Complex Digital Design

Final Lab Report

Junhan Bao, Wentai Ye

March 10, 2024

1 Summary

In the final project, we designed a 128-bit adder with a 3-layer structure, and it takes 5 clock cycles to complete a 512-bit addition. Besides, we designed a 512-bit subtractor, a 512-bit comparator and a function selector. The function selector can select the operation of addition, subtraction or comparison.

2 Technical Description

2.1 Function Selector

Based on the uart_top in Lab2, we added a function selection module because we can now operate subtraction and comparison besides addition. The final state machine diagram of uart_top is as figure[1]. The final state machine diagram of the 512-bit calculator in the above diagram is as figure[2].

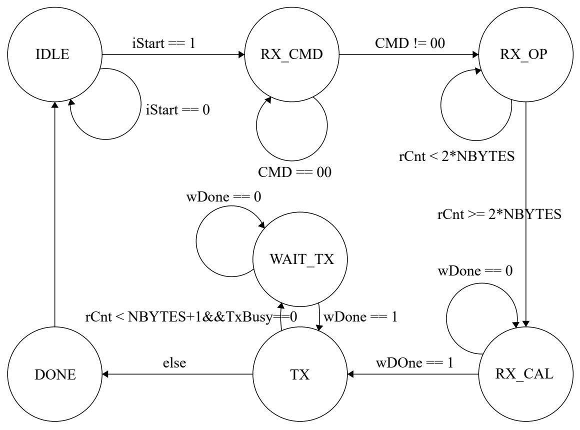


Figure 1: The state machine diagram of uart_top

It first compares the two input operands(iA and iB); if iA is equal to iB, we can derive the sum or difference of the two operands, which is zero without any calculation. If iA is larger than iB, we can derive the sum or difference of the two operands by adding iA and the complement of iB.

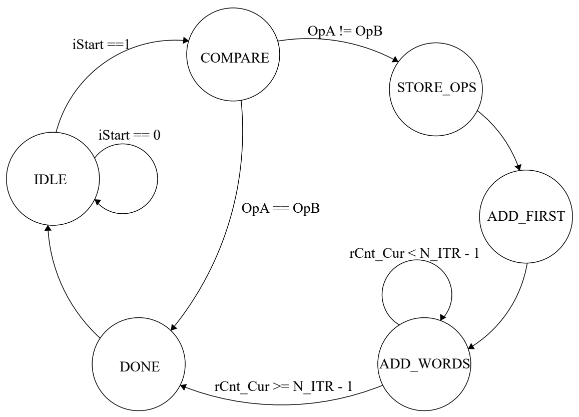


Figure 2: The state machine diagram of 512-bit calculator

2.2 Adder

2.2.1 3-layer adder structure

We designed a 128-bit adder, where the bottom-level module is an 8-bit carry-lookahead adder. First, it is expanded to 32 bits through four layers of carry-select adders and then expanded to 128 bits through four more layers of carry-select adders.

The structure of the top layer is shown in figure[3]. The structure of the medium layer is shown in figure[4]. The structure of the bottom layer (Improved 8-bit carry-lookahead adder) is shown in figure[5].

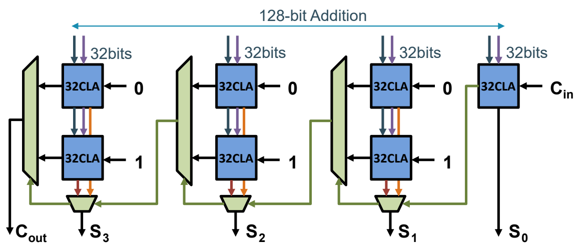


Figure 3: The top layer structure of carry-select adder

If speed is the primary concern, a different design may be more appropriate. If the area is the primary concern, a smaller adder with fewer layers may be better. However, the design we have chosen is a good compromise between speed and area.

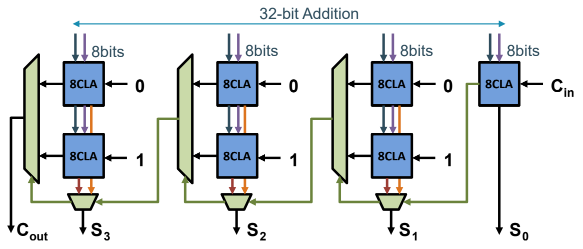


Figure 4: The medium layer structure of carry-select adder

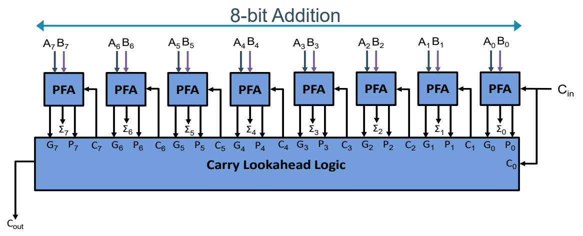


Figure 5: The bottom layer structure of 8-bit carry-look-ahead adder

2.3 Subtractor

The 512-bit subtractor takes two 512-bit inputs, iA and iB, and produces a 512-bit output by first calculating the complement of the smaller operand, denoted as iA. The result is obtained by adding the complement of iA and the other operand iB using the adder that has been designed.

2.4 Comparator

The 512-bit comparator takes in two 512-bit inputs, iA and iB, and outputs a 3-bit result, oRes.

The comparator first computes the bitwise XOR (diff) of the two inputs. It then generates two sets of flags (L1 and L2) by checking if any of the 8-bit chunks in the diff are non-zero.

Next, the comparator selects the largest non-zero flag and corresponding 8-bit chunk in the L2 buffer and saves it in L1buff. The corresponding 64-bit chunk in input iA is also saved in Abuff.

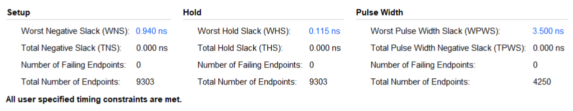
The comparator then selects the largest non-zero flag and corresponding 8-bit chunk in the L1buff and saves it in Asub. The comparator then processes the bits in Asub and L1buff to generate smaller 4-bit and 2-bit flags, and corresponding chunks (A4, L4, A2, L2) until only 1-bit flags (A1, L1) remain.

Finally, the comparator checks the value of L2 to determine the result. If all flags are zero, oRes is set to 3'b010, indicating that the two inputs are equal. If the largest flag in iA is zero, oRes is set to 3'b100, indicating that iA is larger than iB. Otherwise, oRes is set to 3'b100, indicating that iA has a smaller value than iB.

3 Performance Evaluation

3.1 Timing

As shown in the figure[6] of the timing report below, the worst negative slack(WNS) of setup time(WHS) is 0.940 ns, and the worst hold slack is 0.115 ns. The worst delay is 6.840 ns, shown in figure[7].



Setup

Worst Negative Slack (WNS):	0.940 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	9303

Hold

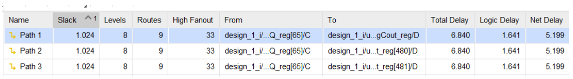
Worst Hold Slack (WHS):	0.115 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	9303

Pulse Width

Worst Pulse Width Slack (WPWS):	3.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	4250

All user specified timing constraints are met.

Figure 6: TimingClosure of the Adder



Name	Slack ^1	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	1.024	8	9	33	design_1_i/...Q_reg[65]/C	design_1_i/u...gCout_reg/D	6.840	1.641	5.199
Path 2	1.024	8	9	33	design_1_i/...Q_reg[65]/C	design_1_i/u...t_reg[480]/D	6.840	1.641	5.199
Path 3	1.024	8	9	33	design_1_i/...Q_reg[65]/C	design_1_i/u...t_reg[481]/D	6.840	1.641	5.199

Figure 7: The worst delay is 6.840 ns

However, in our design, since a 512-bit comparator is also implemented and integrated with a 512-bit calculator, the 128-bit adder is not the slowest combinational circuit but the comparator. The adder's WNS is around 1.0 ns when the comparator is disabled.

It takes 5 clock cycles in total to realize a 512-bit adder and subtractor. One of 5 cycles is used for the comparator, and the rest of the 4 cycles is used for 129-bit add, as figure[8] and

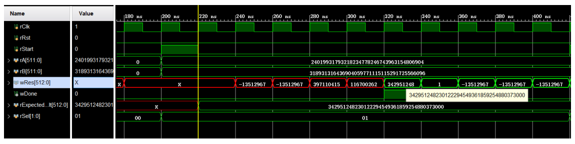


Figure 8: The simulation result of calculator adding function

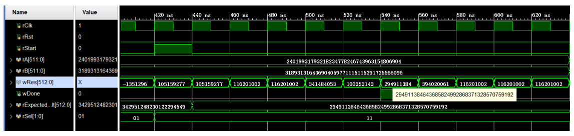


Figure 9: The simulation result of calculator subtraction function

figure[9] show. Another, if the command is ”C”, which means only to compare 2 inputs, it only takes one clock cycle instead, as shown in figure[10].

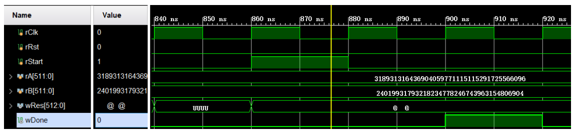
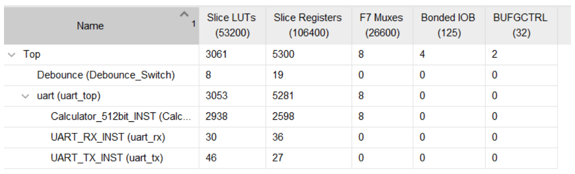


Figure 10: The simulation result of calculator comparing function

3.2 Utilization and Power

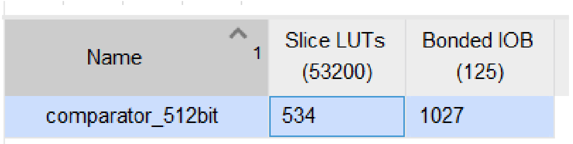
The 128-bit calculator used 2938 LUTs and 2598 FF in total, as shown in the following figure[11] of the utilization report. The 512-



Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	Bonded IOB (125)	BUFGCTRL (32)
Top	3061	5300	8	4	2
Debounce (Debounce_Switch)	8	19	0	0	0
uart (uart_top)	3053	5281	8	0	0
Calculator_512bit_INST (Calc...	2938	2598	8	0	0
UART_RX_INST (uart_rx)	30	36	0	0	0
UART_TX_INST (uart_tx)	46	27	0	0	0

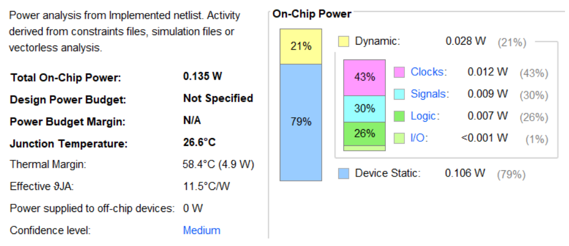
Figure 11: The utilization of 512-bit calculator

bit comparator has used 524 LUTs, as shown in figure[12]. As for the power consumption, its total on-chip power is 0.135W at a medium confidence level, as shown in figure[13].



Name	Slice LUTs (53200)	Bonded IOB (125)
comparator_512bit	534	1027

Figure 12: The utilization of 512-bit comparator



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.135 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.6°C**
Thermal Margin:	58.4°C (4.9 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

On-Chip Power

Dynamic:	0.028 W	(21%)
Clocks:	0.012 W	(43%)
Signals:	0.009 W	(30%)
Logic:	0.007 W	(26%)
I/O:	<0.001 W	(1%)
Device Static:	0.106 W	(79%)

Figure 13: The power report of this design

4 Comparison

Ripple carry adder with ADDER_WIDTH = 16 takes 16 clock cycles in total as shown below. Though it consumes fewer hardware resources, it achieves fewer functions and worse performs.

The WNS is 2.559 ns for ripple adder, as shown below, which is quite a lot, which means it is still possible to make it calculator more bits every clock cycle, for example, 24 bits.

As shown in figure[16], the Ripple adder costs 1336 LUTs, as shown below. It is pretty much less than a 128-bit adder.

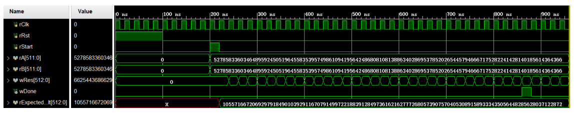
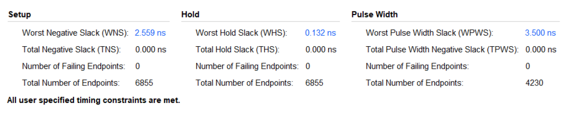


Figure 14: The simulation result of the 16-bit RippleAdder



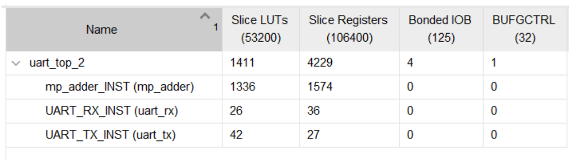
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.559 ns	Worst Hold Slack (WHS):	0.132 ns	Worst Pulse Width Slack (WPWS):	3.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	6855	Total Number of Endpoints:	6855	Total Number of Endpoints:	4230

All user specified timing constraints are met.

Figure 15: The timing closure of the 16-bit RippleAdder

Work	LUTs	WNS(ns)	Power	Required Clock Cycle
This Design	2938	0.940 ns	0.135W	5
16bit-Ripple Adder	1336	2.559 ns	0.145 W	16

Table 1: Comparasion between this design and 16-bit ripple adder



Name	Slice LUTs (53200)	Slice Registers (106400)	Bonded IOB (125)	BUFGCTRL (32)
uart_top_2	1411	4229	4	1
mp_adder_INST (mp_adder)	1336	1574	0	0
UART_RX_INST (uart_rx)	26	36	0	0
UART_TX_INST (uart_tx)	42	27	0	0

Figure 16: The area cost of the 16-bit ripple adder

Compared to the ripple carrier adder, the adder we designed is much faster but larger with more functions. The reason is that the ripple carrier adder is a serial adder, but the adder we designed to put as much logic into parallel operation as possible. The ripple carrier adder needs to wait for the carry signal to be calculated before it can calculate the next bit. In contrast, this design can calculate multiple bits simultaneously, making it much faster. In addition, since more parallel modules are needed, this design costs more hardware resources.

As for the design of the carry-select adder, we found that it could perform best when 4 sub-adder blocks are implemented. If the number of sub-adder blocks goes up, it is better to use a varied-sized carry-select adder.

As for the design of the carry-lookahead adder, the more carries that are calculated beforehand, the more benefits we can gain on calculation speed. We also found that using a multi-input adder calculates faster than the combination of the 2-input adder.

As for an 8-bit carry-lookahead adder, the calculation of carry-out requires the largest amount of computation, performing an AND operation on 9 inputs and then performing an OR on 8 inputs. To reduce the delay caused by data passing through gates, we used multi-input AND gates and multi-input OR gates to perform this operation.

Experimental results have shown that different implementation methods can result in variations in the logic gate generation and impact timing.

For example, when we need a 3-input AND operation, if we write ”iA & iB & iC”, two AND gates will be instantiated, resulting in a time delay caused by two AND gates. However, if we write ”&iA, iB, iC”, one 3-input AND gate will be instantiated, resulting in a time delay caused by only one 3-input AND gate. This causes its calculation speed to be much faster than that of two 2-input AND gates. Therefore, we used this characteristic when designing the carry-lookahead logic circuit to lower the calculation time significantly.

However, the situation does not get better as the input number increases. When it comes to 32 inputs, the speed of calculation gets slower, according to our experiments. Plus, since the 32-input carry-lookahead logic circuit is too long to write and difficult to debug, we did not try the carry-lookahead 32-bit adder.