THE UNIVERSITY OF ALABAMA IN HUNTSVILLE
Department of Electrical and Computer Engineering

CPE 322 Digital Hardware Design Fundamentals

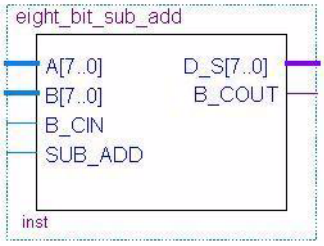
Simulation Assignment #1

RTL & Post FPGA Layout Timing Simulation of two an 8-bit Subtractor/Adder

The purpose of this simulation assignment is to provide an opportunity for students to gain experience using the Quartus II/ModelSim® Computer Aided Design tool suite to perform both RTL (functional) and post-layout timing type simulations on multiple representations of a small digital design.

Subtractor/Adder Module

In this simulation assignment, an eight-bit two's complement subtractor/adder module is modeled in Verilog HDL in two ways which differ from one another in the level of abstraction that is employed. The first design entry method for the subtractor/adder is that of a structural model which is a functional representation that is analogous to the hierarchical schematic capture methodology that was employed in the first part of Laboratory 3 in CPE 324 class. The second design entry method is a behavioral design. The level of abstraction for this type of model is much higher than that of the corresponding structural representation in that the base algorithmic representation of the design is specified but the specific logic configuration needed to implement the algorithm is not. This means that the simulator/synthesis tool has great flexibility in how the actual low-level logic is implemented. Figure 1 illustrates the interface and the base functionality of the subtractor/adder for both type of models.



Function:

```
if (SUB_ADD) = '1' then
  D_S <= A - B - B_Cin --subtract B and B_Cin from A
  if (B>A) B_Cout <= '1' {borrow}
  else B_Cout <= '0' {no borrow}
else { -- if SUB_ADD='0'
  D_S <= (A + B )[7.0]+B_Cin; {lower 8 bits of A + B + B_Cin}
  B_Cout <=  (A+B+B_Cin)[8]; {most significant bit of A+B+B_Cin}
}
```

Figure 1: Eight -Bit Two's Complement Subtractor/Adder Module

The base function of the Subtractor/Adder module is to output the valid two's complement difference or sum between two eight-bit quantities that are present on its *A* and *B* input buses. This module is designed to be cascaded with itself to create proportionately larger size subtraction/addition operations. This is accomplished through the use of the borrow in carry in input, *B_CIN*, and a borrow out /carry out output, *B_COUNT*. The *SUB_ADD* input is the control input that is used to determine whether the input buses A and B will be subtracted or added to one another. Whenever the *SUB_ADD* line is set to a logic high, the *D_S* output produces a two's complement difference of *A-B* and the *B_COUNT* will indicate if a borrow is required at the most significant bit position. Whenever the *SUB_ADD* line is set to a logic low, the *D_S* output produces the sum of *A+B* and the *B_COUNT* line

indicates if a carry needs to generated at the most significant bit position. The module itself can be implemented in a number of ways (such as a ripple or look-ahead type logic) with each style of implementation presenting various complexity/performance trade-offs.

Simulation CAD Tool Environment

In this simulation assignment, students are required to utilize the integrated Quartus II/ModelSim® environment which is described in some detail in manual that is entitled *Design Entry, Simulation, and Emulation using the Altera Quartus II® and ModelSim® CAD Software and the Terasic DE2-115 Rapid Prototyping Platform*, which is available on the UAH CPE 322 Canvas site. Each student is encouraged to perform this simulation using their personalized copy of Quartus II/Altera ModelSim Starter Edition® which is available for free download from Altera® Corporation's website at www.altera.com. Students can also use the computers that have this software installed on them that are present in the UAH ECE's Advanced Logic Design Laboratory in Room 226 Engineering Building when this lab is not being used for a formal CPE 324 laboratory section. This assignment is focused on the simulation aspects only. Logic synthesis and implementation of the design on the Altera Teraasic DE2-115® rapid prototyping platform occurs in Laboratory 3 of CPE 324.

Simulation Types

In this assignment, students are required to perform two types of simulation for both the structural and behavioral models of the Subtractor/Adder Module. The first type of simulation to be employed is called an Register Transfer Level (RTL) or a functional simulation. This type of simulation assumes that all propagation timing delays associated with the logic elements are zero. Such a simulation type is useful to determine if the base function of the design is correct under ideal conditions but it does not incorporate any form of realistic timing considerations that are actually present in a real-world design. The second type of simulation is a timing simulation that does incorporate non-zero timing delays for the elements that make up the design. In such simulations these timing may be entered directly by the user during the design entry phase directly in the hardware description language. We will execute this type of simulation in a future laboratory. It is also often desirable, for timing data to be generated automatically by the CAD tool based upon the manner in which the design will be implemented within the FPGA (or within the Application Specific Integrated Circuit). In such information case, timing information is back annotated into the simulation after a full design has been fully synthesized. The goal is to create a simulation that reflects the worst case timing behavior associated with the targeted implementation. Such timing depends upon such factors as the manufacturing process employed, the expected operational temperature, and the voltage level that is employed. Manufacturers such as Altera will often provide several simulation timing model libraries that can be combined with the post layout information (synthesis, placement, routing) to examine the worst-case scenarios across these various design parameters. To be robust a design should simulate correctly for each of the timing models that are provided by the manufacture. In the case of the targeted Cyclone IV E EP4CE115F29C7 FPGA Altera provides three such timing models (*Slow-7 1.2 V 85°C, Slow-7 1.2V 0°C,* and the *Fast-M 1.2 V 0°C* models)

Targeted FPGA Post-Layout FPGA Pin Assignments

To be accurate, the post-layout timing simulations should incorporate the actual FPGA pin assignments which are associated with the design's inputs and outputs. For this assignment, all post-layout simulations should be made assuming that the pin mappings associated with the Cyclone IV E EP4CE115F29C7 FPGA that maps the switches to the inputs and discrete LED's to the outputs of the

Teraasic DE2-115® platform that were describe in CPE 324 Laboratory 3. These assignments are repeated for convenience in Table 1.

Table 1: Assumed Cyclone IV Pin Location Assignments for Post-layout Simulation

Design I/O		Design I/O	
Design Signal Name	**Cyclone IV Pin Number**	**Design Signal Name**	**Cyclone IV Pin Number**
A[7]	**PIN_AB26**	**B[1]**	**PIN_AB25**
A[6]	**PIN_AD26**	**B[0]**	**PIN_AC25**
A[5]	**PIN_AC26**	**B_CIN**	**PIN_Y24]**
A[4]	**PIN_AB27**	**SUB_ADD**	**PIN_Y23**
A[3]	**PIN_AD27**	**B_COUT**	**PIN_F17**
A[2]	**PIN_AC27**	**D_S[7]**	**PIN_G21**
A[1]	**PIN_AC28t**	**D_S[6]**	**PIN_G22**
A[0]	**PIN_AB28**	**D_S[5]**	**PIN_G20**
B[7]	**PIN_AA22**	**D_S[4]**	**PIN_H21**
B[6]	**PIN_AA23**	**D_S[3]**	**PIN_E24**
B[5]	**PIN_AA24**	**D_S[2]**	**PIN_E25**
B[4]	**PIN_AB23**	**D_S[1]**	**PIN_E22**
B[3]	**PIN_AB24**	**D_S[0]**	**PIN_E21**
B[2]	**PIN_AC24**		

Simulation Stimulus

For each simulation in this assignment the Subtractor/Adder module should be driven with a set of input vectors. The simulation should be driven with a new test vector every 50 ns of simulation time. There should be at least 8 unique test vectors in this set. This stimulus should be the same one that is manually applied to the DE2-115 switches in Laboratory 3 of CPE 324 class. Individual test vectors should excite the simulation in the following manner.

- 1) Add two numbers that do not generate an external carry out request.
- 2) Subtract two numbers that do not generate an external borrow out request.
- 3) Add two numbers in which there is a pending carry input.
- 4) Subtract two numbers when there is a pending borrow input.
- 5) Add two numbers where a carry in propagates all the way from LSB to carry out
- 6) Subtract two numbers where a borrow in propagates all the way from LSB and generates a borrow out request.
- 7) Add two numbers that causes an overflow into the carry bit.
- 8) Subtract a larger twos-complement positive number from a smaller one generating a negative result if the MSB is interpreted as the sign bit.

Structural Verilog HDL Design Implementation

The Verilog HDL design model shown in Figure 2 is a structural (hierarchical) model of a ripple implementation of the two's complement subtractor/adder design that was shown in Figure 1. A soft copy of this listing can be found on the CPE 322 Canvas® website. The purpose of this simulation is not design

creation but is allow each student to develop proficiency with the Quartus II/ModelSim® design tool and to make relevant observations that can be generally applied to designs that the student may create in the future.

```
// Structural Representation of Subtractor/Adder Module
// presented in Part I of Laboratory Experiment
// B. Earl Wells -- University of Alabama Huntsville, 2015
// This version places all component modules in a single file
// Note: it is also possible to include each of the component
// modules in separate files and set up the profile such that
// it references each of these files.

// It is good practice in Quartus to give the project the same
// name as the top-level module (or eight_bit_sub_add in this
// case)

// Top level module
// eight-bit subtractor/adder
module eight_bit_sub_add(D_S,B_COUT,A,B,B_CIN,SUB_ADD);
    output [7:0] D_S;
    output B_COUT;
    input [7:0] A,B;
    input B_CIN,SUB_ADD;
    wire n0;

    four_bit_sub_add C0(D_S[3:0],n0,A[3:0],B[3:0],B_CIN,SUB_ADD);
    four_bit_sub_add C1(D_S[7:4],B_COUT,A[7:4],B[7:4],n0,SUB_ADD);
endmodule

// four-bit subtractor/adder
module four_bit_sub_add(d_s,b_cout,a,b,b_cin,sub_add);
   output [3:0] d_s;
   output b_cout;
   input  [3:0] a,b;
   input b_cin,sub_add;
   wire n0,n1,n2;

   fullsubadd C0(d_s[0],n0,a[0],b[0],b_cin,sub_add);
   fullsubadd C1(d_s[1],n1,a[1],b[1],n0,sub_add);
   fullsubadd C2(d_s[2],n2,a[2],b[2],n1,sub_add);
   fullsubadd C3(d_s[3],b_cout,a[3],b[3],n2,sub_add);
endmodule
```

Figure 2: Structural Verilog Representation of Eight-bit Subtractor/Adder

```
// full subtractor/adder
module fullsubadd(dif_sum,b_cout,x,y,b_cin,sub_add);
   output dif_sum,b_cout;
   input x,y,b_cin,sub_add;
   wire n0,n1,n2,n3,n4;

   not    C0(n0,x);
   mux2_1 C1(n1,x,n0,sub_add);
   and    C2(n2,y,n1);
   and    C3(n3,y,b_cin);
   and    C4(n4,n1,b_cin);
   or     C5(b_cout,n2,n3,n4);
   xor    C6(dif_sum,x,y,b_cin);
endmodule

// 2 to 1 multiplexer
module mux2_1(o,i0,i1,sel);
   output o;
   input i0,i1,sel;
   wire n0,n1,n2;

   not G0(n0,sel);
   and G1(n1,i0,n0);
   and G2(n2,i1,sel);
   or G3(o,n1,n2);
endmodule
```

Figure 2 (continued): Structural Verilog Representation of Eight-bit Subtractor/Adder

Behavioral Verilog HDL Design Implementation

The Verilog HDL design described in Figure 3 below represents a behavioral (or a data flow) type Verilog model of the two's complement adder/subtractor design. A soft copy of this listing can be found on the Angel page associated with the laboratory.

```
// Behavioral Representation of Adder/Subtractor Module
// presented in Part II of Laboratory 3 Experiment
// B. Earl Wells -- University of Alabama Huntsville, 2015
// Single -- Top-Level Module
module eight_bit_sub_add(D_S,B_COUT,A,B,B_CIN,SUB_ADD);
   output reg [7:0] D_S;
   output reg B_COUT;
   input [7:0] A,B;
   input B_CIN,SUB_ADD;

   reg [8:0] Cbuf;

   always @(A or B or B_CIN or SUB_ADD) begin
      if (SUB_ADD) begin
         Cbuf = A - B - B_CIN;
      end
      else begin
         Cbuf = A + B + B_CIN;
      end
      D_S = Cbuf[7:0];
      B_COUT = Cbuf[8];
   end

endmodule
```

Figure 3 Verilog Behavioral Representation of Eight-bit Subtractor/Adder

Assignment

For both (structural and behavioral) Verilog models of the subtractor/adder module students are to perform an RTL type simulation and three post-layout type simulations using the three Altera supplied timings (*low-7 1.2 V 85°C*, *Slow-7 1.2V 0°C*, and the *Fast-M 1.2 V 0°C*)). The post layout simulation should assume that the targeted FPGA is a Cyclone IV E EP4CE115F29C7 and that the input/output pin assignments are those that are expressed in Table 1. There should be a total of eight (8) simulations that are executed. For each simulation, are to excite the simulation with the same set of test vectors. (These test vectors are to be developed in a manner that was outlined in the *Simulation Stimulus* section of this assignment.) The simulations should be executed in a manner that allows the response of the input/output signals to be viewed in both graphical form (i.e. a waveform) as well as a textual listing.

An electronic copy of this simulation assignment (in Adobe PDF®, or MS Word® format) should be produced and submitted on the CPE 322 course Canvas® web page. It should include the following items.

1 Stimulus input used to verify the functionality of the design (include ModelSim stimulus generation commands or the test bench stimulus generation commands used to generate the necessary inputs).

2 Complete waveforms (screen shots placed within the document) and listing files for each simulation with a clear indication of the specific simulation to which they apply.

3 Discussion of general observations and conclusions that include answers to the following questions.

a) For the RTL and post-layout simulations did you observe any glitches in the output that were not consistent with the test vector change at the input? If so which simulations did this occur in and what might be the source of these glitches? If not which simulations and why was the output glitch free?

b) For the RTL simulations were there any significant differences observed between the structural and behavioral models of the Subtractor/adder module? If so what were these differences. Is this result consistent with your expectations? Explain.

c) For the Post layout simulation were there any significant difference observed between the structural and behavioral models of the Subtractor/adder module? If so what were these differences. Is this result consistent with your expectations? Explain.

d) Based upon your worst case post-layout simulation, what is the smallest time period possible between that application of new test vectors that will always give the output time to respond to the input.

e) What are the general merits of viewing the simulation output data as a waveform? What are the disadvantages?

f) What are the general merits of viewing the simulation output data as a textual listing? What are the disadvantages?

Due date for this assignment is 11:55 PM February 4, 2015