M4 SystemVerilog and FPGAs

1. Toy lift and controller state machine

The object of this exercise is to learn how a SystemVerilog description of a state machine can be synthesized to a Field Programmable Gate Array. The state machine is a controller for a toy lift (see Figure 1) and the FPGA is Altera Cyclone III on the Altera DE0 Development and Education Board (Figure 2).

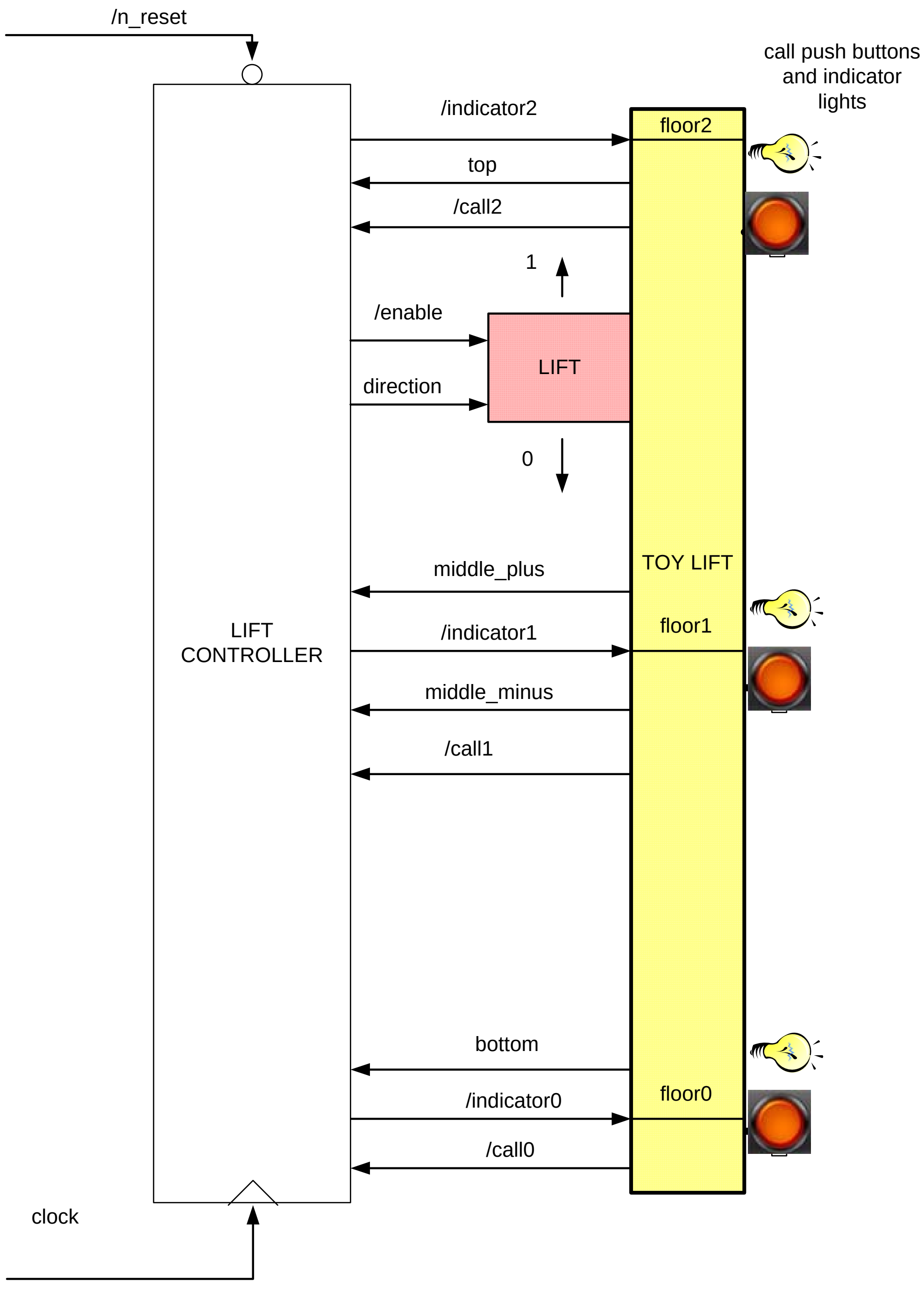


Figure 1. Toy lift and its control signals.

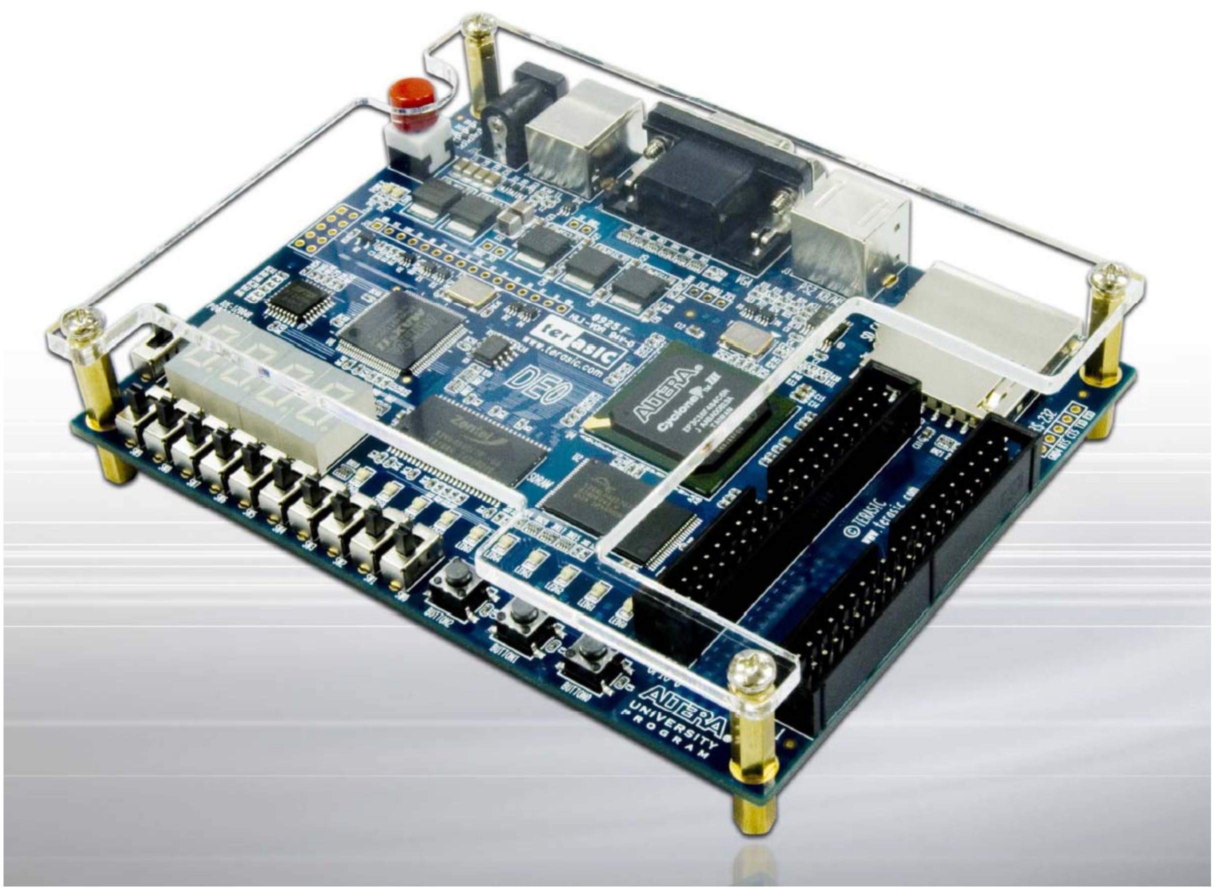


Figure 2. Altera DE0 Development and Education Board with Cyclone III EP3C16 FPGA .

The controller should be implemented on the FPGA as explained in Section 2 below as a state machine described in SystemVerilog. Sample SystemVerilog code the controller and a testbench is provided but you are encouraged to write your own controller. Don't forget that the FPGA is clocked at 50MHz! The design must be a synchronous state machine with an asynchronous reset. The operation of the controller and the functions of the control signals are explained in the sample SystemVerilog code in Figure 3. The controller has the following sensor inputs which are activated by the lift position: **top** – asserted when the lift reaches floor 2, **middle_plus** – asserted when the lift reaches floor 1 from above, **middle_minus** – asserted when the lift reaches floor 1 from below and **bottom** asserted when the lift reaches floor 0. The active low call inputs **call0**, **call1** and **call2** are asserted when the user presses a corresponding call button and deasserted when the lift reaches the corresponding floor. The controller uses two outputs to control the lift motor: active low **enable** switches the motor on, and **direction** (0- down, 1 – up) specifies in which direction the lift should move. The active low signals **indicator0**, **indicator1** and **indicator2** are asserted by the controller when a corresponding call signal is received and deasserted when the call is handled, i.e. when the lift reaches the desired floor. Additionally, the controller has 10 **led** outputs which are used to display information about its status and operation on the LED array of the ALTERA DE0 FPGA development board. The lift controller code in Figure 3. below can be downloaded from the ELEC2014 notes site (file lift_controller.sv). In addition, sample testbench files are provided on the notes web site: lift_testbench.sv and lift_testmodule.sv. Those two files define the clock, n_reset and sample lift operation waveforms for Modelsim simulations and should not be synthesised.

RTL Simulation

Start the Modelsim simulator, create a project consisting of the files lift_testbench.sv, lift_testmodule.sv and lift_controller.sv and simulate to familiarise yourself with the controller's operation. Note the time scale disparity between the very fast clock (50MHz) and the simulated input waveforms which operate in microseconds. To get meaningful results you need to simulate over an interval of several hundred microseconds.

```
module  lift_controller  (input  logic  clock,  n_reset,  top,  middle_plus,
middle_minus, bottom, call0, call1, call2,
  output logic direction, enable, indicator0, indicator1, indicator2,
  output logic led[9:0] // LEDs display info about controller's behaviour
 );

  // SystemVerilog enumerated type for state machine states
  enum {to0, floor0, floor1, floor2, downto1, upto1, to2} state;

always_ff @( posedge clock,
            negedge n_reset)// always_ff means synthesisee to flip-flops
  begin: seq  // note that this block also contains next state comb logic
    if (~n_reset)
      state <= to0; //use non-blocking assignments for registered signals
    else
      case (state)
       floor0:
        if (!call1)
        state <= upto1;
        else if (!call2)
        state <= to2;

      floor1:
        if (!call0)
          state <= to0;
        else if (!call2)
          state <= to2;

      floor2:
          if (!call0)
           state <= to0;
        else if (!call1)
           state <= downto1;

      upto1:
        if (!middle_minus && !middle_plus)
          state <= floor1;

      downto1:
        if (!middle_minus && !middle_plus)
          state <= floor1;

      to0:
        if (!bottom)
          state <= floor0;

      to2:
        if (!top)
          state <= floor2;

    endcase
  end
```

```
always_comb    //  always_comb  -  combinational  logic  for  output  signals;
begin: com
    // use blocking assignments for combinatorial signals
    direction = '0; // 0 is down
    indicator0 = '1; // active low
    indicator1 = '1; // active low
    indicator2 = '1; // active low
    enable = '1;

      // leds display information about the controller's status
      led[0] = (state == floor0);
      led[1] = (state == floor1);
      led[2] = (state == floor2);
      led[3] = (state == to0);  // response to call0
      led[4] = (state == downto1);  // response to call1 if on floor2
      led[5] = (state == upto1);  // response to call1 if on floor0
      led[6] = (state == to2);  // response to call2
      led[7] =  top | middle_plus; // LEDs 7 and 8 show where the lift is:
      led[8] =  top | middle_minus; // 00-bottom, 01-mid-, 10-mid+, 11-top
      led[9] = ~n_reset; // this led lights up if reset is active

    case (state) // this case statement handles motor control & indicators
       upto1:
        if (middle_minus || middle_plus)
          begin
            enable = '0;
              direction = '1;
              indicator1 = '0;
          end
      downto1:
        if (middle_minus || middle_plus)
          begin
            enable = '0;
              direction = '0;
              indicator1 = '0;
          end
      to0:
        if (bottom)
          begin
            enable = '0;
              direction = '0;
              indicator0 = '0;
          end
      to2:
        if (top)
          begin
            enable = '0;
              direction = '1;
              indicator2 = '0;
          end
      default: ;

    endcase
  end
```

Figure 3. SystemVerilog description of lift controller state machine.

2. Synthesis and FPGA programming walk-through with Altera Quartus II

This laboratory uses two commercial design tools: Modelsim for simulation and Altera Quartus II for synthesis and FPGA programming. If any of the files used in your work need to be modified, you also need to use a text editor, such as Scite, or any other editor available on the lab machines. Save your project files in a directory you have a write access to, e.g. on your home drive.

RTL Synthesis

Start Altera Quartus II available from the CAD menu. The general operation of this tool is similar to that of Synplify Pro which you have used in previous laboratory experiments. Altera Quartus II is a dedicated application specifically for FPGAs and Programmable Logic Devices manufactured by Altera. It is a comprehensive and user-friendly tool which performs RTL synthesis from high-level descriptions in a number of Hardware Description Languages, including SystemVerilog, and which connects directly to the development board via a USB link for programming. When you start Quartus, the startup screen shown in Fig. 4 appears.

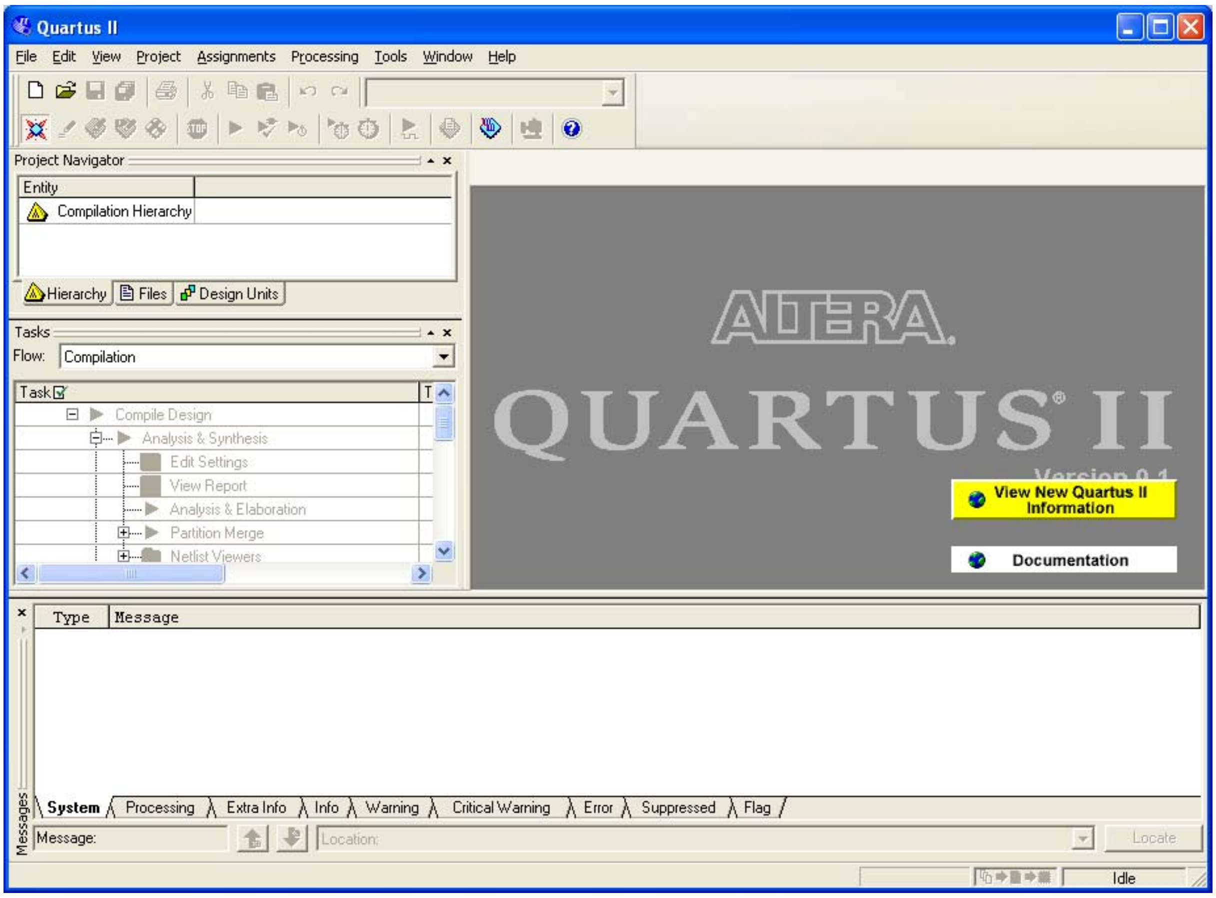
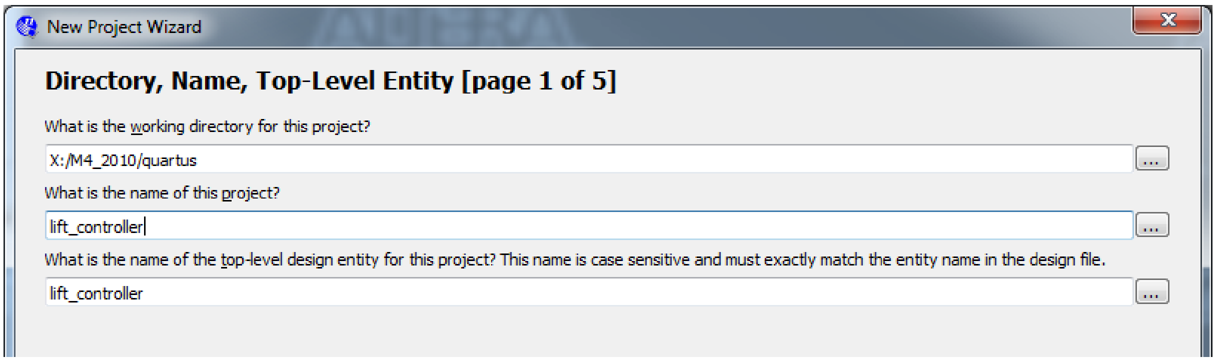


Figure 4. Altera Quartus II startup screen.

Click on File->New Project Wizard. Click Next. Select a working directory for the project (make sure you have a write permission for your working directory), type a project name and specify the name of the top-level design module, ie. **lift_controller** (Figure 5).



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

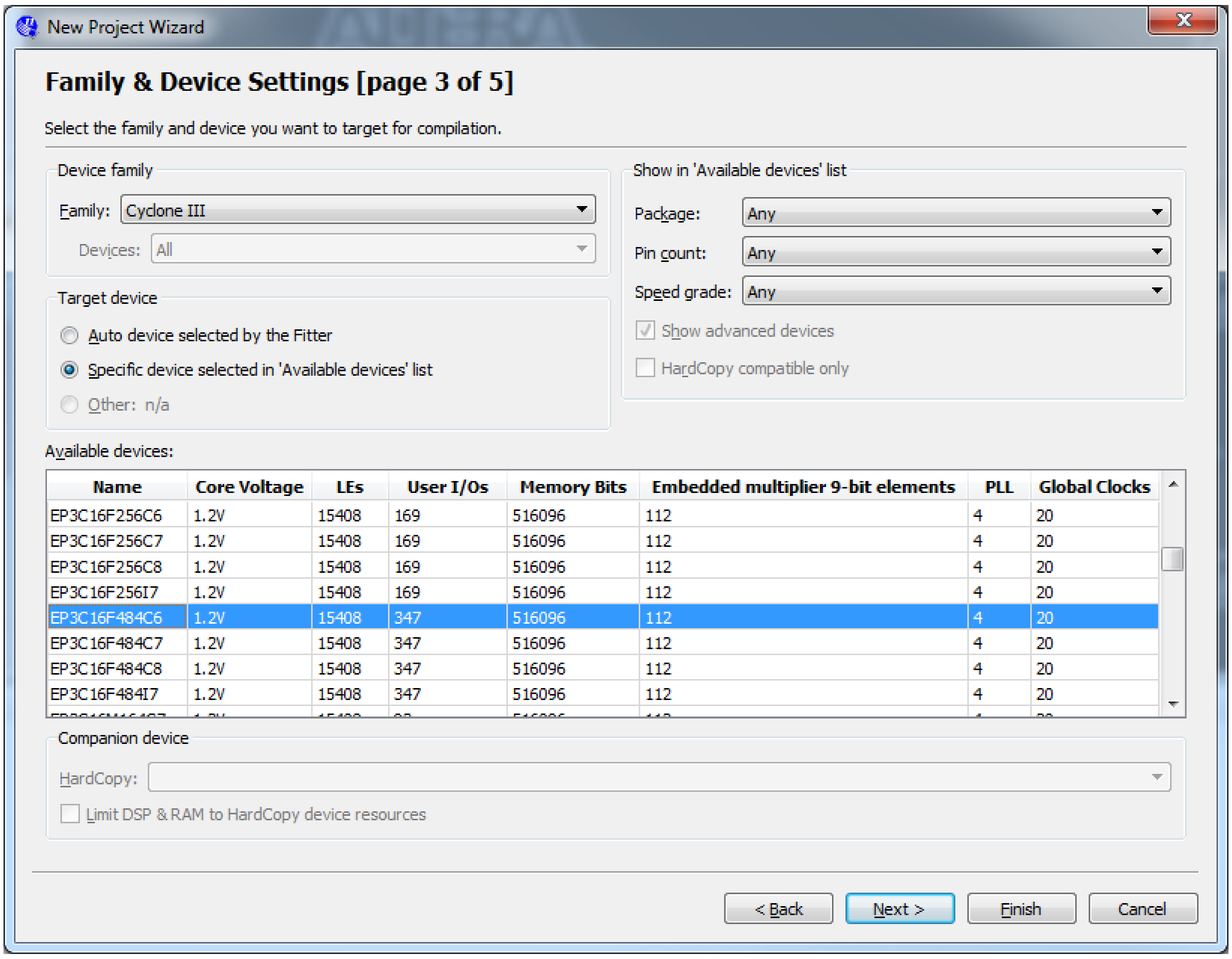
What is the working directory for this project?
X:/M4_2010/quartus

What is the name of this project?
lift_controller

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.
lift_controller

Figure 5. Quartus New Project Wizard.

Click Next, navigate to the directory with your lift controller SystemVerilog module (lift_controller.sv) and add it to the project. Click Next to bring up the Device Settings Window (Figure 6).



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: Any
Pin count: Any
Speed grade: Any
[✓] 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	Global Clocks
EP3C16F256C6	1.2V	15408	169	516096	112	4	20
EP3C16F256C7	1.2V	15408	169	516096	112	4	20
EP3C16F256C8	1.2V	15408	169	516096	112	4	20
EP3C16F256I7	1.2V	15408	169	516096	112	4	20
EP3C16F484C6	1.2V	15408	347	516096	112	4	20
EP3C16F484C7	1.2V	15408	347	516096	112	4	20
EP3C16F484C8	1.2V	15408	347	516096	112	4	20
EP3C16F484I7	1.2V	15408	347	516096	112	4	20

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

Figure 6. Device Settings window.

Select Cyclone III as the device family ant the device EP3C16F484C6 Click Next->Next->Finish. The main Quartus window now shows the project details in the Project Navigator. Now click the Analysis & Synthesis button (Figure 7).

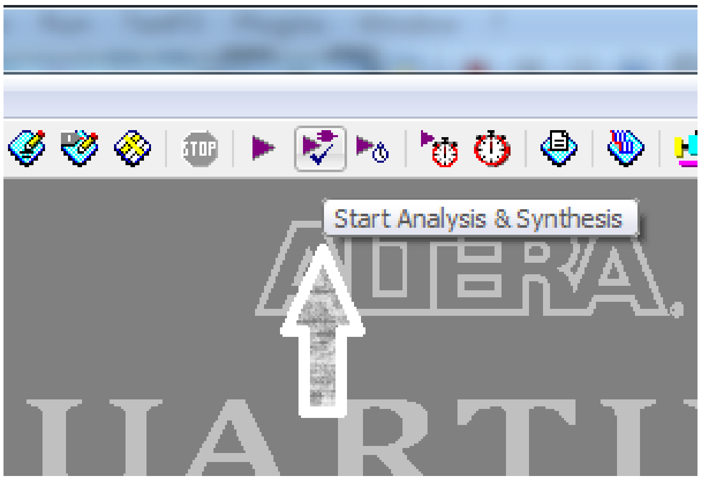


Figure 7. Start Analysis and Synthesis button.

This is a fast run as it does not involve full placement and routing for the FPGA and is intended to reveal any syntax errors or other problems with the SystemVerilog code. If successful, the analysis and synthesis process creates a number of files, including the pin allocation file lift_controller.qsf. You will need to edit this file to provide the pin assignment stipulated by the IDC 40-pin ribbon cable which connects the GPIO 1 (J5) connector on the Altera DE0 board with the toy lift. The required pin definitions are provided in Figure 8 and also in the file lift_pin_asssignment.qsf downloadable from the ELEC2014 notes web site.

```
set_location_assignment PIN_G21 -to clock
set_location_assignment PIN_R16 -to bottom
set_location_assignment PIN_Y17 -to call0
set_location_assignment PIN_U15 -to call1
set_location_assignment PIN_W15 -to call2
set_location_assignment PIN_R14 -to direction
set_location_assignment PIN_T12 -to enable
set_location_assignment PIN_T9 -to indicator0
set_location_assignment PIN_T10 -to indicator1
set_location_assignment PIN_R12 -to indicator2
set_location_assignment PIN_T16 -to middle_minus
set_location_assignment PIN_AA7 -to middle_plus
set_location_assignment PIN_T14 -to top
set_location_assignment PIN_J6 -to n_reset
set_location_assignment PIN_B1 -to led[9]
set_location_assignment PIN_B2 -to led[8]
set_location_assignment PIN_C2 -to led[7]
set_location_assignment PIN_C1 -to led[6]
set_location_assignment PIN_E1 -to led[5]
set_location_assignment PIN_F2 -to led[4]
set_location_assignment PIN_H1 -to led[3]
set_location_assignment PIN_J3 -to led[2]
set_location_assignment PIN_J2 -to led[1]
set_location_assignment PIN_J1 -to led[0]
```

Figure 8. File lift_pin_assignment.qsf.

You can either copy the text in Figure 8 and paste it directly into the file lift_controller.qsf which has just been created by Quartus, or download the file lift_pin_assignment.qsf from the notes web site, save it in your project directory and type the following line at the end of the Quartus lift_controller.qsf file:

```
source lift_pin_assignment.qsf
```

Now run the full synthesis as well as placement and routing by clicking the Start Compilation button in Quartus (Figure 9):

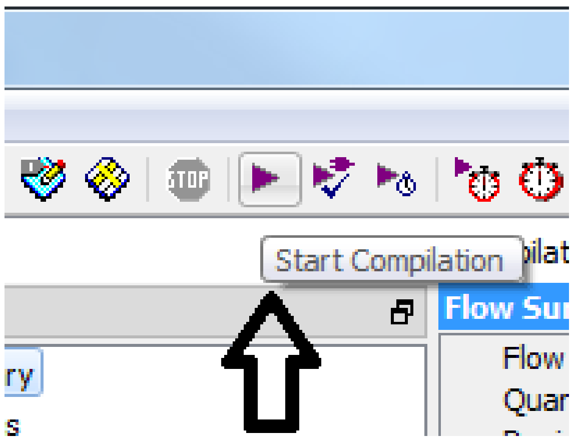
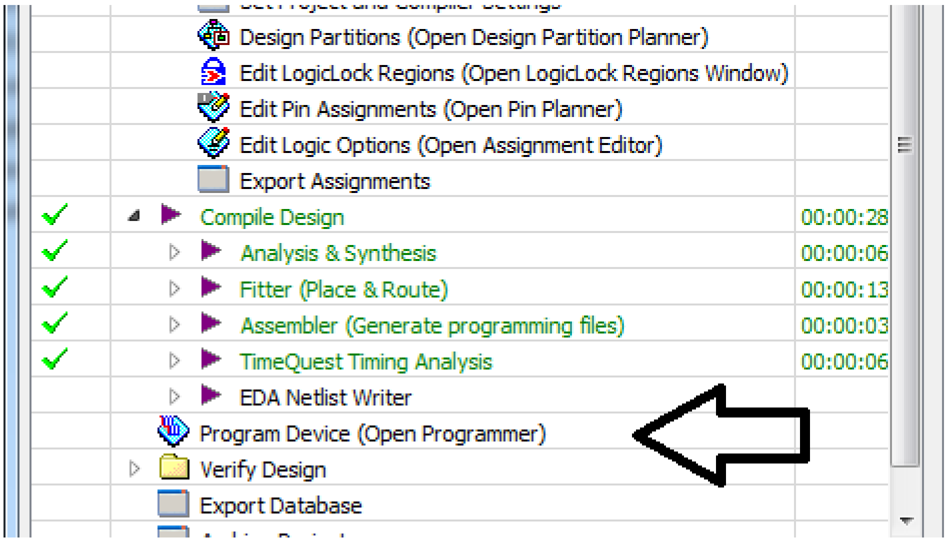


Figure 9. Start Compilation button.

FPGA programming

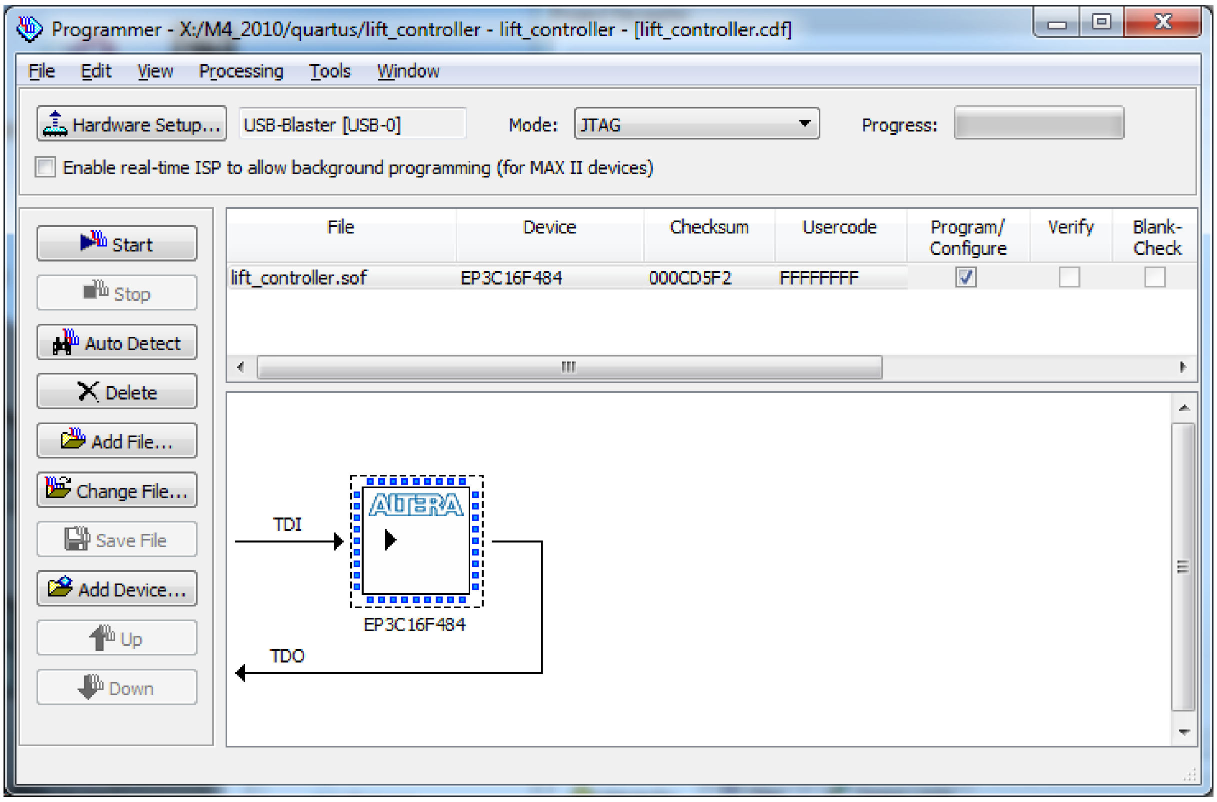
On successful compilation, plug in the Altera DE0 board to a USB socket on your machine and click the Program Device (Open Programmer) task in the Compilation pane (Figure 10).



Design Partitions (Open Design Partition Planner)
Edit LogicLock Regions (Open LogicLock Regions Window)
Edit Pin Assignments (Open Pin Planner)
Edit Logic Options (Open Assignment Editor)
Export Assignments
✓ Compile Design 00:00:28
✓ Analysis & Synthesis 00:00:06
✓ Fitter (Place & Route) 00:00:13
✓ Assembler (Generate programming files) 00:00:03
✓ TimeQuest Timing Analysis 00:00:06
EDA Netlist Writer
Program Device (Open Programmer)
Verify Design
Export Database

Figure 10. Program Device task.

The programmer window pops up as shown in Figure 11. Make sure that the FPGA placement and route file lift_controller.sof exists and click Start to program the FPGA.


Programmer - X:/M4_2010/quartus/lift_controller - lift_controller - [lift_controller.cdf]

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

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

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check
lift_controller.sof	EP3C16F484	000CD5F2	FFFFFFFF	✓		

Figure 11. Quartus programmer.

Now connect the Altera DE0 board with the toy lift using the ribbon cable provided. Plug in the ribbon cable to the GPIO1 IDC 40-pin connector on the DE0. There are two IDC connectors on the Altera DE0 board, GPIO0 and GPIO1. The pin assignment provided in Figure 8 in the lift_pin_assignment.qsf file is valid only for the GPIO1 connector so make sure that you plug in the ribbon cable to that connector.

The pin assignments also connect the active low asynchronous reset signal **n_reset** to the switch SW0 and the controller's led outputs to the LED array on the Altera Board. Note that when the switch SW0 is in the OFF position, the controller remains in the reset state. To start the controller, move the switch SW0 to the ON position.

Play with the toy lift by pressing call buttons and observing the motion of the lift. Also observe the LEDs on the Altera board to make sure that the toy lift behaviour matches the operation of the controller.

Further work

Cyclone III is a very powerful FPGA with more than 15,000 logic blocks. You can therefore extend the original lift controller design to include a more complex behaviour, such as queueing and handling multiple calls. Consider such modifications if you have time. You can also easily alter the logic expressions that drive the LEDs in the lift_controller.sv module to observe different signals on the LEDs if you wish.

tjk
2/12/2010