ECE385 Experiment #4

Eric Meyers, Ryan Helsdingen

Section ABG; TAs: Ben Delay, Shuo Liu

February 17th, 2016

emeyer7, helsdin2

I. Introduction

THE purpose of this lab was to get a brief introduction to System Verilog and Quartus by creating and analyzing a total of 4 designs. The first is an 8 bit serial logic processor that was created during last lab (Lab 3) using TTL chips on a breadboard. The next three being different configurations of 16-bit adders: A carry-ripple adder, a carry-lookahead adder, and a carry-select adder. The team then analyzed the performance of these three adders by selecting different metrics to compare as well as analyzed the performance of the serial logic processor compared to the design used in Lab 3.

II. Schematic of Logic Processor

Please refer to Figure 2 in "Section XI: Figures" to view the Schematic of the 8-bit Logic Processor designed in the Pre-Lab of this Experiment.

III. Design Simulations of Logic Processor

Please refer to Figure 3 in "Section XI: Figures" to view the Annotated RTL Simulation Output of the 8-bit Logic Processor designed in the Pre-Lab of this Experiment.

IV. Written Description of Adder Circuit

The three adders used in this lab were the carry-ripple adder (CRA), carry-select adder (CSA), and the carry-lookahead adder (CLA). All three designs manipulated in some way or another the full adder. As you can see in the full adder block diagram (Fig. 6) and full adder truth table (Table IV, the full adder has two logic inputs, a carry-in bit input, and two outputs including a sum bit and a carry-out bit. The full adders function is to add all three input bits and send any access bits to the carry-out bit. In other words, the sum and carry-out bit form a two bit sum of the three inputs with the sum being the less significant bit.

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The first adder designed, the CRA, was the most straightfoward approach to an N-bit binary adder. It consisted of a chain of full adders connected by their carry bits. The first adder would have a carry-in bit of logic 0. It would take in the least significant bit of both register A and B (A[0] and B[0]) and add the two, outputting the least significant sum bit (S[0]) and a carry-out bit that would link to the carry-in bit of the next full adder. This chain would continue for all 16 bits. While this design is simple, long computation time is its biggest issue. Later full adders must wait for a signal in their carry-in bits in order to function correctly. One adder alone, has several gates the signal must go through before a carry-out signal is generated.

The next adder designed was the CLA. It brought in a method that would reduce the wait time that the CRA suffered from. This method took the CRA design and replaced the carry-out bit of the full adder with both a generating (G) bit and a propagating (P) bit. Each full adder would then take in input bits A and B and do a little educated guesswork with them. The G bit checked for two high bits, guaranteeing that the carry-in of the next full adder would be high. It followed G=A·B. The P bit checked for only one of the two to be high. It followed P=A+B, and would serve as a potential for a logic 1 to be sent to the next full adder. The last possible result of A and B inputs, that is, both are low, would output a low P and G bit giving no chance for the next carry-in bit to be logic 1. The carry-in bit of the next adder thus

held a logical expression of $C_{i+1} = G_i + (P_i \cdot C_i)$. This new method cuts down the computation time compared to the CRA by predetermining the next carry-in bit if inputs bits A and B share the same signal. Unfortunately the CLA uses more logic gates at the cost of area and power consumption.

The final adder designed was the CSA. This design was the least-straightforward of the three designs. Each bit required two full-adders, a 2-to-1 mux, an AND gate, and an OR gate. Both full-adders would take in the same A and B input bits. One full-adder would simulate a logic 0 for the carry-in bit and send its sum bit to the 0 input on the 2-to-1 mux. Likewise, the other full-adder would simulate a logic 1 for the carry-in bit and send its sum output bit the 1 input of the 2-to-1 mux. The output of the mux is the sum bit for that one-bit CSA and would be selected through the value of the carry-in bit for the one-bit CSA. The carry-out of the full-adder with the logic 1 carry-in gets AND'd with the carry-in of the one-bit CSA and that result gets OR'd with the carry-out of the full-adder with the logic 0 carry-in. The result is the carry-out bit of the one-bit CSA which gets sent to the carry-in bit of the next one-bit CSA. This design cuts down on computation time by precomputing both possibilities of carry-in bits. However, this design uses the most logic gates which increases both power consumption and area.

V. Purpose of Modules

For the CRA, only full-adders were used to add each bit. The need for modules was not necessary for this design, so it manipulated the flat approach. However, for the CLA and CSA, 4-bit modules were created to optimize speed. The number of gates per bit of adding was quite higher for these two designs. A flat approach would have been impractical especially for 16 bit addition or greater. Instead, a hierarchical approach was used. Both the CLA and CSA used 4-bit modules.

The CLA 4-bit module contained four copies of the one-bit CLA. The carry-in bit of the least significant bit acted as the carry-in bit of the module and the carry-out bit of the most significant bit acted as the carry-out bit of the module. The module took in four input bits for A and B sending them to their respective one-bit CLA inputs and outputted four sum bits, one from each one-bit CLA. Connecting the modules was a simple wire connecting the carry-out of one module to the carry-in of the next high significant 4-bit module. The 16-bit CLA had therefore 4x4-bit modules connected.

The same process was used with the CSA except that each module contained (4) 1-bit CSA. Also the need for guessing the carry-in bit on the least significant 4-bit module was unnecessary since it was guaranteed to be logic 0, so a 4-bit full-adder replaced it.

VI. State Machine

The State Machine for the Serial Bit Logic Processor Control Unit is shown in Figure 5 in "Section XI: Figures" of this document.

The State Machine is broken down into a total of ten states labeled A through J, with A being the initial/halt state and J being the final state. PUT MORE HERE

VII. Schematic Block Diagrams

All block diagrams are shown in Figures 6 to 13 found in "Section XI: Figures" of this document. Block diagrams show in order the CRA basic full adder block digram, then the CLA block diagrams, and finally the CSA block diagrams. For the CLA and CSA block digrams, first the 4x4-bit system block digram is shown, then the block diagram for a 4-bit module is shown, and lastly the one-bit block diagram is shown.

VIII. Design Analysis Comparison

The following table displays the values the team received upon analyzing the metrics requested in the pre-lab.

Metric	Ripple	Lookahead	Select
Memory(BRAM)	0	0	0
Frequency (MHz)	62.81	64.526	61.55
Power (mW)	156.65	156.39	156.30

The following figure

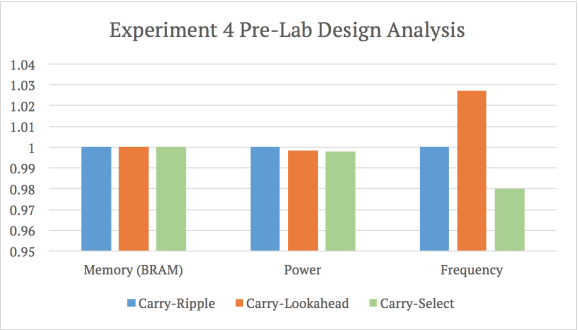

Fig. 1: Area, Power, Frequency Comparison

IX. Post-Lab

1) Compare the usage of LUT, Memory, and Flip-Flop of your bit-serial logic processor exercise in the IQT with your TTL design in Lab 3. Make an educated guess of the usage of these resources for TTL assuming

the processor is extended to 8-bit. Which design is better, and why?

Answer: The System Verilog design used in Lab 4 clearly dominates the total usage of Logic Elements in Lab 3 almost by a factor of three. This is because the system used in Lab4 is very efficient and optimizes/reduces the amount of Logic Elements used as neccessary. However, this was not done in Lab 3 efficiently. If the team attempted to do Lab 3 again, a lower LE count would have been achieved but no where near the efficiency of using Quartus.

8-bit Processor Comparison

	Lab4 - System Verilog	Lab3 - TTL Hardware
LUT (#)	67	186
Memory (#)	0	0
FlipFlop (#)	0	3

TABLE I: Processor Area Comparison

2) For the adders, refer to the Design Resources and Statistics in IQT.30-32 and complete the following design statistics table for each adder. This is more comprehensive than the above design analysis and is required for every SystemVerilog circuit.

These numbers are shown below in Table II .

Adder Comparison

	Ripple Adder	Lookahead Adder	Select Adder
LUT (#)	114	114	114
DSP (#)	0	0	0
Memory (#)	0	0	0
Flip-Flop (#)	6	5	6
Frequency (MHz)	62.81	64.526	61.55
Static Power (mW)	98.55	98.55	98.55
Dynamic Power (mW)	3.67	3.39	3.28
Total Power (mW)	156.65	156.39	156.3

TABLE II: Processor Area Comparison

Observe the data plot and provide explanation to the data, i.e., does each resource breakdown comparison from the plot makes sense? Are they complying with the theoretical design expectations, e.g., the maximum operating frequency of the carry-lookahead adder is higher than the carry-ripple adder? Which design consumes more power than the other as you expected, why?

Answer: The frequency of the Lookahead Adder was, indeed, faster than that of the Ripple Adder or the Select Adder. This makes sense because, although the Lookahead Adder consumes more area through logic elements (must connect each adder together with some logic so that the next adder can see the carry bits), it is faster which is shown in the frequency analysis.

The only result that was somewhat surprising to the group was the fact that the Select Adder was slower than the Ripple Adder. The team was not sure why this was the case because the Ripple Adder should *usually* be slower than the Ripple Adder however, not in this case. This might be attributed to the fact that.....

X. Conclusion

Overall, the team successfully demonstrated all three different adder configurations (Carry-Ripple, Carry-Lookahead, and Carry-Select) during the demo period for full credit. These adders were analyzed in the previous sections and it is evident that the Lookahead Adder was the fastest out of all three, and the Select Adder consumed the least amount of power out of all three.

The serial bit logic processor created in the pre-lab outperformed the bit logic processor constructed in lab 3 in several manners. For one, the total number of logic elements on the System Verilog Processor was $\approx$ 60 while the TTL Lab 3 total logic elements was exceeding $\approx$ 180. This is almost a factor of three reduction because of Quartus optimization techniques and improvements to the circuit.

Overall, the team gained plenty of insight to the benefits of using System Verilog vs. using TTL chips on breadboards. The complexity decreased significantly and breaking sequential and combinational logic into modules through System Verilog proves to be a very effective.

XI. Figures

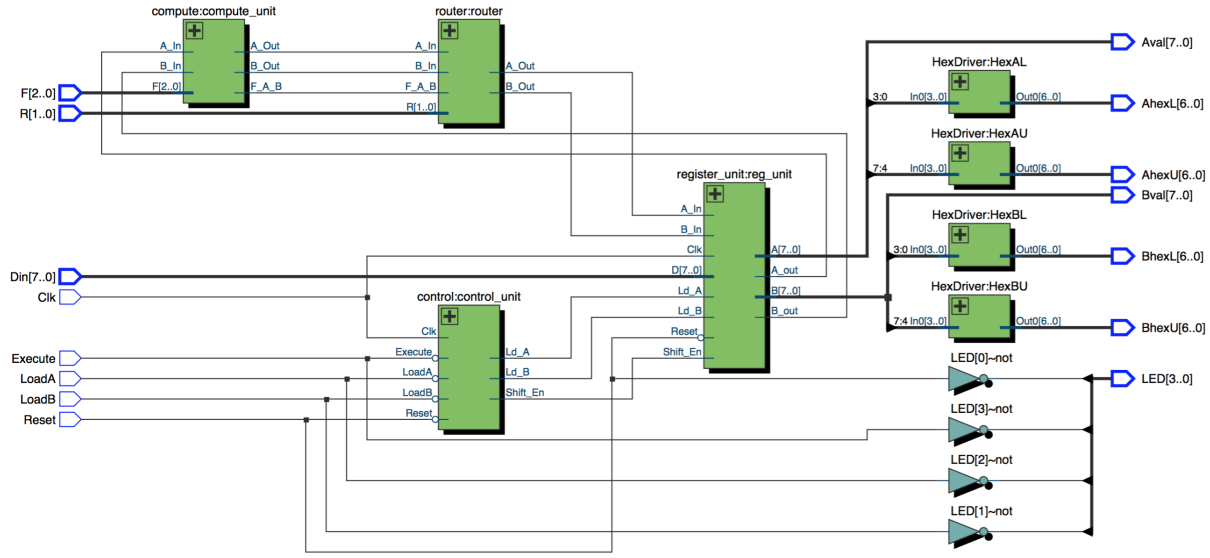

Fig. 2: Serial Logic Processor Schematic

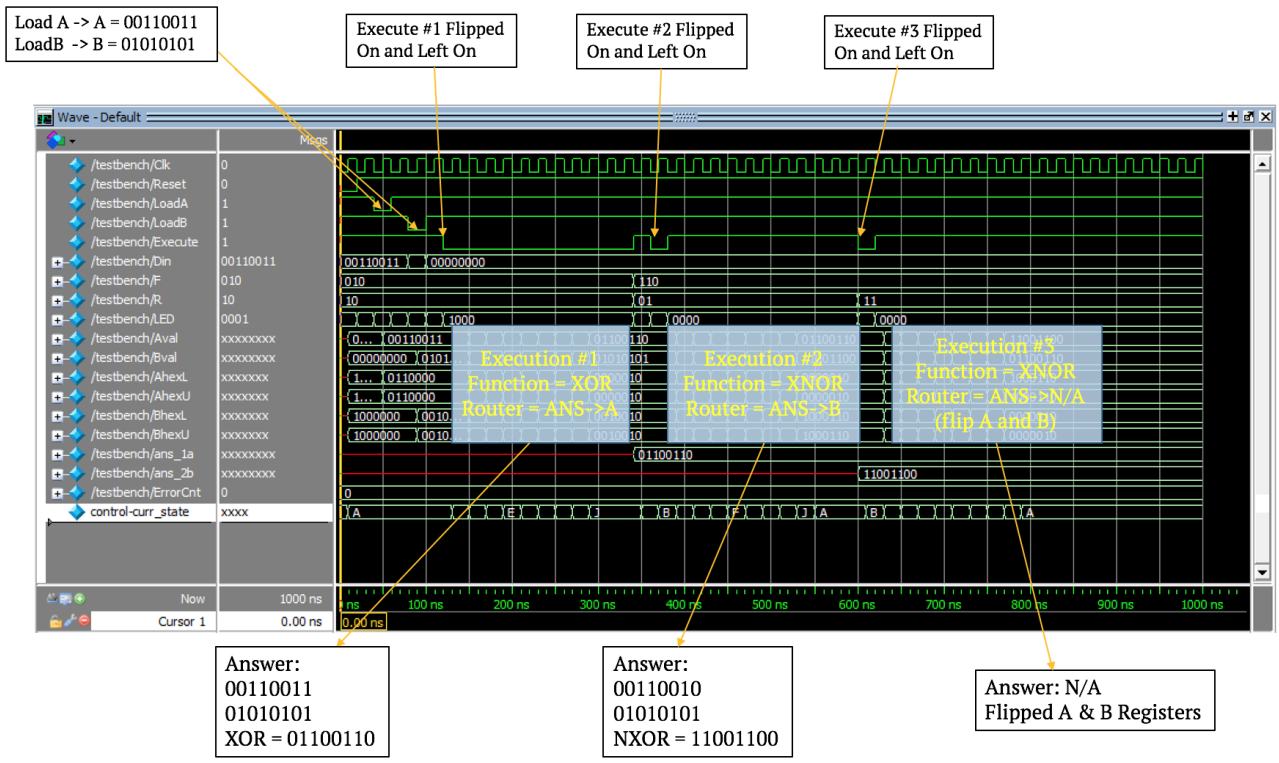

Fig. 3: Serial Logic Processor RTL Simulation Output

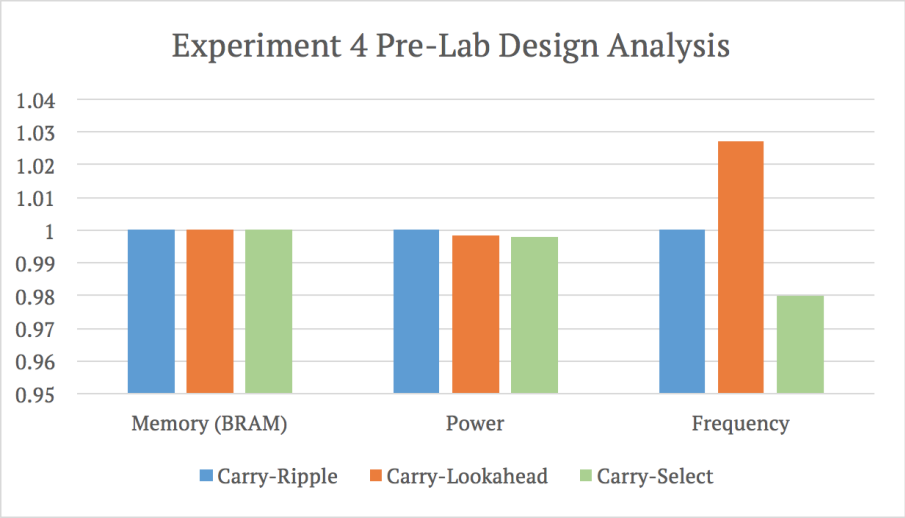

Fig. 4: Pre-Lab Adder Design Analysis (Area, Power, Frequency)

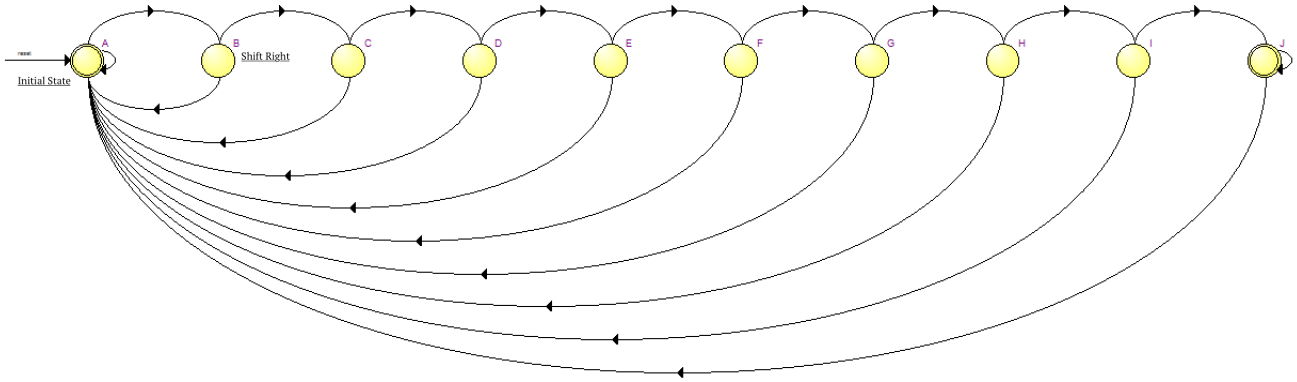

Fig. 5: State Machine for Serial Bit Logic Processor

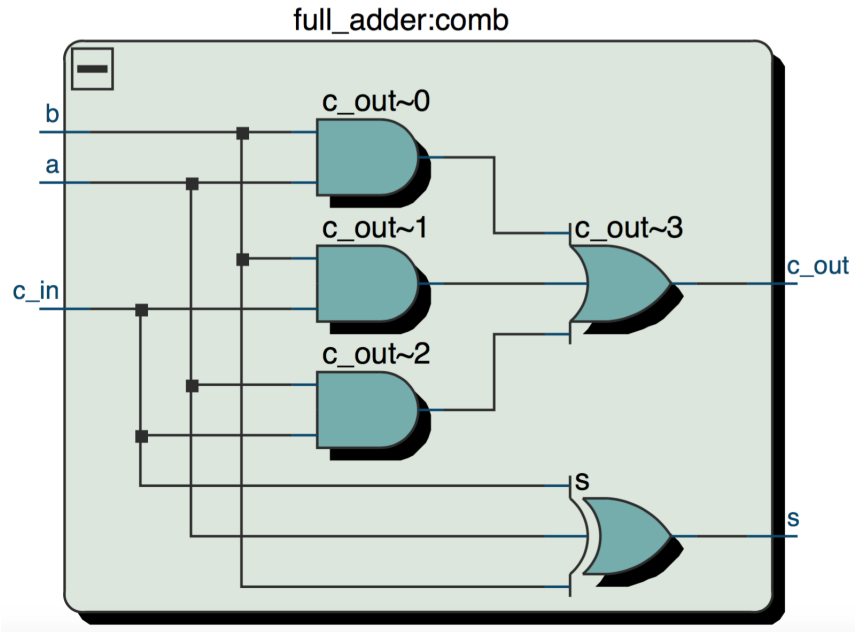

Fig. 6: Full-Adder Block Schematic

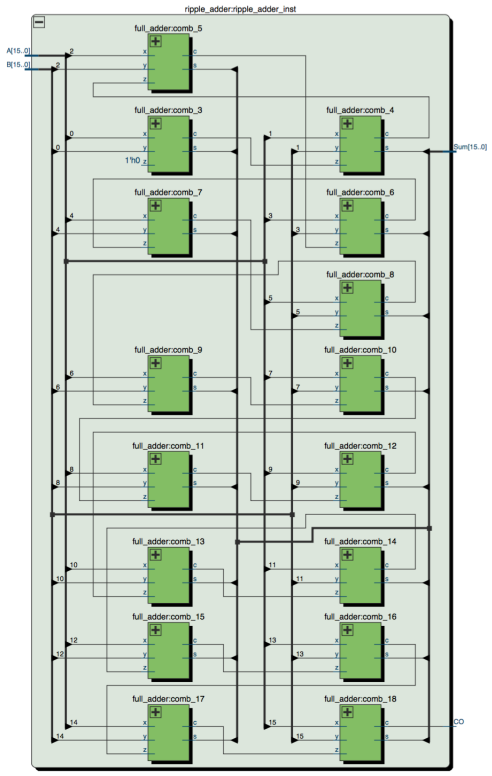

Fig. 7: Carry Ripple Block Schematic

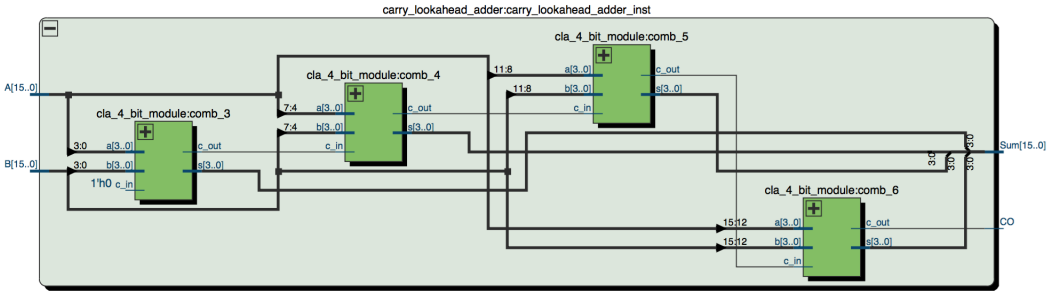

Fig. 8: Carry Lookahead (CLA) Block Schematic

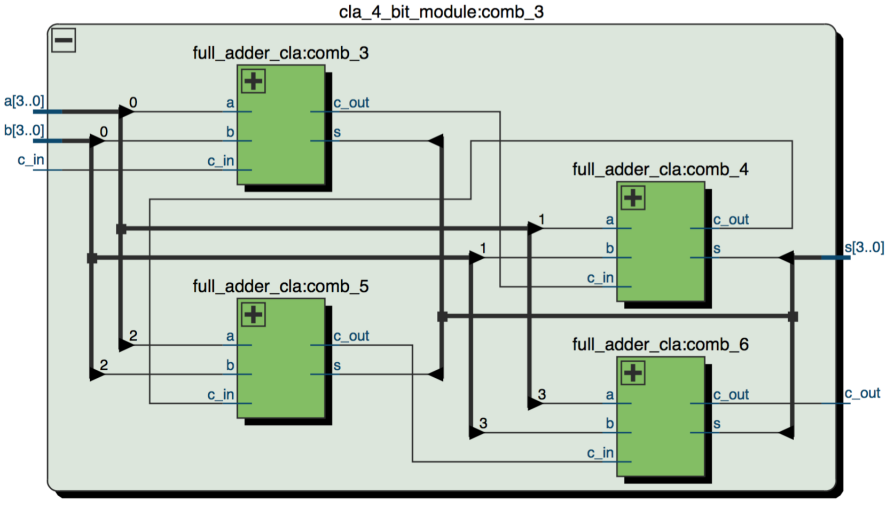

Fig. 9: Four-Bit CLA Block Schematic

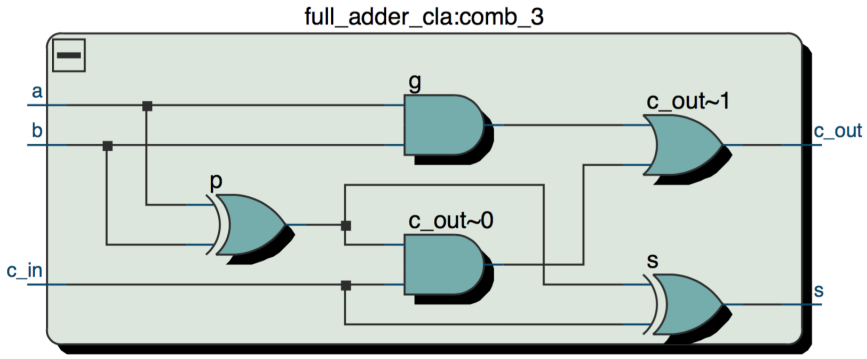

Fig. 10: One-Bit CLA Block Schematic

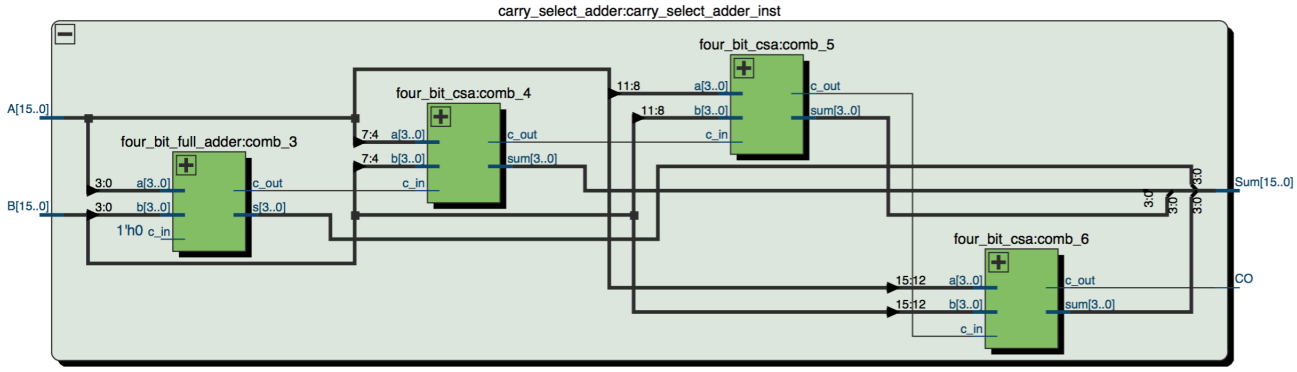

Fig. 11: Carry Select (CSA) Block Schematic

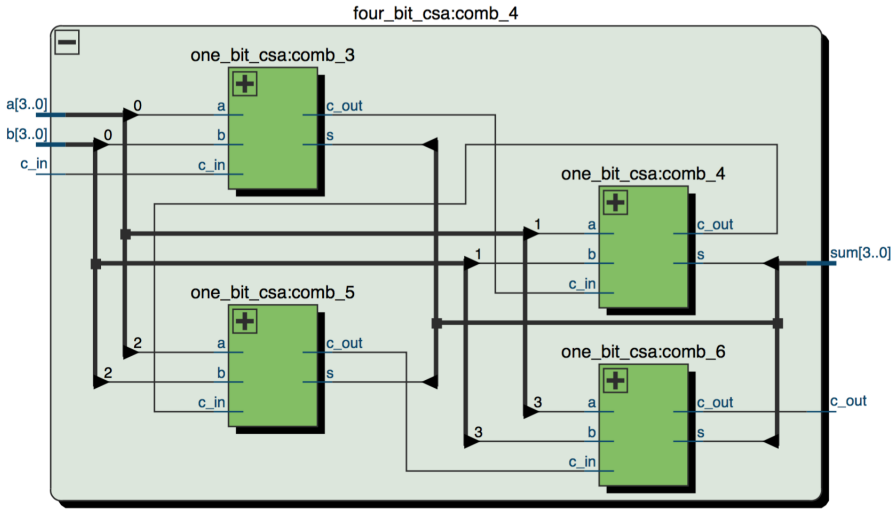

Fig. 12: Four-Bit CSA Block Schematic

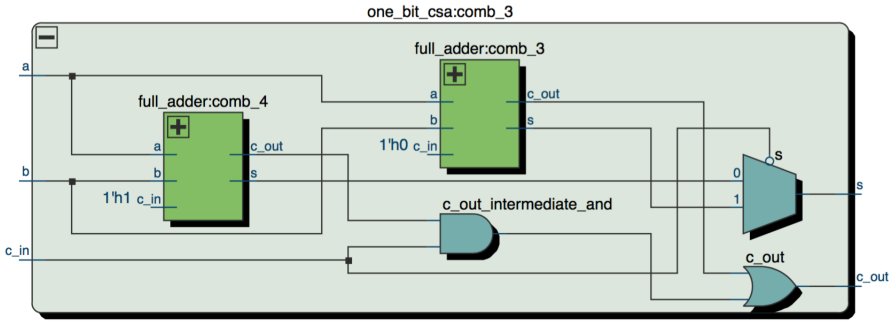

Fig. 13: One-Bit CSA Block Schematic