ECE 385

Experiment #4

Spring 2018

Adders

Sahil Shah/sahils2

Samir Kumar/samirkk2

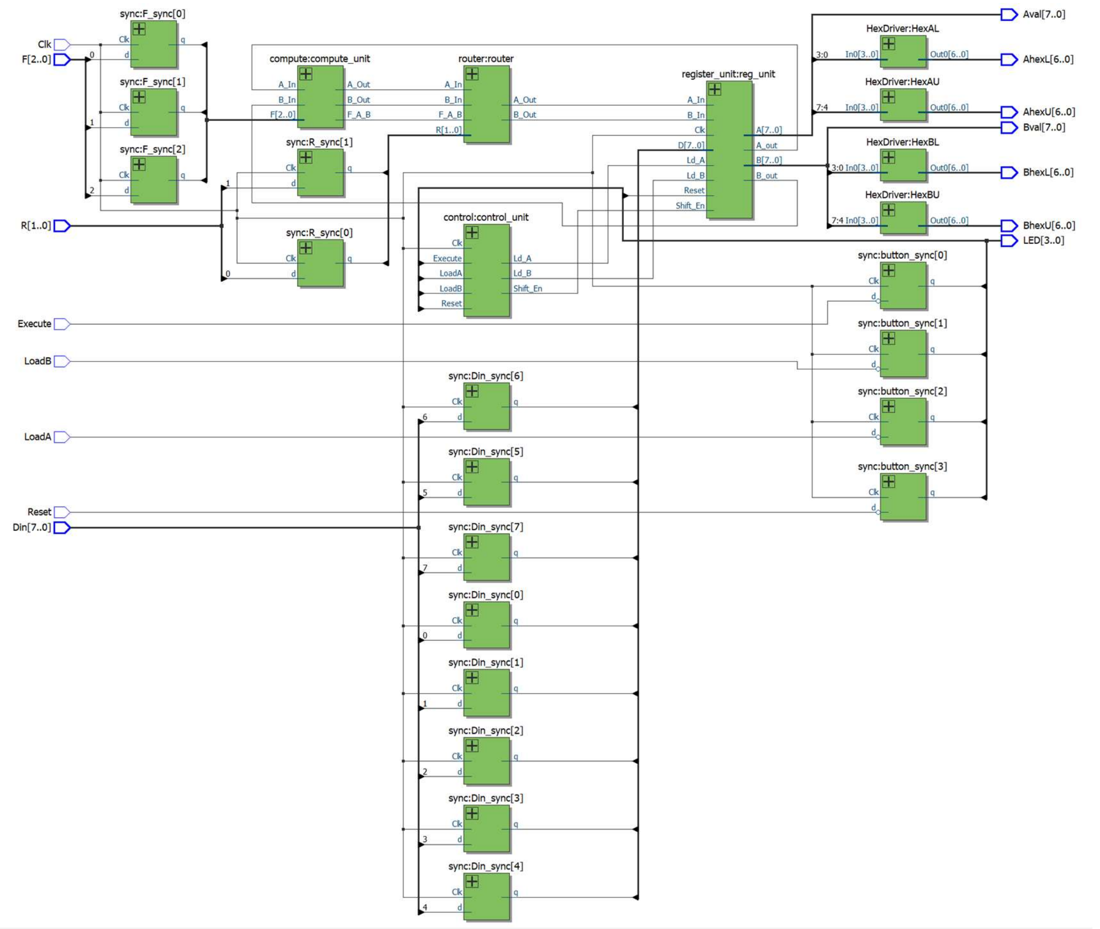
ABC - Tues, 11:30-2:20

Vibhakar Vemulapati

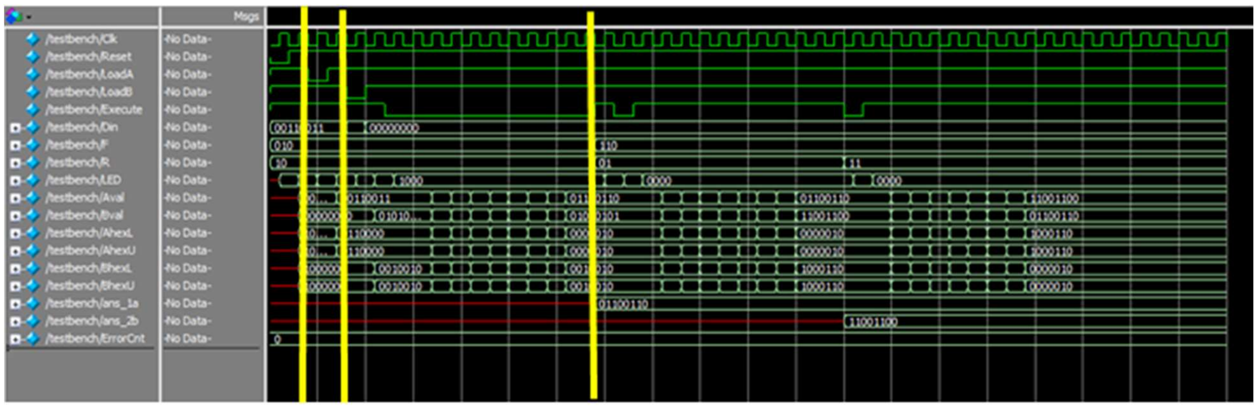
Introduction: This lab is an introduction to the Quartus design software and the SystemVerilog HDL (hardware design language). In this lab, we will extend the 4-bit serial logic processor to 8-bits and design a ripple-carry adder, a carry-lookahead adder, and a carry-select adder using SystemVerilog.

Bit-Serial Logic Processor:

Schematic Block Diagram:



Design simulation:



It is important to note that this simulation makes use of active low buttons and switches (0 is on and 1 is off). The first yellow line marks when LoadA turns on (the waveform for LoadA goes to 0), the value in Din is loaded into Aval and when LoadB turns on (marked by the second yellow line), the value in Din is put into Bval. Then when the Execute switch is toggled, which can be seen with the signal Execute going to 0 at the third yellow line, the register unit shifts, which is seen in the values of Aval and Bval.

Adders:

Description:The adders accepted 3 inputs: A, B, and the carry in (cin). Depending on the type of adder, cin was used in different ways. Each adder had at least 2 outputs: Sum and Carry out. 16 full adders were needed to implement each adder.

Ripple Carry: The carry out of the previous full adder was fed into the carry in of the next one. 16 consecutive full adders designed in this fashion make up the ripple-carry adder.

Carry-Lookahead (CLA): Every bit in a CLA generates a carry out immediately, depending the values of A and B. A carry out is *generated* if A and B are 1 (A AND B). The CLA bit will *propagate* the carry in if either A or B is 1 (A XOR B).

Carry-Select Adder: There are four, 4-bit adders in what is called a hierarchical design (the CLA was also designed this way). One of the 4-bit adders consists of two sets of four full adders. One set with a carry in of 1 and one set with a carry in of 0. These 4-bit adders use the ripple carry design, therefore there is a delay for the next 4-bit adder to receive the carry in. For this reason, each 4-bit adder computes the sum with a carry in of 1 and a carry in of 0. The correct sum and carry out are chosen depending on the actual carry in.

Module Description:

- full_adder: single bit full adder with 3 inputs: two operands and the carry in bit. The outputs are the sum and the carry out bit. The sum value is a triple XOR of the 3 inputs. The carry out is '1' of any two of the inputs is 1.
- 16bit_ripple_adder: combination of sixteen full_adder modules. Inputs include 16-bit inputs and a single bit carry in bit. Each bit of the 16-bit input is fed into each full_adder and the carry out is fed into the carry in of the next full_adder.
- 4bitCLA: this module the lowest part of the 4x4 hierarchical CLA design. The 4-bit CLA has 4-bit inputs and a carry in bit. The outputs are a 4-bit sum and the group propagate and generate signal. First, the group propagate signal is a logical AND of the propagate signal generated by each single bit adder. This signal is simply the result of ANDing the input bits (a and b). Next, the 4-bit CLA generate signal will be high if the most significant generate bit of the CLA is high or if a lower significant bit is high and the propagate signal(s) before it was also high. The final logic looks like this: G = g3 OR (g2 AND p3) OR (g1 AND p3 AND p2) OR (g0 AND p3 AND p2 AND p1). The group propagate

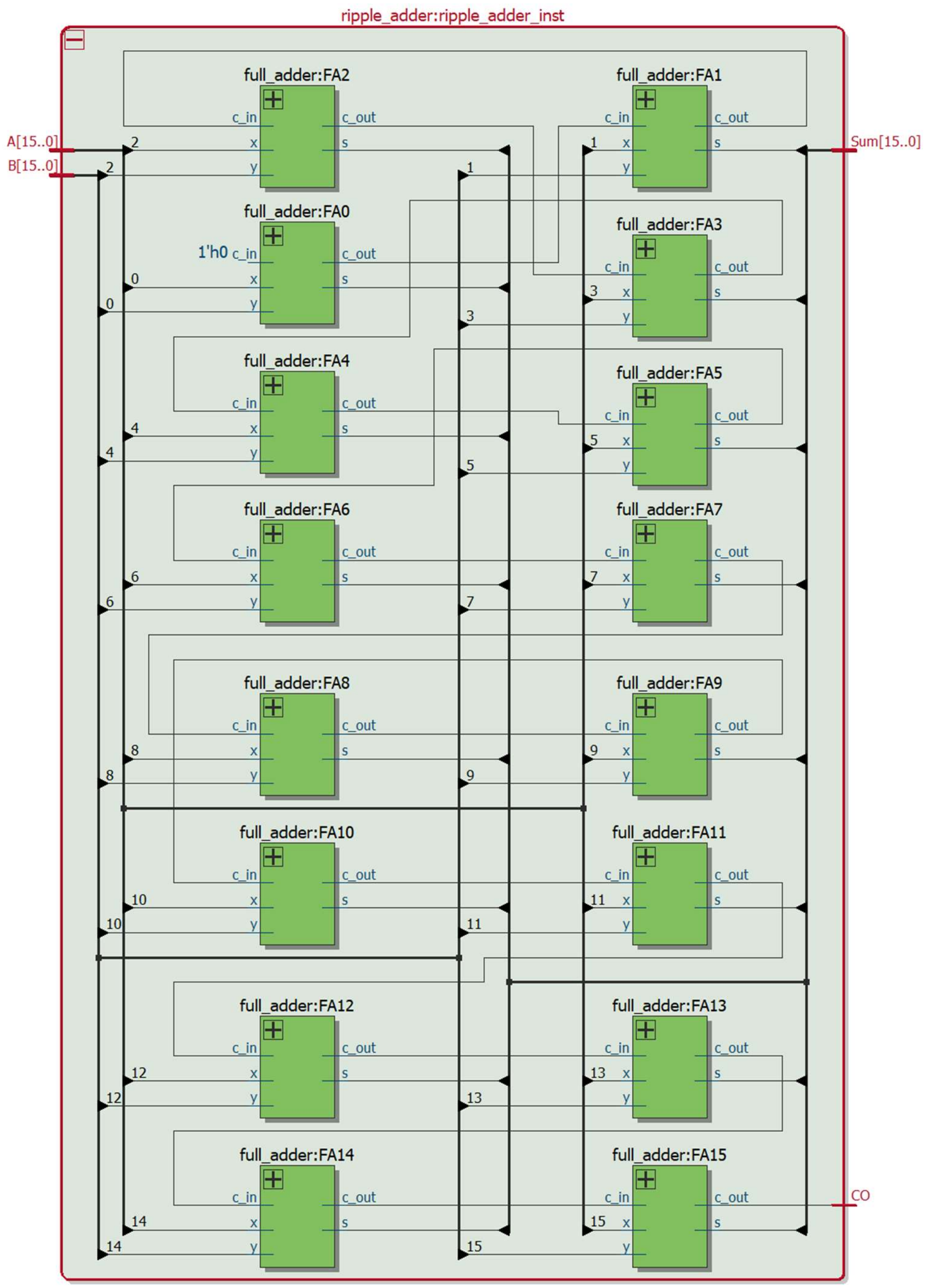
simply passes the carry in bit if high. The group generate signal will output a carry out using the above equation.

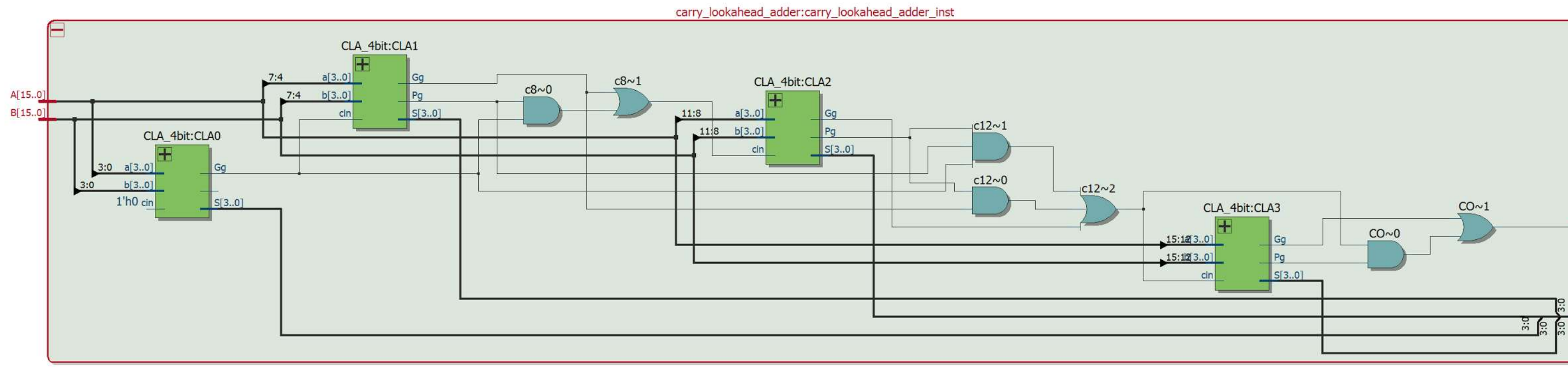
- Carry_lookahead_adder: This module used four 4bit CLAs and had similar inputs and outputs. They included two 16-bit inputs and a carry-in. The outputs included the sum (16-bit), the carry out, and the final propagate and generate signals. Depending on the group generate and propagate bits of each 4-bit module, each carry was determined.
- Carry_select_4bit: This module is also part of the 4x4 hierarchical design of this adder. It takes 3-bit operand input and outputs two sums and carry outs. One of them the initial carry in of 1 and the other with 0. Each 4-bit adder ripples the carries through.
- Carry_select_adder: This adder used four 4-bit CSA modules. Its inputs include the 16-bit operands, where individual quarters were distributed to each 4-bit module, and the carry in. Outputs included the sum and the carry out. The first 4-bit module used the actual carry in and selected the correct sum and carry out. The correct carry out of the first module is then rippled into the second 4-bit CSA. Because the two possible scenarios have been calculated already, the module only needs to select the correct sum and carry out based on the carry in, which is the carry out of the previous 4-bit CSA. This behavior is repeated for each CSA

Schematic Block Diagrams

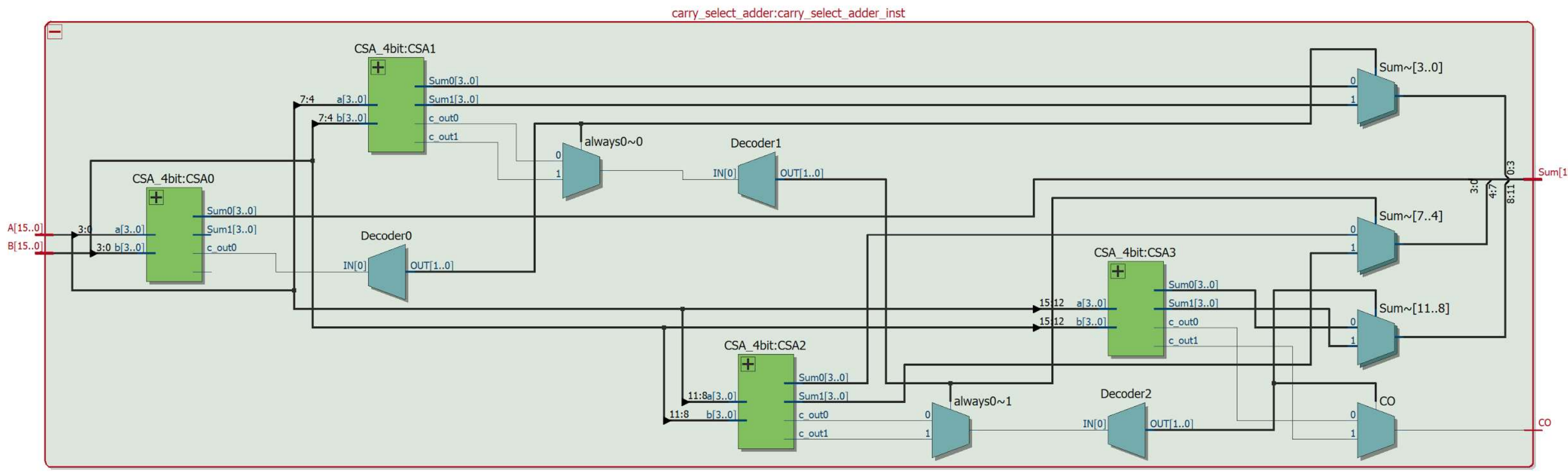
- Ripple

Adder:





- Carry Lookahead Adder:



- Carry Select Adder:

Design Analysis:

ACTUAL DATA

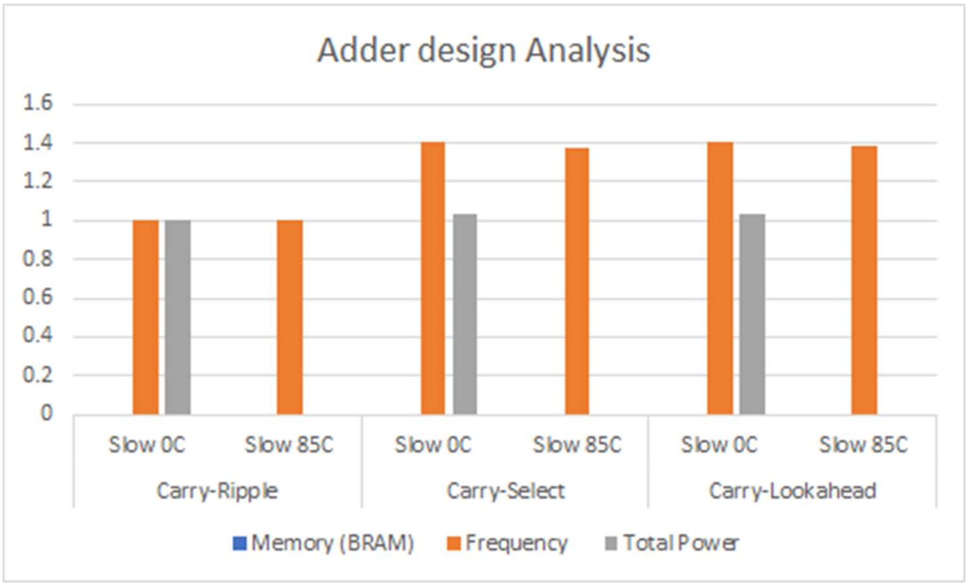
	Carry-Ripple		Carry-Select		Carry-Lookahead	
	Slow 0C	Slow 85C	Slow 0C	Slow 85C	Slow 0C	Slow 85C

Memory (BRAM)	0		0		0	
Frequency (MHz)	66.57	61.99	93.55	85.3	93.6	85.63
Total Power (mW)	156.37		161.9		161.99	

NORMALIZED FOR PLOT:

	Carry-Ripple		Carry-Select		Carry-Lookahead	
	Slow 0C	Slow 85C	Slow 0C	Slow 85C	Slow 0C	Slow 85C
Memory (BRAM)	0		0		0	
Frequency	1	1	1.4053	1.3760	1.4060	1.3813
Total Power	1		1.0354		1.0359	

PLOT:

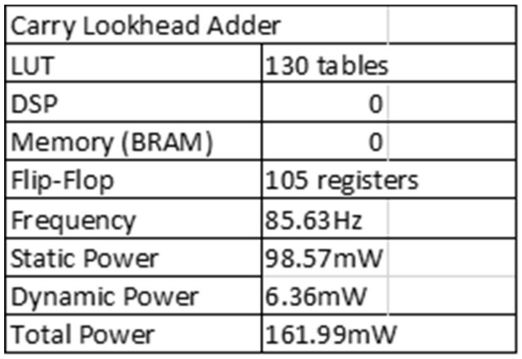


Post-Lab Questions

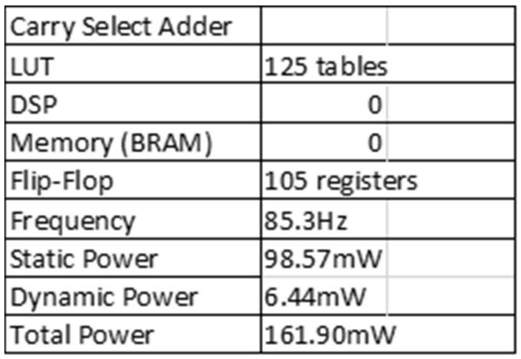
1. The TTL design is better because less logic elements are used in this design, thereby making it more efficient. As seen in Resource Usage Summary in the IQT, the SystemVerilog design using much more logic and memory than the circuit in Lab 3. The

FPGA uses around 2600 logic elements, while our TTL design uses about 700-900. Less elements means a smaller circuit footprint as well as a cheaper cost to make.

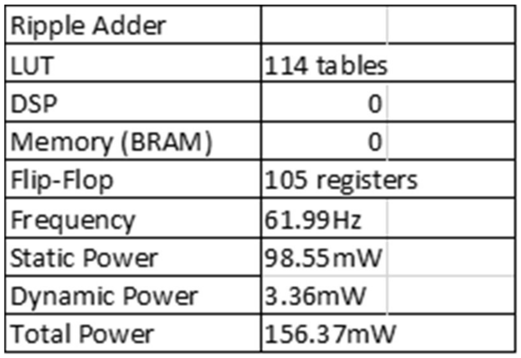
2.



Carry Lookhead Adder	
LUT	130 tables
DSP	0
Memory (BRAM)	0
Flip-Flop	105 registers
Frequency	85.63Hz
Static Power	98.57mW
Dynamic Power	6.36mW
Total Power	161.99mW



Carry Select Adder	
LUT	125 tables
DSP	0
Memory (BRAM)	0
Flip-Flop	105 registers
Frequency	85.3Hz
Static Power	98.57mW
Dynamic Power	6.44mW
Total Power	161.90mW



Ripple Adder	
LUT	114 tables
DSP	0
Memory (BRAM)	0
Flip-Flop	105 registers
Frequency	61.99Hz
Static Power	98.55mW
Dynamic Power	3.36mW
Total Power	156.37mW

The tables above show the resources used by each of the adders. This analysis confirms our initial analysis of the three adders because it shows that both the carry lookahead and carry select adders operate at higher frequencies than the ripple adder. This is expected because of the lack of waiting for the carry to ripple through each full adder. In addition, the CLA and CSA uses more total power than the ripple adder which is due to the increased number of logic elements used to implement it.

Conclusion: The design of this lab was given to us so the only challenges what we faced was learning the SystemVerilog syntax and how to operate Quartus and ModelSim. With regards to the lab itself however, we three different adder designs, all with different implementations. The ripple adder has the simplest design and thus uses the least number of logic elements and using the least amount of power. However, this implementation of the adder operates slowly so when the speed of a system is important, the ripple adder is not the best choice. The lookahead and select adders both run faster than the ripple adder with the lookahead adder being the fastest. However, the power consumption of the lookahead adder proved to be the greatest of the three. The select adder consumed slightly less power and operated slightly slower than the lookahead adder but was still much faster than the ripple adder.