KU Leuven

Bachelor of Engineering Technology

Complex Digital Design

Hardware Accelerator for Arithmetic Operations

CHAIRI Mahmoud
MATTIACCI Esteban

May 6, 2023

1 Introduction

The main goal of this project was to improve the multi-precision-adder design by designing a new combinational adder, more efficient than the current 16-bit Ripple Carry Adder. The new design is expected to perform the addition in fewer clock cycles, therefore, decreasing the overall latency.

2 Features

For our design, we have adopted a hybrid approach that combines the Carry Lookahead Adder with the Carry Select Adder. Specifically, we have utilized the Carry Lookahead Adder within the blocks of the Carry Select Adder. All of our modules have been implemented in a generalized manner, allowing them to be N bits adders that can be instantiated for any given values of `ADDER_WIDTH` and `BLOCK_SIZE`. This approach has improved the versatility and robustness of the code while simplifying the testing of various values.

Our design can handle `ADDER_WIDTH` values up to 128 bits with blocks of 11 bits. In this case, our adder requires only six clock cycles to complete the addition operation. We have prioritized the development of a robust and efficient adder and, as such, our implementation supports only addition. However, if the operands are given in two's complement form, our system can also be used to perform subtraction.

3 Technical Description

Our adder design is a hybrid of two well-known adder architectures: the Carry Select Adder and the Carry Look Ahead Adder. The Carry Select Adder reduces the latency of an addition operation by computing the addition between subsets of the operands for both the carry-one and carry-zero cases. The carry then propagates through the circuit and selects the correct result, resulting in improved performance at the expense of increased circuit area. To perform the pre-computations of each subset pair from the two operands, we utilized the Carry Look Ahead Adder, which accelerates the propagation of the carry in the circuit by using carry look-ahead logic.

By combining these two approaches, we were able to achieve a significant improvement in terms of latency while incurring an acceptable cost in terms of area, which is well-suited to the context of this project. Our adder design allows for efficient computation of an addition operation and is capable of handling `ADDER_WIDTH` values up to 128 bits with blocks of 11 bits.

The following diagram showcases a single block. M-bit operands will be processed, one bit per PFA adder. Each sequence of PFA adders will receive a carry of either 1 or 0. Two multiplexers then decide which one will be the final output. This decision is based on the prior carry.

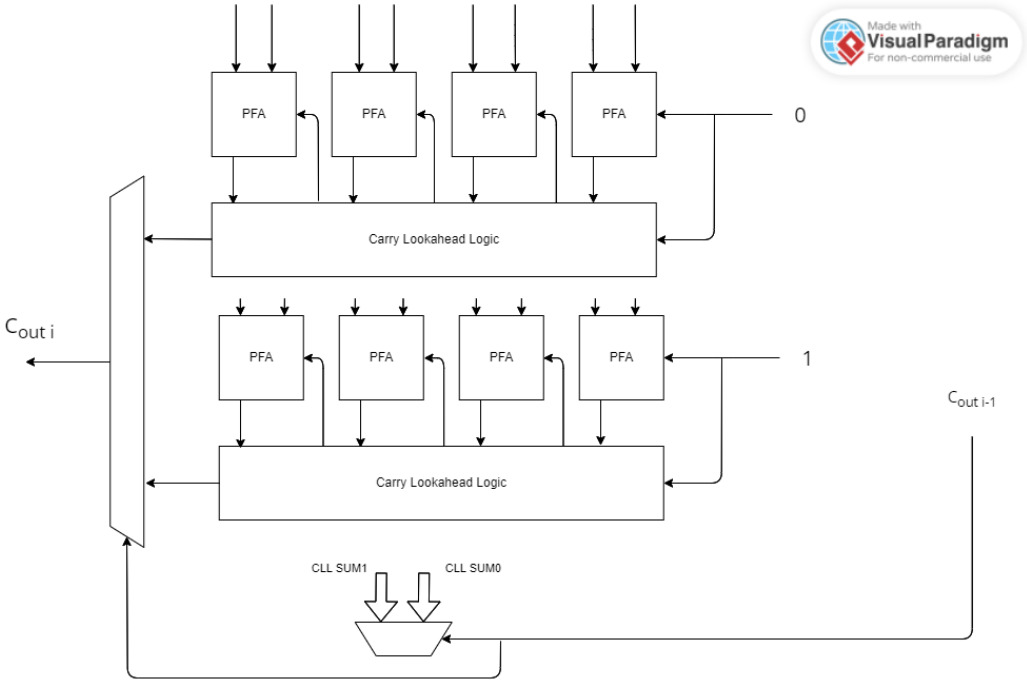


Figure 1: One block module from Carry Select Adder

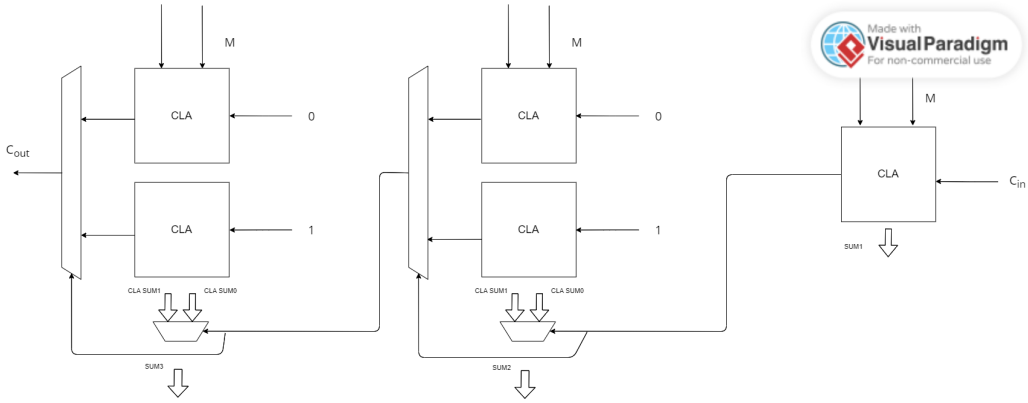
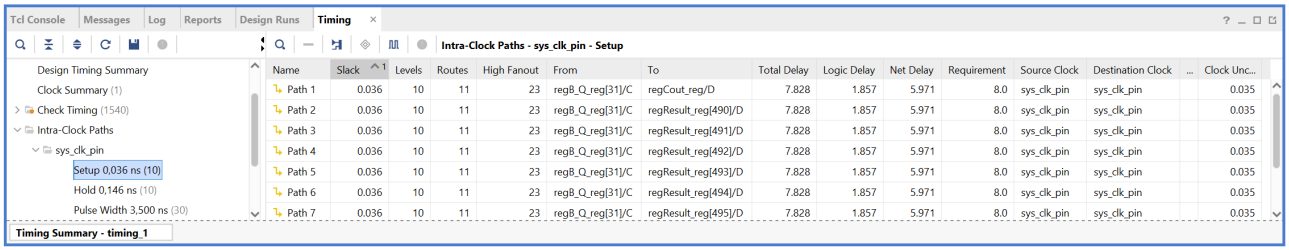


Figure 2: Overview of the whole adder

In the high-up diagram, an overall overview of the design is given, for 3 Blocks receiving M sized bit operands.

4 Performance evaluation

For an `ADDER_WIDTH` of 128 and a `BLOCK_SIZE` of 11, we obtain the following reports in Vivado :



Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	...	Clock Unc...
Path 1	0.036	10	11	23	regB_Q_reg[31]/C	regCout_reg/D	7.828	1.857	5.971	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 2	0.036	10	11	23	regB_Q_reg[31]/C	regResult_reg[490]/D	7.828	1.857	5.971	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 3	0.036	10	11	23	regB_Q_reg[31]/C	regResult_reg[491]/D	7.828	1.857	5.971	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 4	0.036	10	11	23	regB_Q_reg[31]/C	regResult_reg[492]/D	7.828	1.857	5.971	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 5	0.036	10	11	23	regB_Q_reg[31]/C	regResult_reg[493]/D	7.828	1.857	5.971	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 6	0.036	10	11	23	regB_Q_reg[31]/C	regResult_reg[494]/D	7.828	1.857	5.971	8.0	sys_clk_pin	sys_clk_pin		0.035
Path 7	0.036	10	11	23	regB_Q_reg[31]/C	regResult_reg[495]/D	7.828	1.857	5.971	8.0	sys_clk_pin	sys_clk_pin		0.035

Figure 3: Timing report for the hybrid adder with `ADDER_WIDTH` of 128 and a `BLOCK_SIZE` of 11

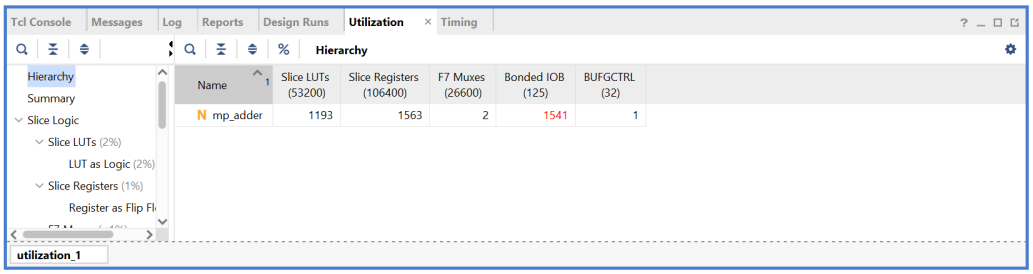


Figure 4: Utilization report for the hybrid adder with `ADDER_WIDTH` of 128 and a `BLOCK_SIZE` of 11

As we can see, the timing constraints are met.

5 Comparison and Discussion

We had the following timing and utilization reports for the 16-bits Ripple Carry Adder :

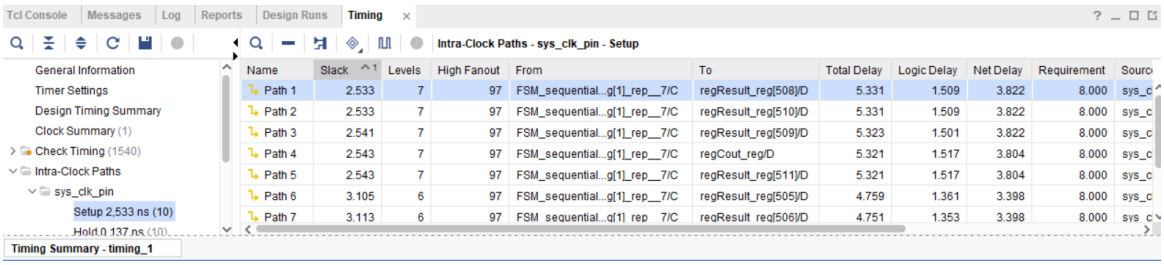


Figure 5: Timing report for the 16-bits Ripple Carry Adder

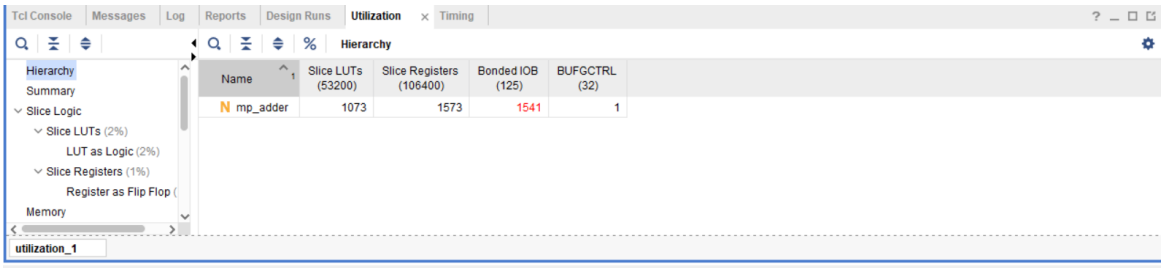


Figure 6: Utilization report for the 16-bits Ripple Carry Adder

Looking back to the Ripple Carry Adder, which could take a maximum of 16 bits to meet the timing constraints, we increased the possible `ADDER_WIDTH` by 8 times and still obtain a positive slack. Thus, the total clock cycles needed for the full addition are now reduced from 34 to 6. The trade-off for this increase in performance is that more area is needed on the FPGA to implement the Hybrid Carry Adder. This is expected, since an increase in speed in any design, also will lead to an increase in area usage.