Innovate Asia 2008
SOPC Builder and NIOS II

- ◆Build SOPC for DE1
- ◆NIOS II Programming
- ◆Example: SDCARD Music Player

Build SOPC for DE1

◆New SOPC System

◆Add CPU/Component

◆Clock Setting

◆Specify Connection

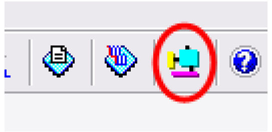
◆Adjust Base Address/Interrupt Number

◆Adjust Arbitration

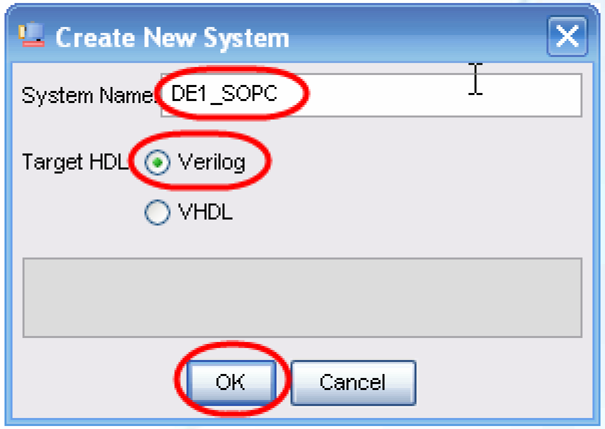
◆Generate Code

New SOPC System

- Click SOPC Builder ICON under Quartus II



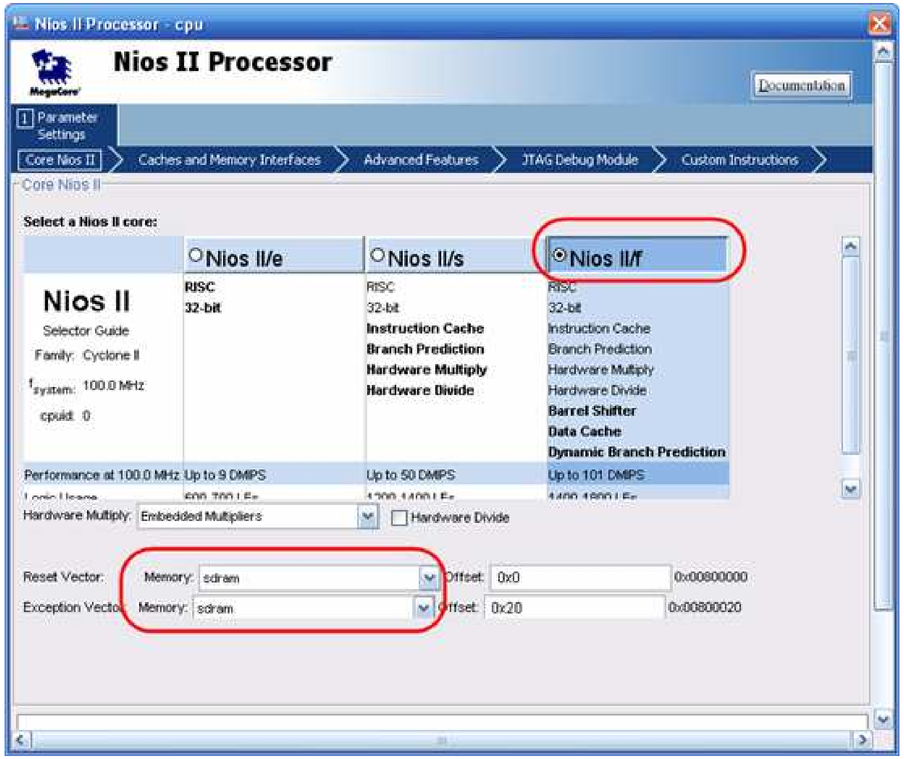
- Input Project Name



System Name: DE1_SOPC

Target HDL: ● Verilog ○ VHDL

Add CPU



Core Nios II

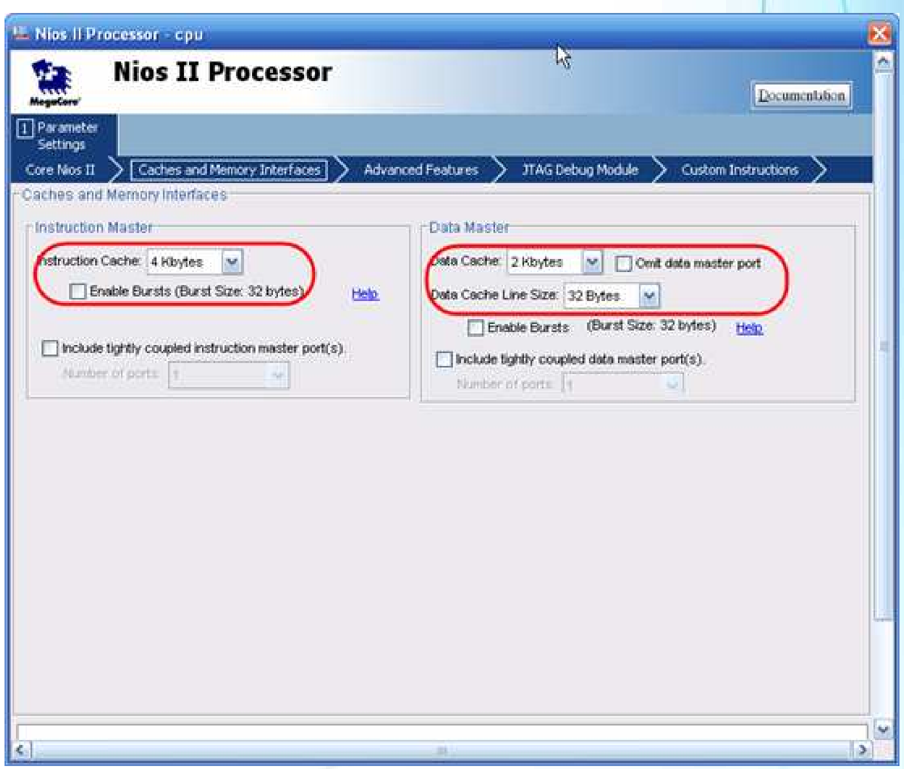
Select a Nios II core:

Nios II Selector Guide Family: Cyclone II f_{system}: 100.0 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 100.0 MHz	Up to 9 DMIPS	Up to 50 DMIPS	Up to 101 DMIPS
Logic Usage	600-700 LEs	1200-1400 LEs	1400-1800 LEs

Hardware Multiply: Embedded Multipliers ☐ Hardware Divide

Reset Vector: Memory: sdram Offset: 0x0 0x00800000

Exception Vector: Memory: sdram Offset: 0x20 0x00800020



Caches and Memory Interfaces

Instruction Master

Instruction Cache: 4 Kbytes

☐ Enable Bursts (Burst Size: 32 bytes)

☐ Include tightly coupled instruction master port(s).

Number of ports: 1

Data Master

Data Cache: 2 Kbytes ☐ Omit data master port

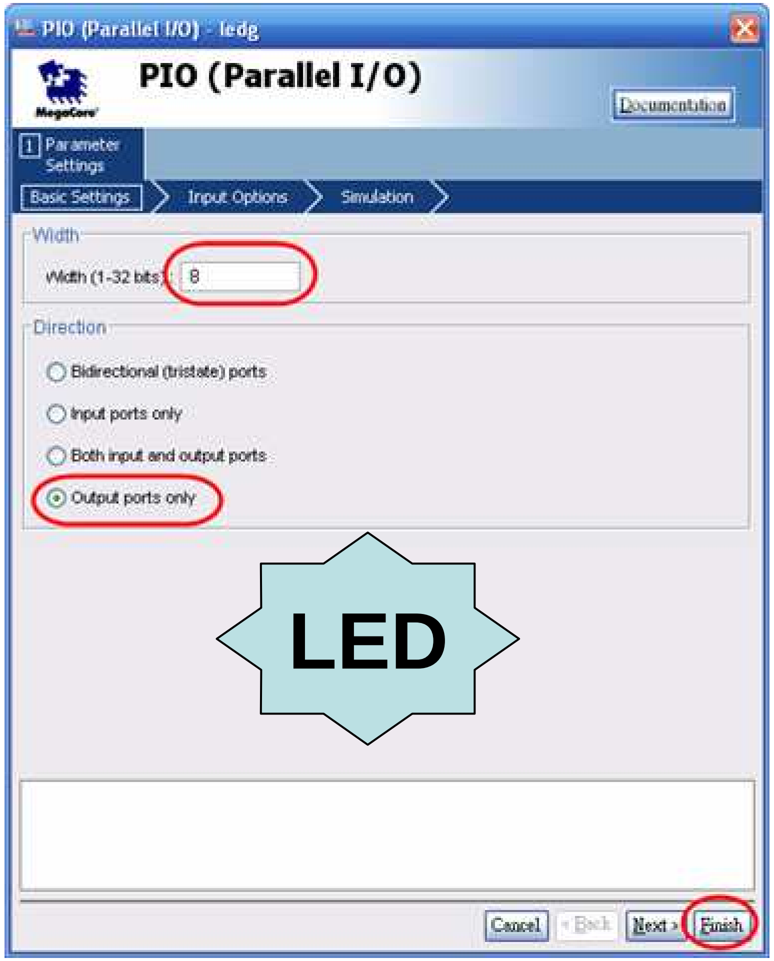
Data Cache Line Size: 32 Bytes

☐ Enable Bursts (Burst Size: 32 bytes)

☐ Include tightly coupled data master port(s).

Number of ports: 1

Add GPIO Controller for LED/7SEG/SWITCH/BUTTON



PIO (Parallel I/O) - ledg

PIO (Parallel I/O)

Width

Width (1-32 bits): 8

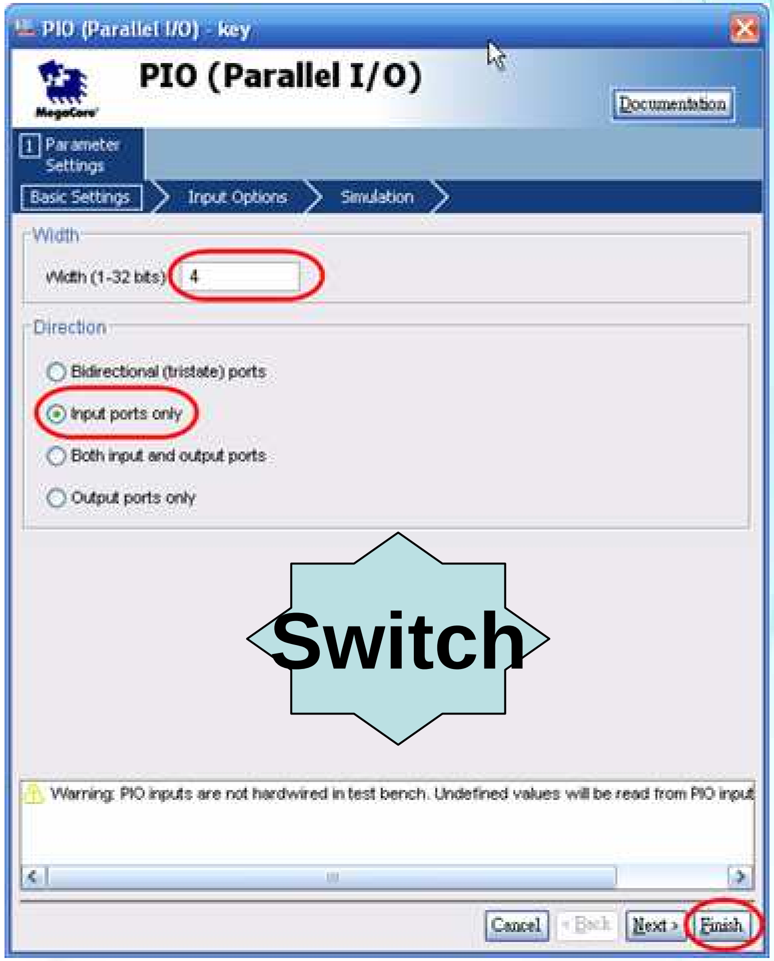
Direction

○ Bidirectional (tristate) ports

○ Input ports only

○ Both input and output ports

● Output ports only



PIO (Parallel I/O) - key

PIO (Parallel I/O)

Width

Width (1-32 bits): 4

Direction

○ Bidirectional (tristate) ports

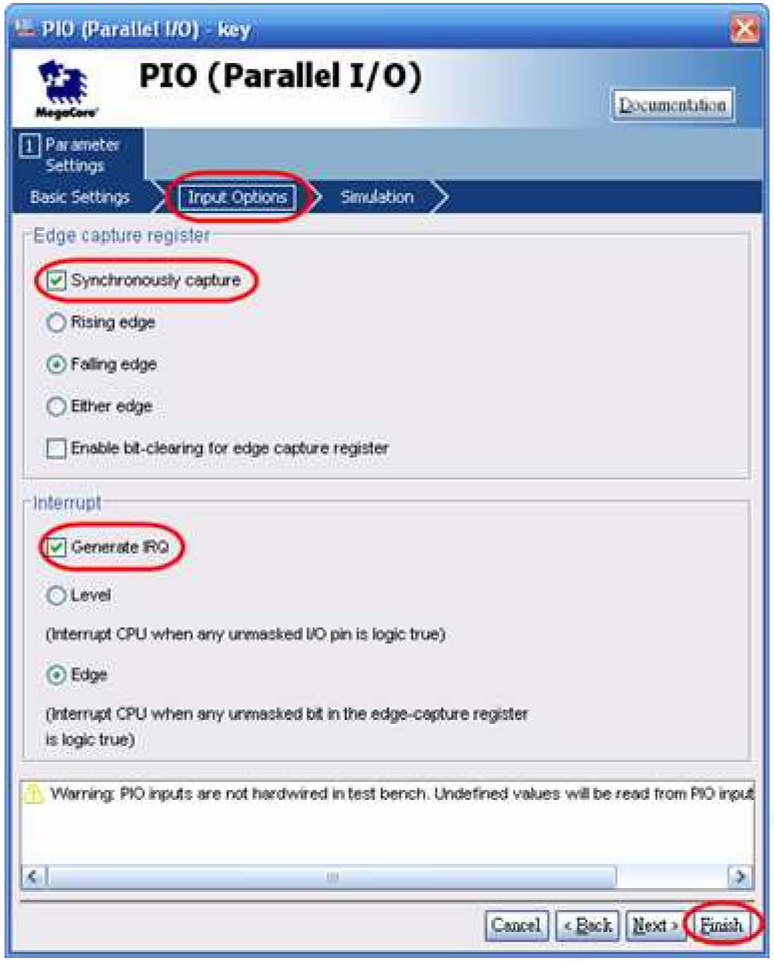
● Input ports only

○ Both input and output ports

○ Output ports only

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

Enable Interrupt for Input GPIO



PIO (Parallel I/O) - key

PIO (Parallel I/O)

Edge capture register

☑ Synchronously capture

○ Rising edge

● Falling edge

○ Either edge

□ Enable bit-clearing for edge capture register

Interrupt

☑ Generate IRQ

○ Level

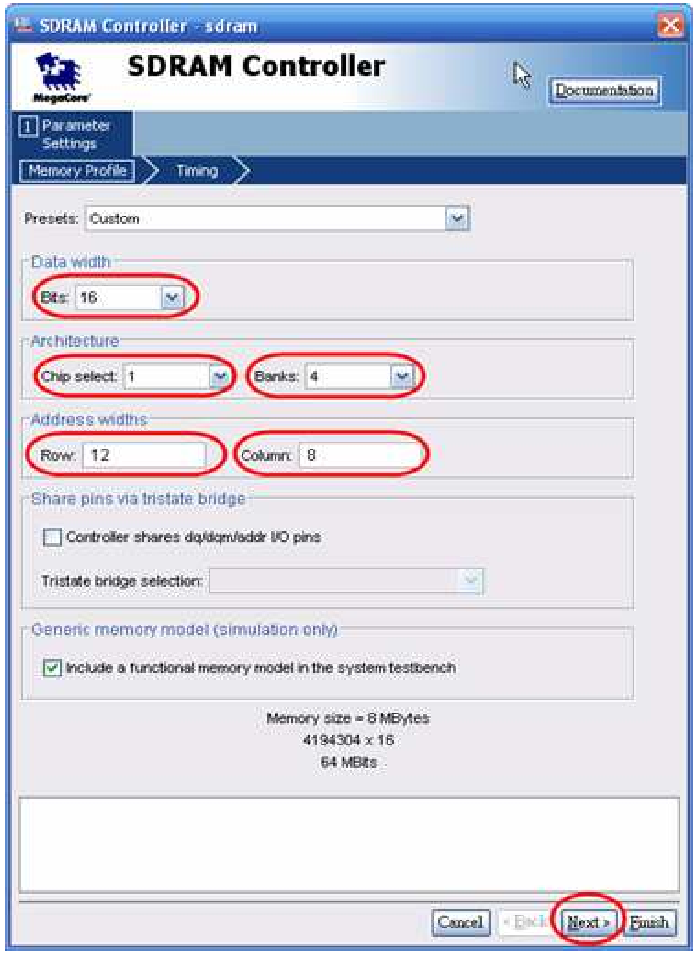
(Interrupt CPU when any unmasked I/O pin is logic true)

● Edge

(Interrupt CPU when any unmasked bit in the edge-capture register is logic true)

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

ADD SDRAM



Presets: Custom

Data width

Bits: 16

Architecture

Chip select: 1

Banks: 4

Address widths

Row: 12

Column: 8

Share pins via tristate bridge

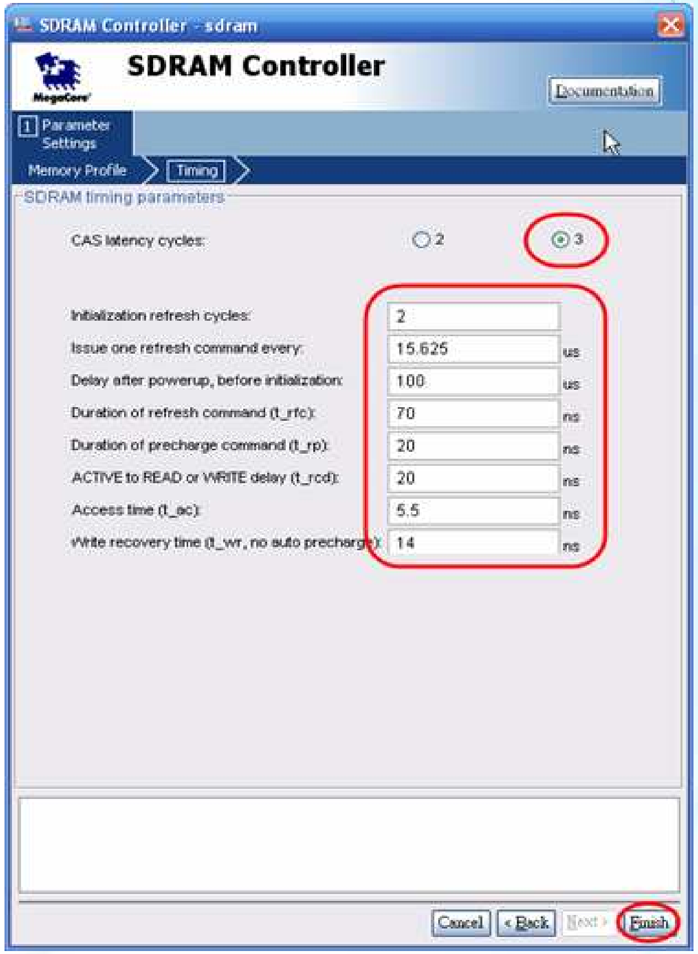
☐ Controller shares dq/dqm/addr I/O pins

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



SDRAM timing parameters

CAS latency cycles: ○ 2 ● 3

Initialization refresh cycles: 2

Issue one refresh command every: 15.625 us

Delay after powerup, before initialization: 100 us

Duration of refresh command (t_rfc): 70 ns

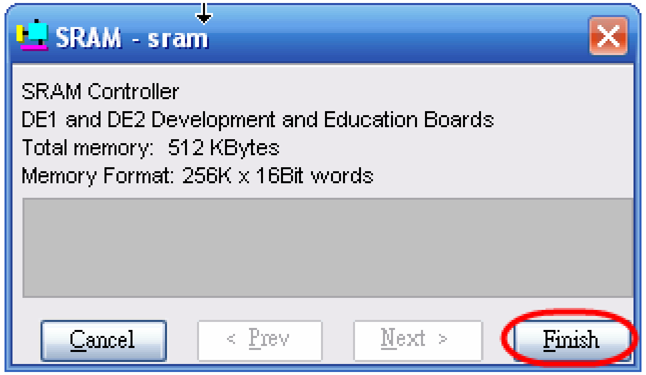
Duration of precharge command (t_rp): 20 ns

ACTIVE to READ or WRITE delay (t_rcd): 20 ns

Access time (t_ac): 5.5 ns

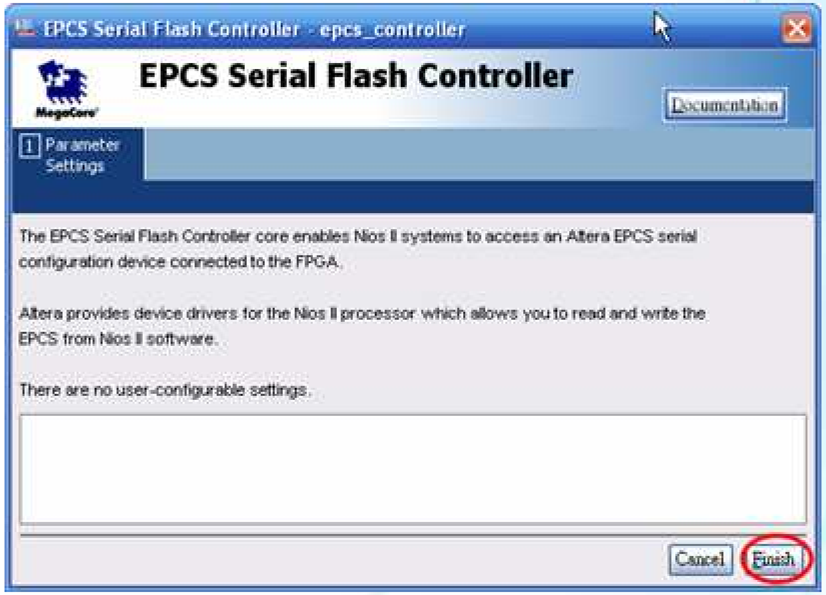
Write recovery time (t_wr, no auto precharge): 14 ns

Add SRAM/EPCS



SRAM - sram

SRAM Controller
DE1 and DE2 Development and Education Boards
Total memory: 512 KBytes
Memory Format: 256K x 16Bit words



EPCS Serial Flash Controller - epcs_controller

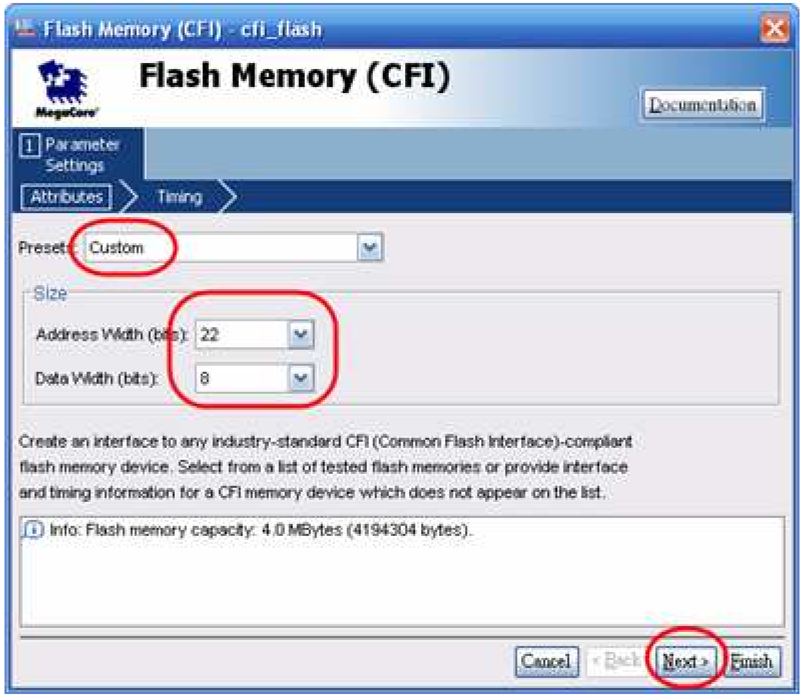
EPCS Serial Flash Controller

The EPCS Serial Flash Controller core enables Nios II systems to access an Altera EPCS serial configuration device connected to the FPGA.

Altera provides device drivers for the Nios II processor which allows you to read and write the EPCS from Nios II software.

There are no user-configurable settings.

Add FLASH



Presets: Custom

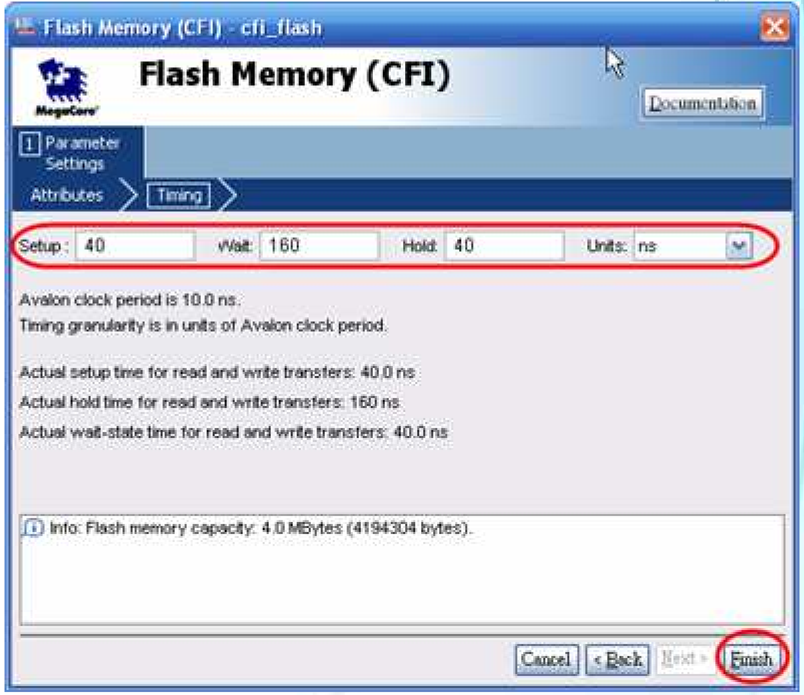
Size

Address Width (bits): 22

Data Width (bits): 8

Create an interface to any industry-standard CFI (Common Flash Interface)-compliant flash memory device. Select from a list of tested flash memories or provide interface and timing information for a CFI memory device which does not appear on the list.

Info: Flash memory capacity: 4.0 MBytes (4194304 bytes).



Setup : 40 Wait: 160 Hold: 40 Units: ns

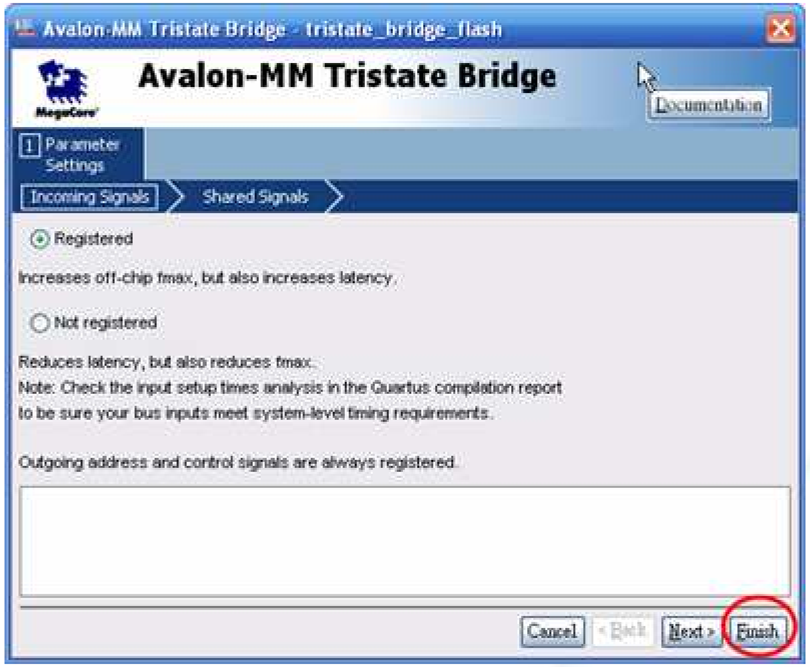
Avalon clock period is 10.0 ns.
Timing granularity is in units of Avalon clock period.

Actual setup time for read and write transfers: 40.0 ns
Actual hold time for read and write transfers: 160 ns
Actual wait-state time for read and write transfers: 40.0 ns

Info: Flash memory capacity: 4.0 MBytes (4194304 bytes).

Add Tristate Bridge for Flash

- Bridge and Adapters→Memory Mapped->Avalon-MM Tristate Bridge



● Registered

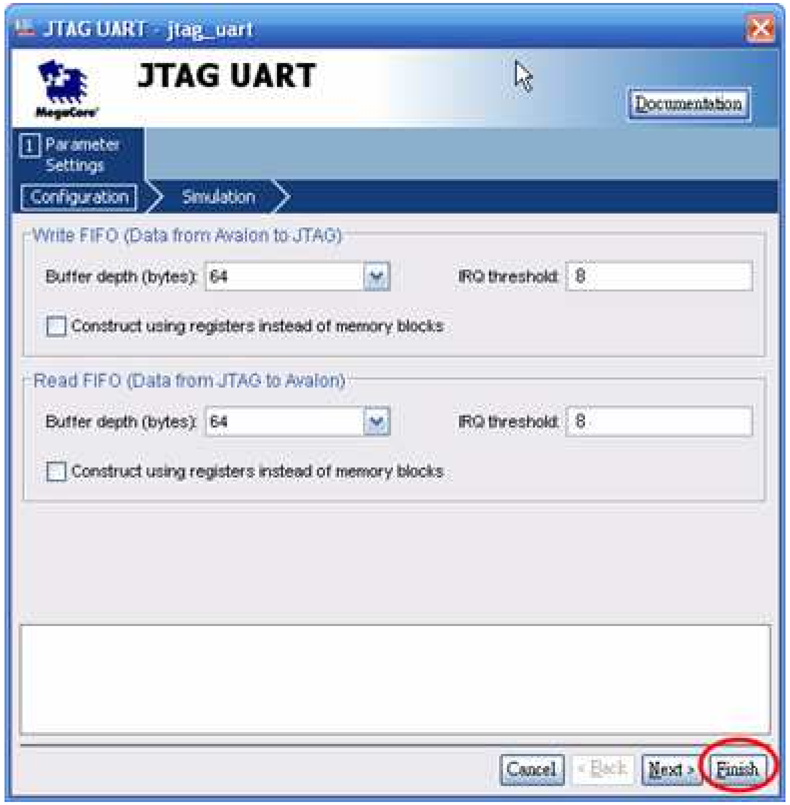
Increases off-chip fmax, but also increases latency.

○ Not registered

Reduces latency, but also reduces fmax.
Note: Check the input setup times analysis in the Quartus compilation report
to be sure your bus inputs meet system-level timing requirements.

Outgoing address and control signals are always registered.

JTAG-UART/UART



Write FIFO (Data from Avalon to JTAG)

Buffer depth (bytes): 64

IRQ threshold: 8

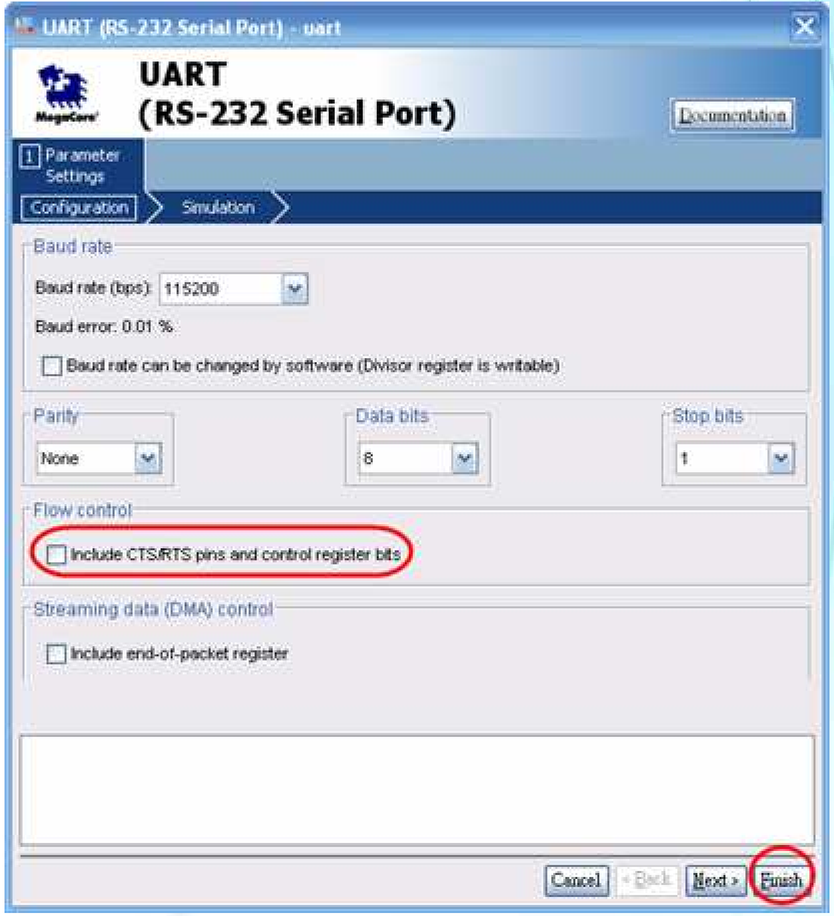
□ Construct using registers instead of memory blocks

Read FIFO (Data from JTAG to Avalon)

Buffer depth (bytes): 64

IRQ threshold: 8

□ Construct using registers instead of memory blocks



Baud rate

Baud rate (bps): 115200

Baud error: 0.01 %

□ Baud rate can be changed by software (Divisor register is writable)

Parity: None

Data bits: 8

Stop bits: 1

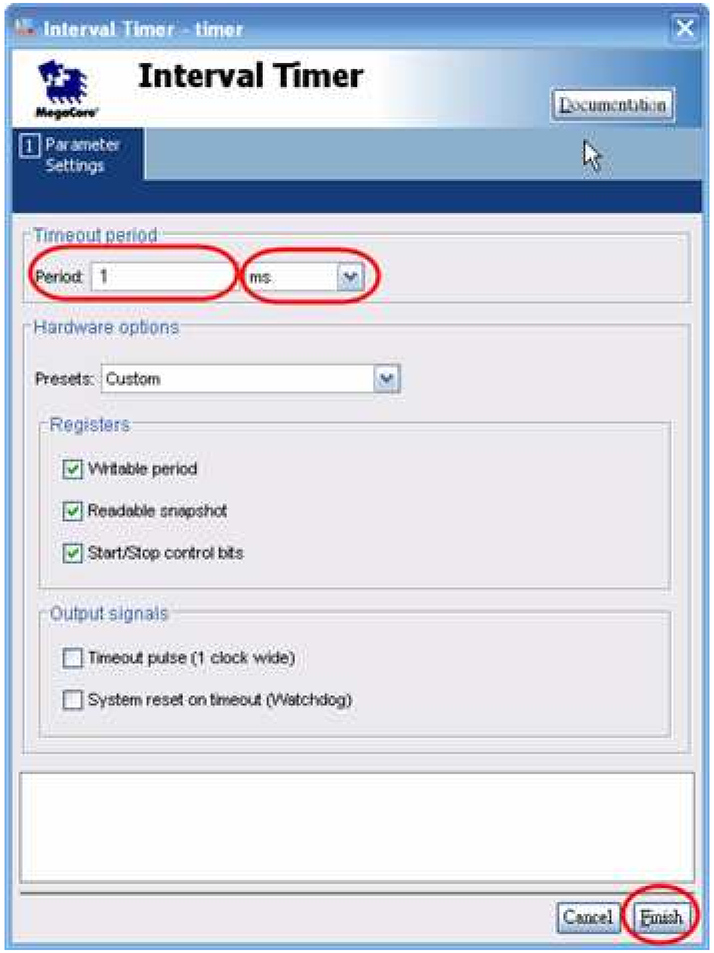
Flow control

□ Include CTS/RTS pins and control register bits

Streaming data (DMA) control

□ Include end-of-packet register

Timer/Time-Stamp/sysid



Interval Timer - timer

Interval Timer

Timeout period

Period: 1 ms

Hardware options

Presets: Custom

Registers

☑ Writable period

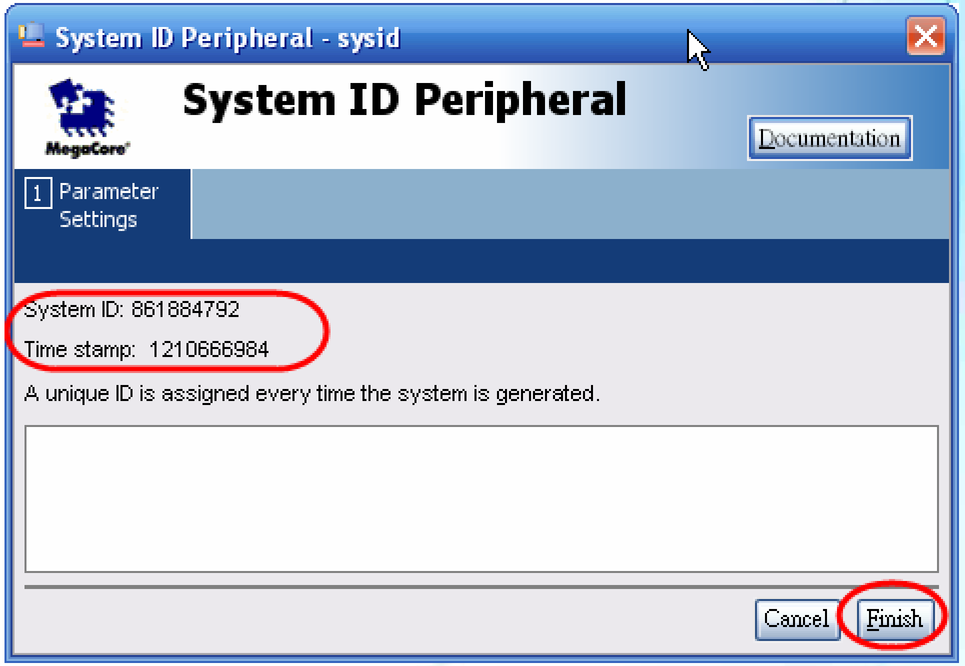
☑ Readable snapshot

☑ Start/Stop control bits

Output signals

□ Timeout pulse (1 clock wide)

□ System reset on timeout (Watchdog)



System ID Peripheral - sysid

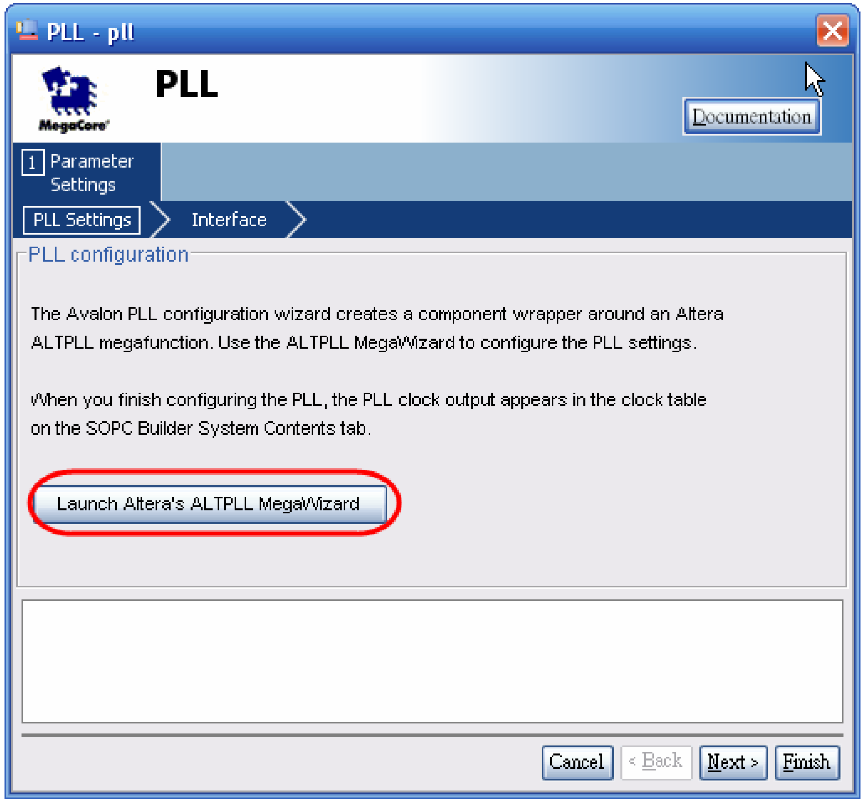
System ID Peripheral

System ID: 861884792

Time stamp: 1210666984

A unique ID is assigned every time the system is generated.

Add PLL



PLL - pll

PLL

Documentation

1 Parameter Settings

PLL Settings > Interface >

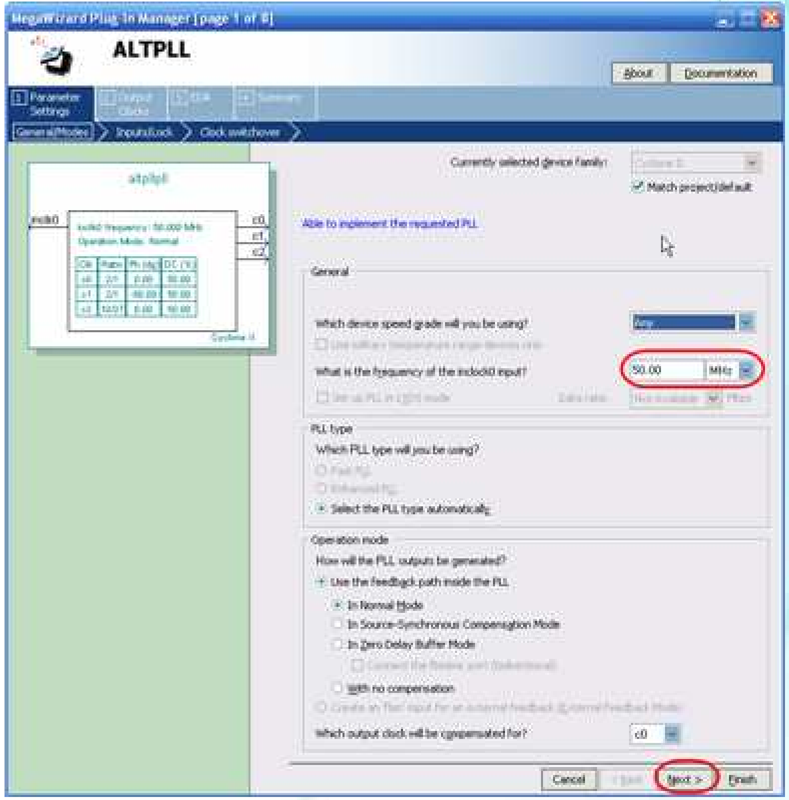
PLL configuration

The Avalon PLL configuration wizard creates a component wrapper around an Altera ALTPLL megafunction. Use the ALTPLL MegaWizard to configure the PLL settings.

When you finish configuring the PLL, the PLL clock output appears in the clock table on the SOPC Builder System Contents tab.

Launch Altera's ALTPLL MegaWizard

Cancel | < Back | Next > | Finish



MegaWizard Plug-In Manager [page 1 of 8]

ALTPLL

About | Documentation

1 Parameter Settings | 2 Output Clocks | 3 EDA | 4 Summary

General/Modes > Inputs/Lock > Clock switchover >

altpllpll
inclk0
c0
c1
c2
inclk0 frequency: 50.000 MHz
Operation Mode: Normal
Clk | Ratio | Ph (dg) | DC (%)
c0 | 2/1 | 0.00 | 50.00
c1 | 2/1 | -90.00 | 50.00
c2 | 10/21 | 0.00 | 50.00
Cyclone II

Currently selected device family: Cyclone II

☑ Match project/default

Able to implement the requested PLL

General

Which device speed grade will you be using? Any

☐ Use military temperature range devices only

What is the frequency of the inclock0 input? 50.00 MHz

☐ Set up PLL in LVDS mode   Data rate: Not Available Mbps

PLL type

Which PLL type will you be using?

○ Fast PLL

○ Enhanced PLL

● Select the PLL type automatically

Operation mode

How will the PLL outputs be generated?

● Use the feedback path inside the PLL

● In Normal Mode

○ In Source-Synchronous Compensation Mode

○ In Zero Delay Buffer Mode

☐ Connect the fbmimic port (bidirectional)

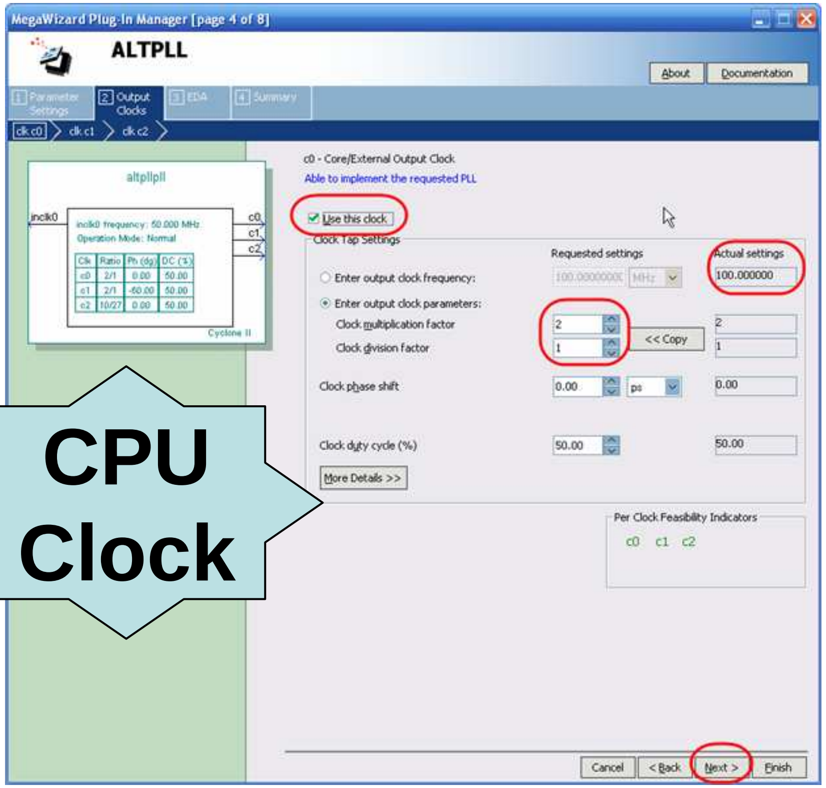
○ With no compensation

○ Create an 'fbin' input for an external feedback (External Feedback Mode)

Which output clock will be compensated for? c0

Cancel | < Back | Next > | Finish

Add PLL (2)



MegaWizard Plug-In Manager [page 4 of 8]

ALTPLL

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	2/1	0.00	50.00
c1	2/1	-60.00	50.00
c2	10/27	0.00	50.00

c0 - Core/External Output Clock

Able to implement the requested PLL

☑ Use this clock

Clock Tap Settings

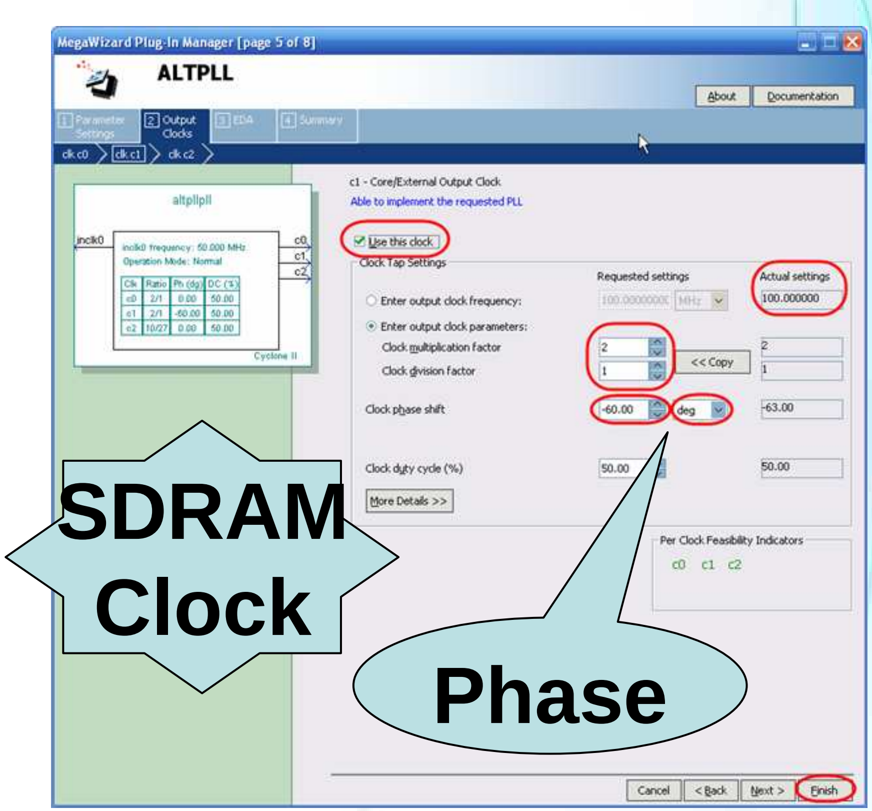
	Requested settings		Actual settings
○ Enter output clock frequency:	100.0000000	MHz	100.000000
● Enter output clock parameters:			
Clock multiplication factor	2	<< Copy	2
Clock division factor	1		1
Clock phase shift	0.00	ps	0.00
Clock duty cycle (%)	50.00		50.00

More Details >>

Per Clock Feasibility Indicators

c0 c1 c2

CPU Clock



MegaWizard Plug-In Manager [page 5 of 8]

ALTPLL

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	2/1	0.00	50.00
c1	2/1	-60.00	50.00
c2	10/27	0.00	50.00

c1 - Core/External Output Clock

Able to implement the requested PLL

☑ Use this clock

Clock Tap Settings

	Requested settings		Actual settings
○ Enter output clock frequency:	100.0000000	MHz	100.000000
● Enter output clock parameters:			
Clock multiplication factor	2	<< Copy	2
Clock division factor	1		1
Clock phase shift	-60.00	deg	-63.00
Clock duty cycle (%)	50.00		50.00

More Details >>

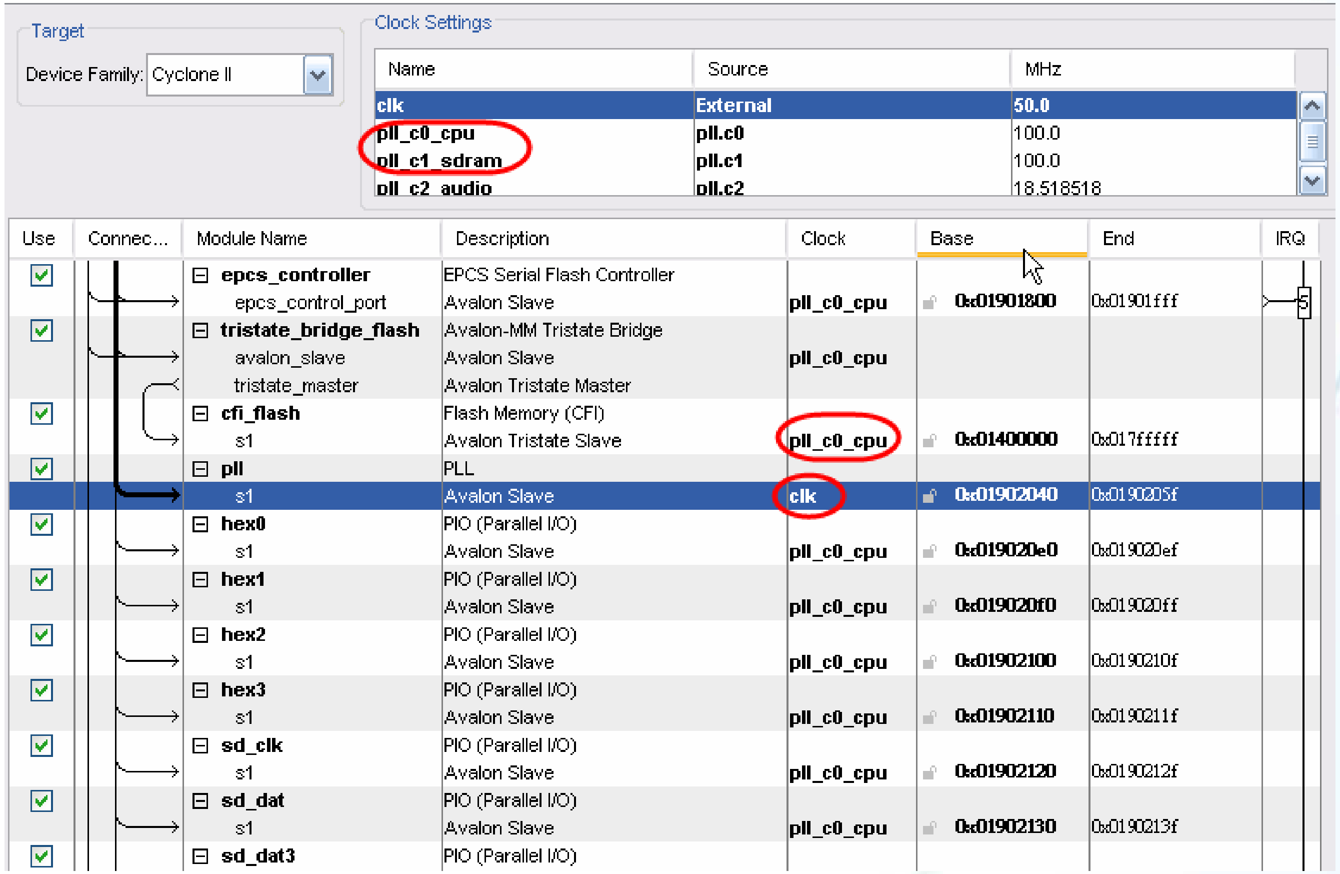
Per Clock Feasibility Indicators

c0 c1 c2

SDRAM Clock

Phase

Clock Setting



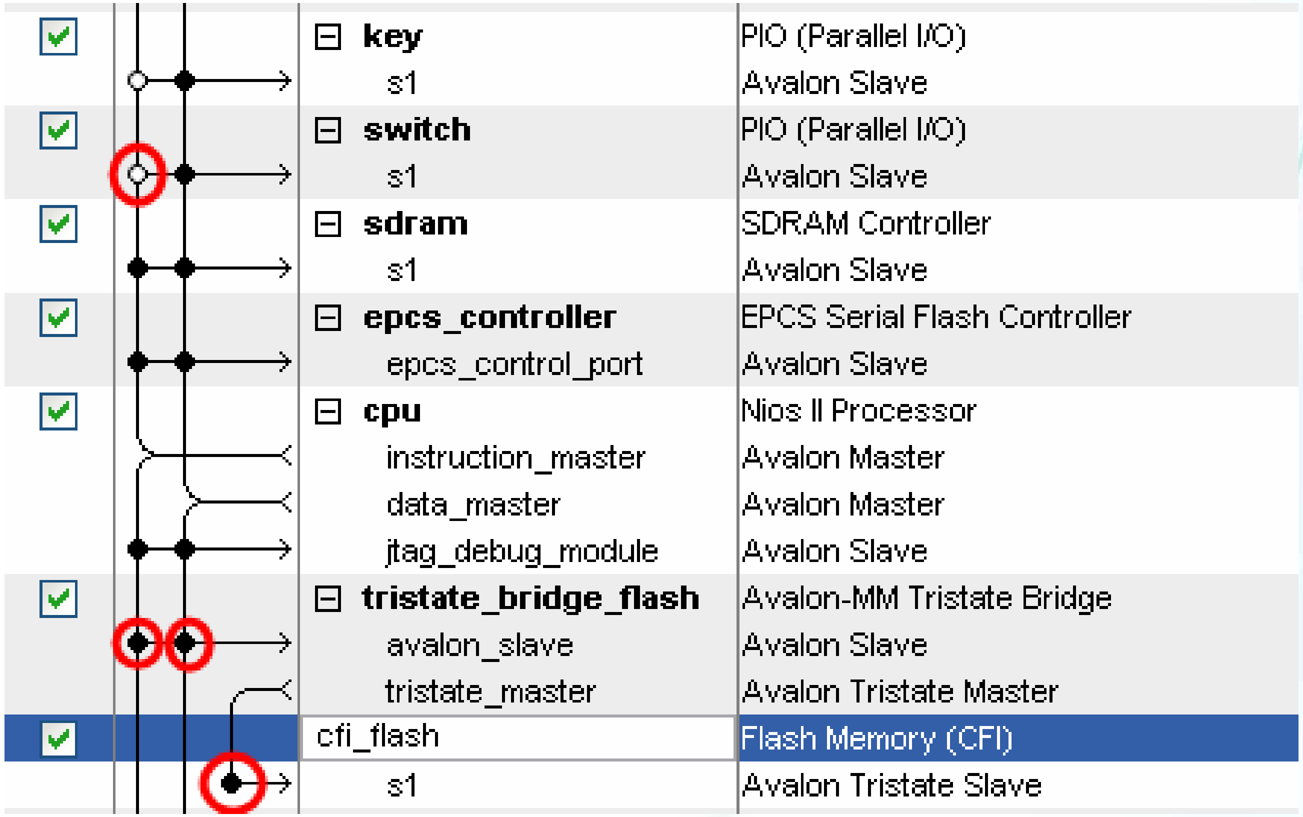
Device Family: Cycione II

Name	Source	MHz
clk	External	50.0
pll_c0_cpu	pll.c0	100.0
pll_c1_sdram	pll.c1	100.0
pll_c2_audio	pll.c2	18.518518

Use	Connec...	Module Name	Description	Clock	Base	End	IRQ
☑		⊟ epcs_controller	EPCS Serial Flash Controller				
		epcs_control_port	Avalon Slave	pll_c0_cpu	0x01901800	0x01901fff	5
☑		⊟ tristate_bridge_flash	Avalon-MM Tristate Bridge				
		avalon_slave	Avalon Slave	pll_c0_cpu			
		tristate_master	Avalon Tristate Master				
☑		⊟ cfi_flash	Flash Memory (CFI)				
		s1	Avalon Tristate Slave	pll_c0_cpu	0x01400000	0x017fffff	
☑		⊟ pll	PLL				
		s1	Avalon Slave	clk	0x01902040	0x0190205f	
☑		⊟ hex0	PIO (Parallel I/O)				
		s1	Avalon Slave	pll_c0_cpu	0x019020e0	0x019020ef	
☑		⊟ hex1	PIO (Parallel I/O)				
		s1	Avalon Slave	pll_c0_cpu	0x019020f0	0x019020ff	
☑		⊟ hex2	PIO (Parallel I/O)				
		s1	Avalon Slave	pll_c0_cpu	0x01902100	0x0190210f	
☑		⊟ hex3	PIO (Parallel I/O)				
		s1	Avalon Slave	pll_c0_cpu	0x01902110	0x0190211f	
☑		⊟ sd_clk	PIO (Parallel I/O)				
		s1	Avalon Slave	pll_c0_cpu	0x01902120	0x0190212f	
☑		⊟ sd_dat	PIO (Parallel I/O)				
		s1	Avalon Slave	pll_c0_cpu	0x01902130	0x0190213f	
☑		⊟ sd_dat3	PIO (Parallel I/O)				

Connection

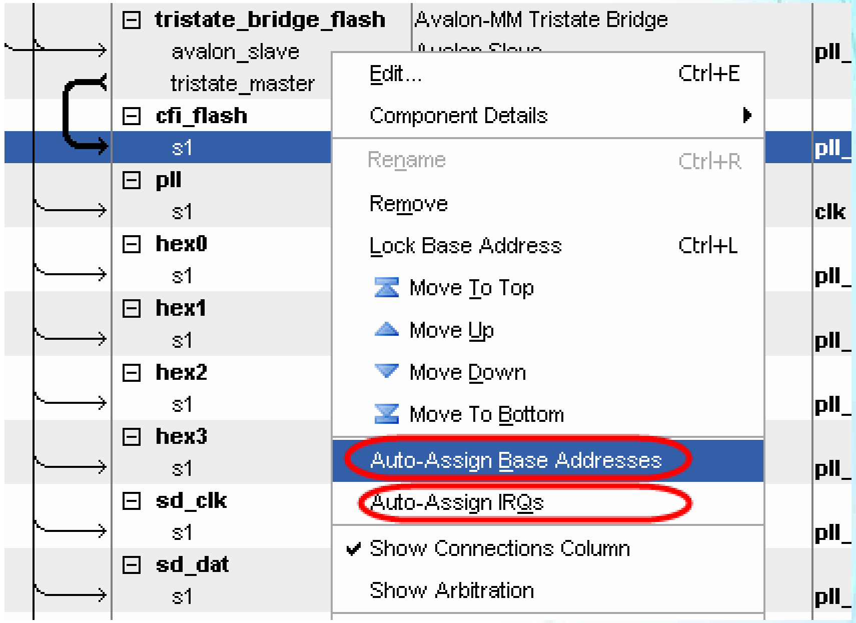
- Master Port →Slave Port

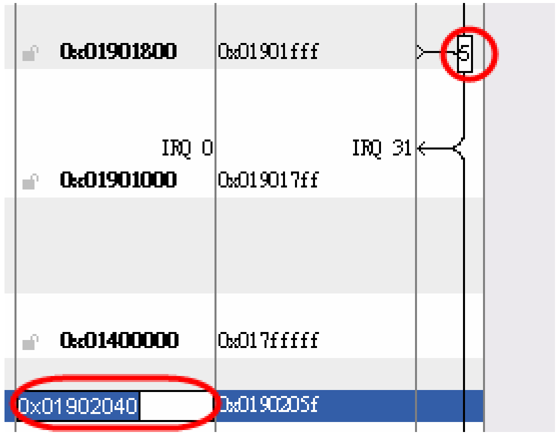


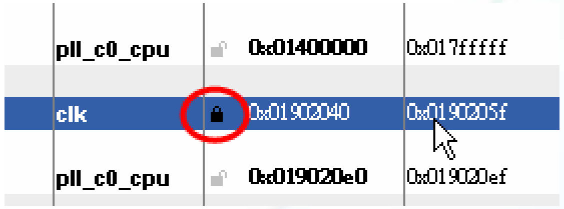
☑	⊟ **key**	PIO (Parallel I/O)
	s1	Avalon Slave
☑	⊟ **switch**	PIO (Parallel I/O)
	s1	Avalon Slave
☑	⊟ **sdram**	SDRAM Controller
	s1	Avalon Slave
☑	⊟ **epcs_controller**	EPCS Serial Flash Controller
	epcs_control_port	Avalon Slave
☑	⊟ **cpu**	Nios II Processor
	instruction_master	Avalon Master
	data_master	Avalon Master
	jtag_debug_module	Avalon Slave
☑	⊟ **tristate_bridge_flash**	Avalon-MM Tristate Bridge
	avalon_slave	Avalon Slave
	tristate_master	Avalon Tristate Master
☑	cfi_flash	Flash Memory (CFI)
	s1	Avalon Tristate Slave

Adjust Base Address and Interrupt Number

- Apply Auto
- Adjust in manual
- Lock Address

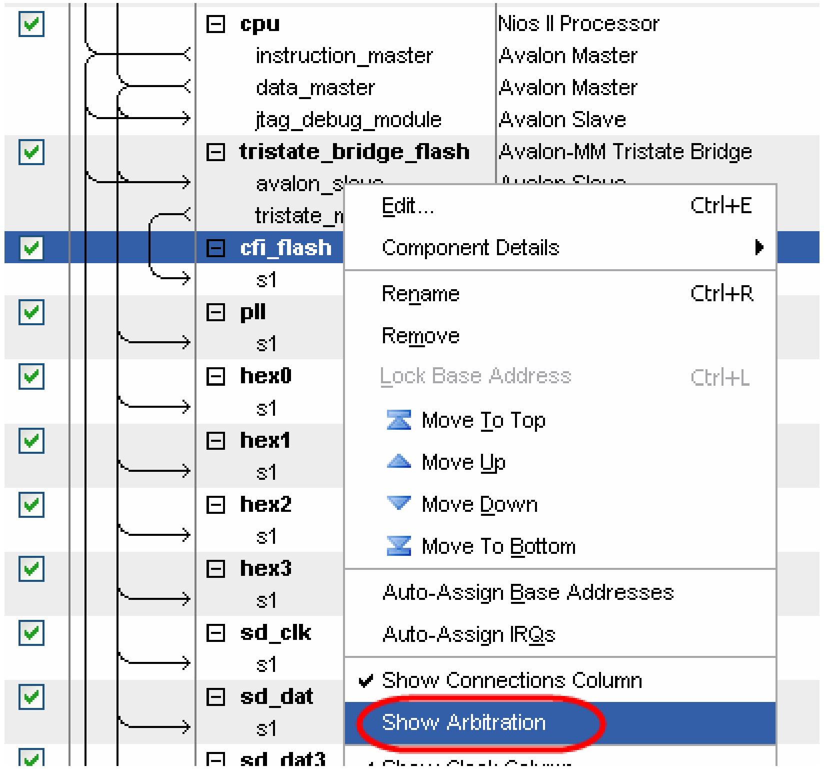




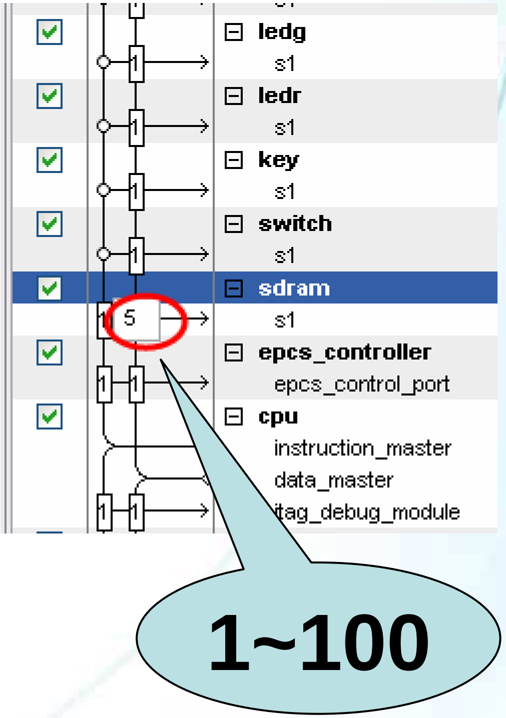


pll_c0_cpu		0x01400000	0x017fffff
clk		0x01902040	0x0190205f
pll_c0_cpu		0x019020e0	0x019020ef

Adjust Arbitration

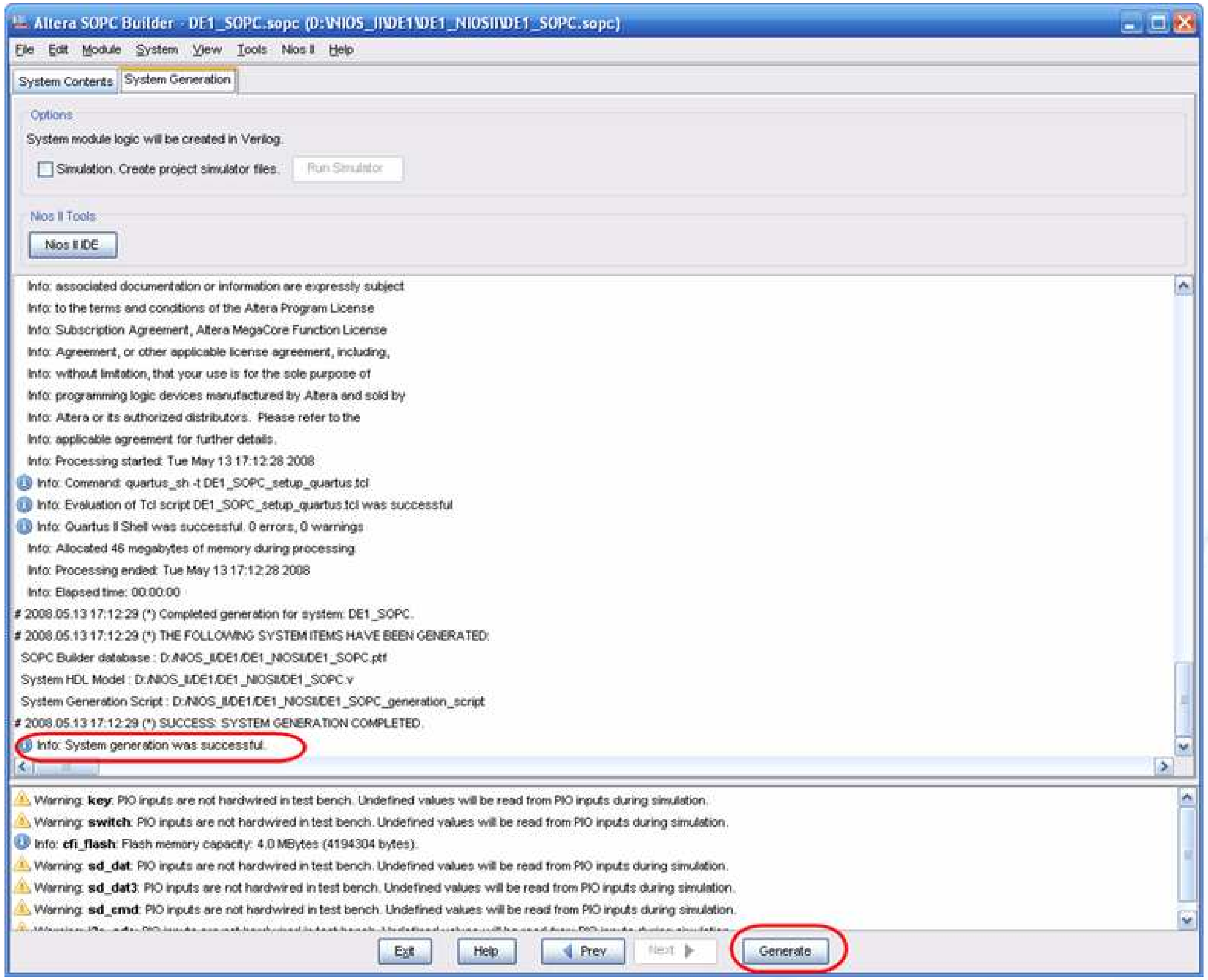


☑ ⊟ cpu — Nios II Processor
instruction_master — Avalon Master
data_master — Avalon Master
jtag_debug_module — Avalon Slave
☑ ⊟ tristate_bridge_flash — Avalon-MM Tristate Bridge
avalon_slave — Avalon Slave
tristate_m
☑ ⊟ cfi_flash
s1
☑ ⊟ pll
s1
☑ ⊟ hex0
s1
☑ ⊟ hex1
s1
☑ ⊟ hex2
s1
☑ ⊟ hex3
s1
☑ ⊟ sd_clk
s1
☑ ⊟ sd_dat
s1
☑ ⊟ sd_dat3



☑ ⊟ ledg
s1
☑ ⊟ ledr
s1
☑ ⊟ key
s1
☑ ⊟ switch
s1
☑ ⊟ sdram
s1
☑ ⊟ epcs_controller
epcs_control_port
☑ ⊟ cpu
instruction_master
data_master
jtag_debug_module

Generate Code


Altera SOPC Builder - DE1_SOPC.sopc (D:\NIOS_II\DE1\DE1_NIOSII\DE1_SOPC.sopc)
File Edit Module System View Tools Nios II Help
System Contents
System Generation
Options
Nios II Tools
Nios II IDE
Run Simulator
Exit
Help
Prev
Next
Generate

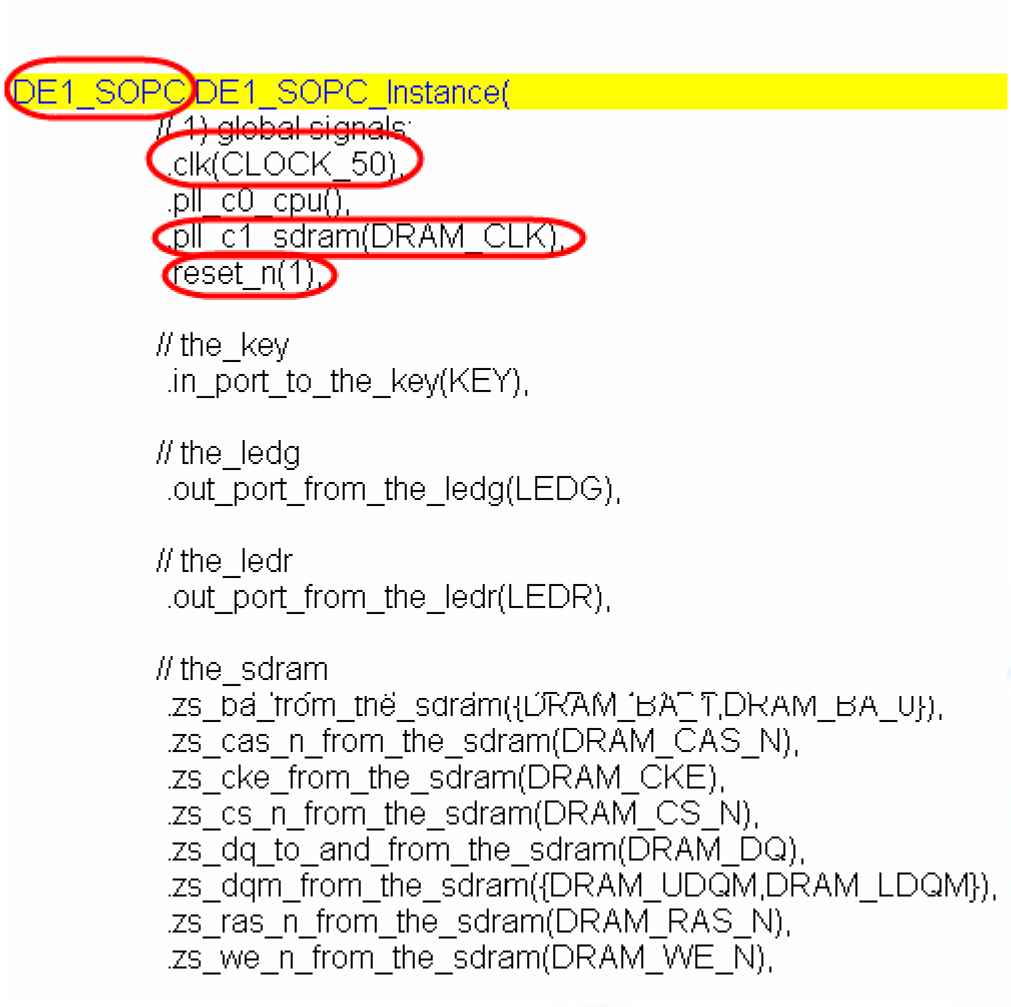
System module logic will be created in Verilog.

☐ Simulation. Create project simulator files.

```
Info: associated documentation or information are expressly subject
Info: to the terms and conditions of the Altera Program License
Info: Subscription Agreement, Altera MegaCore Function License
Info: Agreement, or other applicable license agreement, including,
Info: without limitation, that your use is for the sole purpose of
Info: programming logic devices manufactured by Altera and sold by
Info: Altera or its authorized distributors.  Please refer to the
Info: applicable agreement for further details.
Info: Processing started: Tue May 13 17:12:28 2008
Info: Command: quartus_sh -t DE1_SOPC_setup_quartus.tcl
Info: Evaluation of Tcl script DE1_SOPC_setup_quartus.tcl was successful
Info: Quartus II Shell was successful. 0 errors, 0 warnings
Info: Allocated 46 megabytes of memory during processing
Info: Processing ended: Tue May 13 17:12:28 2008
Info: Elapsed time: 00:00:00
# 2008.05.13 17:12:29 (*) Completed generation for system: DE1_SOPC.
# 2008.05.13 17:12:29 (*) THE FOLLOWING SYSTEM ITEMS HAVE BEEN GENERATED:
 SOPC Builder database : D:/NIOS_II/DE1/DE1_NIOSII/DE1_SOPC.ptf
 System HDL Model : D:/NIOS_II/DE1/DE1_NIOSII/DE1_SOPC.v
 System Generation Script : D:/NIOS_II/DE1/DE1_NIOSII/DE1_SOPC_generation_script
# 2008.05.13 17:12:29 (*) SUCCESS: SYSTEM GENERATION COMPLETED.
Info: System generation was successful.
```

```
Warning: key: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
Warning: switch: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
Info: cfi_flash: Flash memory capacity: 4.0 MBytes (4194304 bytes).
Warning: sd_dat: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
Warning: sd_dat3: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
Warning: sd_cmd: PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.
```

Instantiate SOPC in Quartus Top



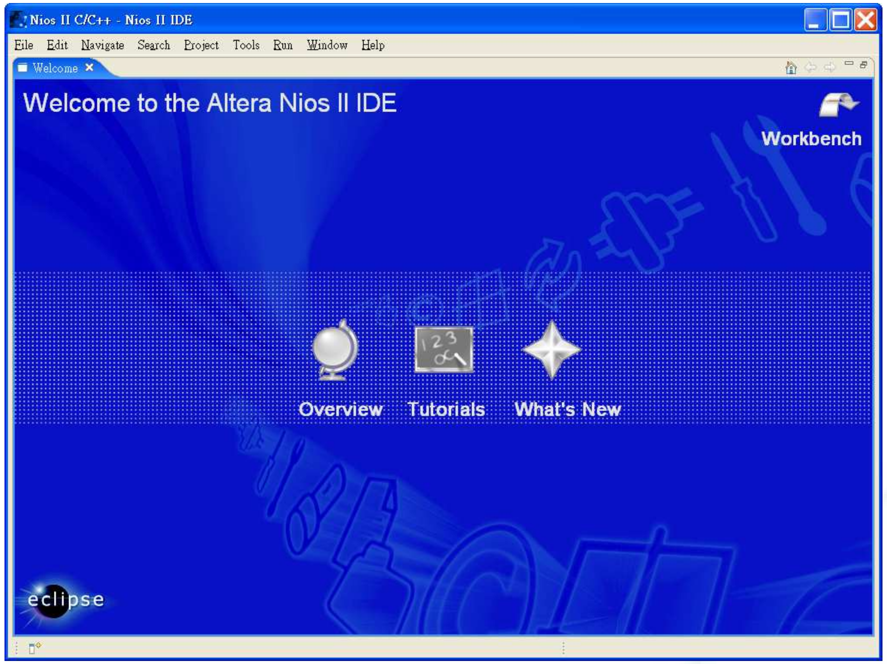
```
DE1_SOPC DE1_SOPC_Instance(
                // 1) global signals:
                 .clk(CLOCK_50),
                 .pll_c0_cpu(),
                 .pll_c1_sdram(DRAM_CLK),
                 .reset_n(1),

                // the_key
                 .in_port_to_the_key(KEY),

                // the_ledg
                 .out_port_from_the_ledg(LEDG),

                // the_ledr
                 .out_port_from_the_ledr(LEDR),

                // the_sdram
                 .zs_ba_from_the_sdram({DRAM_BA_1,DRAM_BA_0}),
                 .zs_cas_n_from_the_sdram(DRAM_CAS_N),
                 .zs_cke_from_the_sdram(DRAM_CKE),
                 .zs_cs_n_from_the_sdram(DRAM_CS_N),
                 .zs_dq_to_and_from_the_sdram(DRAM_DQ),
                 .zs_dqm_from_the_sdram({DRAM_UDQM,DRAM_LDQM}),
                 .zs_ras_n_from_the_sdram(DRAM_RAS_N),
                 .zs_we_n_from_the_sdram(DRAM_WE_N),
```

NIOS II Programming

- New NIOS II Project
- NIOS II System API

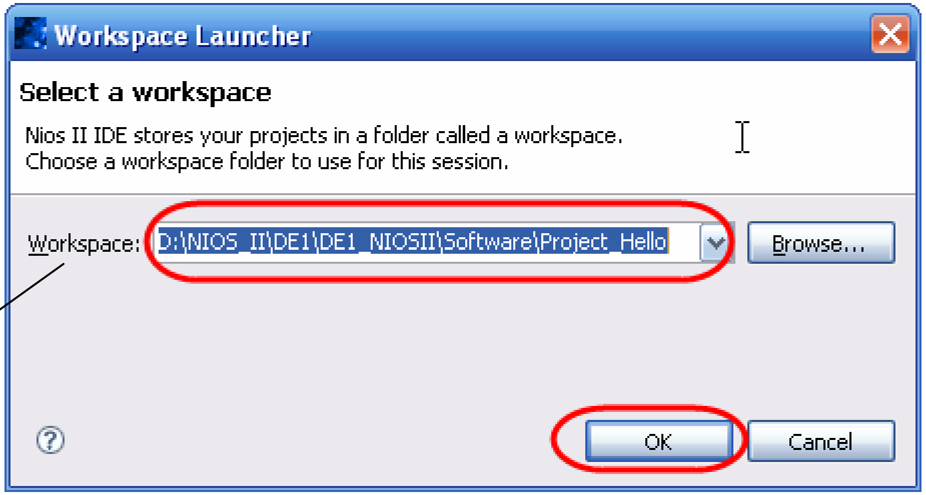
Start NIOS II IDE 7.2

- Windows選單“開始→所有程式→Altera→NIOS II EDS 7.2→NIOS II IDE 7.2”


Nios II C/C++ - Nios II IDE
File Edit Navigate Search Project Tools Run Window Help
Welcome
Welcome to the Altera Nios II IDE
Workbench
Overview
Tutorials
What's New
eclipse

Setup Workspace

- 選單“File→Switch Workspace…”



Workspace Launcher

Select a workspace

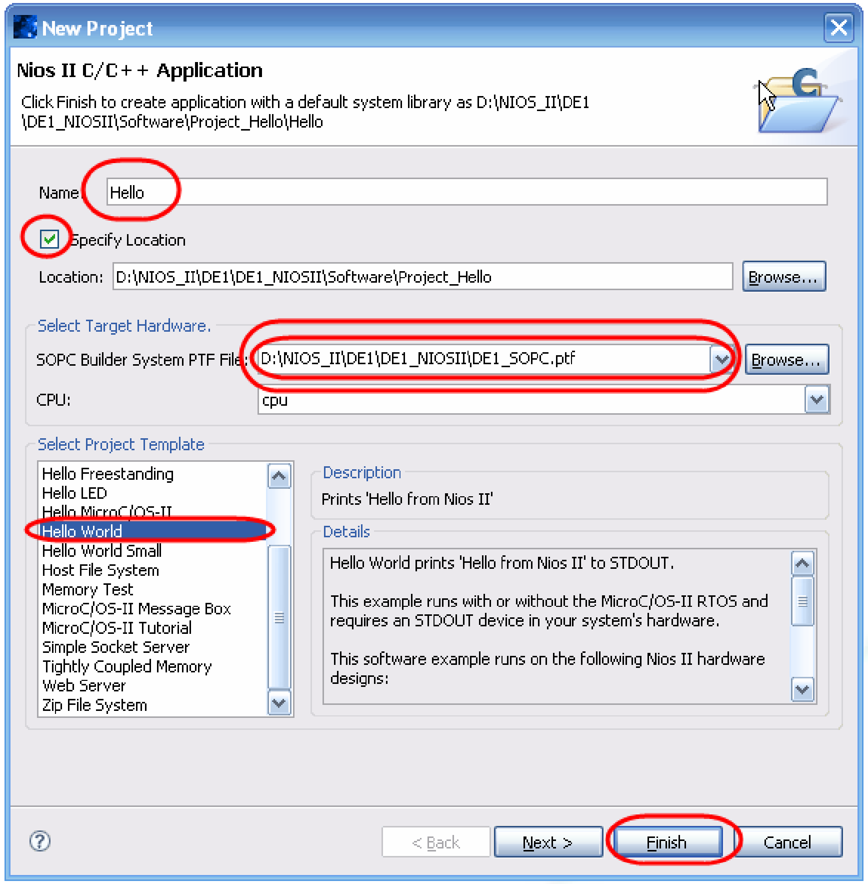
Nios II IDE stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: D:\NIOS_II\DE1\DE1_NIOSII\Software\Project_Hello

D:\NIOS_II\DE1\ software\project_hello

New Nios Project

- 選單“File→New→Nios II C/C++ Application”



Nios II C/C++ Application

Click Finish to create application with a default system library as D:\NIOS_II\DE1\DE1_NIOSII\Software\Project_Hello\Hello

Name: Hello

☑ Specify Location

Location: D:\NIOS_II\DE1\DE1_NIOSII\Software\Project_Hello

Select Target Hardware.

SOPC Builder System PTF File: D:\NIOS_II\DE1\DE1_NIOSII\DE1_SOPC.ptf

CPU: cpu

Select Project Template

- Hello Freestanding
- Hello LED
- Hello MicroC/OS-II
- Hello World
- Hello World Small
- Host File System
- Memory Test
- MicroC/OS-II Message Box
- MicroC/OS-II Tutorial
- Simple Socket Server
- Tightly Coupled Memory
- Web Server
- Zip File System

Description

Prints 'Hello from Nios II'

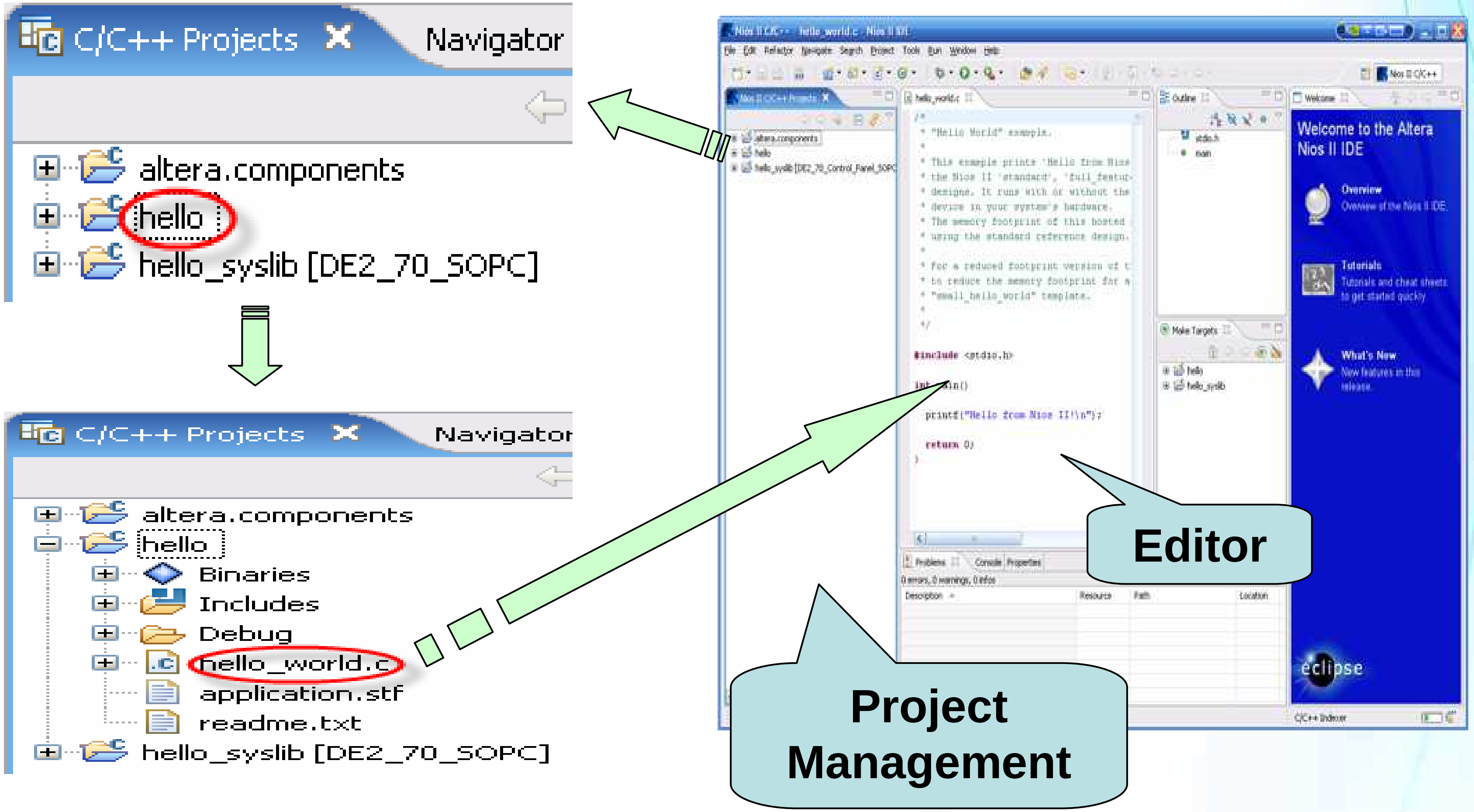
Details

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.

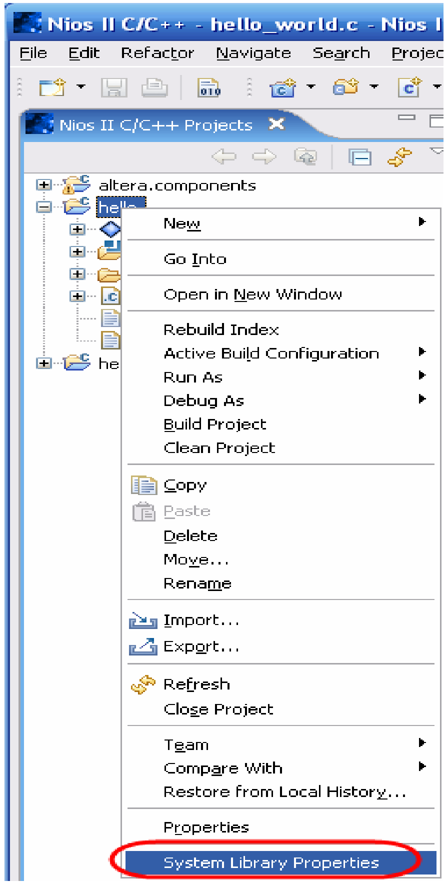
This software example runs on the following Nios II hardware designs:

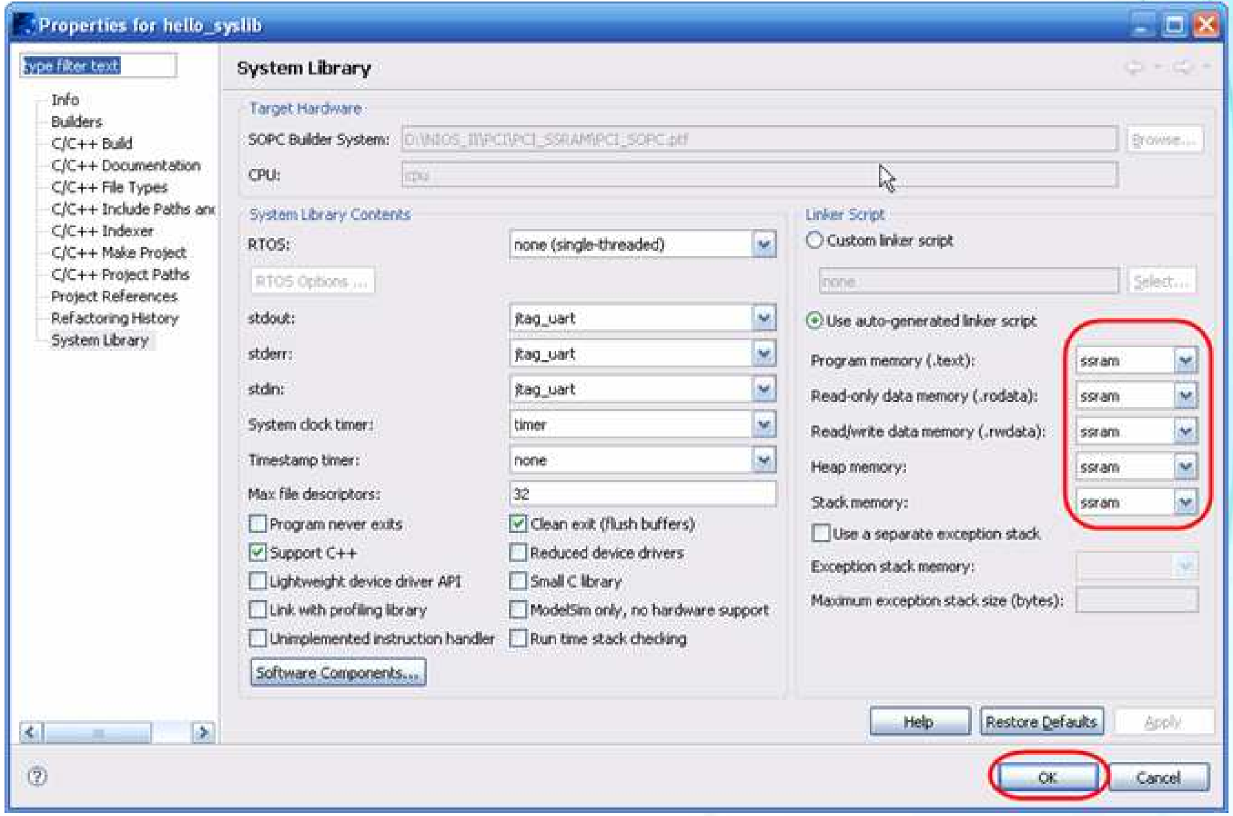
Hello Project



```
/*
 * "Hello World" example.
 *
 * This example prints 'Hello from Nios
 * the Nios II 'standard', 'full_featu
 * designs. It runs with or without the
 * device in your system's hardware.
 * The memory footprint of this hosted
 * using the standard reference design.
 *
 * For a reduced footprint version of t
 * to reduce the memory footprint for a
 * "small_hello_world" template.
 *
 */

#include <stdio.h>

int main()
{
  printf("Hello from Nios II!\n");

  return 0;
}
```

Project Configuration





System Library

Target Hardware

SOPC Builder System: D:\NIOS_II\PCI\PCI_SSRAM\PCI_SOPC.ptf

CPU: cpu

System Library Contents

RTOS: none (single-threaded)

stdout: jtag_uart

stderr: jtag_uart

stdin: jtag_uart

System clock timer: timer

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

Program memory (.text): ssram

Read-only data memory (.rodata): ssram

Read/write data memory (.rwdata): ssram

Heap memory: ssram

Stack memory: ssram

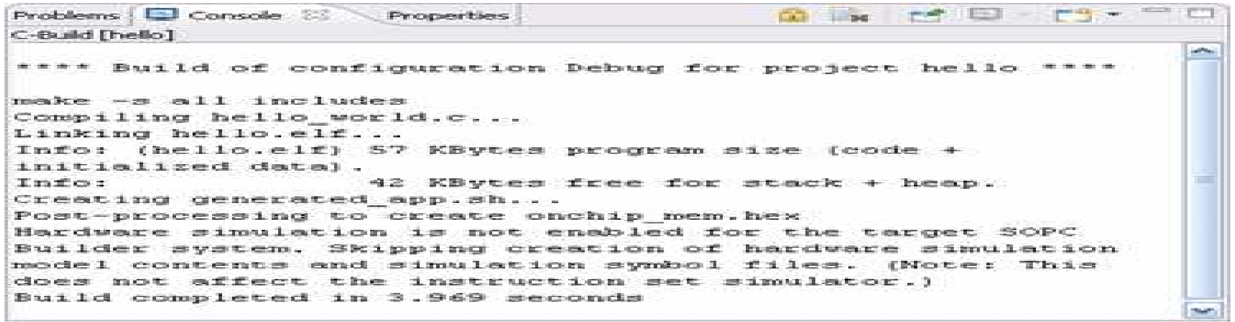
□ Use a separate exception stack

Exception stack memory:

Maximum exception stack size (bytes):

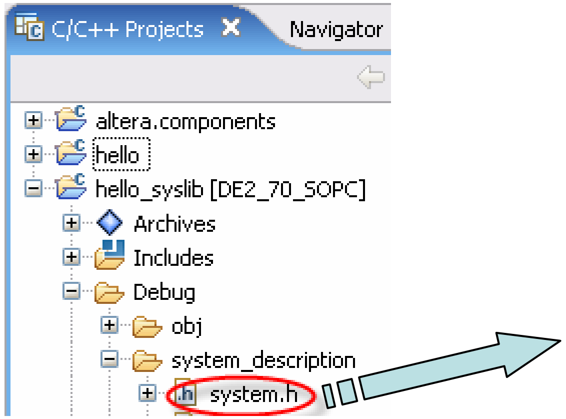
Compile

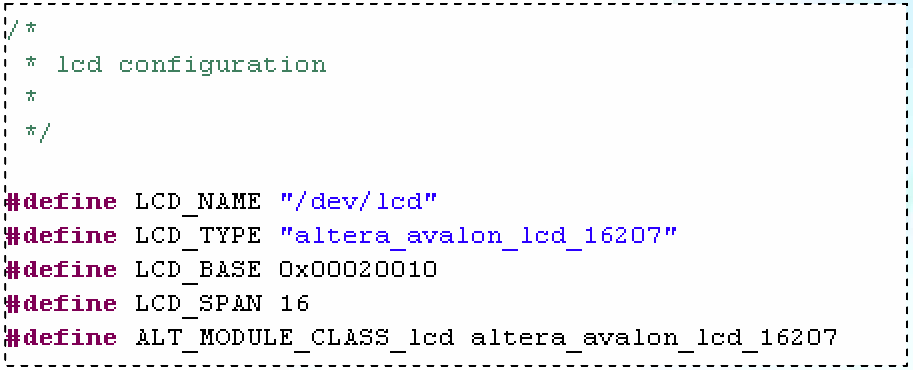
- Menu “Project→Build All”



```
**** Build of configuration Debug for project hello ****

make -s all includes
Compiling hello_world.c...
Linking hello.elf...
Info: (hello.elf) 57 KBytes program size (code +
initialized data).
Info:                42 KBytes free for stack + heap.
Creating generated_app.sh...
Post-processing to create onchip_mem.hex
Hardware simulation is not enabled for the target SOPC
Builder system. Skipping creation of hardware simulation
model contents and simulation symbol files. (Note: This
does not affect the instruction set simulator.)
Build completed in 3.969 seconds
```

- Sytem.h

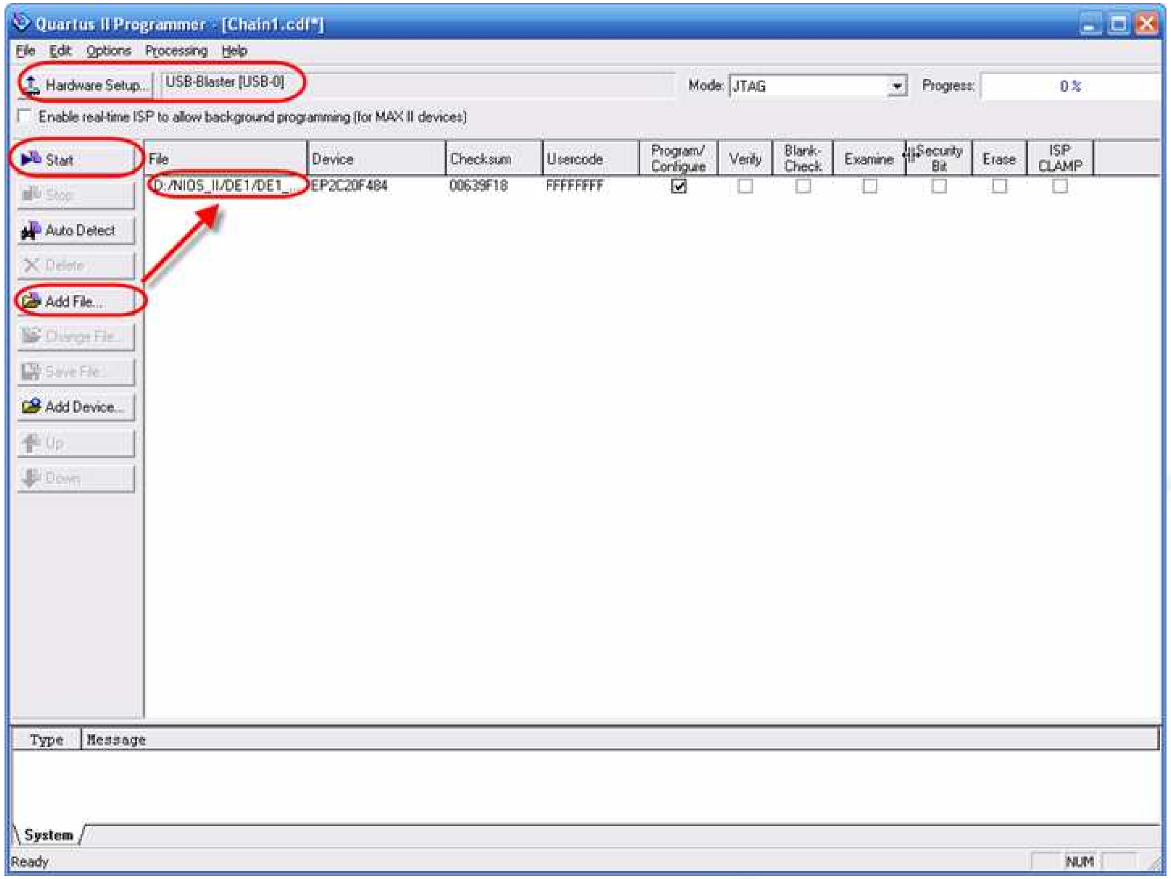




```
/*
 * lcd configuration
 *
 */

#define LCD_NAME "/dev/lcd"
#define LCD_TYPE "altera_avalon_lcd_16207"
#define LCD_BASE 0x00020010
#define LCD_SPAN 16
#define ALT_MODULE_CLASS_lcd altera_avalon_lcd_16207
```

Download Hardware .SOF

- Menu “Tools→Quartus II Progammer”



USB-Blaster [USB-0]

Mode: JTAG

Progress: 0 %

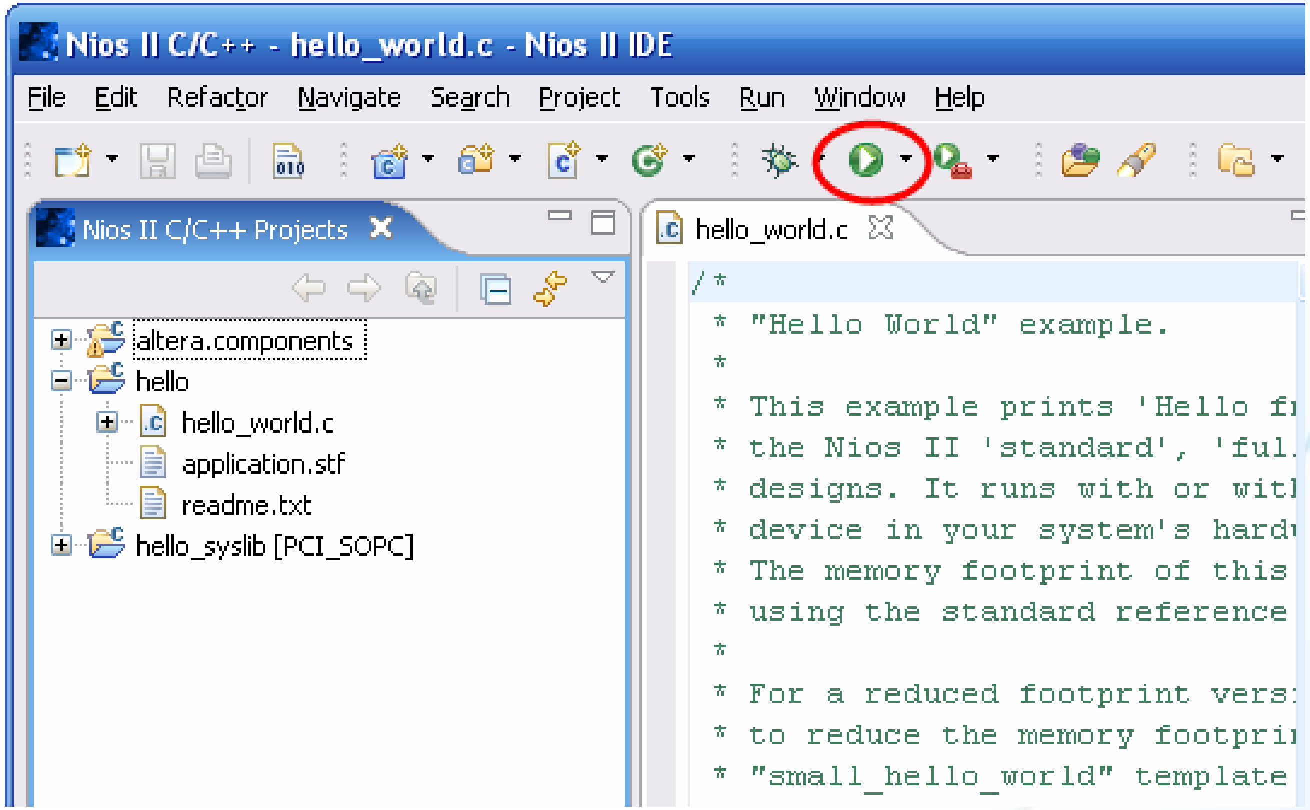
□ Enable real-time ISP to allow background programming (for MAX II devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
D:/NIOS_II/DE1/DE1_...	EP2C20F484	00639F18	FFFFFFFF	☑	□	□	□	□	□	□

Ready

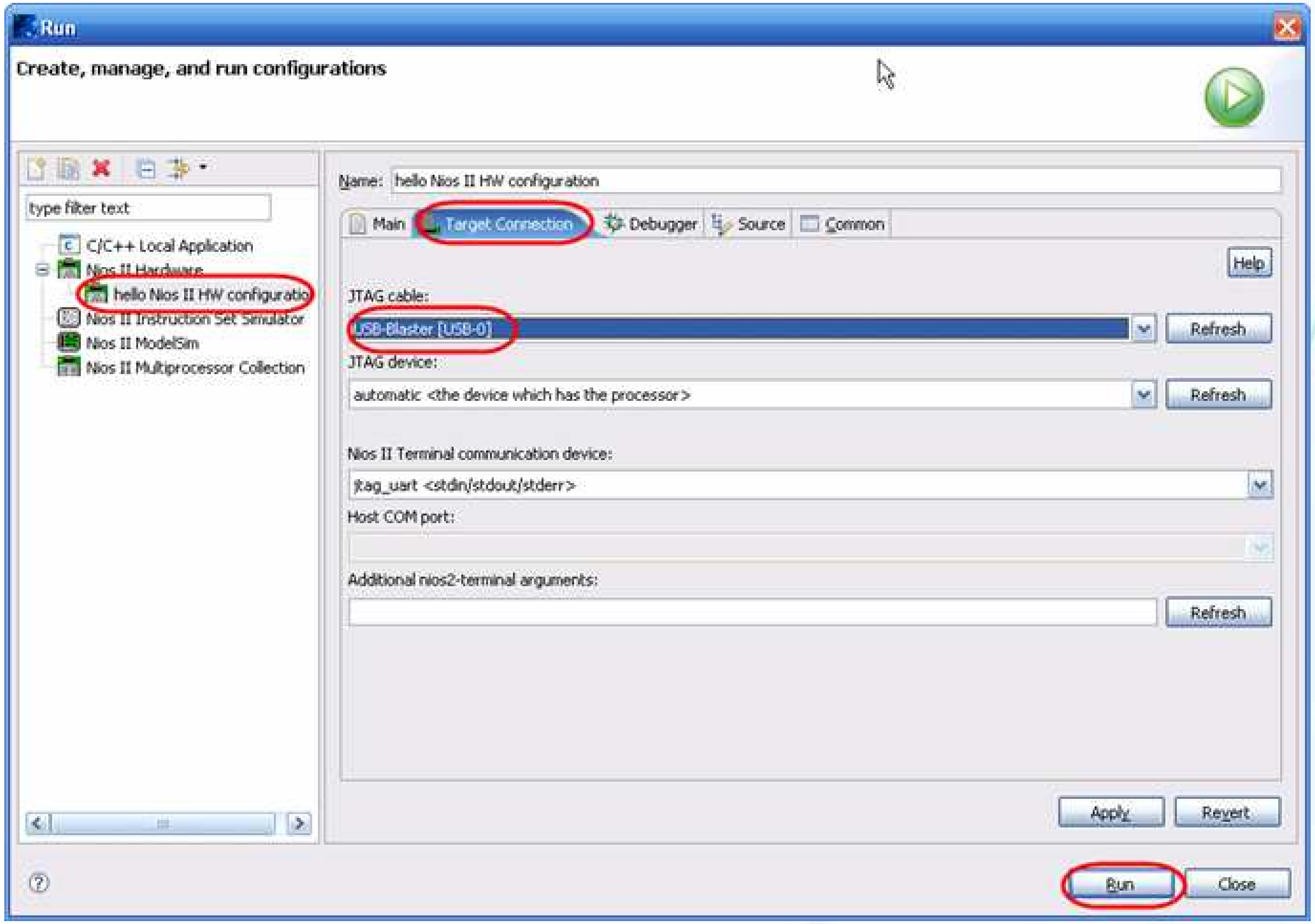
NUM

Run



```
/*
 * "Hello World" example.
 *
 * This example prints 'Hello fr
 * the Nios II 'standard', 'full
 * designs. It runs with or with
 * device in your system's hardw
 * The memory footprint of this
 * using the standard reference
 *
 * For a reduced footprint versi
 * to reduce the memory footprin
 * "small_hello_world" template
```

Run Configuration



Create, manage, and run configurations

Name: hello Nios II HW configuration

JTAG cable: USB-Blaster [USB-0]

JTAG device: automatic <the device which has the processor>

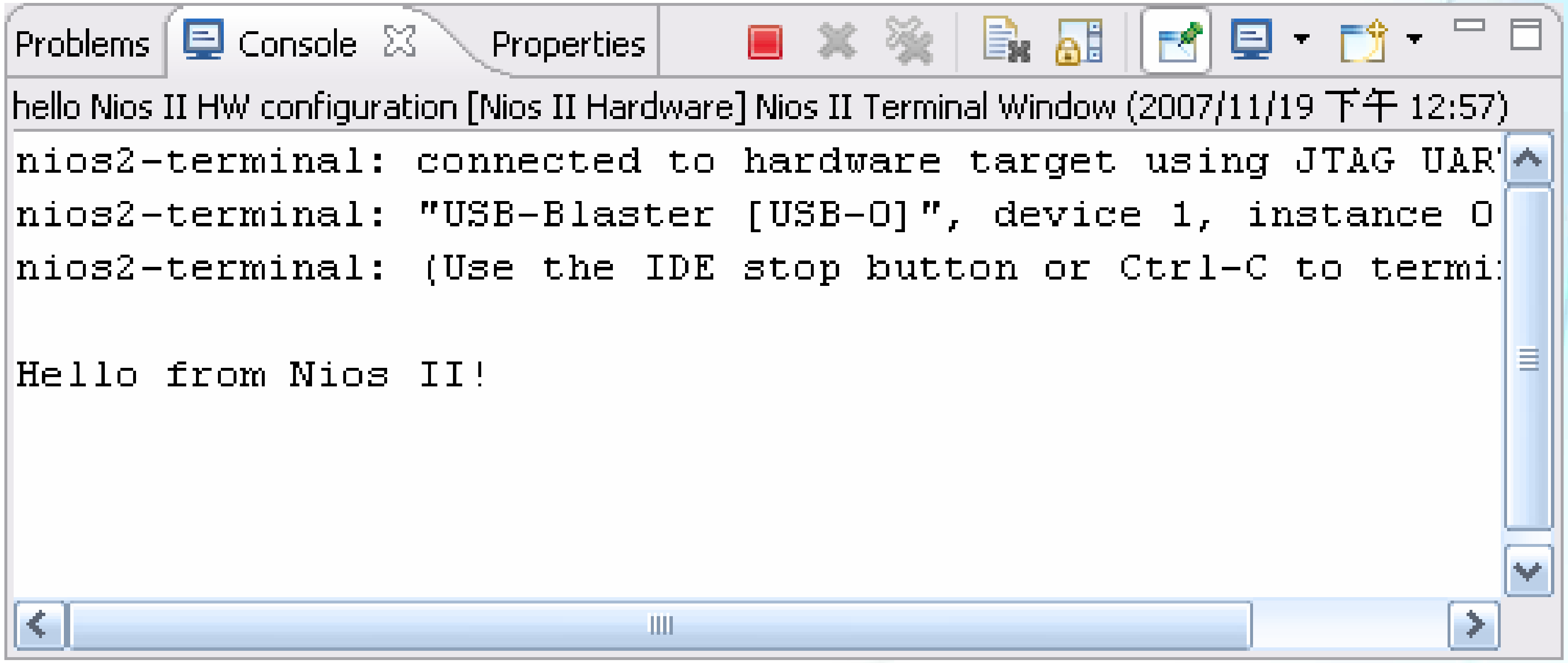
Nios II Terminal communication device: jtag_uart <stdin/stdout/stderr>

Host COM port:

Additional nios2-terminal arguments:

Result

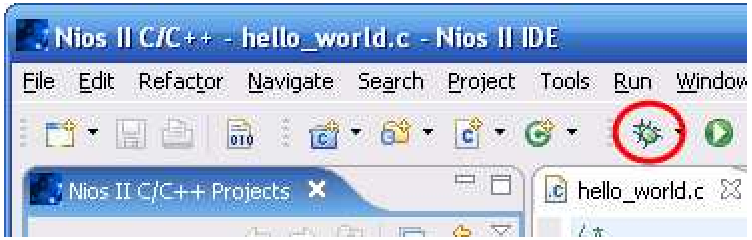
- “Hello from Nios II!” appears in Consol Window

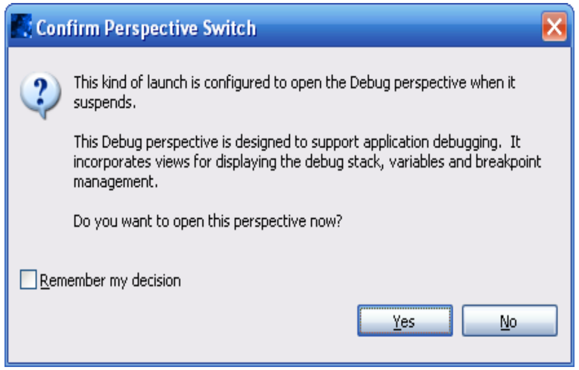


hello Nios II HW configuration [Nios II Hardware] Nios II Terminal Window (2007/11/19 下午 12:57)

```
nios2-terminal: connected to hardware target using JTAG UAR
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to termi

Hello from Nios II!
```

Debug


Nios II C/C++ - hello_world.c - Nios II IDE
File Edit Refactor Navigate Search Project Tools Run Window
Nios II C/C++ Projects
hello_world.c

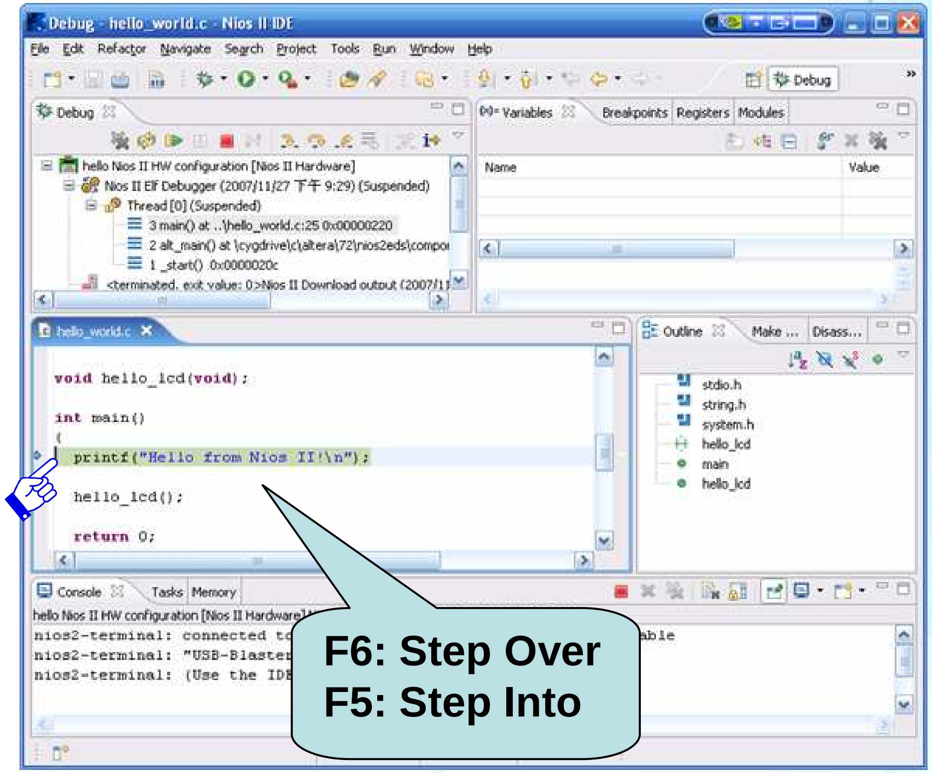


This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

☐ Remember my decision



```
hello Nios II HW configuration [Nios II Hardware]
  Nios II Elf Debugger (2007/11/27 下午 9:29) (Suspended)
    Thread [0] (Suspended)
      3 main() at ..\hello_world.c:25 0x00000220
      2 alt_main() at \cygdrive\c\altera\72\nios2eds\compo
      1 _start() 0x0000020c
  <terminated, exit value: 0>Nios II Download output (2007/11
```

```
void hello_lcd(void);

int main()
{
  printf("Hello from Nios II!\n");

  hello_lcd();

  return 0;
```

```
hello Nios II HW configuration [Nios II Hardware]
nios2-terminal: connected to                    able
nios2-terminal: "USB-Blaster
nios2-terminal: (Use the IDE
```

F6: Step Over
F5: Step Into

NIOS II System API

Altera Data Type

- alt_32: singed 32-bit integer
- alt_u32: unsigned 32-bit integer
- alt_16: singed 16-bit integer
- alt_u16: unsigned 16-bit integer
- alt_8: singed 8-bit integer
- alt_u8: unsigned 8-bit integer
- defined in “alt_types.h”

Access Hardware

- Access Memory:
  - Pointer (Data Cache Enabled)
- Access Device Register:
  - IORD, IOWR (Data Cache Disabled)
- Access by HAL API:
  - open, write, read, close (and fcntl)
  - alt_xxx

PIO-LED Example

LED閃爍

```
void test1_led(void){
    alt_u32 led_mask=0;
    while(1){
        // green led control
        IOWR_ALTERA_AVALON_PIO_DATA(
                        PIO_GREEN_LED_BASE, led_mask);
        // red led control
        IOWR_ALTERA_AVALON_PIO_DATA(
                        PIO_RED_LED_BASE, led_mask);
        // toggle led
        led_mask ^= 0xFFFFFFFF;

        // sleep 0.2 second
        usleep(200*1000);
    } // while
}
// PIO_GREEN_LED_BASE & PIO_RED_LED_BASE defined in system.h
```

PIO-SWITCH Example

Switch狀態顯示

```
void test2_switch(void){
    alt_u32 mask;
    while(1){
        // (switch up) active-high
        mask = IORD_ALTERA_AVALON_PIO_DATA(PIO_SWITCH_BASE);

         // high-active
        IOWR_ALTERA_AVALON_PIO_DATA(PIO_RED_LED_BASE, mask);
    }
}
```

PIO-IRQ Example (1)

Pushbutton IRQ Enable

```
void test3_irq_pushbutton(void){
    static alt_u8 led_indicate=0x00;
    // init led indicator
    IOWR_ALTERA_AVALON_PIO_DATA(PIO_GREEN_LED_BASE, led_indicate);

    // enable interrupt, 4-button
    IOWR_ALTERA_AVALON_PIO_IRQ_MASK(PIO_BUTTON_BASE, 0x0F);

    // Reset the edge capture register
    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PIO_BUTTON_BASE,0);

    // register ISR
    if ((alt_irq_register(PIO_BUTTON_IRQ, (void *)&led_indicate, pushbutton_isr) != 0))
        printf("[pushbutton]register button IRQ fail\n");
    else
        printf("[pushbutton]register button IRQ success\n");
}
```

PIO-IRQ Example (2)

Pushbutton ISR

```
void pushbutton_isr(void* context, alt_u32 id){
    alt_u8 pushbutton_mask;
    alt_u8 *pled_indicate = (alt_u8*)context;

    // get the edge capture mask
    pushbutton_mask = IORD_ALTERA_AVALON_PIO_EDGE_CAP(
                                    PIO_BUTTON_BASE) & 0x0F;  // 4-button

    // Reset the edge capture register
    IOWR_ALTERA_AVALON_PIO_EDGE_CAP(PIO_BUTTON_BASE,0);

    //  update led indicator
   *pled_indicate ^= pushbutton_mask;
    IOWR_ALTERA_AVALON_PIO_DATA(
                    PIO_GREEN_LED_BASE, *pled_indicate);
}
```

Timer Example

Time Measurement

```
void test_timer(void){
    alt_u32 time_start, time_elapsed, ticks_per_second;

    // check hardware
    ticks_per_second = alt_ticks_per_second();
    if (ticks_per_second == 0){
        printf("timer hardware not works well\n");
        return;
    }

    // measure time
    time_start = alt_nticks();
    usleep(1*1000*1000); // sleep 1 second
    time_elapsed = alt_nticks() - time_start;
    printf("[timer test]time elapsed:%.3f seconds\n",
                    (float)time_elapsed/(float)ticks_per_second);

}
```

Alarm Example

LED Blink

```
#define ALRAM_DUR   (alt_ticks_per_second()/2)

alt_u32 alarm_callback(void *context){
    static alt_u8 led_mask = 0xFF;
    IOWR_ALTERA_AVALON_PIO_DATA(PIO_GREEN_LED_BASE, led_mask);
    led_mask ^= 0xFF;
    return ALRAM_DUR;

}

void test_alarm(void){
    int result;
    static alt_alarm alarm;
    result = alt_alarm_start (&alarm, ALRAM_DUR, alarm_callback, NULL);
    if (result != 0)
        printf("[alarm test] failed to start alarm\n");

    // call alt_alram_stop(&alarm) to stop it.
}
```

Timestamp Example

Time Measure

```
void test_timestamp(void){
    alt_u32 timestamp_freq;
    timestamp_freq = alt_timestamp_freq();
    if (timestamp_freq == 0){
        printf("timestamp hardware not works well\n");
        return;
    }
    printf("[timestamp]timestamp_freq = %ld\n", timestamp_freq);
    //
    alt_timestamp_start();
    usleep(1*1000*1000); // sleep 1 second
    printf("[timestamp]timestamp 1:%.3f seconds\n",
                        (float)alt_timestamp()/(float)timestamp_freq);
    usleep(500*1000); // sleep 0.5 second
    printf("[timestamp]timestamp 2:%.3f seconds\n",
                        (float)alt_timestamp()/(float)timestamp_freq);

}
```

UART Example (1)

UART Write & None-Blocking Read

```
void test8_uart(void){
    int uart, result;
    char szHello[] = "\r\nHello from Nios II Uart, please input:\r\n";
    char szRead[1];

    // open uart
    uart = open(UART_NAME, O_ACCMODE);  // UART_NAME defined in system.h
    if (!uart){
        printf("failed to open uart\n");
        return;
    }

    // write uart
    if (write(uart, szHello, strlen(szHello)) != strlen(szHello)){
        printf("failed to write uart");
        close(uart);
        return;
    }
```

UART Example (2)

UART Write & None-Blocking Read

```
 // none-blocking read
    fcntl(uart, F_SETFL, O_ACCMODE | O_NONBLOCK);
    while(result >= 0){
        result = read(uart, szRead, sizeof(szRead));
        if (result == -1){
            printf("failed to read uart");
        }else if (result > 0){
            printf("%c", szRead[0]);
        }
    }
    fcntl(uart, F_SETFL, O_ACCMODE);
    close(uart);

}
```

Memory Access Example

Write & Read

```
void test8_memory(void){
    int i;
    const int test_num = 8;
    alt_u32 data32;
    volatile alt_u32 *pSDRAM = (alt_u32 *)SDRAM_U1_BASE;

    for(i=0;i<test_num;i++){
        *(pSDRAM+i) = i;
    }

    for(i=0;i<test_num;i++){
        data32 = *(pSDRAM+i);
        printf("*(pSDRAM+%d)=%08lXh\n", i, data32);
    }
}
```

Flash Erase Example (1)

Erase Flash

```
void test10_flash_erase(void){
   alt_flash_fd* fd_flash;
   flash_region *regions_flash=0,*nextreg;
   int number_of_regions_flash=0;
   int error_code, r, i, offset;
   alt_u32 length, block_index;

   fd_flash = alt_flash_open_dev(CFI_FLASH_NAME);
   if (fd_flash){
      error_code = alt_get_flash_info(fd_flash,&regions_flash, &number_of_regions_flash);
      if (error_code == 0){
         block_index = 0;
         nextreg = regions_flash;
         for(r=0;r<number_of_regions_flash && !error_code;r++){
            printf("=== region %d, size=%d, offset=%08lX, block_num=%d, block_size=%d\n",
                 r, nextreg->region_size, (alt_u32)nextreg->offset,
                 nextreg->number_of_blocks, nextreg->block_size);
            offset = nextreg->offset;
```

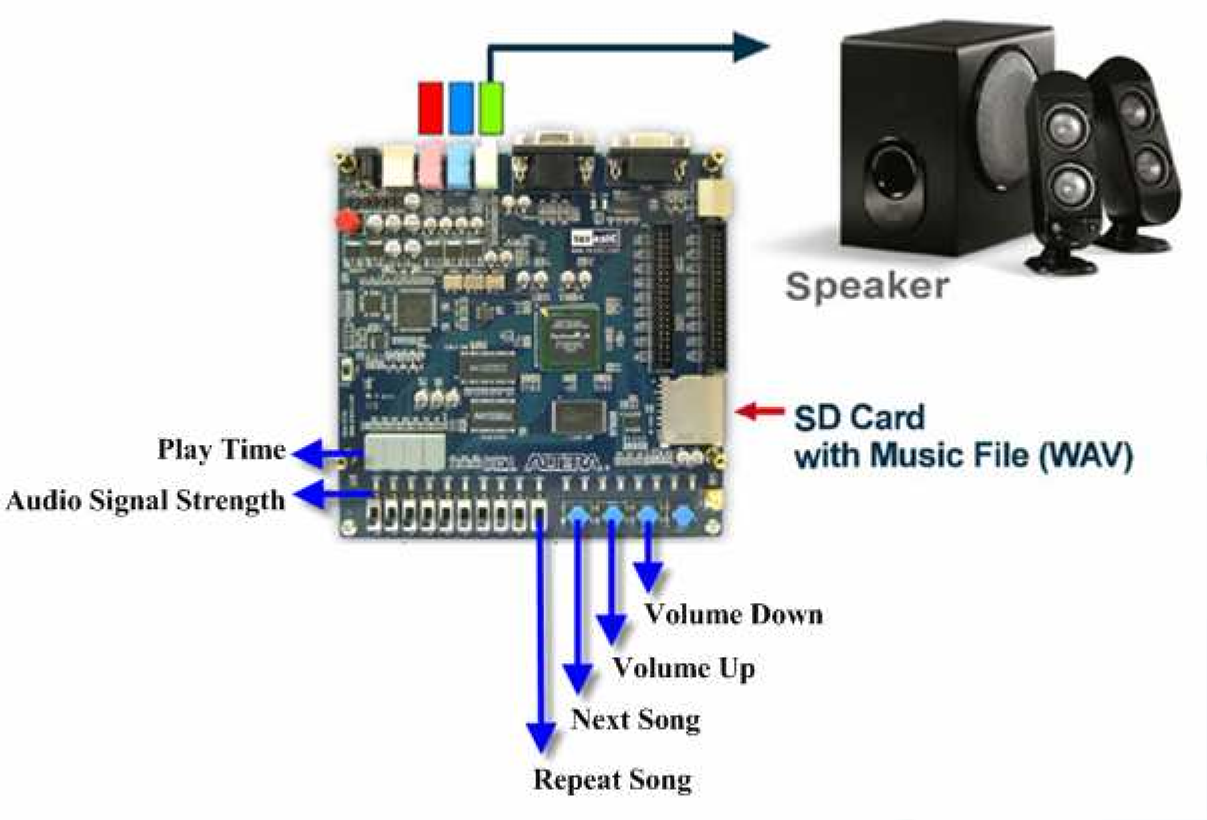
Flash Erase Example (2)

```
                for(i=0;i<nextreg->number_of_blocks && !error_code;i++){
                    length = nextreg->block_size;
                    error_code = alt_erase_flash_block(fd_flash, offset, length);
                    if (error_code)
                        printf("faied to erase flash block  %d\n", block_index);
                    else
                        printf("erase block %d success\n", block_index);
                    offset += length;
                    block_index++;
                } // for i
                nextreg++;
            } // or r
            printf("faied to get flash info\n");
        }
        alt_flash_close_dev(fd_flash);
    }else{
        printf("failed to open flash\n");
    }
}
```

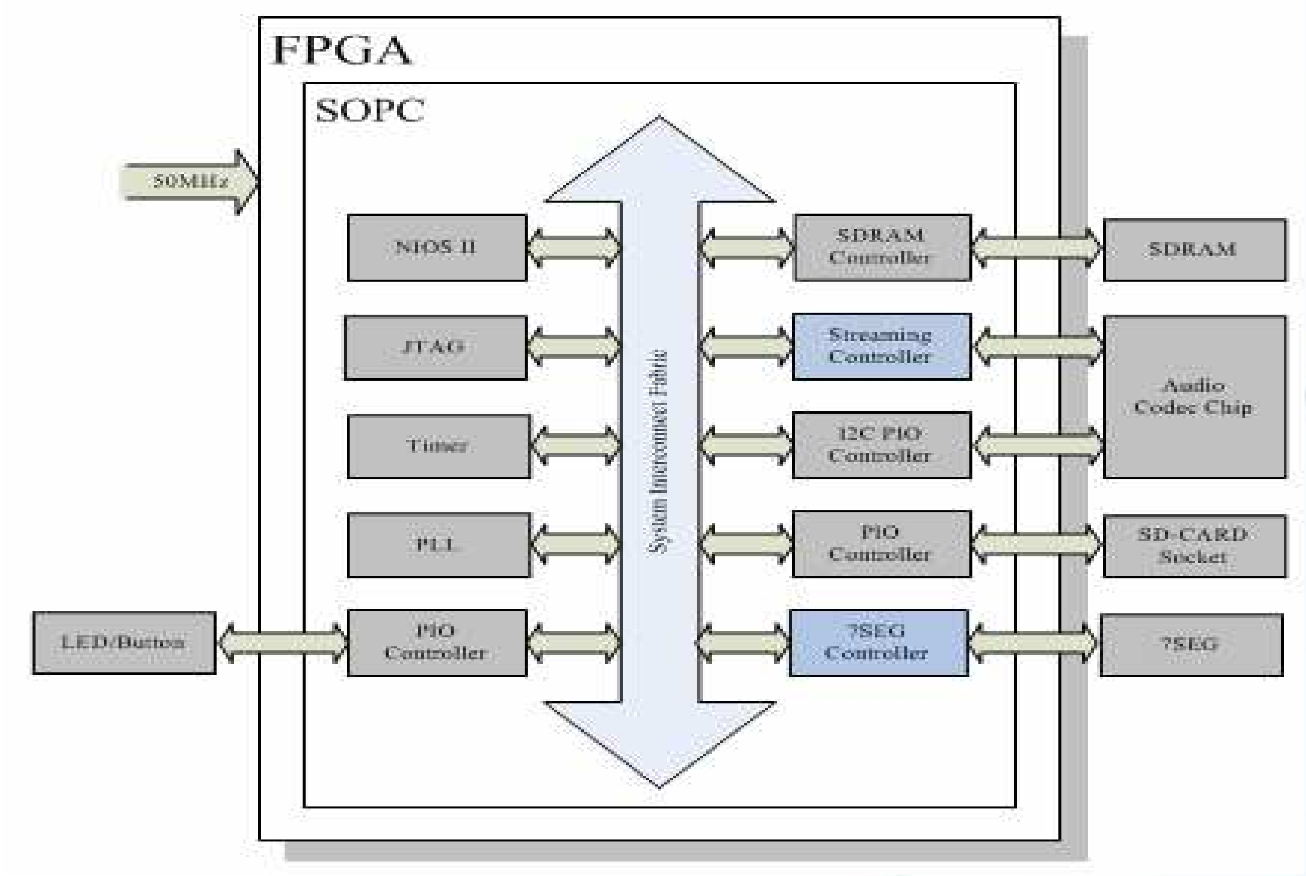
Example
SDCARD Music Player

- ◆SOPC Hardware Design
- ◆NIOS II C Program Design

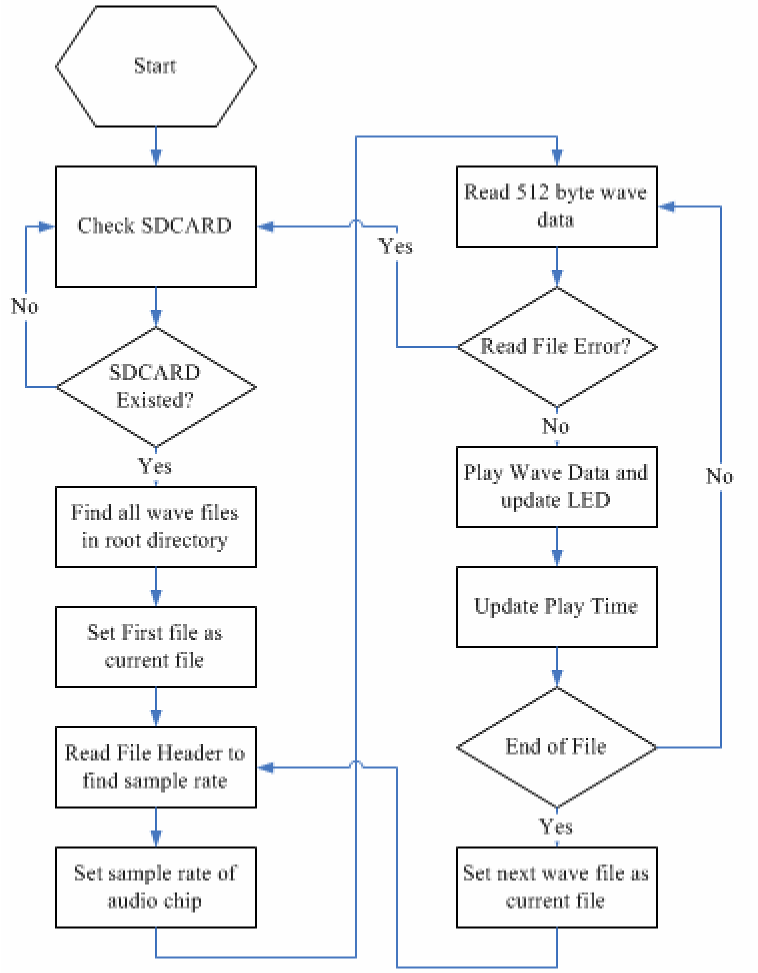
SDCARD Music Player – Function



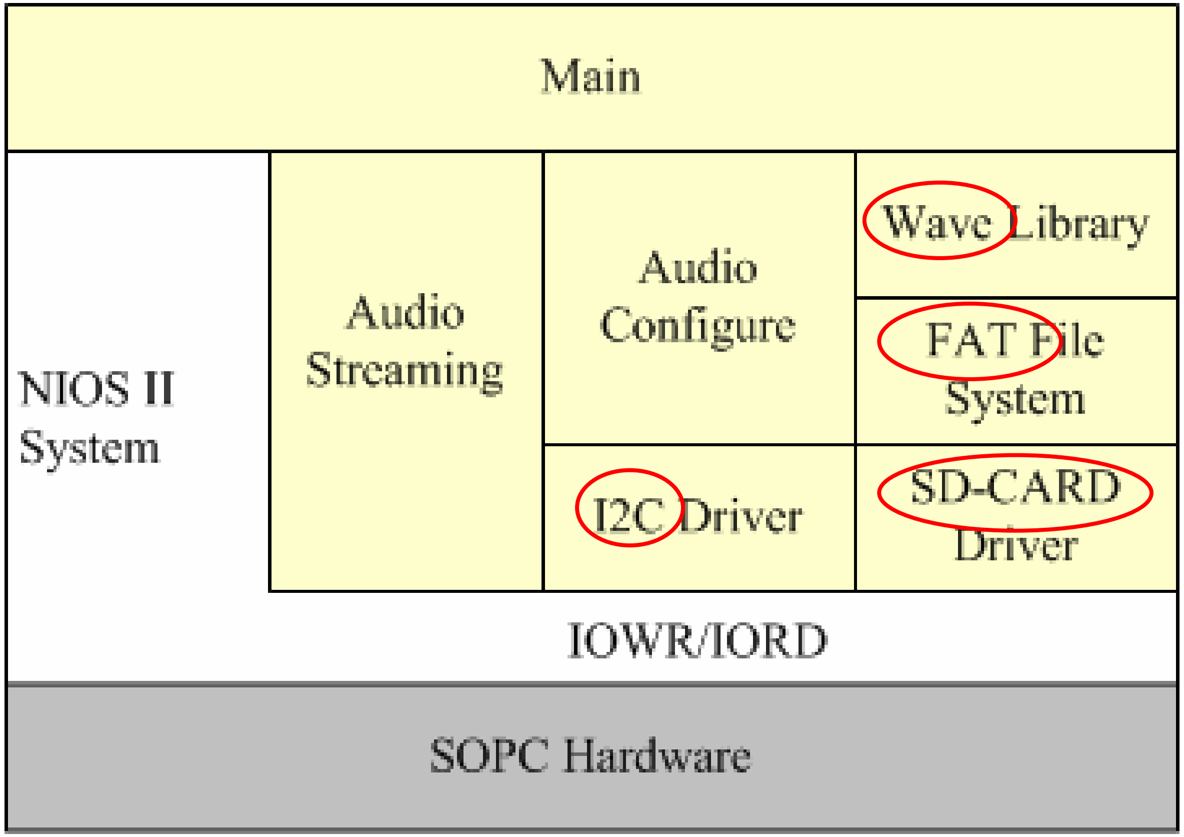
Music Player – SOPC



Program Flow Chart

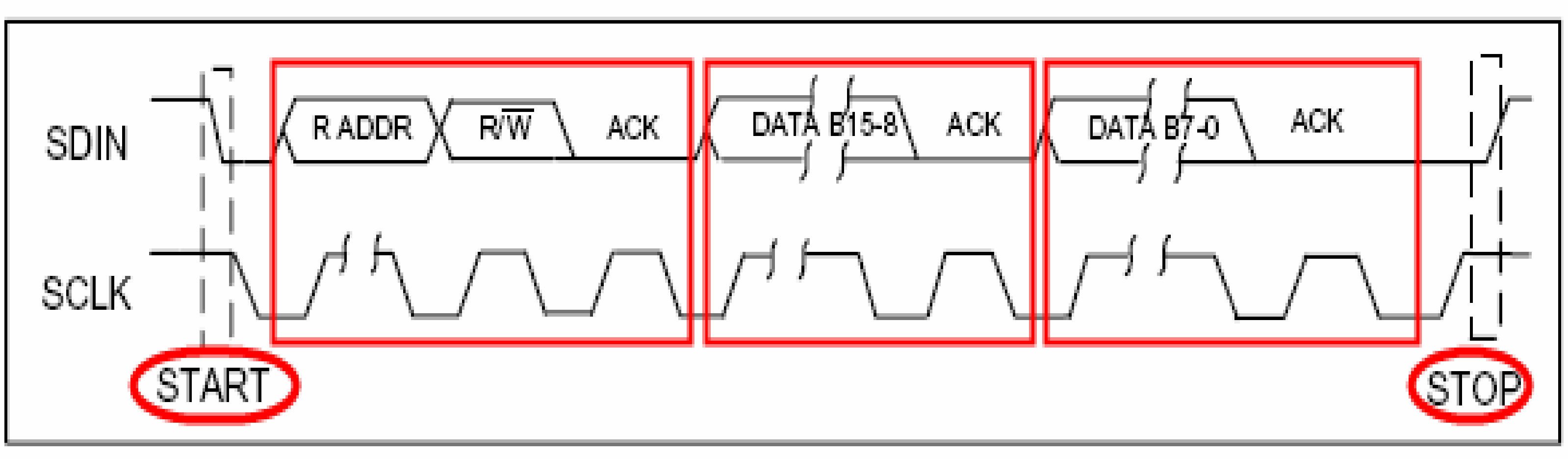

Start
Check SDCARD
No
SDCARD Existed?
Yes
Find all wave files in root directory
Set First file as current file
Read File Header to find sample rate
Set sample rate of audio chip
Read 512 byte wave data
Yes
Read File Error?
No
Play Wave Data and update LED
Update Play Time
No
End of File
Yes
Set next wave file as current file

Music Player – NIOS II Program



I2C Protocol

- Start/Stop: Change at SCLK High
- Data: Change at SCLK Low



I2C Implementation

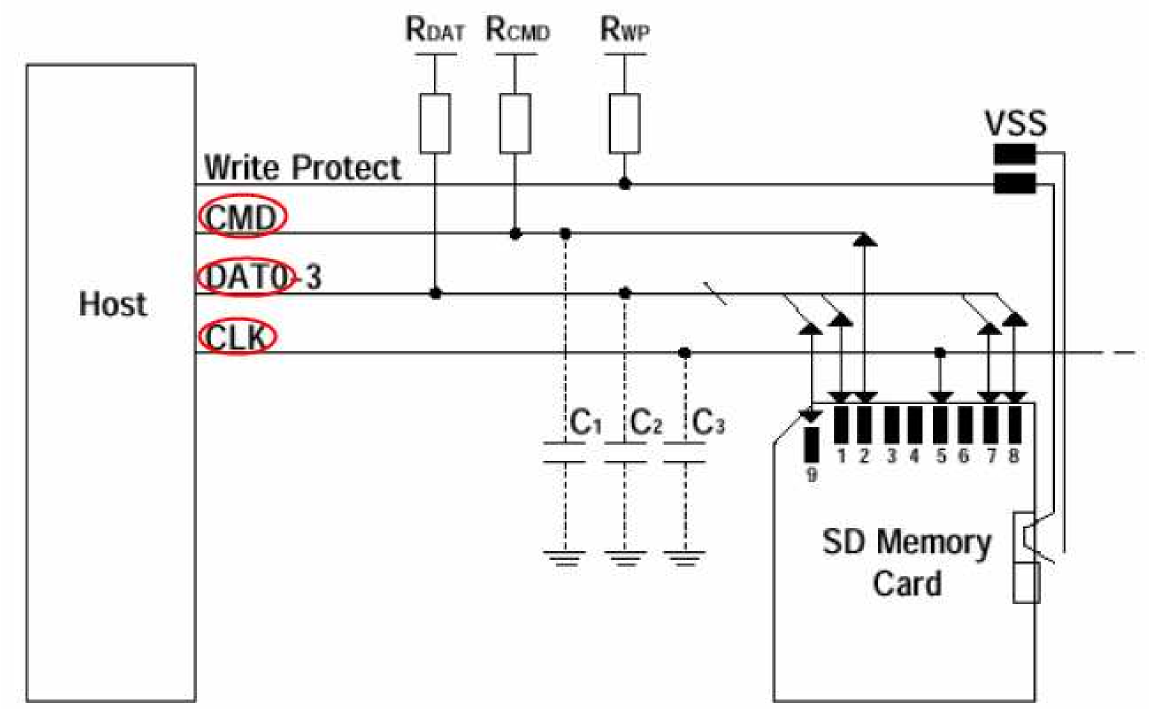
- NIOS Implement I2C protocol
- Use two PIO controllers
  - I2C Clock
  - I2C Data

- I2C Clock Implement (output pin)
  - IORD_ALTERA_AVALON_PIO_DATA

- I2C Data Implement: (inout pin)
  - IOWR_ALTERA_AVALON_PIO_DIRECTION
  - IORD_ALTERA_AVALON_PIO_DATA
  - IOWR_ALTERA_AVALON_PIO_DATA

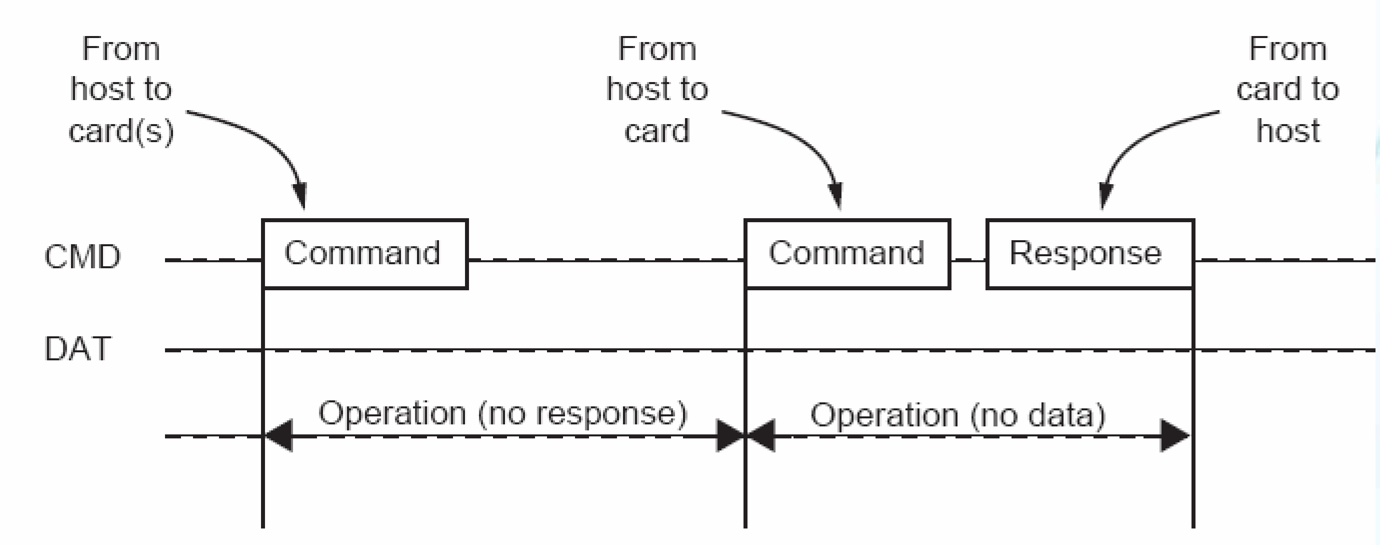
SDCARD Interface

- SD 1-Bit Protocol: CMD, DAT0, CLK



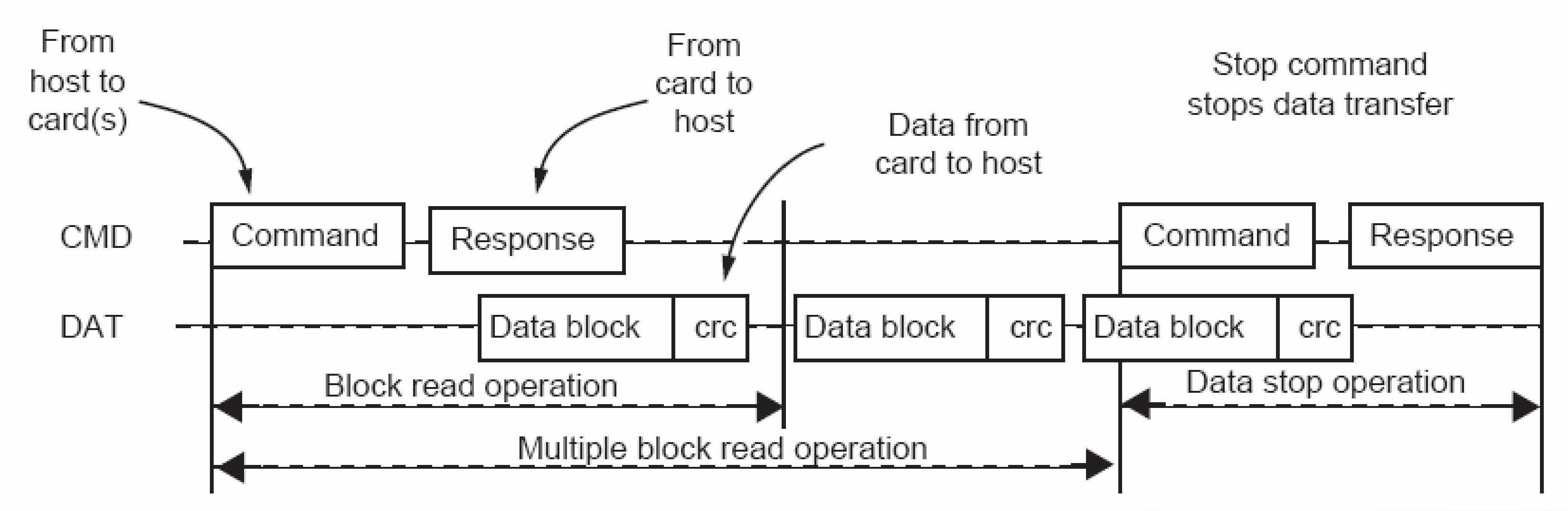
SDCARD – Bus Protocol (1)

- No Data: 1).No response 2). Response



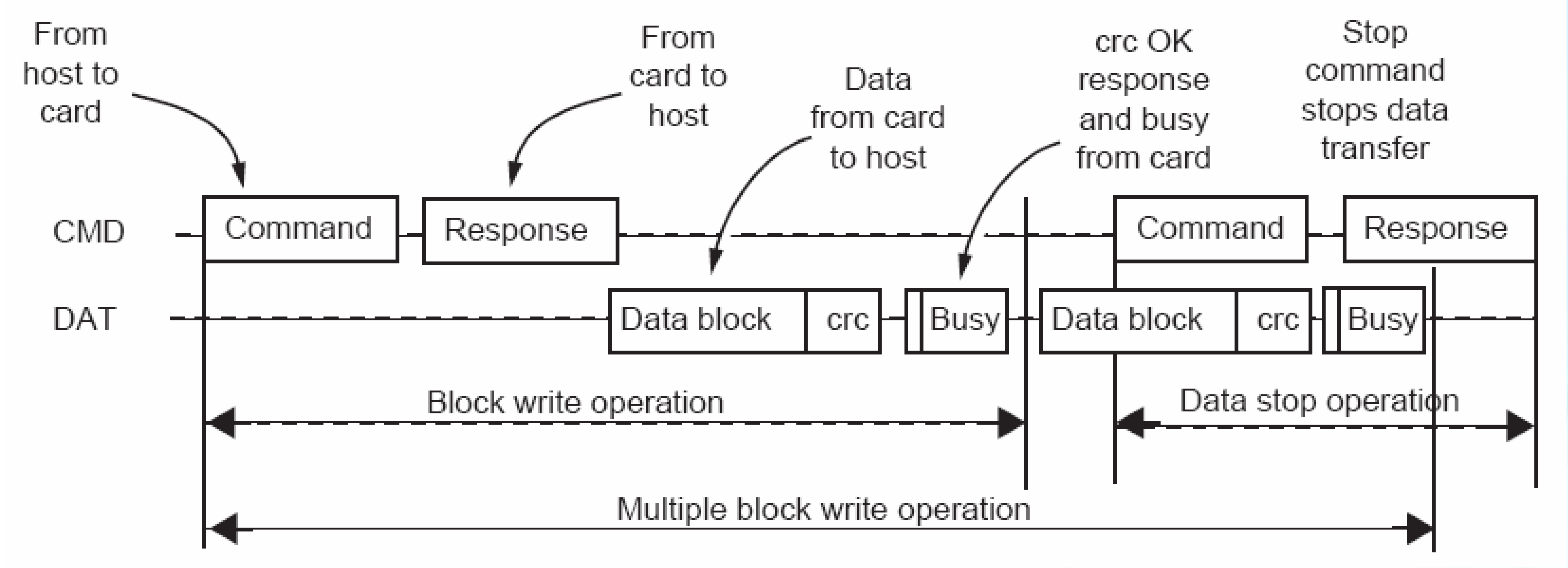
SDCARD – Bus Protocol (2)

- Multiple Block Read



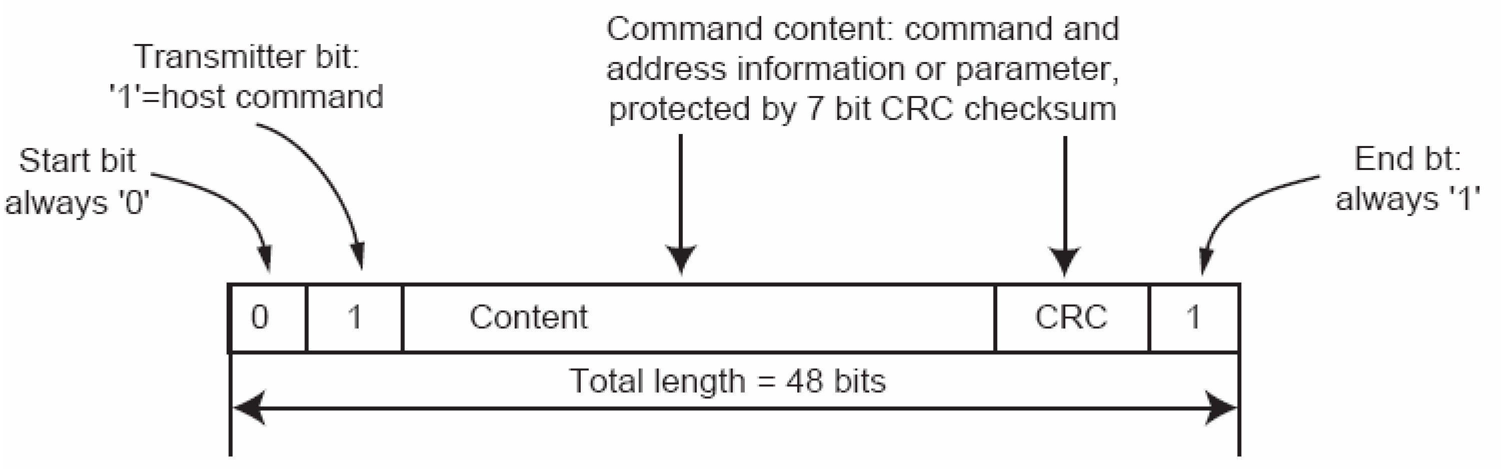
SDCARD – Bus Protocol (3)

- Multiple Block Write



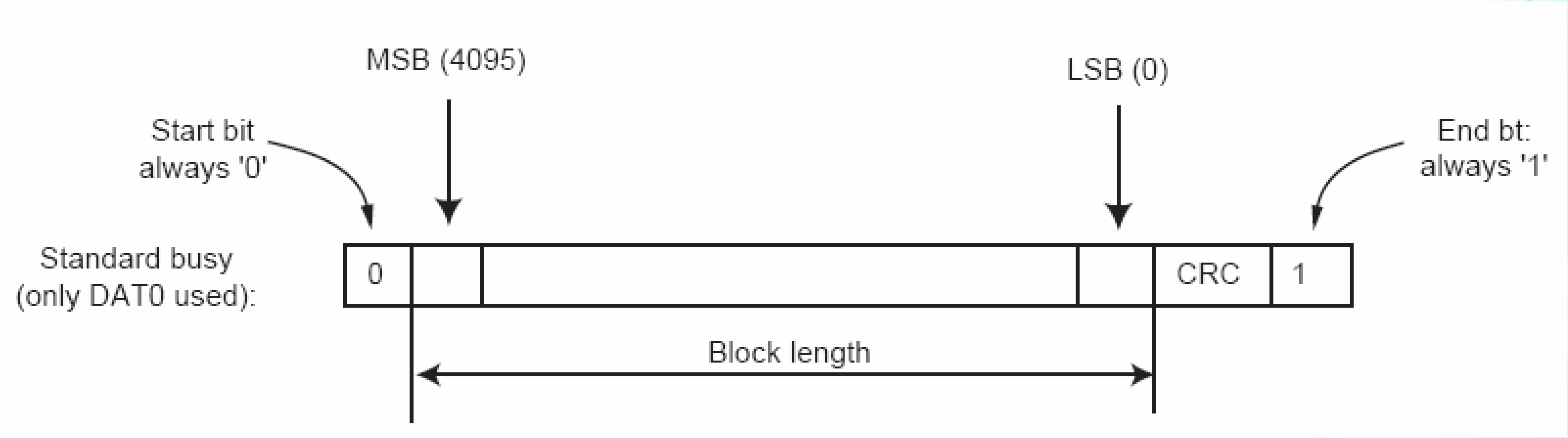
SDCARD – Command Token Format

- Token Size: Fixed to 6 bytes

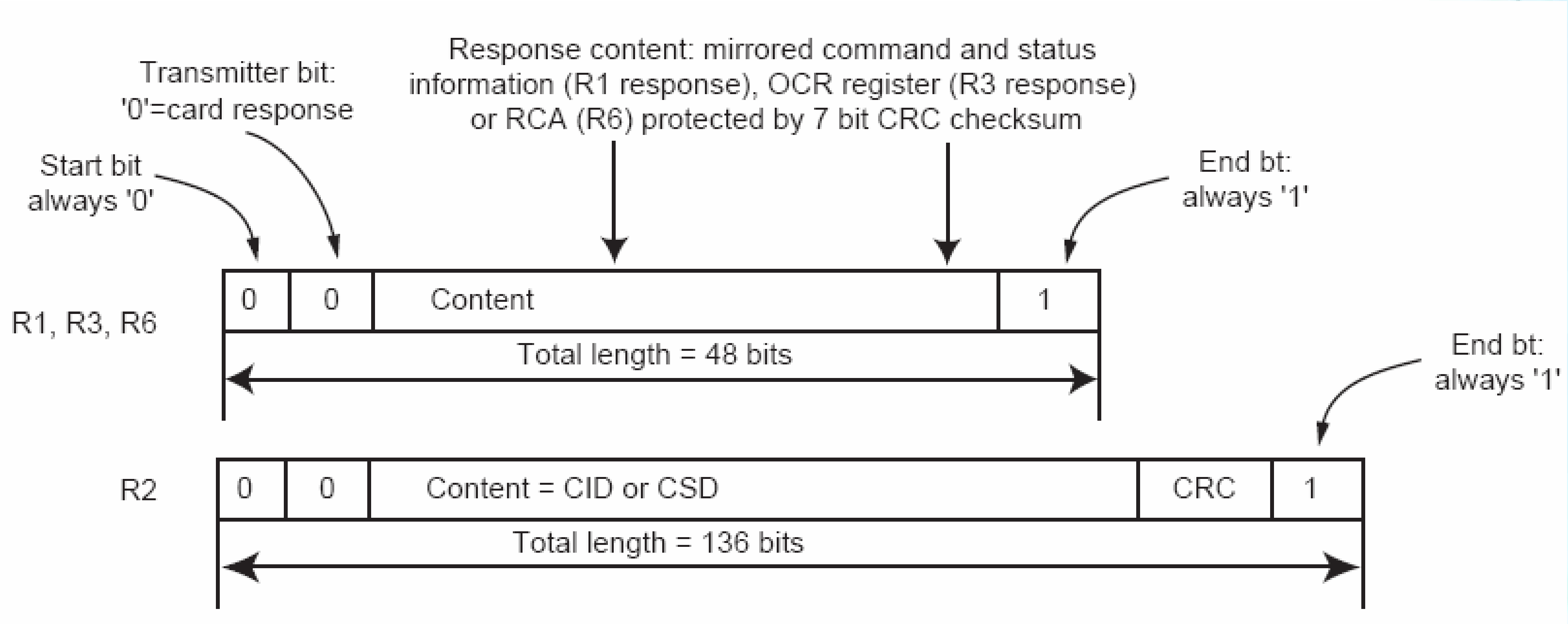


SDCARD – Data Packet Format

- Size: 1~512 Bytes



SDCARD – Response Token Format



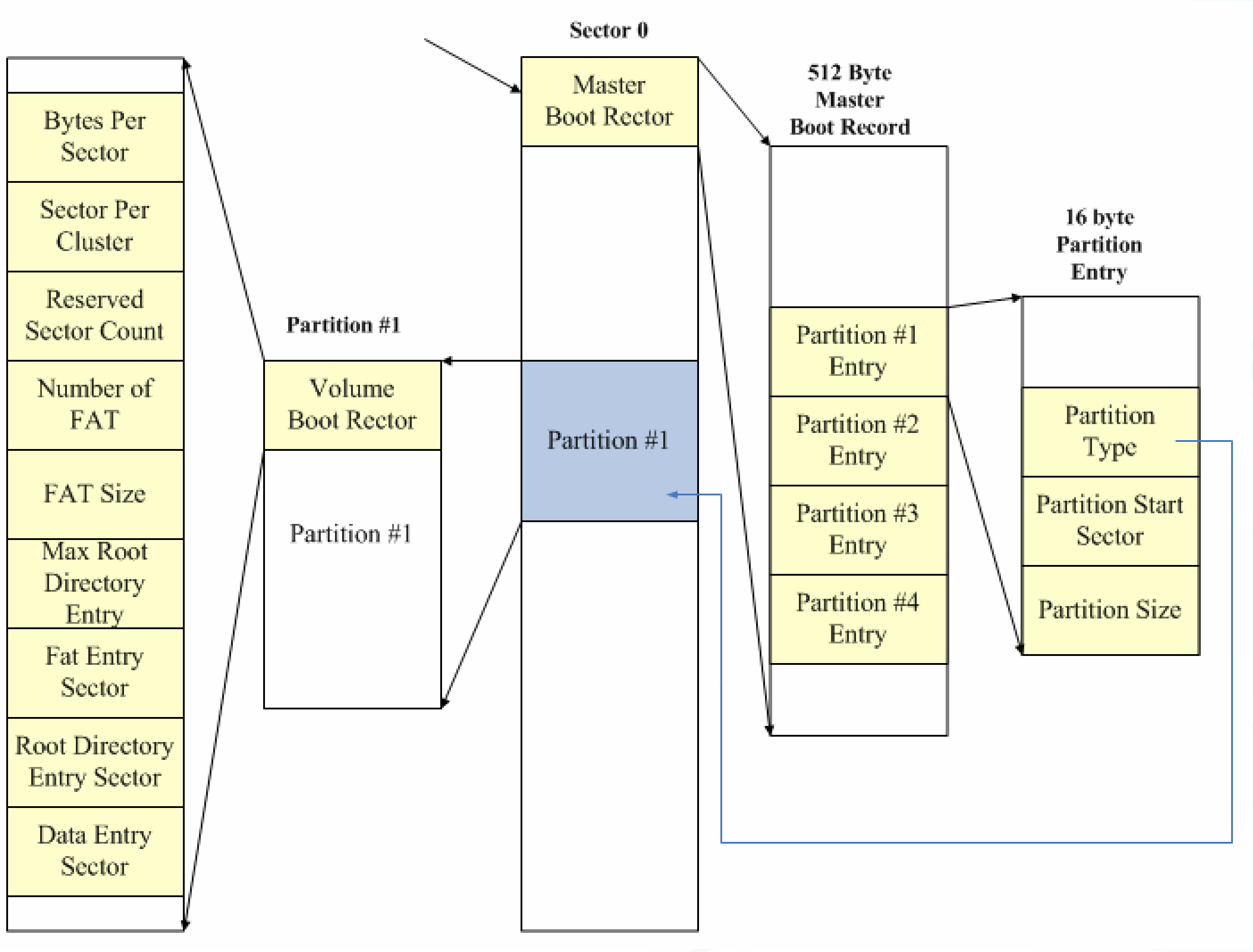
SDCAR Driver Implementation

- NIOS II Implement 1-bit SD mode protocol
- Use three PIO controllers
  - Clock
  - Command
  - Data

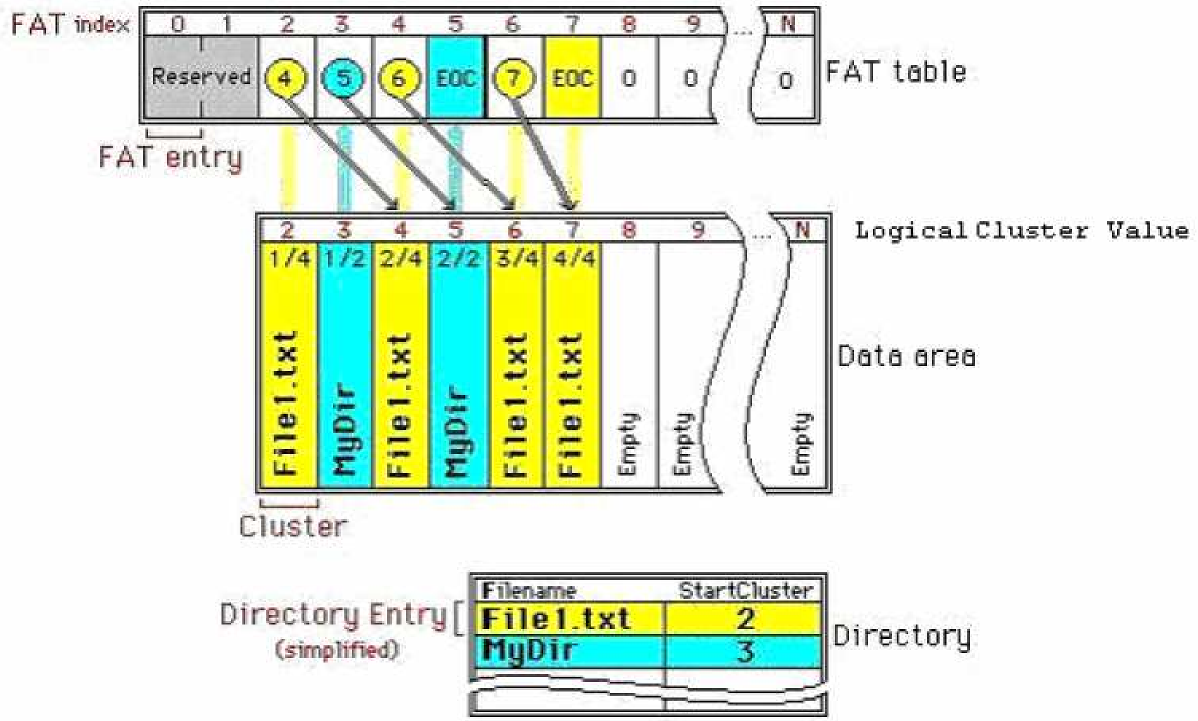
- Clock and Command Implement (output pin)
  - IORD_ALTERA_AVALON_PIO_DATA

- Data Implement: (inout pin)
  - IOWR_ALTERA_AVALON_PIO_DIRECTION
  - IORD_ALTERA_AVALON_PIO_DATA
  - IOWR_ALTERA_AVALON_PIO_DATA

FAT System


Sector 0
Master Boot Rector
Partition #1
512 Byte Master Boot Record
Partition #1 Entry
Partition #2 Entry
Partition #3 Entry
Partition #4 Entry
16 byte Partition Entry
Partition Type
Partition Start Sector
Partition Size
Partition #1
Volume Boot Rector
Partition #1
Bytes Per Sector
Sector Per Cluster
Reserved Sector Count
Number of FAT
FAT Size
Max Root Directory Entry
Fat Entry Sector
Root Directory Entry Sector
Data Entry Sector

FAT – File

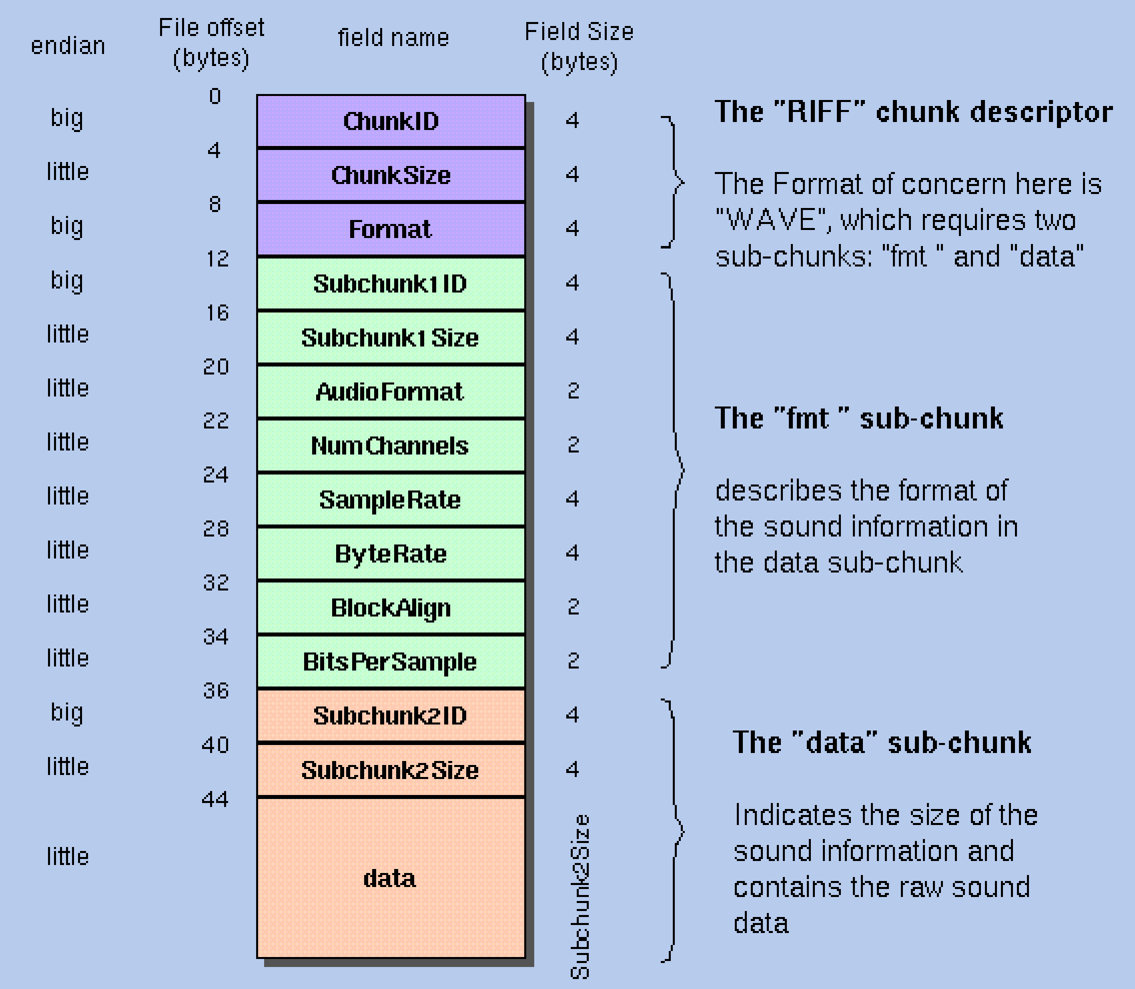


0	1	2	3	4	5	6	7	8	9	...	N
Reserved		4	5	6	EOC	7	EOC	0	0		0

2	3	4	5	6	7	8	9	...	N
1/4	1/2	2/4	2/2	3/4	4/4				
File1.txt	MyDir	File1.txt	MyDir	File1.txt	File1.txt	Empty	Empty		Empty

Filename	StartCluster
File1.txt	2
MyDir	3

.WAV – File Format



endian	File offset (bytes)	field name	Field Size (bytes)
big	0	ChunkID	4
little	4	ChunkSize	4
big	8	Format	4
big	12	Subchunk1ID	4
little	16	Subchunk1Size	4
little	20	AudioFormat	2
little	22	NumChannels	2
little	24	SampleRate	4
little	28	ByteRate	4
little	32	BlockAlign	2
little	34	BitsPerSample	2
big	36	Subchunk2ID	4
little	40	Subchunk2Size	4
little	44	data	Subchunk2Size

The "RIFF" chunk descriptor

The Format of concern here is "WAVE", which requires two sub-chunks: "fmt " and "data"

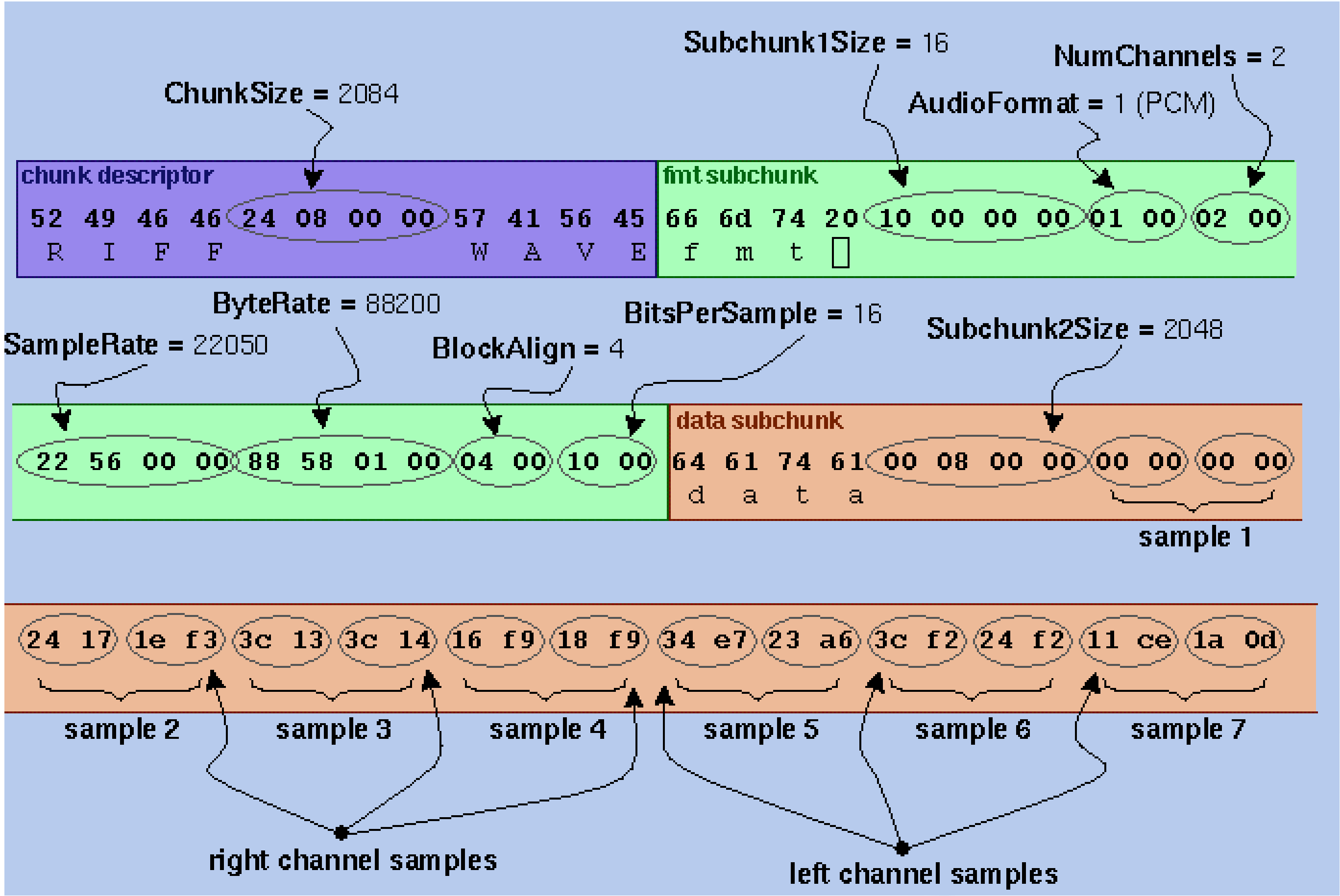
The "fmt " sub-chunk

describes the format of the sound information in the data sub-chunk

The "data" sub-chunk

Indicates the size of the sound information and contains the raw sound data

.WAV – Example



ChunkSize = 2084

Subchunk1Size = 16

AudioFormat = 1 (PCM)

NumChannels = 2

chunk descriptor

52 49 46 46 24 08 00 00 57 41 56 45

R I F F W A V E

fmt subchunk

66 6d 74 20 10 00 00 00 01 00 02 00

f m t

SampleRate = 22050

ByteRate = 88200

BlockAlign = 4

BitsPerSample = 16

Subchunk2Size = 2048

22 56 00 00 88 58 01 00 04 00 10 00

data subchunk

64 61 74 61 00 08 00 00 00 00 00 00

d a t a

sample 1

24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d

sample 2 sample 3 sample 4 sample 5 sample 6 sample 7

right channel samples

left channel samples

THANK YOU!