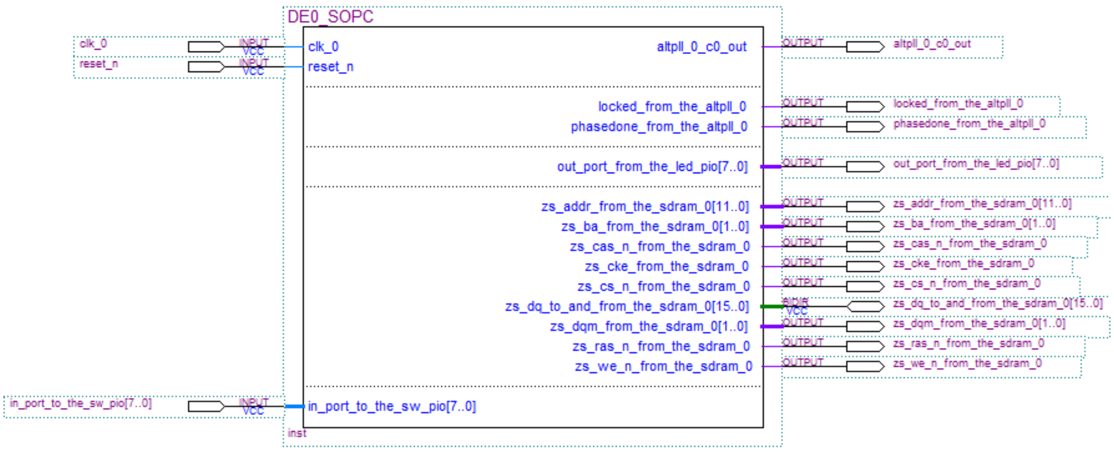

Symbol
Libraries:
Project
DE0_SOPC
c:/altera/10.0/quartus/libraries/
Name:
DE0_SOPC
Repeat-insert mode
Insert symbol as block
Launch MegaWizard Plug-In
MegaWizard Plug-In Manager...
DE0_SOPC
clk_0
reset_n
altpll_0_c0_out
locked_from_the_altpll_0
phasedone_from_the_altpll_0
out_port_from_the_led_pio[7..0]
zs_addr_from_the_sdram_0[11..0]
zs_ba_from_the_sdram_0[1..0]
zs_cas_n_from_the_sdram_0
zs_cke_from_the_sdram_0
zs_cs_n_from_the_sdram_0
zs_dq_to_and_from_the_sdram_0[15..0]
zs_dqm_from_the_sdram_0[1..0]
zs_ras_n_from_the_sdram_0
zs_we_n_from_the_sdram_0
in_port_to_the_sw_pio[7..0]
inst
OK
Cancel

Fig.15 Add DE0_SOPC symbol

3-1-3 Right-click **DE0_SOPC** symbol and select “Generate pins for Symbol ports”:




DE0_SOPC
clk_0
reset_n
altpll_0_c0_out
locked_from_the_altpll_0
phasedone_from_the_altpll_0
out_port_from_the_led_pio[7..0]
zs_addr_from_the_sdram_0[11..0]
zs_ba_from_the_sdram_0[1..0]
zs_cas_n_from_the_sdram_0
zs_cke_from_the_sdram_0
zs_cs_n_from_the_sdram_0
zs_dq_to_and_from_the_sdram_0[15..0]
zs_dqm_from_the_sdram_0[1..0]
zs_ras_n_from_the_sdram_0
zs_we_n_from_the_sdram_0
in_port_to_the_sw_pio[7..0]
inst

3-1-4 Rename Symbol ports:

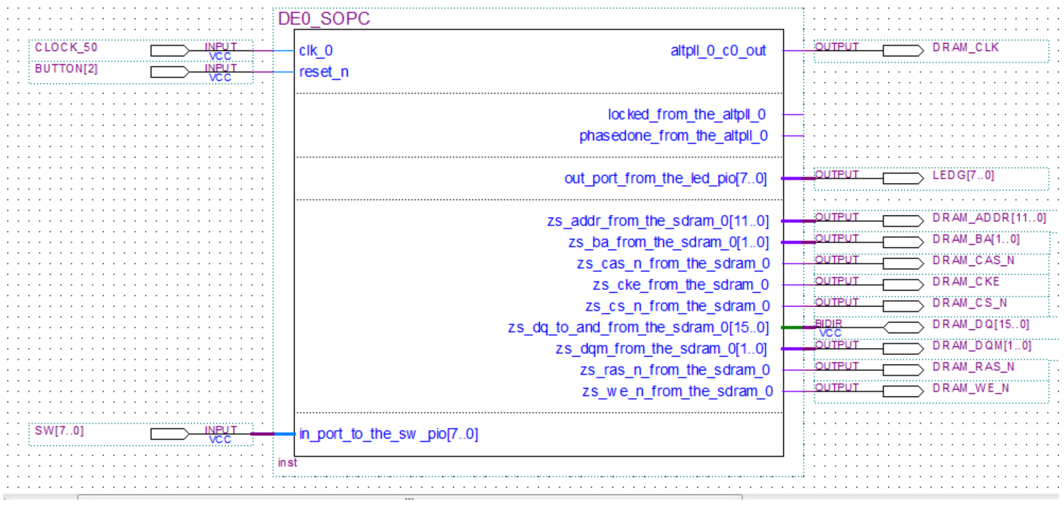
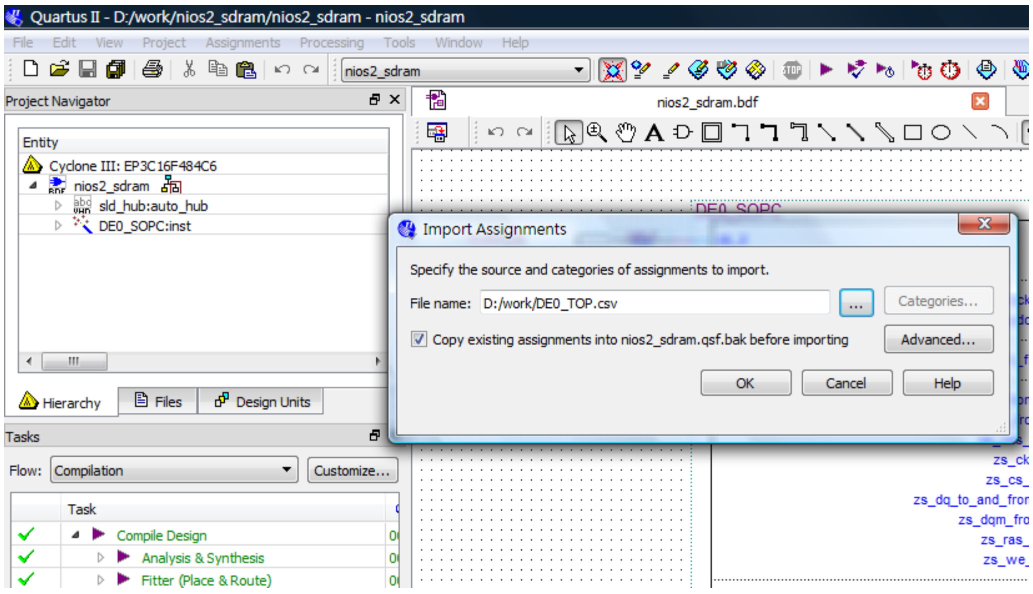


Fig.17 Rename Symbol ports

3-2 Compilier the Design.

4. Pin Assignments & Download

4-1 Choose **Assignments** > **Import Assignments**



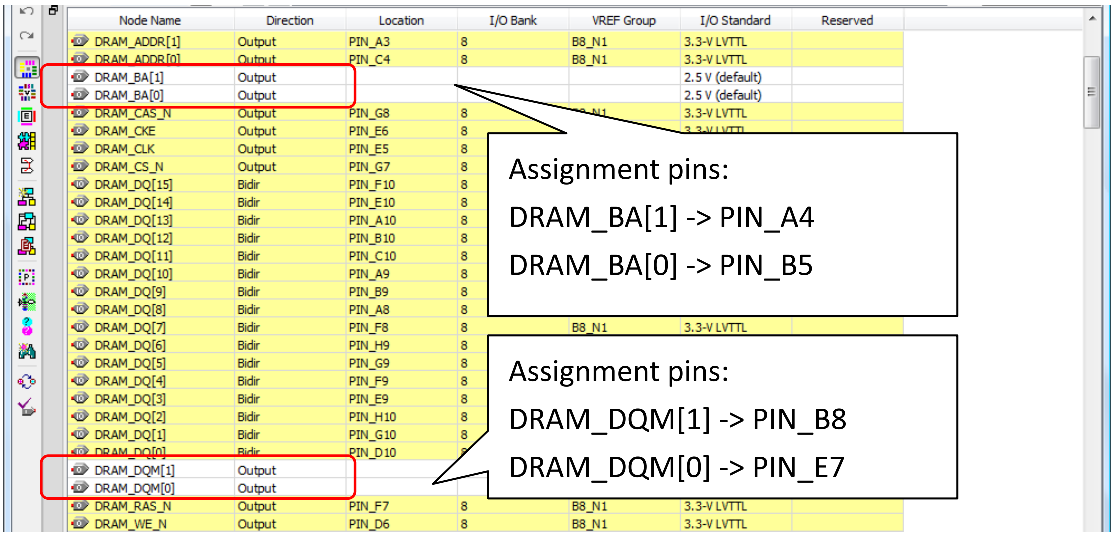
Specify the source and categories of assignments to import.

File name: D:/work/DE0_TOP.csv

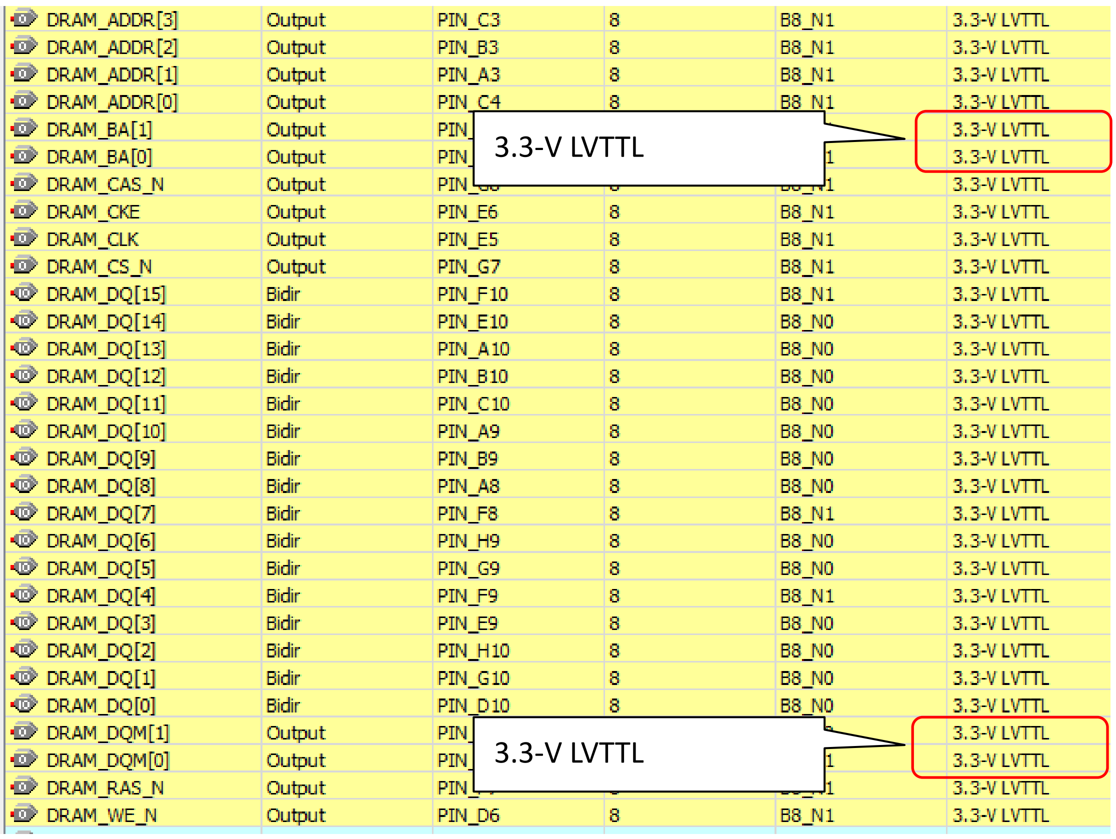
[✓] Copy existing assignments into nios2_sdram.qsf.bak before importing

Fig.20 Specify assignment source

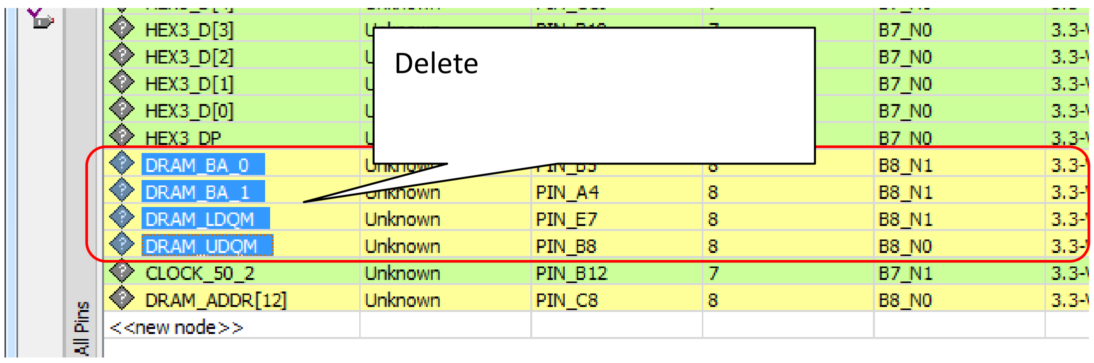
4-2 Modify the pin assignments of DRAM_BA[1..0] & DRAM_DQM[1..0]



Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
DRAM_ADDR[1]	Output	PIN_A3	8	B8_N1	3.3-V LVTTL	
DRAM_ADDR[0]	Output	PIN_C4	8	B8_N1	3.3-V LVTTL	
DRAM_BA[1]	Output				2.5 V (default)	
DRAM_BA[0]	Output				2.5 V (default)	
DRAM_CAS_N	Output	PIN_G8	8	B8_N1	3.3-V LVTTL	
DRAM_CKE	Output	PIN_E6	8		3.3-V LVTTL	
DRAM_CLK	Output	PIN_E5	8			
DRAM_CS_N	Output	PIN_G7	8			
DRAM_DQ[15]	Bidir	PIN_F10	8			
DRAM_DQ[14]	Bidir	PIN_E10	8			
DRAM_DQ[13]	Bidir	PIN_A10	8			
DRAM_DQ[12]	Bidir	PIN_B10	8			
DRAM_DQ[11]	Bidir	PIN_C10	8			
DRAM_DQ[10]	Bidir	PIN_A9	8			
DRAM_DQ[9]	Bidir	PIN_B9	8			
DRAM_DQ[8]	Bidir	PIN_A8	8			
DRAM_DQ[7]	Bidir	PIN_F8	8	B8_N1	3.3-V LVTTL	
DRAM_DQ[6]	Bidir	PIN_H9	8			
DRAM_DQ[5]	Bidir	PIN_G9	8			
DRAM_DQ[4]	Bidir	PIN_F9	8			
DRAM_DQ[3]	Bidir	PIN_E9	8			
DRAM_DQ[2]	Bidir	PIN_H10	8			
DRAM_DQ[1]	Bidir	PIN_G10	8			
DRAM_DQ[0]	Bidir	PIN_D10	8			
DRAM_DQM[1]	Output					
DRAM_DQM[0]	Output					
DRAM_RAS_N	Output	PIN_F7	8	B8_N1	3.3-V LVTTL	
DRAM_WE_N	Output	PIN_D6	8	B8_N1	3.3-V LVTTL	



DRAM_ADDR[3]	Output	PIN_C3	8	B8_N1	3.3-V LVTTL
DRAM_ADDR[2]	Output	PIN_B3	8	B8_N1	3.3-V LVTTL
DRAM_ADDR[1]	Output	PIN_A3	8	B8_N1	3.3-V LVTTL
DRAM_ADDR[0]	Output	PIN_C4	8	B8_N1	3.3-V LVTTL
DRAM_BA[1]	Output	PIN_			3.3-V LVTTL
DRAM_BA[0]	Output	PIN_		1	3.3-V LVTTL
DRAM_CAS_N	Output	PIN_		1	3.3-V LVTTL
DRAM_CKE	Output	PIN_E6	8	B8_N1	3.3-V LVTTL
DRAM_CLK	Output	PIN_E5	8	B8_N1	3.3-V LVTTL
DRAM_CS_N	Output	PIN_G7	8	B8_N1	3.3-V LVTTL
DRAM_DQ[15]	Bidir	PIN_F10	8	B8_N1	3.3-V LVTTL
DRAM_DQ[14]	Bidir	PIN_E10	8	B8_N0	3.3-V LVTTL
DRAM_DQ[13]	Bidir	PIN_A10	8	B8_N0	3.3-V LVTTL
DRAM_DQ[12]	Bidir	PIN_B10	8	B8_N0	3.3-V LVTTL
DRAM_DQ[11]	Bidir	PIN_C10	8	B8_N0	3.3-V LVTTL
DRAM_DQ[10]	Bidir	PIN_A9	8	B8_N0	3.3-V LVTTL
DRAM_DQ[9]	Bidir	PIN_B9	8	B8_N0	3.3-V LVTTL
DRAM_DQ[8]	Bidir	PIN_A8	8	B8_N0	3.3-V LVTTL
DRAM_DQ[7]	Bidir	PIN_F8	8	B8_N1	3.3-V LVTTL
DRAM_DQ[6]	Bidir	PIN_H9	8	B8_N0	3.3-V LVTTL
DRAM_DQ[5]	Bidir	PIN_G9	8	B8_N0	3.3-V LVTTL
DRAM_DQ[4]	Bidir	PIN_F9	8	B8_N1	3.3-V LVTTL
DRAM_DQ[3]	Bidir	PIN_E9	8	B8_N0	3.3-V LVTTL
DRAM_DQ[2]	Bidir	PIN_H10	8	B8_N0	3.3-V LVTTL
DRAM_DQ[1]	Bidir	PIN_G10	8	B8_N0	3.3-V LVTTL
DRAM_DQ[0]	Bidir	PIN_D10	8	B8_N0	3.3-V LVTTL
DRAM_DQM[1]	Output	PIN_			3.3-V LVTTL
DRAM_DQM[0]	Output	PIN_		1	3.3-V LVTTL
DRAM_RAS_N	Output	PIN_		1	3.3-V LVTTL
DRAM_WE_N	Output	PIN_D6	8	B8_N1	3.3-V LVTTL

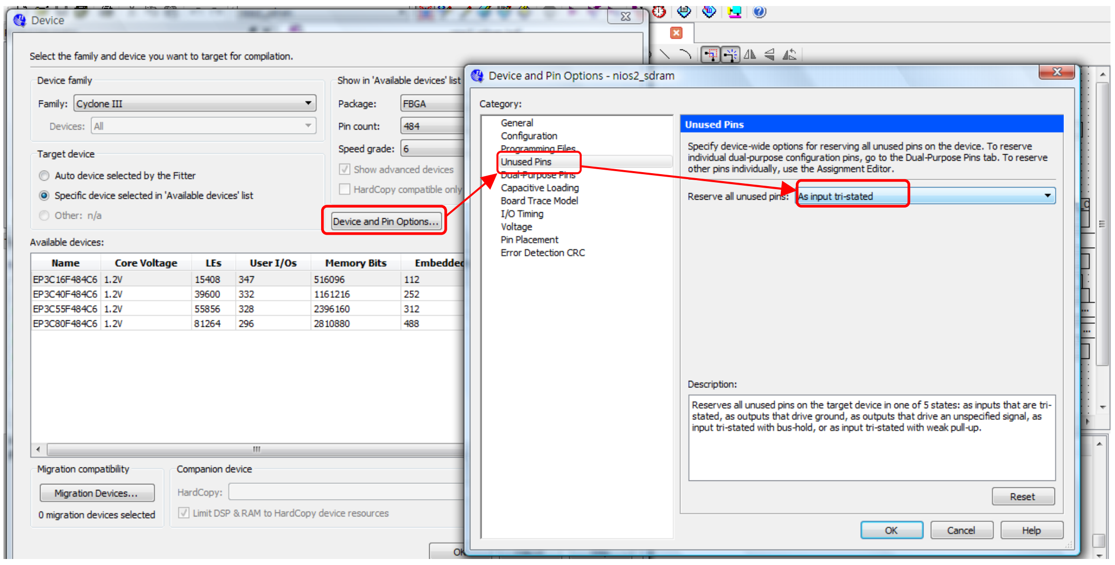


HEX3_D[3]	U			B7_N0	3.3-
HEX3_D[2]	U			B7_N0	3.3-
HEX3_D[1]	U			B7_N0	3.3-
HEX3_D[0]	U			B7_N0	3.3-
HEX3_DP	U			B7_N0	3.3-
DRAM_BA_0				B8_N1	3.3-
DRAM_BA_1	Unknown	PIN_A4	8	B8_N1	3.3-
DRAM_LDQM	Unknown	PIN_E7	8	B8_N1	3.3-
DRAM_UDQM	Unknown	PIN_B8	8	B8_N0	3.3-
CLOCK_50_2	Unknown	PIN_B12	7	B7_N1	3.3-
DRAM_ADDR[12]	Unknown	PIN_C8	8	B8_N0	3.3-
<<new node>>					

4-3 Compiler the design.

4-3-1 Select Assignments > Device > Device and Pin Options

(在 Quartus II 編譯 project 前，先選擇【Assignments】/【Device】開啟 Settings 視窗以進行 Unused Pins 的設定，並將未使用的腳位都設定為高阻抗輸入。)



Device

Select the family and device you want to target for compilation.

Device family

Family: Cyclone III

Devices: All

Target device

Auto device selected by the Fitter

[✓ Specific device selected in 'Available devices' list]

Other: n/a

Show in 'Available devices' list

Package: FBGA

Pin count: 484

Speed grade: 6

Show advanced devices

HardCopy compatible only

Device and Pin Options...

Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded
EP3C16F484C6	1.2V	15408	347	516096	112
EP3C40F484C6	1.2V	39600	332	1161216	252
EP3C55F484C6	1.2V	55856	328	2396160	312
EP3C80F484C6	1.2V	81264	296	2810880	488

Migration compatibility

Migration Devices...

0 migration devices selected

Companion device

HardCopy:

Limit DSP & RAM to HardCopy device resources

Device and Pin Options - nios2_sdram

Category:

General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Capacitive Loading
Board Trace Model
I/O Timing
Voltage
Pin Placement
Error Detection CRC

Unused Pins

Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve other pins individually, use the Assignment Editor.

Reserve all unused pins: As input tri-stated

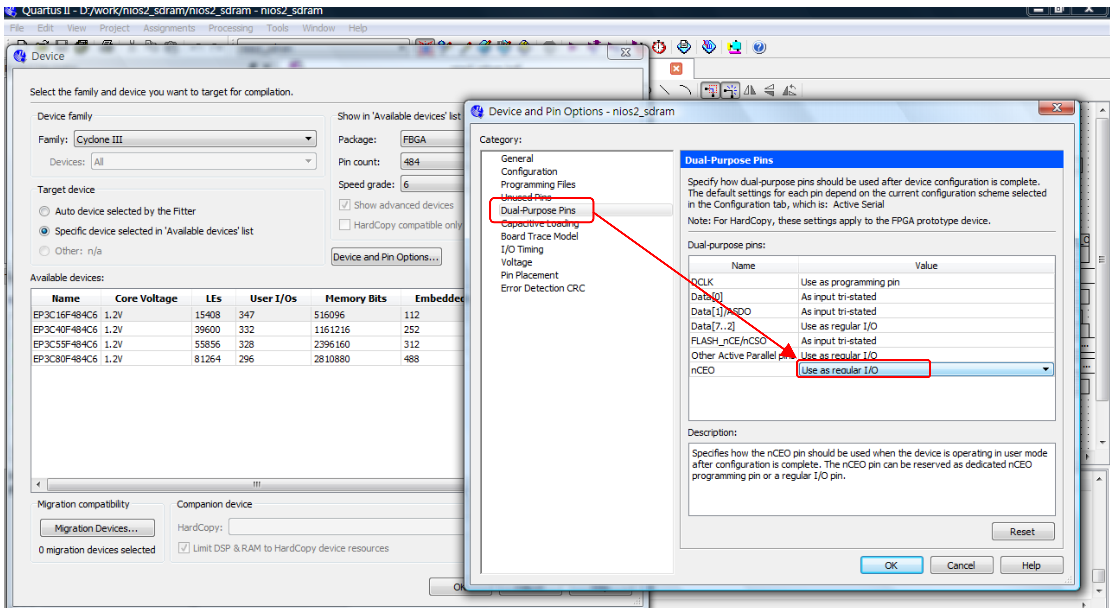
Description:

Reserves all unused pins on the target device in one of 5 states: as inputs that are tri-stated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri-stated with bus-hold, or as input tri-stated with weak pull-up.

Reset

OK Cancel Help

Setting Unused Pins



Quartus II - D:/work/nios2_sdram/nios2_sdram - nios2_sdram

Device

Select the family and device you want to target for compilation.

Device family

Family: Cyclone III

Devices: All

Target device

Auto device selected by the Fitter

[✓ Specific device selected in 'Available devices' list]

Other: n/a

Show in 'Available devices' list

Package: FBGA

Pin count: 484

Speed grade: 6

Show advanced devices

HardCopy compatible only

Device and Pin Options...

Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded
EP3C16F484C6	1.2V	15408	347	516096	112
EP3C40F484C6	1.2V	39600	332	1161216	252
EP3C55F484C6	1.2V	55856	328	2396160	312
EP3C80F484C6	1.2V	81264	296	2810880	488

Migration compatibility

Migration Devices...

0 migration devices selected

Companion device

HardCopy:

Limit DSP & RAM to HardCopy device resources

Device and Pin Options - nios2_sdram

Category:

General
Configuration
Programming Files
Unused Pins
Dual-Purpose Pins
Capacitive Loading
Board Trace Model
I/O Timing
Voltage
Pin Placement
Error Detection CRC

Dual-Purpose Pins

Specify how dual-purpose pins should be used after device configuration is complete. The default settings for each pin depend on the current configuration scheme selected in the Configuration tab, which is: Active Serial

Note: For HardCopy, these settings apply to the FPGA prototype device.

Dual-purpose pins:

Name	Value
DCLK	Use as programming pin
Data[0]	As input tri-stated
Data[1]/ASDO	As input tri-stated
Data[7..2]	Use as regular I/O
FLASH_nCE/nCSO	As input tri-stated
Other Active Parallel pins	Use as regular I/O
nCEO	Use as regular I/O

Description:

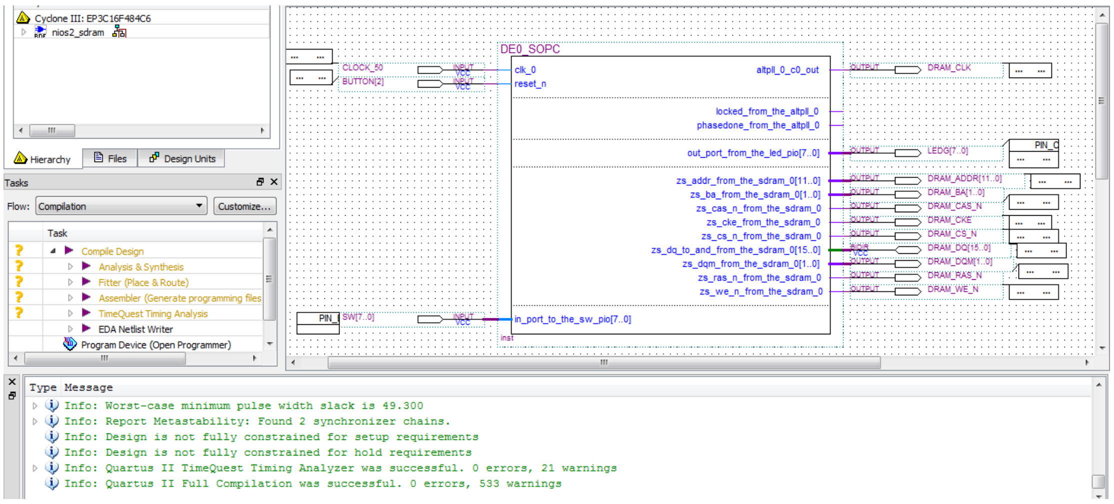
Specifies how the nCEO pin should be used when the device is operating in user mode after configuration is complete. The nCEO pin can be reserved as dedicated nCEO programming pin or a regular I/O pin.

Reset

OK Cancel Help

Setting Dual-Purpose Pins

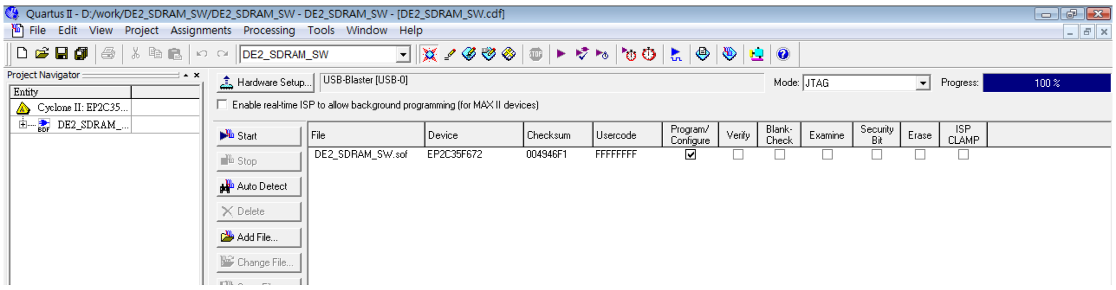
4-3-2 Start Compilation



```
Type Message
Info: Worst-case minimum pulse width slack is 49.300
Info: Report Metastability: Found 2 synchronizer chains.
Info: Design is not fully constrained for setup requirements
Info: Design is not fully constrained for hold requirements
Info: Quartus II TimeQuest Timing Analyzer was successful. 0 errors, 21 warnings
Info: Quartus II Full Compilation was successful. 0 errors, 533 warnings
```

Fig.21 Compiler the design

4-3-3 download the hardware



Quartus II - D:/work/DE2_SDRAM_SW/DE2_SDRAM_SW - DE2_SDRAM_SW - [DE2_SDRAM_SW.cdf]

Hardware Setup... USB-Blaster [USB-0] Mode: JTAG Progress: 100 %

☐ Enable real-time ISP to allow background programming (for MAX II devices)

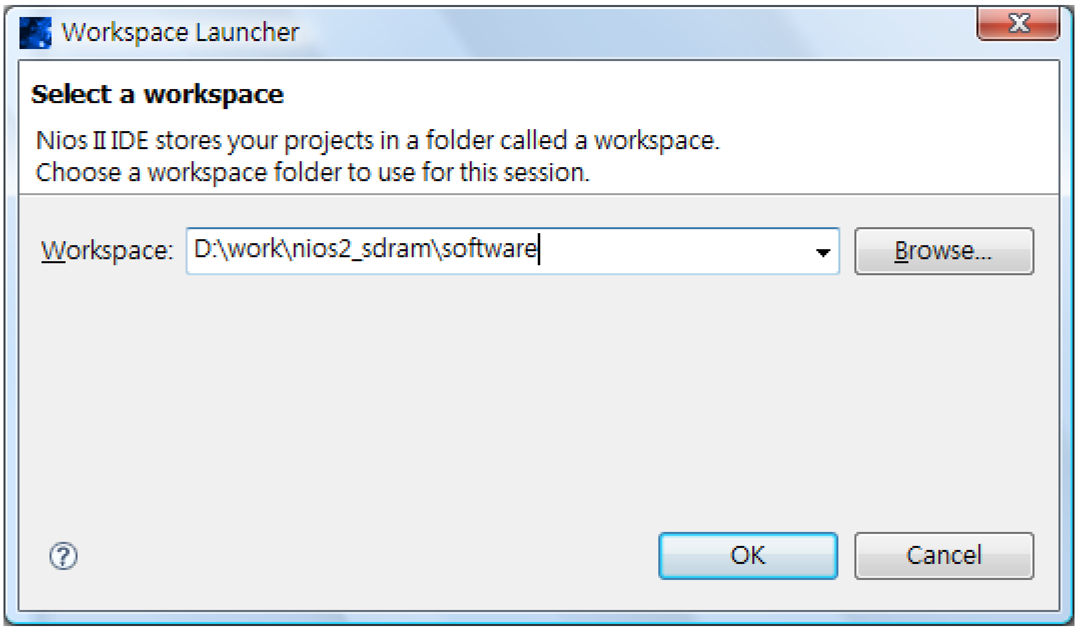
File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
DE2_SDRAM_SW.sof	EP2C35F672	004946F1	FFFFFFFF	☑	☐	☐	☐	☐	☐	☐

Fig.22 download the hardware

三、軟體設計

1. Create a New Nios II IDE Project

1-1 Choose **File** > **switch workspace，設定工作空間於專案下**



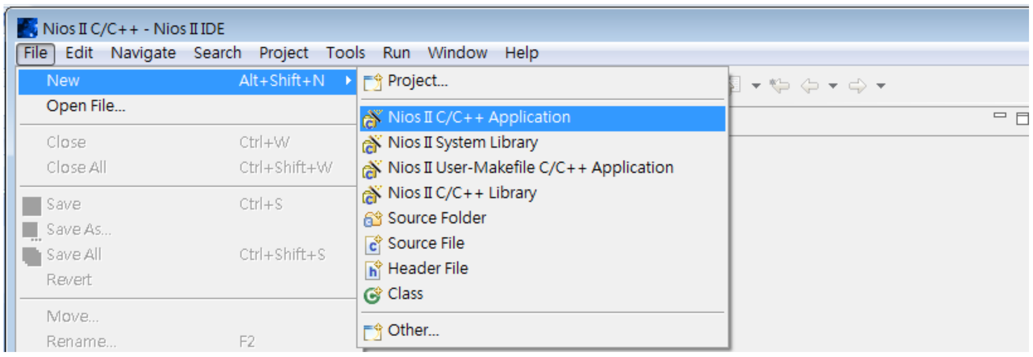
Workspace Launcher

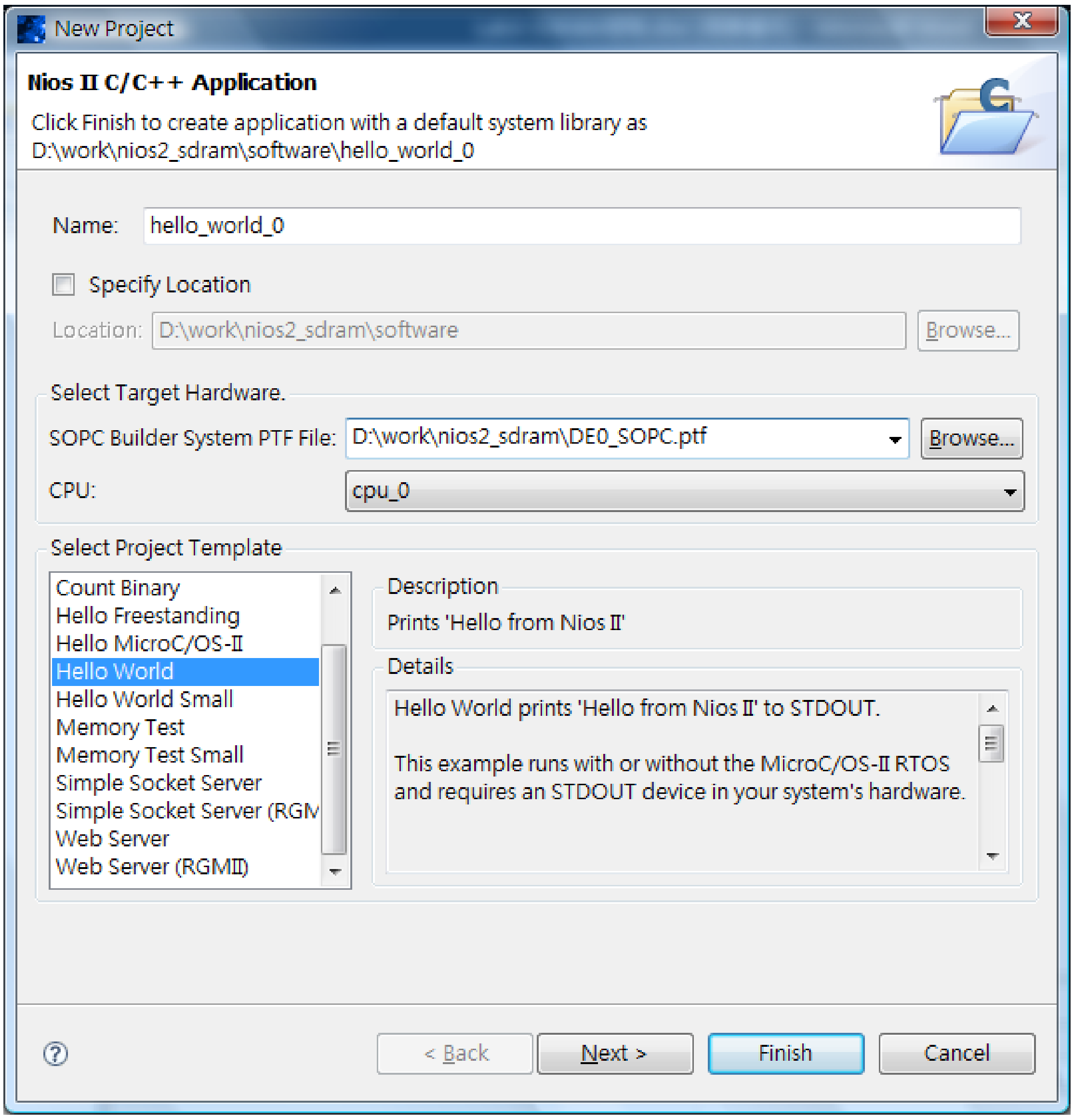
Select a workspace

Nios II IDE stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: D:\work\nios2_sdram\software

1-2 Choose **File** >**New** >**Nios C/C++ Application**. The first page of New Project wizard opens.




New Project

Nios II C/C++ Application

Click Finish to create application with a default system library as D:\work\nios2_sdram\software\hello_world_0

Name: hello_world_0

Specify Location

Location: D:\work\nios2_sdram\software

Select Target Hardware.

SOPC Builder System PTF File: D:\work\nios2_sdram\DE0_SOPC.ptf

CPU: cpu_0

Select Project Template

- Count Binary
- Hello Freestanding
- Hello MicroC/OS-II
- Hello World
- Hello World Small
- Memory Test
- Memory Test Small
- Simple Socket Server
- Simple Socket Server (RGM
- Web Server
- Web Server (RGMII)

Description

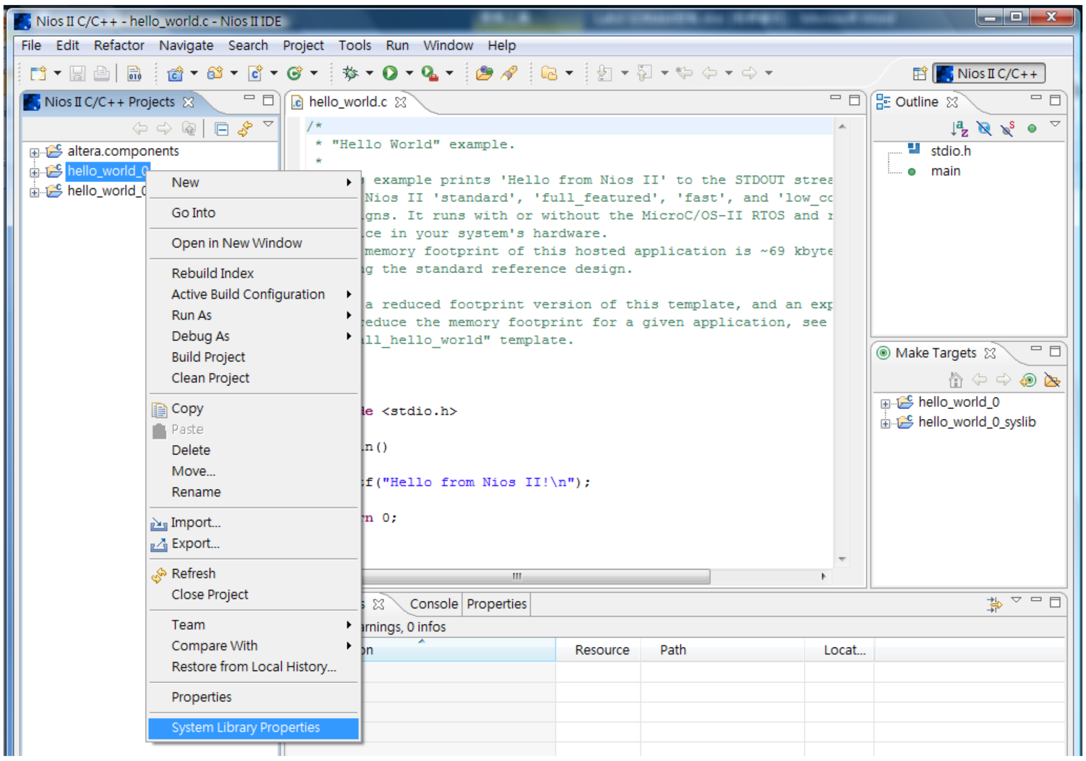
Prints 'Hello from Nios II'

Details

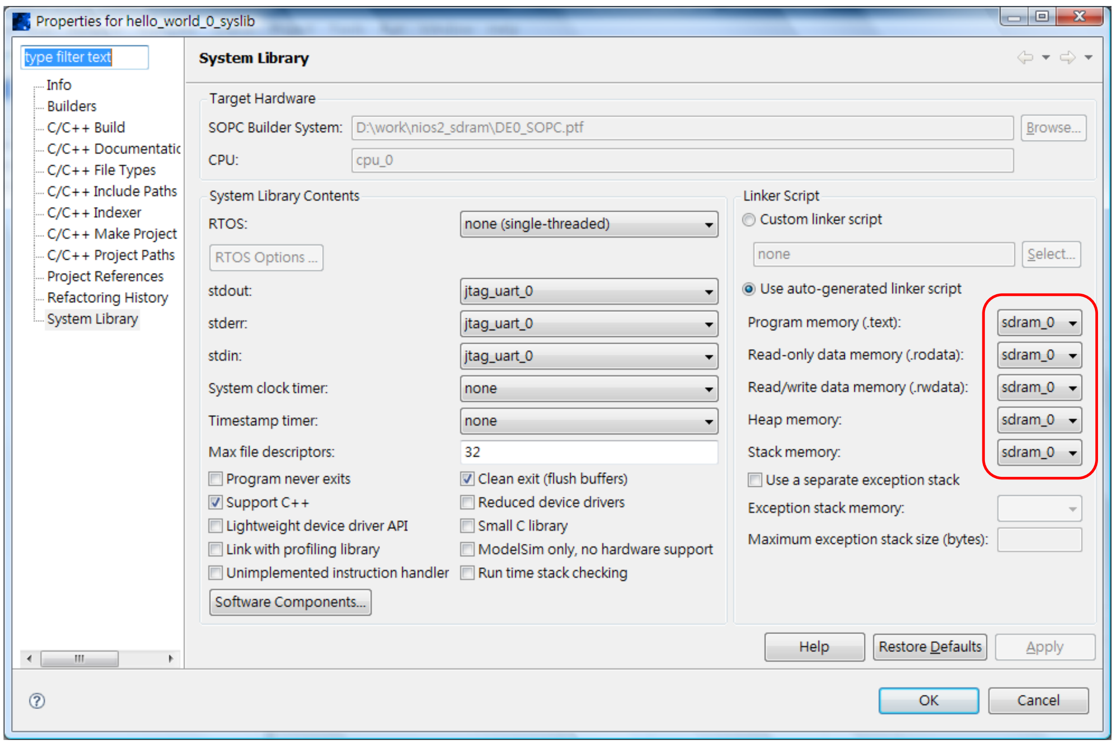
Hello World prints 'Hello from Nios II' to STDOUT.

This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware.

1-3 Right-click and select Select "**System Library Properties":**



```
/*
 * "Hello World" example.
 *
  example prints 'Hello from Nios II' to the STDOUT strea
 Nios II 'standard', 'full_featured', 'fast', and 'low_cc
 gns. It runs with or without the MicroC/OS-II RTOS and r
 ce in your system's hardware.
 memory footprint of this hosted application is ~69 kbyte
 g the standard reference design.

 a reduced footprint version of this template, and an exp
 educe the memory footprint for a given application, see
 ll_hello_world" template.

 e <stdio.h>

 n()

 f("Hello from Nios II!\n");

 n 0;
```



System Library

Target Hardware

SOPC Builder System: D:\work\nios2_sdram\DE0_SOPC.ptf

CPU: cpu_0

System Library Contents

RTOS:	none (single-threaded)
stdout:	jtag_uart_0
stderr:	jtag_uart_0
stdin:	jtag_uart_0
System clock timer:	none
Timestamp timer:	none
Max file descriptors:	32

- [ ] Program never exits
- [✓] Clean exit (flush buffers)
- [✓] Support C++
- [ ] Reduced device drivers
- [ ] Lightweight device driver API
- [ ] Small C library
- [ ] Link with profiling library
- [ ] ModelSim only, no hardware support
- [ ] Unimplemented instruction handler
- [ ] Run time stack checking

Linker Script

Custom linker script [✓ Use auto-generated linker script]

none

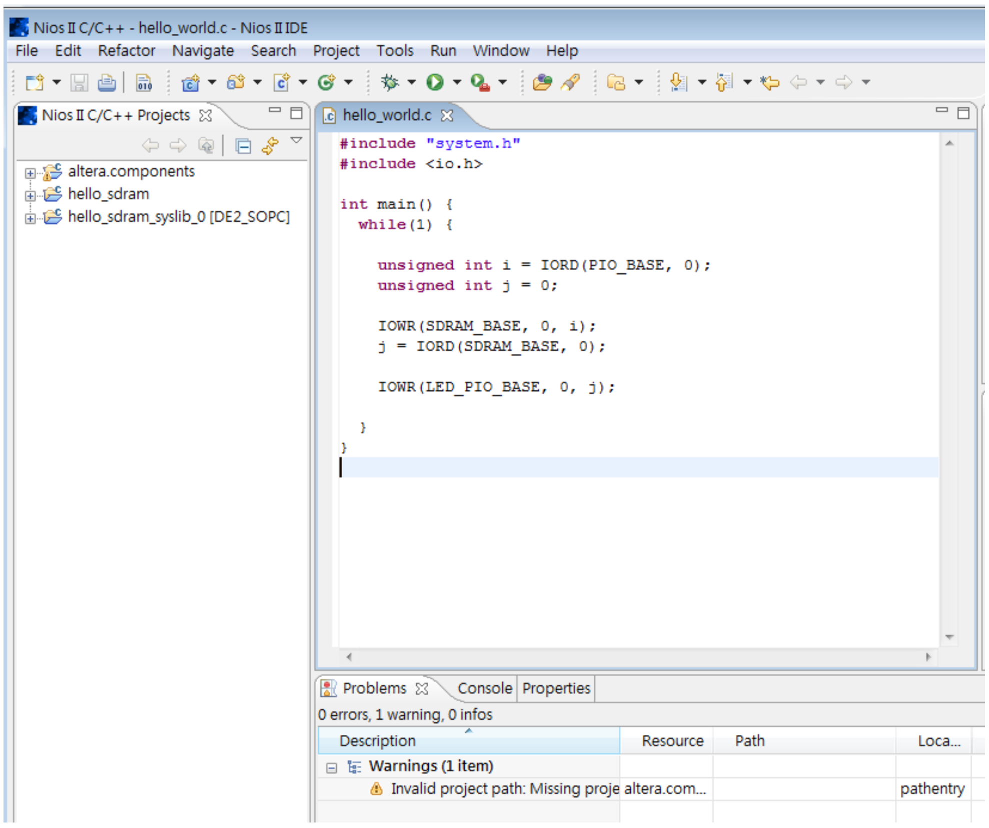
Program memory (.text):	sdram_0
Read-only data memory (.rodata):	sdram_0
Read/write data memory (.rwdata):	sdram_0
Heap memory:	sdram_0
Stack memory:	sdram_0

- [ ] Use a separate exception stack

Exception stack memory:

Maximum exception stack size (bytes):

1-4 hello_world.c


Nios II C/C++ - hello_world.c - Nios II IDE
File Edit Refactor Navigate Search Project Tools Run Window Help
Nios II C/C++ Projects
altera.components
hello_sdram
hello_sdram_syslib_0 [DE2_SOPC]
hello_world.c
Problems
Console
Properties

```
#include "system.h"
#include <io.h>

int main() {
  while(1) {

    unsigned int i = IORD(PIO_BASE, 0);
    unsigned int j = 0;

    IOWR(SDRAM_BASE, 0, i);
    j = IORD(SDRAM_BASE, 0);

    IOWR(LED_PIO_BASE, 0, j);

  }
}
```

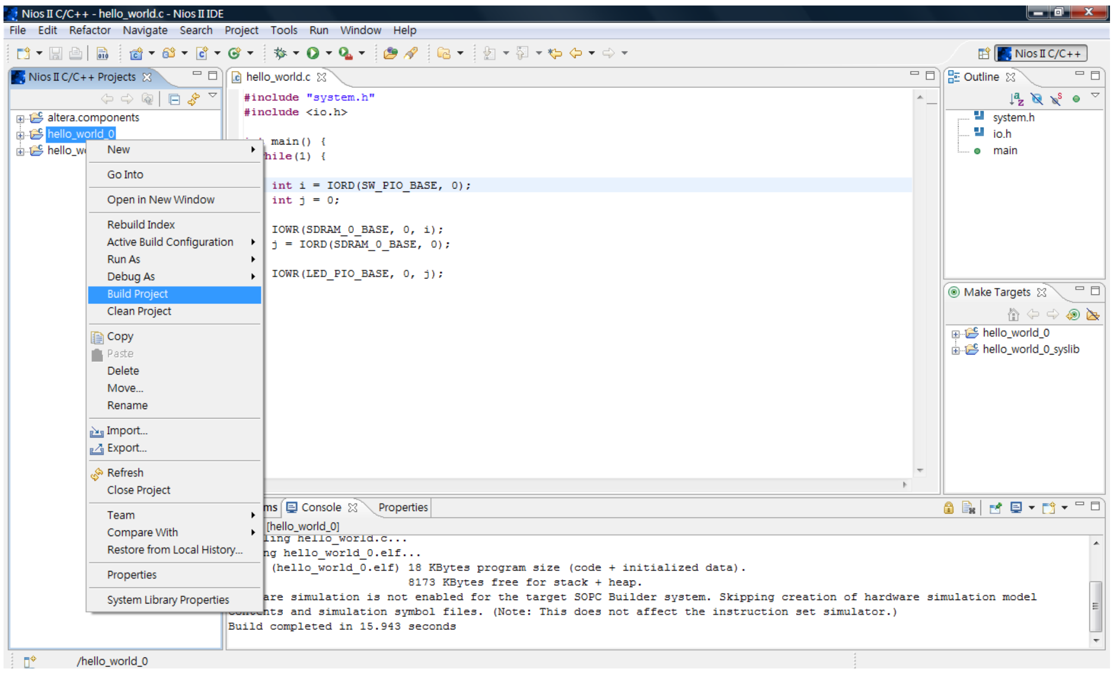
0 errors, 1 warning, 0 infos

Description	Resource	Path	Loca...
Warnings (1 item)			
Invalid project path: Missing proje	altera.com...		pathentry

Fig.25 hello_world.c

```
#include "system.h"
#include <io.h>

int main() {
  while(1) {

    int i = IORD(SW_PIO_BASE, 0);
    int j = 0;

    IOWR(SDRAM_0_BASE, 0, i);
    j = IORD(SDRAM_0_BASE, 0);

    IOWR(LED_PIO_BASE, 0, j);

  }
  return 0;
}
```

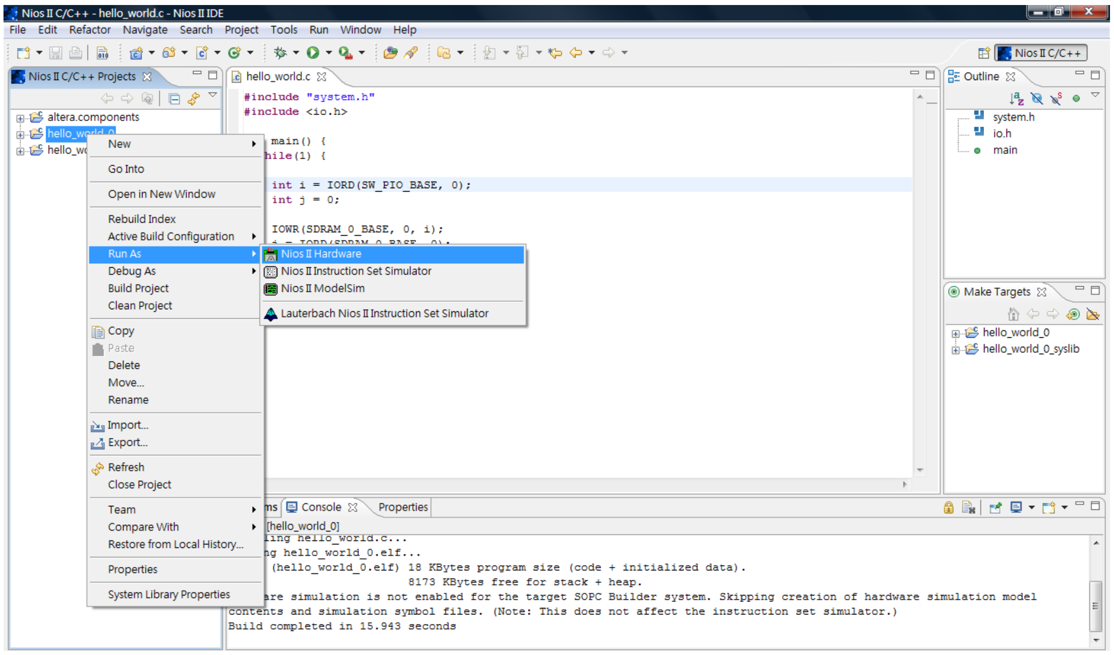
1-5 Build Project



```
#include "system.h"
#include <io.h>

    main() {
   hile(1) {

    int i = IORD(SW_PIO_BASE, 0);
    int j = 0;

    IOWR(SDRAM_0_BASE, 0, i);
    j = IORD(SDRAM_0_BASE, 0);

    IOWR(LED_PIO_BASE, 0, j);
```

```
[hello_world_0]
 ling hello_world.c...
 ng hello_world_0.elf...
  (hello_world_0.elf) 18 KBytes program size (code + initialized data).
                      8173 KBytes free for stack + heap.
 are simulation is not enabled for the target SOPC Builder system. Skipping creation of hardware simulation model
 ents and simulation symbol files. (Note: This does not affect the instruction set simulator.)
Build completed in 15.943 seconds
```

1-6 Run As > Nios II Hardware



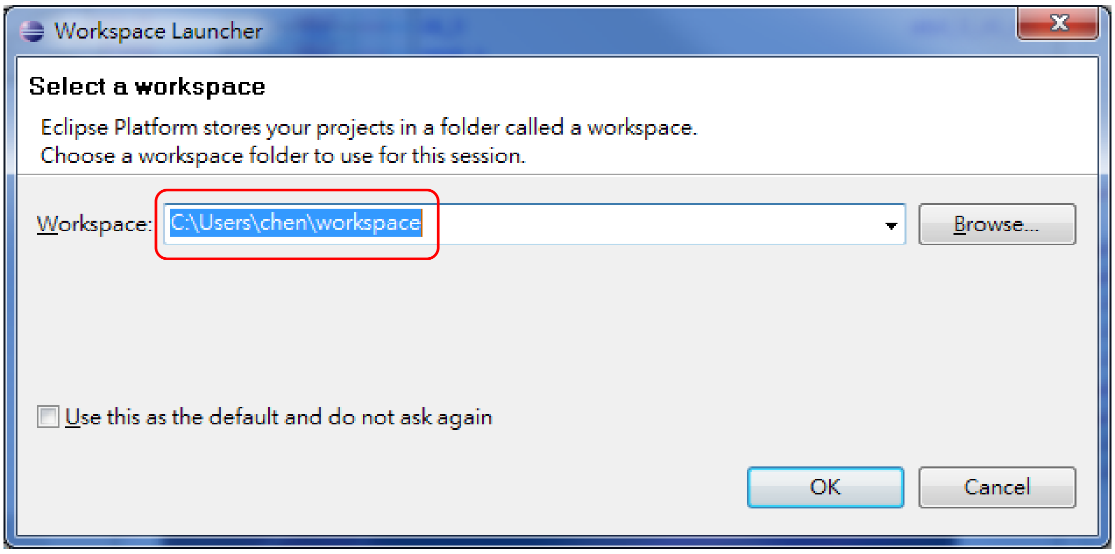
```
#include "system.h"
#include <io.h>

    main() {
   hile(1) {

    int i = IORD(SW_PIO_BASE, 0);
    int j = 0;

    IOWR(SDRAM_0_BASE, 0, i);
```

```
[hello_world_0]
 ling hello_world.c...
 ng hello_world_0.elf...
  (hello_world_0.elf) 18 KBytes program size (code + initialized data).
                      8173 KBytes free for stack + heap.
 are simulation is not enabled for the target SOPC Builder system. Skipping creation of hardware simulation model
 ents and simulation symbol files. (Note: This does not affect the instruction set simulator.)
Build completed in 15.943 seconds
```

新版 Nios II 軟體的使用

1. Create a New Nios II EDS

1-1 接受預設之 Workspace，無需再更改。



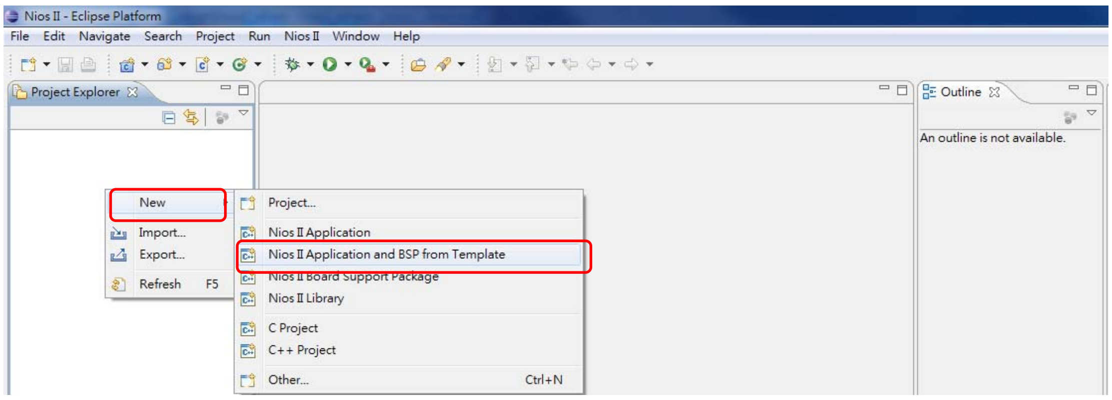
Workspace Launcher

Select a workspace

Eclipse Platform stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

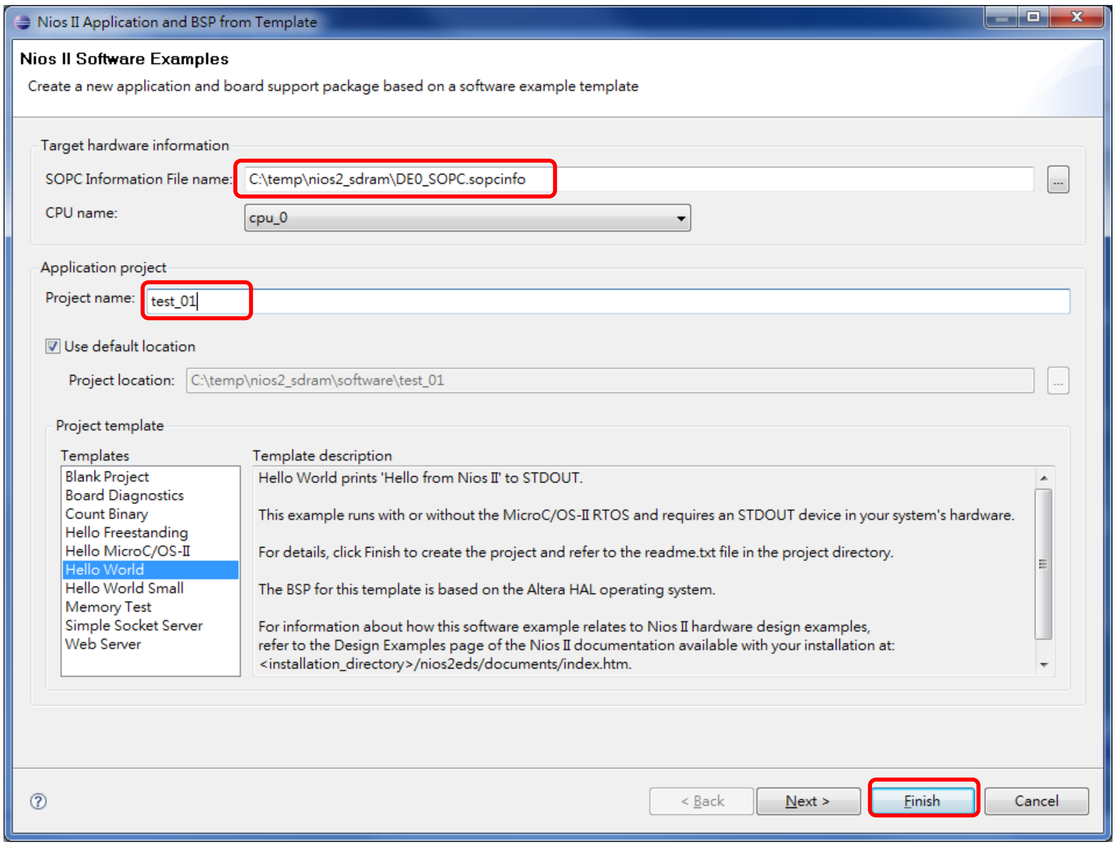
Workspace: C:\Users\chen\workspace

1-2 按滑鼠右鍵，Choose **New** >**Nios II Application and BSP from Template**.



Nios II - Eclipse Platform

An outline is not available.



Nios II Application and BSP from Template

Nios II Software Examples

Create a new application and board support package based on a software example template

Target hardware information

SOPC Information File name: C:\temp\nios2_sdram\DE0_SOPC.sopcinfo

CPU name: cpu_0

Application project

Project name: test_01

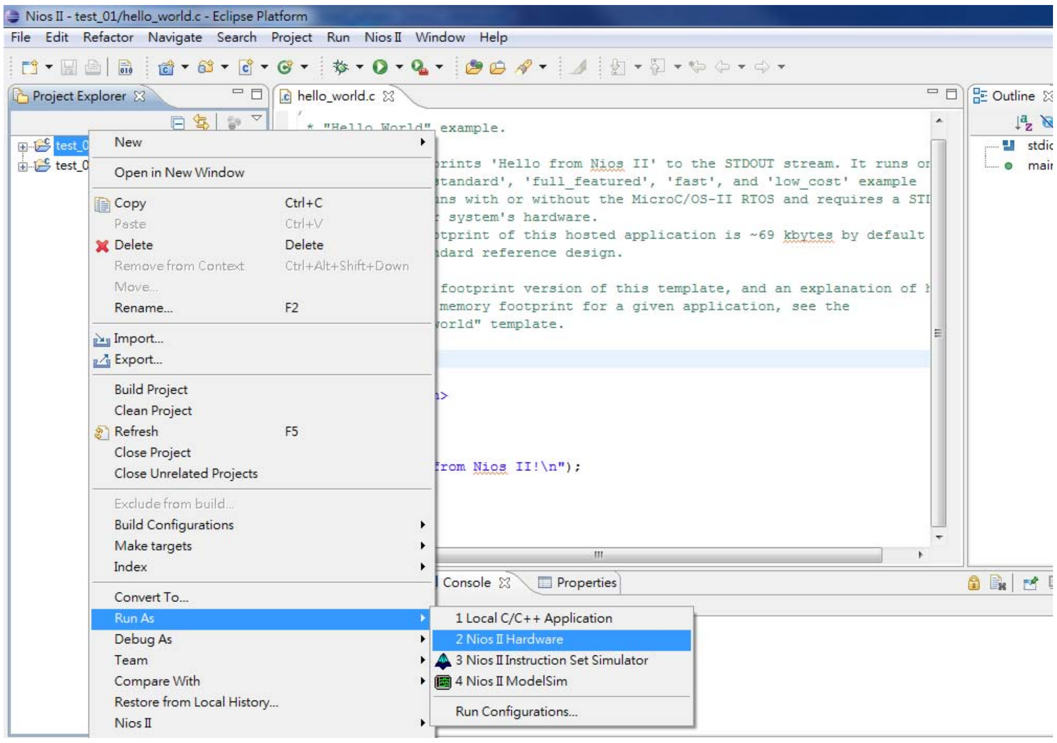
☑ Use default location

Project location: C:\temp\nios2_sdram\software\test_01

Project template

Templates	Template description
Blank Project Board Diagnostics Count Binary Hello Freestanding Hello MicroC/OS-II Hello World Hello World Small Memory Test Simple Socket Server Web Server	Hello World prints 'Hello from Nios II' to STDOUT. This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware. For details, click Finish to create the project and refer to the readme.txt file in the project directory. The BSP for this template is based on the Altera HAL operating system. For information about how this software example relates to Nios II hardware design examples, refer to the Design Examples page of the Nios II documentation available with your installation at: <installation_directory>/nios2eds/documents/index.htm.

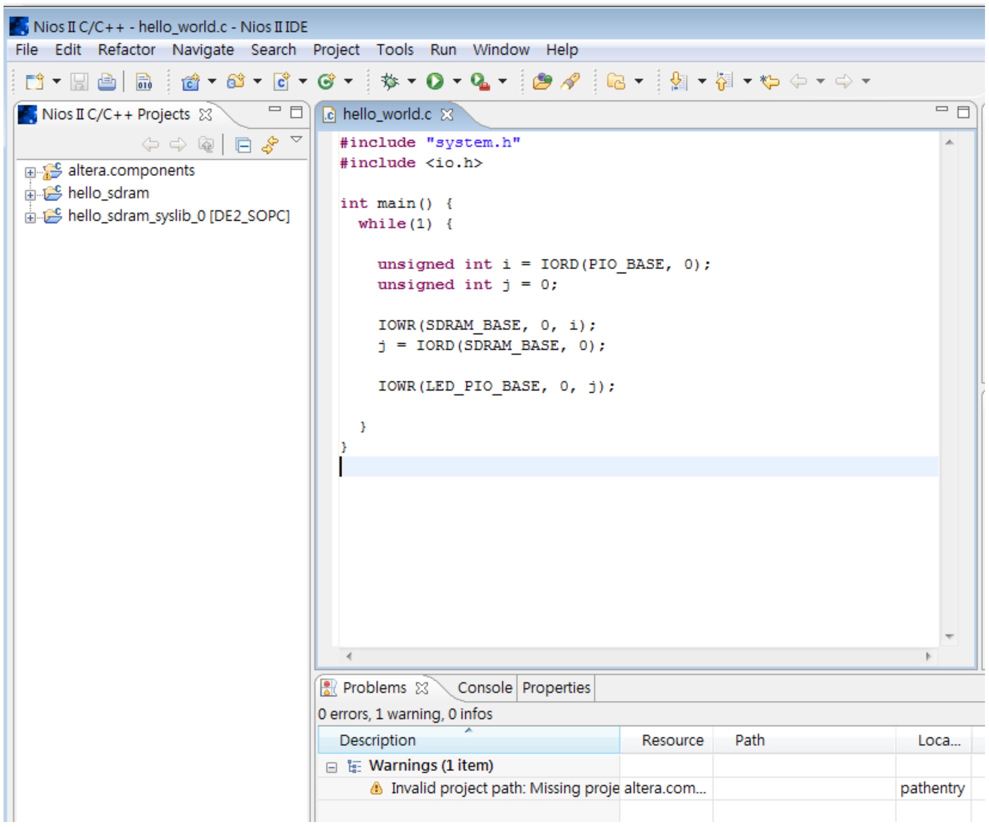
測試 Hello world.c



Nios II - test_01/hello_world.c - Eclipse Platform

```
 * "Hello World" example.
 ...
 rints 'Hello from Nios II' to the STDOUT stream. It runs o
 tandard', 'full_featured', 'fast', and 'low_cost' example
 ns with or without the MicroC/OS-II RTOS and requires a STI
  system's hardware.
 tprint of this hosted application is ~69 kbytes by default
 dard reference design.

 footprint version of this template, and an explanation of
 memory footprint for a given application, see the
 orld" template.

 >

 rom Nios II!\n");
```

1-3 修改 hello_world.c 程式內容



Nios II C/C++ - hello_world.c - Nios II IDE

Nios II C/C++ Projects: altera.components, hello_sdram, hello_sdram_syslib_0 [DE2_SOPC]

hello_world.c

```
#include "system.h"
#include <io.h>

int main() {
  while(1) {

    unsigned int i = IORD(PIO_BASE, 0);
    unsigned int j = 0;

    IOWR(SDRAM_BASE, 0, i);
    j = IORD(SDRAM_BASE, 0);

    IOWR(LED_PIO_BASE, 0, j);

  }
}
```

Problems | Console | Properties

0 errors, 1 warning, 0 infos

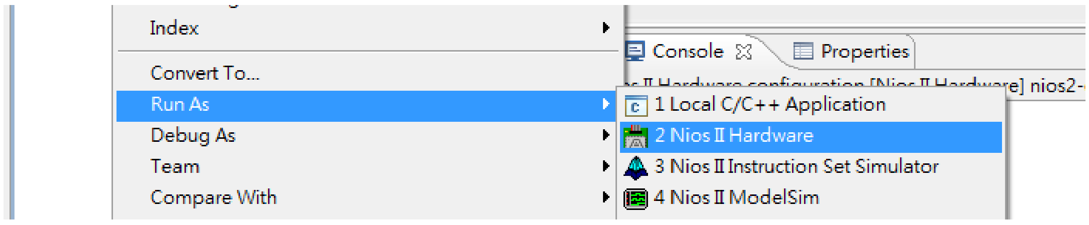
Description	Resource	Path	Loca...
Warnings (1 item)			
Invalid project path: Missing proje	altera.com...		pathentry

Fig.25 hello_world.c

```
#include "system.h"
#include <io.h>

int main() {
  while(1) {

    int i = IORD(SW_PIO_BASE, 0);
    int j = 0;

    IOWR(SDRAM_0_BASE, 0, i);
    j = IORD(SDRAM_0_BASE, 0);

    IOWR(LED_PIO_BASE, 0, j);

  }
  return 0;
}
```

1-5 Build Project

1-6 Run As > Nios II Hardware

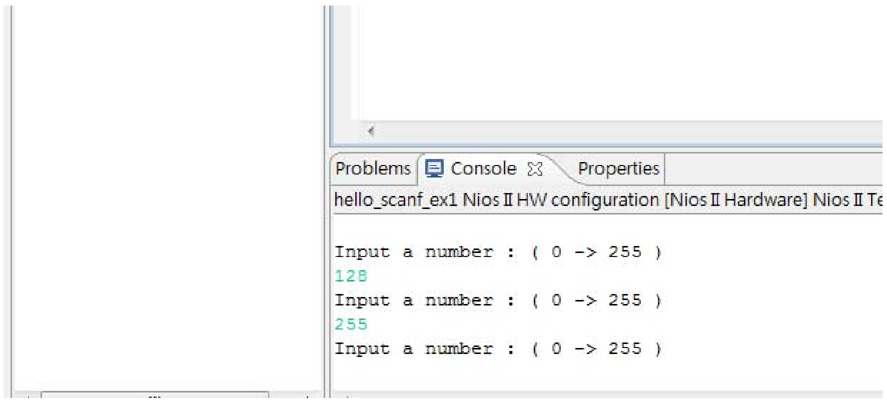


四、結論與探討

隨堂練習一：

1. 試以 printf 及 scanf 等敘述設計一個程式，首先在 console 上顯示 “input a number: (0->255)”，然後抓回使用者所輸入的數值，最後將值顯示在 LED 上。

```
#include <stdio.h>
#include <system.h>
#include <io.h>

int main(void)
{ int data;

  while(1)
  { printf("Input a number : ( 0 -> 255 )\n");
    scanf("%d",&data);
    IOWR( ?????? ); }
  return 0;
}
```



hello_scanf_ex1 Nios II HW configuration [Nios II Hardware] Nios II Te

```
Input a number : ( 0 -> 255 )
128
Input a number : ( 0 -> 255 )
255
Input a number : ( 0 -> 255 )
```

2. 利用一個 switch statement 讓使用者輸入選擇，來模擬一個簡單的選擇目錄，以執行兩個數字的相加、加減和加乘功能。

```
#include <stdio.h>
#include <system.h>
#include <io.h>

int main(void)
{ int menu,data1,data2,total;

  while(1){
  printf(???);
  scanf(???);

  printf(???);
  scanf(???);

  printf(???);
  scanf(???);

  switch(menu)
  {
    case 1:
      ????????????
    case 2:
     ????????????
    case 3:
     ????????????
    default:
      ????????????

  }

  if (menu == 1)
      printf(????????????);
  else if(menu == 2)
      printf(????????????);
  else if (menu == 3)
      printf(????????????);
  }

  return 0;
}
```

隨堂練習二：

1. 右旋功能的 LED 跑馬燈：

```
#include "system.h"
#include "unistd.h"  /* usleep */
#include <io.h>

int main (void)  {
  unsigned char j=0x01;  /* LED var */

  while (1)    /* Loop forever */
  {
    if (j==0x01)
    { j=0x80;
      IOWR(????????????);}
    else
    { IOWR(????????????);}

   usleep(30000);
   }

  return 0;  // End program
}
```

2. 試產生一組可重覆左、右旋交互動作之 LED 跑馬燈。

```
#include <io.h>
#include "unistd.h"
#include "system.h"

int main()
{
  unsigned char led = 0x01;
  char dir = 0;

  while (1)
  {
    if (led & 0x81)
    { dir = (dir ?? 0x01);}
    if (dir)
      { led = ????????????;}
    else
      { led = ????????????;}

    IOWR(LED_PIO_BASE,0,led);
    usleep(500000);
   }

    return 0;
}
```

3. 若將 for 敘述中的“led=led<<1”改為“led=(led<<1)+1”，LED 的移動方式會發生什麼變化？

```
#include <io.h>
#include "unistd.h"   // for usleep
#include "system.h"

int main()
{
  int count,led;

  while(1)
  {
    for(count=0,led=0x01;count<8;count++,led=(led<<1)+1)
    {  IOWR(LED_PIO_BASE,0,led);
       usleep(500000); // Delay 500000 us
     }
   }
     return 0;
}
```

4. 試修改程式以產生一組由右向左逐一點亮，再由左向右逐一點亮的 LED。

隨堂練習三：

1. 試以一維陣列儲放 LED 變化值，再透過 for loop 讀取陣列資料的方式設計 LED 跑馬燈(逐一點亮、聚攏和分開等變化功能)

```
#include "system.h"
#include "unistd.h"   // usleep
#include <io.h>
#include <stdio.h>

unsigned char LED_TBL[42]={
0x00, 0xFF,
0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80,
0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, 0xFF,
0xFF, 0x7F, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x01,
0x81, 0x42, 0x24, 0x18, 0x18, 0x24, 0x42, 0x81,
0x81, 0xC3, 0xE7, 0xFF, 0xFF, 0xE7, 0xC3, 0x81,
};

int main()
{
  unsigned char i;

  while(1)
  {
     ????????????
  }
  return 0;
}
```

2. 試改成利用兩個 for loop 迴圈來讀取二維陣列儲存值的方式實現相同的跑馬燈功能。

```
#include <io.h>
#include <stdio.h>
#include "unistd.h"   // usleep
#include "system.h"

unsigned int LED_TBL[6][8] =
{ ????????????
};

int main()
{
  int x,y;
  while(1)
  {
    ????????????
  }
  return 0;
}
```