Duke ECE496 – Spring 2013 – Project Part 3: Adder and Shifter

100 Points. Due electronically by 11:59pm on February 8.

In this part of the project, you will implement a 32-bit 2's complement adder and a 32-bit barrel shifter. These will be components in the ALU of your processor.

Project Part 3a: 32-bit Carry-Select-Lookahead Adder/Subtractor (60 points)

The adder uses a combination of ripple-carry, carry-lookahead and carry-select techniques. The lower 16-bit addition (adding bits 0-15 of the addends) is performed by two 8-bit carry-lookahead adders chained in ripple-carry fashion. The upper 16-bit addition (adding bits 16-31 of the addends) is performed using two 16-bit ripple-carry-lookahead adders used in carry-select fashion. Thus you will be using three identical 16-bit ripple-carry-lookahead adders for the entire 32-bit adder, as illustrated in Figure 1. You do not need to account for underflow or overflow.

In addition to the addends, the adder also takes a single-bit input ctrl_subtract which, if high, denotes that 2's complement subtraction should be performed instead of addition (data_sum = data_addendA – data_addendB). You will implement your adder in Structural VHDL using the Quartus II software. You should create one VHDL (adder.vhd) or Block-Diagram (adder.bdf) file that has exactly the same format as the diagram in Figure 2. This top-level adder.vhd or adder.bdf file is likely to refer to other lower-level files (e.g., 8bitCLA, etc.). The top-level file adder.vhd or adder.bdf file is what you will then use later in the semester when you need a fast adder for your processor's ALU. Figure 2 is a screenshot of the adder component in Quartus, and it shows the signal names that you MUST use in your design to facilitate testing and grading.

After implementing your adder, you should test it thoroughly to verify that it works correctly. One test waveform is provided for your adder in the file test_adder_s13.vwf. In addition, this assignment will be graded by running additional tests that are not provided, so do not assume that you can ignore bugs that do not manifest themselves on the one test that is provided.

Project Part 3b: 32-bit Logarithmic Barrel Shifter/Rotator (40 points)

The shifter/rotator uses the barrel shifting method discussed in lecture for faster delay and lower area. In addition to the operand and shift/rotate amount, the shifter/rotator also takes two other single-bit inputs: ctrl_right and ctrl_shift. If high, ctrl_right denotes that a right shift/rotate should be performed instead of a left shift/rotate. Analogously, ctrl_shift denotes a zero-extending or zero-filling shift when high, and a rotation if low. Remember that during right rotations the LSBs (least significant bits) take place of the MSBs (most significant bits) and the opposite happens for left rotations. You will implement your logarithmic barrel shifter/rotator in Structural VHDL using the Quartus II software. You should create one VHDL (shifter.vhd) or Block-Diagram (shifter.bdf) file that has exactly the same format as the diagram in Figure 3. This top-level shifter.vhd or shifter.bdf file is likely to refer to other lower-level files (e.g., mux32, etc.). The top-level file shifter.vhd or shifter.bdf file is what you will then use later in

the semester when you need a shifter for your processor's ALU. Figure 3 is a screenshot of the shifter component in Quartus, and it shows the signal names that you MUST use in your design to facilitate testing and grading.

After implementing your shifter/rotator, you should test it thoroughly to verify that it works correctly. One test waveform is provided for your shifter in the file test_shifter_s13.vwf. In addition, this assignment will be graded by running additional tests that are not provided, so do not assume that you can ignore bugs that do not manifest themselves on the one test that is provided.

Submitting This Assignment

To submit this assignment, create a Quartus Archive (Project → Archive Project) named project3.qar of all the files needed to implement your design. Make sure that your top-level files are named adder.vhd or adder.bdf and shifter.vhd or shifter.bdf. Names of lower-level files are unrestricted, but be sure to include them along with your top-level design entity in the Quartus Archive file. Submit your Quartus Archive to Sakai along with all group members' names and NetIDs.

Figure 1: 32-bit Tiered Carry-Select-Lookahead Adder (CSLA)

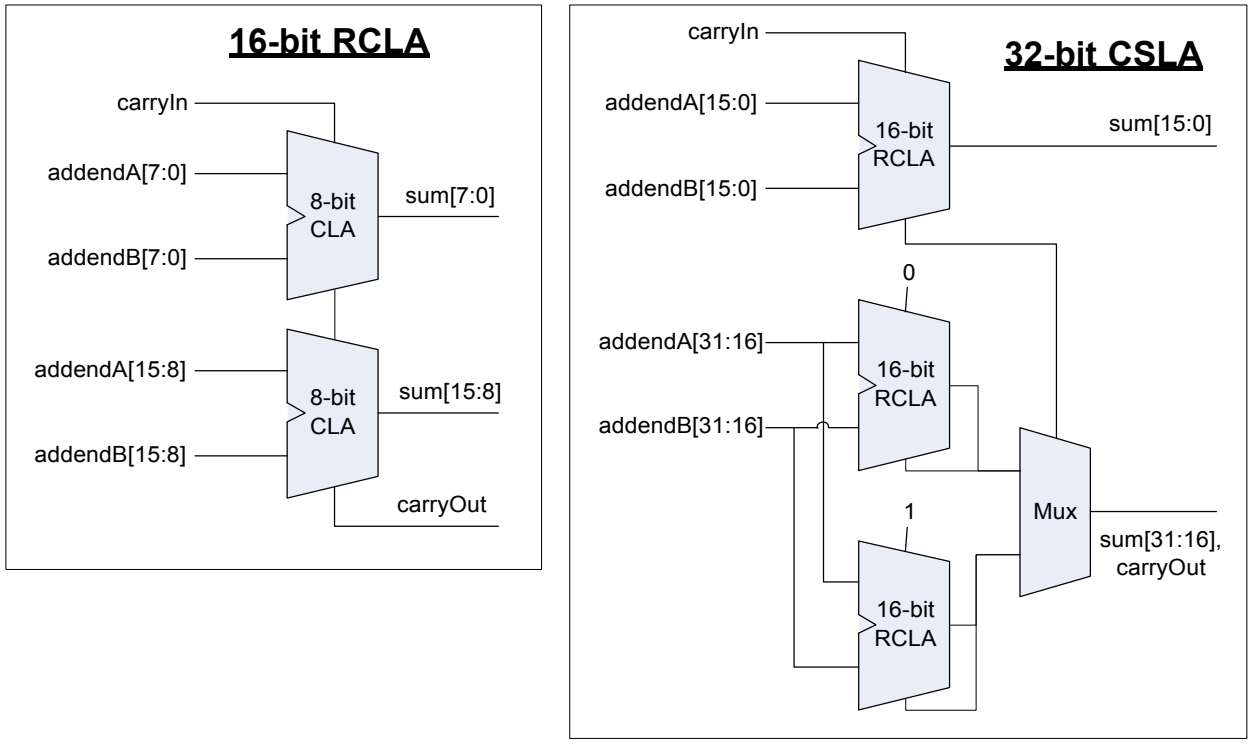

Figure 2: adder VHDL and BDF screenshots

```
ENTITY adder IS
    PORT (  data_addendA, data_addendB  : IN STD_LOGIC_VECTOR(31 DOWNTO 0); -- 32bit inputs
            ctrl_subtract   : IN STD_LOGIC; -- subtraction control (NOT add / subtract)
            data_sum    : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);    -- 32bit sum output
            data_carryout   : OUT STD_LOGIC);   -- carry output
END adder;
```

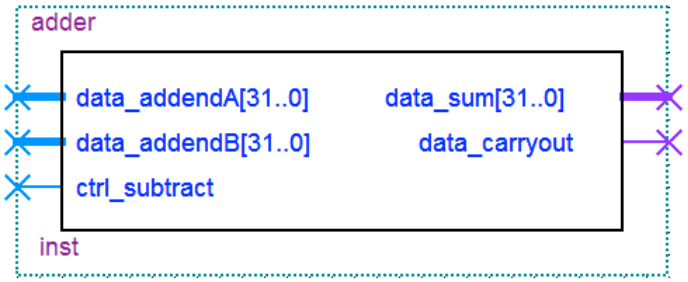

Figure 3: shifter VHDL screenshot

```
ENTITY shifter IS
    PORT ( data_A : IN STD_LOGIC_VECTOR(31 DOWNTO 0); -- 32bit input
           ctrl_right: IN STD_LOGIC;    -- shift/rotate direction (right/NOT left)
           ctrl_shift: IN STD_LOGIC;    -- shift/NOT rotate
           ctrl_shamt: IN STD_LOGIC_VECTOR(4 DOWNTO 0);  -- 5bit unsigned shift/rotate amount
           data_S: OUT STD_LOGIC_VECTOR(31 DOWNTO 0)); -- 32bit output
END shifter
```