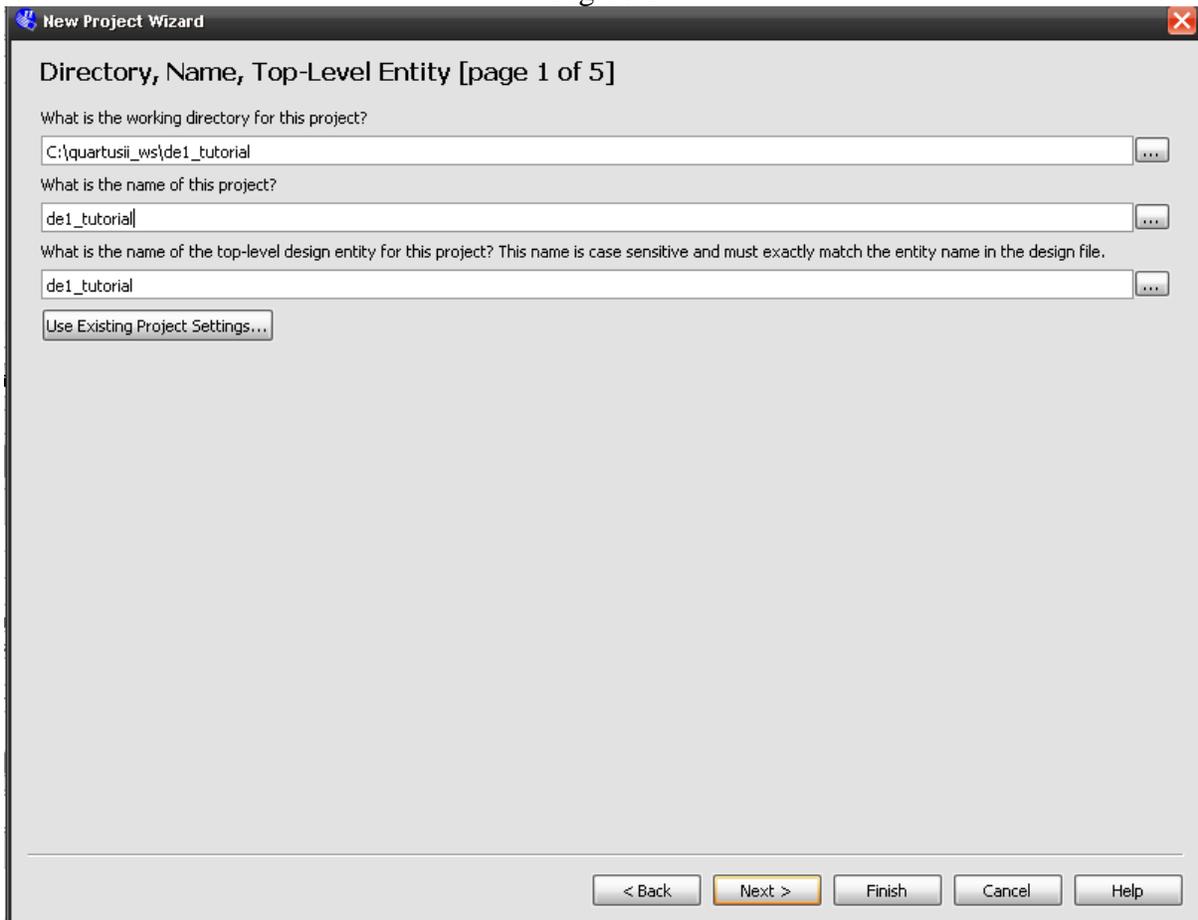


Altera DE1 Qsys Tutorial

1. Start the Quartus II software (Windows Platform v12.0 Service Pack 2).
2. Create a new project: From the Quartus II toolbar, select File > New > New Quartus II Project; click Next to skip the Introduction page. In the Directory, Name, Top-Level Entity page, create a working directory (without spaces) then append `\de1_tutorial` to the end (this creates a project directory). Enter `de1_tutorial` as the project name, which is also the name of the top level entity. Confirm that the page is the same as in Figure 1; click Next and Yes when asked to create the project directory.

Figure 1



3. Click Next to skip the Add Files page.

4. In the Family & Device Settings page under Target device, select the option Specific device selected in 'Available devices' list then choose *EP2C20F484C7* as the target device. Confirm that the page is the same as in Figure 2; click Finish to create the project.

Figure 2

New Project Wizard [page 3 of 5]

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

Device Family

Family: Cyclone II
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: Any
Pin count: Any
Speed grade: Any
Name filter:
 Show advanced devices HardCopy compatible only

Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit elements	PLL	I/O
EP2C20F484C7	1.2V	18752	315	239616	52	4	16
EP2C20F484C8	1.2V	18752	315	239616	52	4	16
EP2C20F484I8	1.2V	18752	315	239616	52	4	16
EP2C20Q240C8	1.2V	18752	142	239616	52	4	16
EP2C35F484C6	1.2V	33216	322	483840	70	4	16
EP2C35F484C7	1.2V	33216	322	483840	70	4	16
EP2C35F484C8	1.2V	33216	322	483840	70	4	16

Companion device

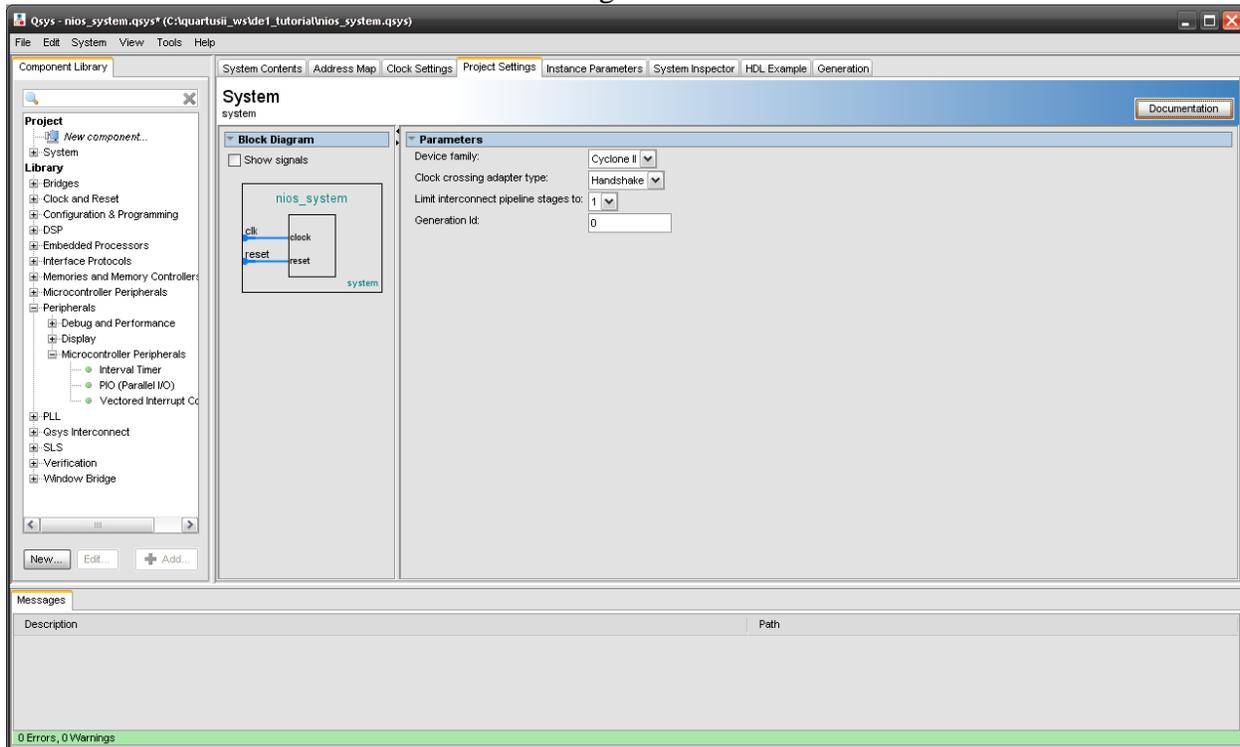
HardCopy:
 Limit DSP & RAM to HardCopy device resources

< Back **Next >** Finish Cancel Help

5. From the Quartus II toolbar, select Tools > Qsys. Then, from the Qsys toolbar, select File > Save As... enter *nios_system* as the file name (this will be the name of the system generated by Qsys).

6. Within Qsys, confirm that the Project Settings tab is the same as Figure 3.

Figure 3



Note: Steps 7-28 use the Qsys Components Library to build *nios_system* from individual modules.

7. Within Qsys, where it says Components Library, select Embedded Processors > Nios II Processor; click Add.

8. Within the Nios II MegaCore page, select the Nios II/e core; click Finish.

9. Within Qsys, where it says Components Library, select Memories and Memory Controllers > On-Chip > On-Chip Memory (RAM or ROM); click Add.

10. Within the On-Chip Memory MegaCore page; click Finish.

11. Within Qsys, where it says Components Library, select Interface Protocols > Serial > JTAG UART; click Add.

12. Within the JTAG UART MegaCore page; click Finish.

13. Within Qsys, where it says Components Library, select Peripherals > Microcontroller Peripherals > Interval Timer; click Add.

14. Within the Interval Timer MegaCore page; click Finish.

15. Within Qsys Builder, where it says Components Library, select Memories and Memory Controllers > External Memory Interfaces > SDRAM Interfaces > SDRAM Controller; click Add.

16. Within the SDRAM Controller MegaCore Parameter Settings page, change it to look like Figure 4; click Finish.

Figure 4

SDRAM Controller - sdram

SDRAM Controller

Parameter Settings

Memory Profile Timing

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

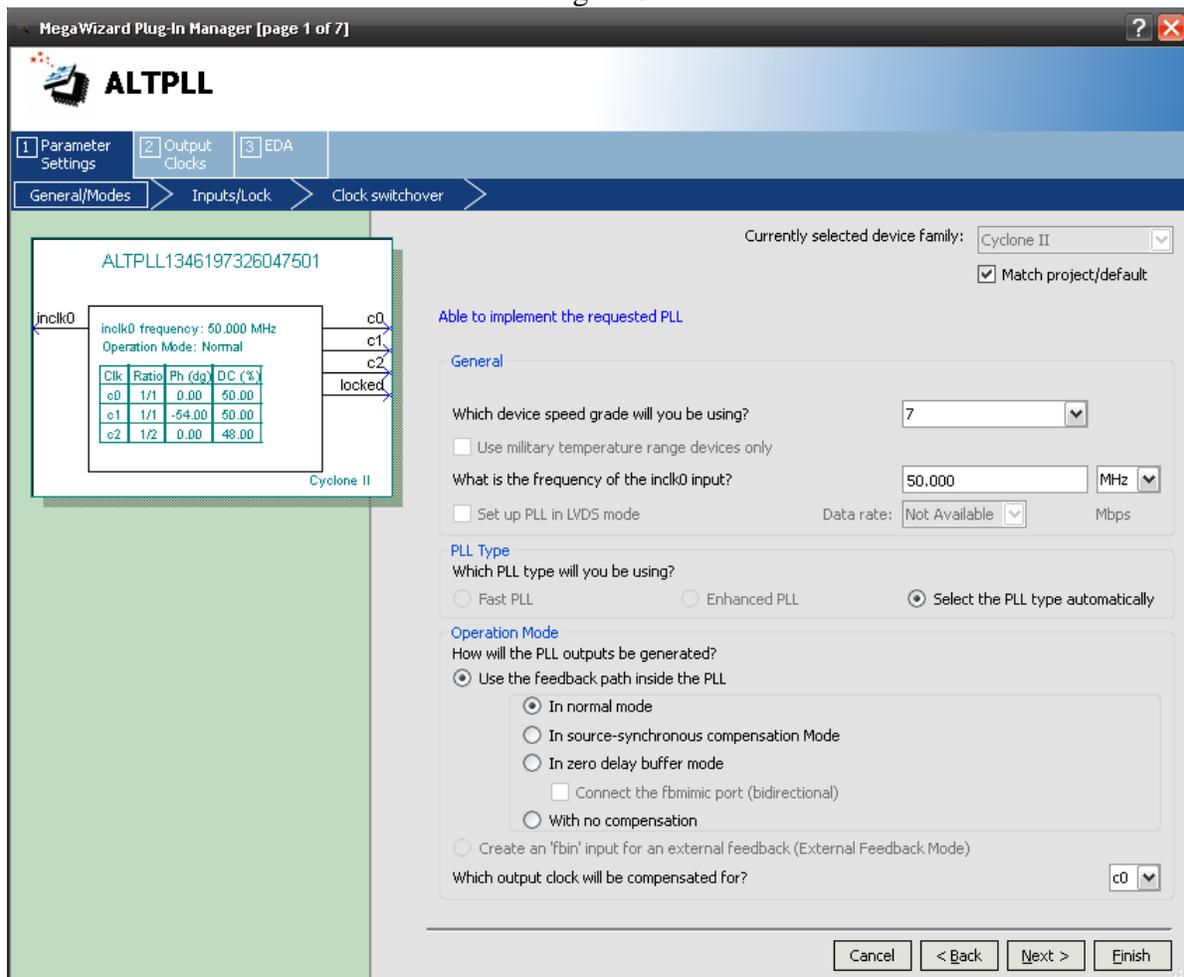
Cancel < Back Next > Finish

Note: This system will use the DE1's 50 MHz clock source (external clock source), however it will need an ALTPLL to create a -3ns shifted clock signal *c1* for the SDRAM controller/SDRAM chip (Accounts for clock skew on the DE1 board) and a 25 MHz clock source for the toggling red LEDs.

17. Within Qsys, where it says Components Library, select PLL > Avalon ALTPLL; click Add.

18. Within the MegaWizard Plug-In Manager change the Parameter Settings > General/Modes to the settings in Figure 5; click Next.

Figure 5



19. Within the MegaWizard Plug-In Manger change the Parameter Settings > Inputs/Lock tab to the settings in Figure 6; click Next.

Figure 6

MegaWizard Plug-In Manager [page 2 of 7]

ALTPLL

1 Parameter Settings 2 Output Clocks 3 EDA

General/Modes > Inputs/Lock > Clock switchover >

ALTPLL1346198695416563

inclk0

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00
c1	1/1	-54.00	50.00
c2	1/2	0.00	48.00

Cyclone II

c0
c1
c2
locked

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
- Hold 'locked' output low for 25000 cycles after the PLL initializes

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

Cancel < Back **Next >** Finish

20. Within the MegaWizard Plug-In Manager leave the Parameter Settings > Clock switchover tab unchanged; click Next.

21. Within the MegaWizard Plug-In Manager change the Output Clocks > clk c0 tab to the settings in Figure 7; click Next.

Figure 7

The screenshot displays the MegaWizard Plug-In Manager window for the ALTPLL component. The window title is "MegaWizard Plug-In Manager [page 4 of 7]". The main title bar shows "ALTPLL". The navigation tabs are "1 Parameter Settings", "2 Output Clocks", and "3 EDA". The "Output Clocks" tab is selected, and the "clk c0" sub-tab is active.

On the left, a diagram shows the ALTPLL1346199188909204 block. It has an input "inclk0" and three outputs: "c0", "c1", and "c2". A "locked" signal is also shown. Below the diagram is a table of clock settings:

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00
c1	1/1	-54.00	50.00
c2	1/2	0.00	48.00

The right side of the window shows the "c0 - Core/External Output Clock" configuration. It includes a checkbox "Use this clock" which is checked. Below it are "Requested Settings" and "Actual Settings" for various parameters:

- Enter output clock frequency: 50.00000000 MHz (Requested) / 50.000000 (Actual)
- Enter output clock parameters:
 - Clock multiplication factor: 1 (Requested) / 1 (Actual)
 - Clock division factor: 1 (Requested) / 1 (Actual)
- Clock phase shift: 0.00 (Requested) / 0.00 (Actual)
- Clock duty cycle (%): 50.00 (Requested) / 50.00 (Actual)

A "Description" table is also present:

Description	Val
Primary clock VCO frequency (MHz)	7...
Modulus for M counter	15

At the bottom, there are "Per Clock Feasibility Indicators" for c0, c1, and c2, all shown in green. Navigation buttons at the bottom include "Cancel", "< Back", "Next >", and "Finish".

22. Within the MegaWizard Plug-In Manager change the Output Clocks > clk c1 tab to the settings in Figure 8; click Next.

Figure 8

ALTPLL

1 Parameter Settings | 2 Output Clocks | 3 EDA

clk c0 > **clk c1** > clk c2

ALTPLL1346199188909204

inclk0

inclk0 frequency: 50.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00
c1	1/1	-54.00	50.00
c2	1/2	0.00	48.00

Cyclone II

c1 - Core/External Output Clock
Able to implement the requested PLL

Use this clock

Clock Tap Settings

Enter output clock frequency:
Requested Settings: 50.00000000 MHz | Actual Settings: 50.000000

Enter output clock parameters:
Clock multiplication factor: 1 | Actual Settings: 1
Clock division factor: 1 | Actual Settings: 1

Clock phase shift: -3.00 ns | Actual Settings: -3.00

Clock duty cycle (%): 50.00 | Actual Settings: 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)	7...
Modulus for M counter	15

Per Clock Feasibility Indicators: c0 c1 c2

Cancel < Back **Next >** Finish

23. Within the MegaWizard Plug-In Manager change the Output Clocks > clk c2 tab to the settings in Figure 9; click Next.

Figure 9

The screenshot shows the MegaWizard Plug-In Manager window for the ALTPLL component. The 'Output Clocks' tab is selected, and the 'clk c2' configuration is being viewed. On the left, a summary window displays the PLL ID 'ALTPLL1346265633354266' and a table of clock settings for Cyclone II. The main configuration area for 'c2 - Core/External Output Clock' is shown, with a 'Use this clock' checkbox checked. The 'Requested Settings' and 'Actual Settings' are displayed for various parameters.

Clk	Ratio	Ph. (dg)	DC (%)
c0	1/1	0.00	50.00
c1	1/1	-54.00	50.00
c2	1/2	0.00	50.00

c2 - Core/External Output Clock
Able to implement the requested PLL

Use this clock

Clock Tap Settings

Requested Settings	Actual Settings
Enter output clock frequency: 25.00000000 MHz	25.000000
Enter output clock parameters: Clock multiplication factor: 1	1
Enter output clock parameters: Clock division factor: 1	2
Clock phase shift: 0.00 deg	0.00
Clock duty cycle (%): 50.00	50.00

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

Per Clock Feasibility Indicators: c0 c1 c2

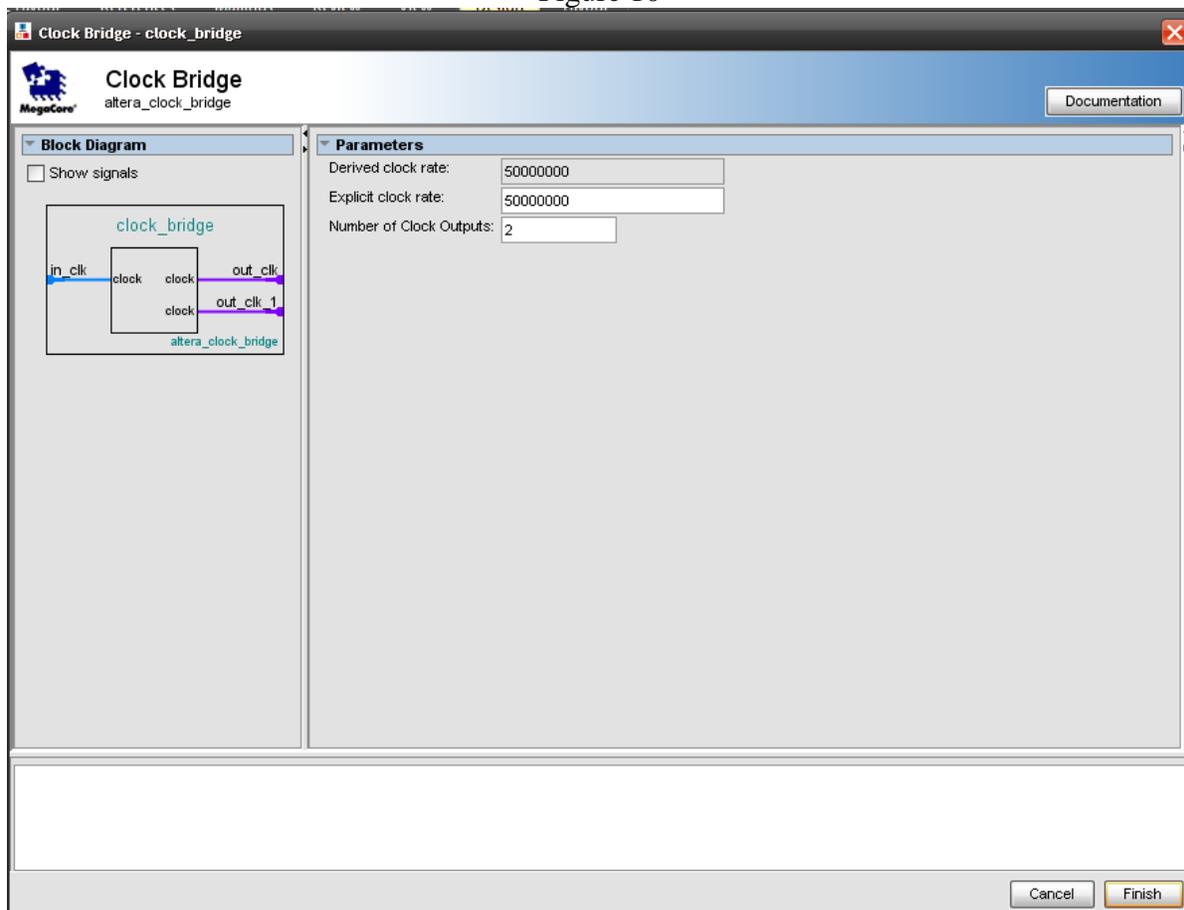
24. Within the MegaWizard Plug-In Manager leave the EDA tab unchanged; click Finish to add the PLL to the system.

Note: In a Qsys system a clock source can be exported or connected internally, but not both. Thus necessitating the use of a clock bridge, the clock bridge allows clock source *cl* to be exported to the SDRAM chip as well as internally connected to the SDRAM controller. The Qsys interconnect will be added in steps 34-36.

25. Within Qsys, where it says Components Library, select Bridges > Clock Bridge; click Add.

26. Within the MegaCore Clock Bridge page, change the settings to look like Figure 10; click Finish.

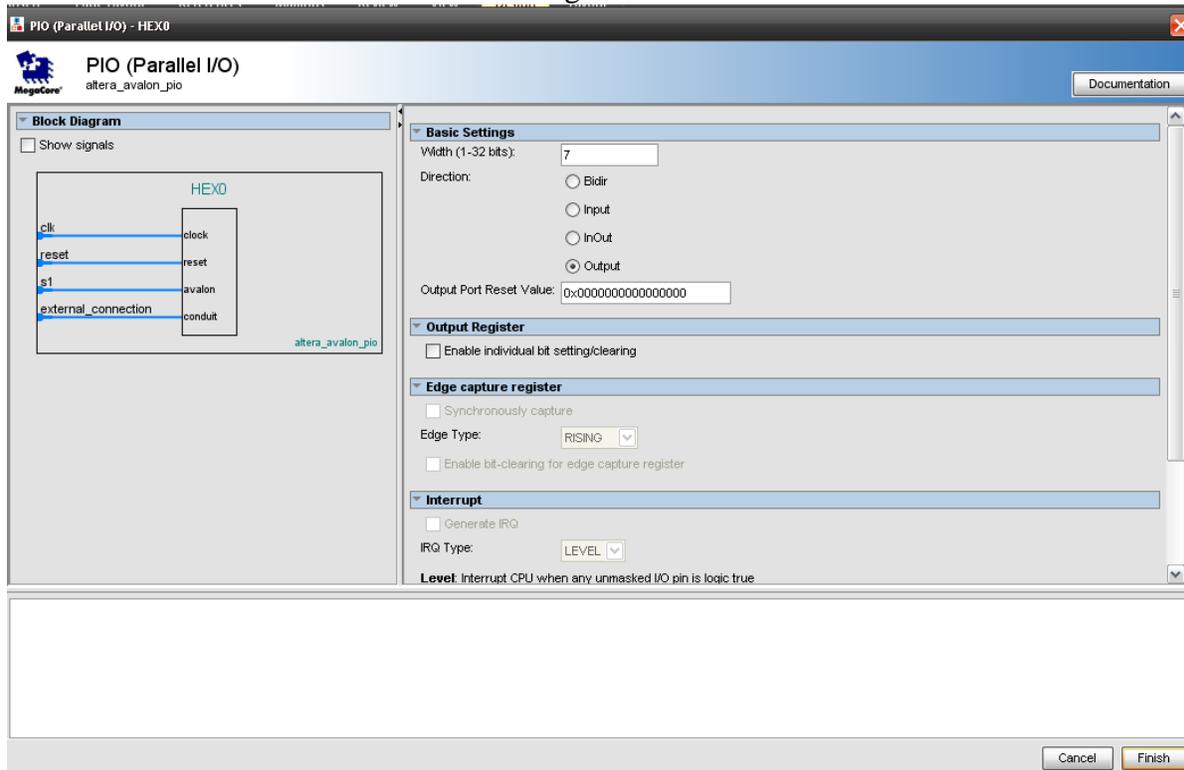
Figure 10



27. Within Qsys, where it says Components Library, select Peripherals > Microcontroller Peripherals > PIO (Parallel I/O); click Add.

28. Within the MegaCore PIO (Parallel I/O) page, change the settings to look like Figure 11; click Finish.

Figure 11



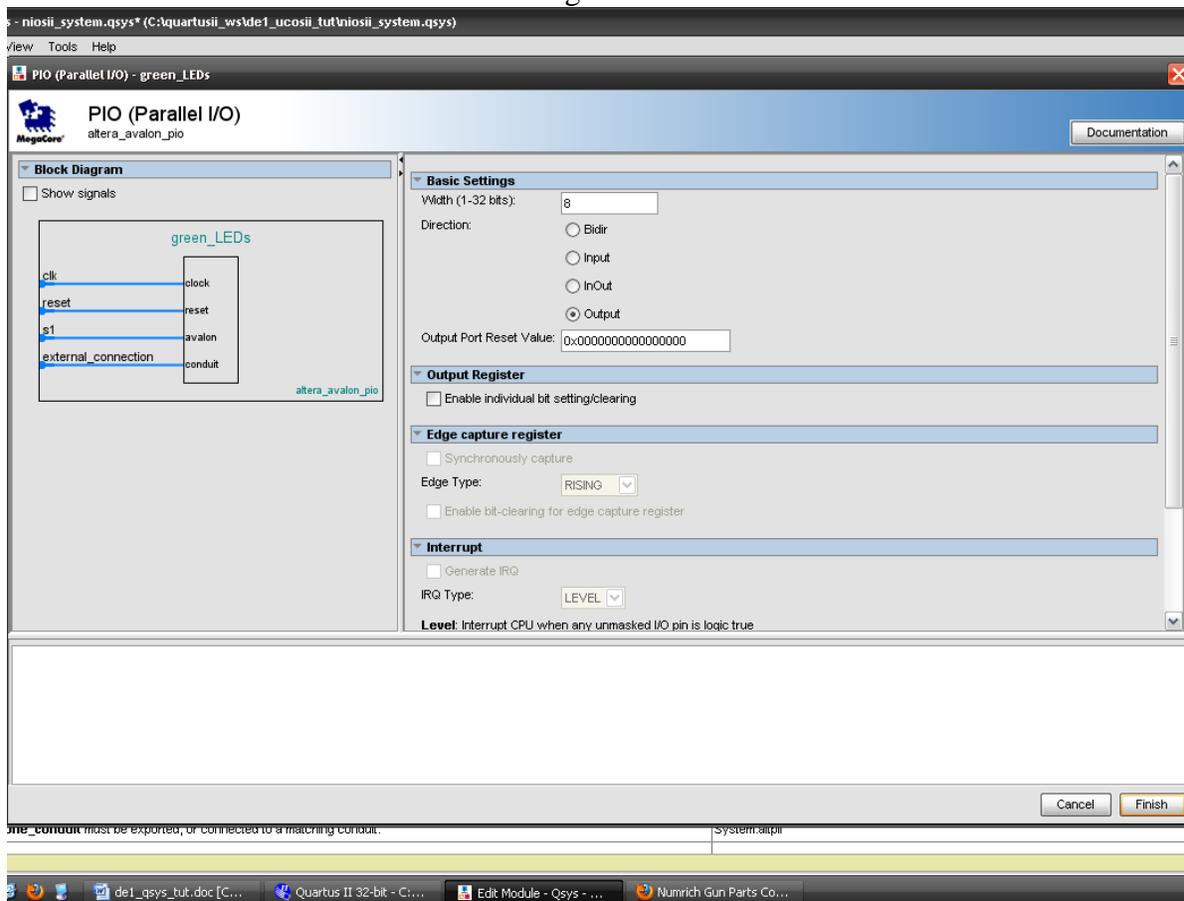
29. Within Qsys, select the newly added component and from the toolbar, select Edit > Rename; type *HEX0* and press enter.

30. Repeat steps 27-29 three more times changing the names to *HEX1*, *HEX2*, and *HEX3*.

31. Within Qsys, where it says Components Library, select Peripherals > Microcontroller Peripherals > PIO (Parallel I/O); click Add.

32. Within the MegaCore PIO (Parallel I/O) page, change the settings to look like Figure 12; click Finish.

Figure 12



33. Within Qsys, select the newly added component and, from the toolbar, select Edit > Rename; type *green_LEDs* and press enter.

34. Export the necessary signals: To export a module's signal, find its row and in the Export column click where it says Click to export. Once the signal name is typed, press enter to add the export; pressing **esc** will cancel the export. The following exports will need to be created:

- green_LEDs > external_connection > (Click and press **enter**)
- HEX0 > external_connection > (Click and press **enter**)
- HEX1 > external_connection > (Click and press **enter**)
- HEX2 > external_connection > (Click and press **enter**)
- HEX3 > external_connection > (Click and press **enter**)
- sdram_0 > wire > (Click and press **enter**)
- altpll_0 > inclk_interface > (Click and press **enter**)
- altpll_0 > c2 > (Click and press **enter**)
- clock_bridge_0 > out_clk_1 > (Click and press **enter**)

35. Create clock connections using the ALTPLL and the Clock Bridge: First, delete the auto generated clock source clk_0 by selecting it and, from the Qsys toolbar, selecting Edit > Remove. Make the following clock connections by clicking on the corresponding empty circles within the Connections column (dark circles indicate connections):

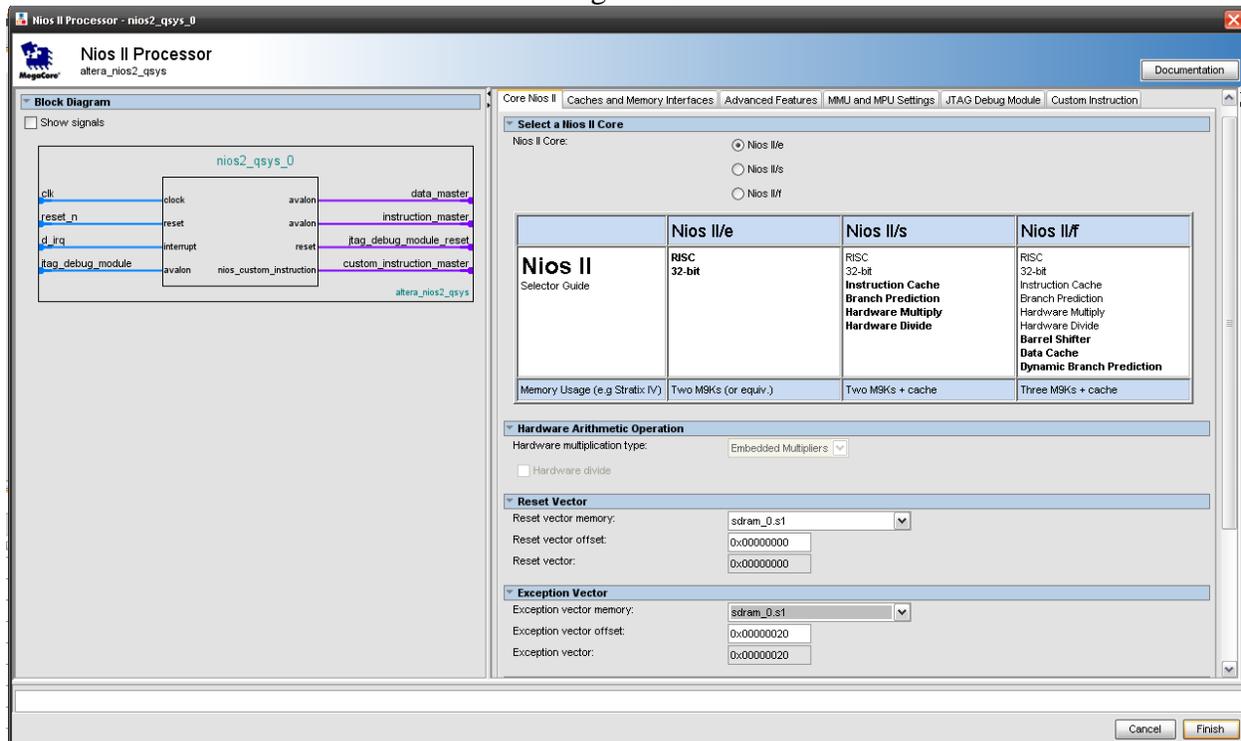
- altpll_0.c0 > nios2_qsys.clk
- altpll_0.c0 > onchip_memory2_0.clk1
- altpll_0.c0 > jtag_uart_0.clk
- altpll_0.c0 > timer_0.clk
- altpll_0.c0 > HEX0.clk
- altpll_0.c0 > HEX1.clk
- altpll_0.c0 > HEX2.clk
- altpll_0.c0 > HEX3.clk
- altpll_0.c0 > green_LEDs.clk
- altpll_0.c1 > clock_bridge_0.in_clk
- clock_bridge_0.out_clk > sdram_0.clk

36. Create master/slave connections using the Nios II Processor: Make the following master/slave connections by clicking on the corresponding empty circles within the Connections column (dark circles indicate connections):

- nios2_qsys_0.data_master > onchip_memory2_0.s1
- nios2_qsys_0.instruction_master > onchip_memory2_0.s1
- nios2_qsys_0.data_master > jtag_uart_0.avalon_jtag_slave
- nios2_qsys_0.data_master > timer_0 > s1
- nios2_qsys_0.data_master > sdram_0.s1
- nios2_qsys_0.data_master > sdram_0.s1
- nios2_qsys_0.data_master > altpll_0.pll_slave
- nios2_qsys_0.data_master > HEX0.s1
- nios2_qsys_0.data_master > HEX1.s1
- nios2_qsys_0.data_master > HEX2.s1
- nios2_qsys_0.data_master > HEX3.s1
- nios2_qsys_0.data_master > green_LEDs.s1

37. Assign reset and exception vectors: Within Qsys, select the Nios II Processor and, from the toolbar, select Edit > Edit... Within the MegaCore page, select *sdram_0.s1* from the Reset vector memory drop-down menu as well as the Exception Vector drop-down menu. The page should look like Figure 13; click Finish.

Figure 13



38. Assign Base Addresses: From the Qsys toolbar, select System > Assign Base Addresses.

39. Create Global Reset Network: From the Qsys toolbar, select System > Create Global Reset Network.

40. Assign interrupt numbers and connect the IRQs for the JTAG UART and the Interval Timer: Within Qsys, find the *jtag_uart_0* row and connect its IRQ in the IRQ column, then select the IRQ number and type 16. Within Qsys, find the *timer_0* row and connect its IRQ in the IRQ column, then select the IRQ number and type 0.

41. Within Qsys, in the Generation tab, click Generate.

42. Within Qsys, in the Address Map tab, copy the base addresses of HEX0, HEX1, HEX2, HEX3, and *green_LEDs* for later use in the application software.

43. Within Qsys, in the HDL Example tab, copy the instantiation of *nios_system* for later use in the Quartus II, Verilog top-level entity.

44. Within Quartus II, select File > New > Verilog HDL File.

45. Within Quartus II, select File > Save As... *de1_tutorial.v*.
46. Within Quartus II, paste the code from Figure 14 into *de1_tutorial.v*.
47. Within Quartus II, select File > New > Verilog HDL File.
48. Within Quartus II, select File > Save As... *toggle.v*.
49. Within Quartus II, paste the code from Figure 15 into *toggle.v*.
50. Within Quartus II, select Assignments > Import Assignments...
DE1_pin_assignments.csv.
51. Within Quartus II, select Processing > Start Compilation.
52. Within Quartus II, select Tools > Programmer; within Programmer click Start.
53. Exit Quartus II.

Figure 14

```

module de1_tutorial(CLOCK_50, LEDG, LEDR, DRAM_CLK, DRAM_CKE,
DRAM_ADDR, DRAM_BA_1, DRAM_BA_0, DRAM_CS_N, DRAM_CAS_N,
DRAM_RAS_N,
DRAM_WE_N, DRAM_DQ, DRAM_UDQM, DRAM_LDQM, HEX3, HEX2, HEX1, HEX0);

input CLOCK_50;
output [7:0] LEDG;
output [9:5] LEDR;
output [6:0] HEX3;
output [6:0] HEX2;
output [6:0] HEX1;
output [6:0] HEX0;
output [11:0] DRAM_ADDR;
output DRAM_BA_1, DRAM_BA_0, DRAM_CAS_N, DRAM_RAS_N, DRAM_CLK;
output DRAM_CKE, DRAM_CS_N, DRAM_WE_N, DRAM_UDQM, DRAM_LDQM;
inout [15:0] DRAM_DQ;

wire toggle_clock;

nios_system u0 (
.pio_0_external_connection_export (LEDG), // pio_0_external_connection.export
.hex3_external_connection_export (HEX3), // hex3_external_connection.export
.hex2_external_connection_export (HEX2), // hex2_external_connection.export
.hex1_external_connection_export (HEX1), // hex1_external_connection.export
.hex0_external_connection_export (HEX0), // hex0_external_connection.export
.sdram_0_wire_addr      (DRAM_ADDR),          //          sdram_0_wire.addr
.sdram_0_wire_ba        ({DRAM_BA_1, DRAM_BA_0}), //          //
.ba
.sdram_0_wire_cas_n     (DRAM_CAS_N),          //          .cas_n
.sdram_0_wire_cke       (DRAM_CKE),           //          .cke
.sdram_0_wire_cs_n     (DRAM_CS_N),           //          .cs_n
.sdram_0_wire_dq        (DRAM_DQ),            //          .dq
.sdram_0_wire_dqm       ({DRAM_UDQM, DRAM_LDQM}), //          //
.dqm
.sdram_0_wire_ras_n     (DRAM_RAS_N),          //          .ras_n
.sdram_0_wire_we_n     (DRAM_WE_N),           //          .we_n
.altpll_0_inclk_interface_clk (CLOCK_50), // altpll_0_inclk_interface.clk
.altpll_0_c2_clk        (toggle_clock),       //          altpll_0_c2.clk
.clock_bridge_0_out_clk_1_clk (DRAM_CLK) // clock_bridge_0_out_clk_1.clk
);

toggle t(toggle_clock, LEDR);

endmodule

```

Figure 15

```
module toggle(clock, counter_out);  
  
input  clock;  
output [9:5] counter_out;  
  
reg [31:0] counter_data;  
  
always @ (posedge clock)  
begin  
    counter_data <= counter_data + 1;  
end  
  
assign counter_out[9] = counter_data[21];  
assign counter_out[5] = counter_data[21];  
  
assign counter_out[8] = counter_data[26];  
assign counter_out[6] = counter_data[26];  
  
assign counter_out[7] = counter_data[27];  
  
endmodule
```