Synchronous 4-Bit Addition Module using Carry Look-Ahead Adder and D Flip-Flops

Sai Sampath Ravikanti International Institute of Information Technology Hyderabad, India
sampath.ravikanti@students.iiit.ac.in
2023102033

***Abstract*—This paper presents the design and implementation of a synchronous 4-bit addition module. The module integrates a carry look-ahead adder (CLA) with D flip-flops for synchronized, high-speed operations. The proposed design is implemented in NGSpice and Verilog, with layout done in MAGIC software. It achieves extremely low delay while optimizing parameters like area and power consumption.**

***Index Terms*—D flip-flop, 4-bit carry look-ahead adder, power-efficient, minimum delay.**

I. Introduction

Addition is a fundamental operation in Arithmetic Logic Units (ALUs). Various designs, such as ripple adders and carry select adders, are commonly used for addition. However, these circuits often suffer from significant propagation delays due to the dependency of higher bit calculations on the carry-out of lower bits.

The Carry Look-Ahead Adder (CLA) addresses this issue by calculating the carry and sum directly from inputs. To synchronize the CLA with an external clock, high-speed D flip-flops are used for input and output bits.

II. D Flip-Flop

Proposed Structure: To construct an efficient D flip-flop, we use **Complementary Pass-Transistor Logic** (CPL). This logic employs PMOS and NMOS transistors as switches synchronized with the clock. CPL circuits consume less power and are faster than CMOS counterparts, making them ideal for high-speed applications.

The latch design in CPL includes an NMOS and PMOS transistor with their drains and sources interconnected, followed by a static CMOS inverter to restore voltage levels. This ensures proper cascading for constructing a D flip-flop. The proposed structure utilizes a total of 10 transistors, comprising 5 PMOS and 5 NMOS transistors.

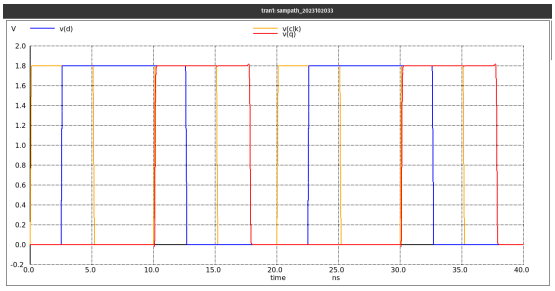

Fig. 1: Output of the D flip-flop.

A. NGSpice Simulations

1) Clock-to-Q Delay: The clock-to-Q delay of the flip-flop is evaluated using the NGSpice whose output is given in Figure 15.

```
tpd_rise           =  1.374924e-10 targ=  5.287492e-09 trig=  5.150000e-09
tpd_fall           =  1.177288e-10 targ=  1.526773e-08 trig=  1.515000e-08
total_prop_delay   =  1.27611e-10
```

Fig. 2: Clock to Q delay obtained from NGSpice

From simulations, the clock-to-Q delay is:

$$t_{\text{pcq}} = 137.49\,\text{ps}$$

To achieve the least delay, PMOS and NMOS transistor dimensions are optimized to balance rise and fall times. The chosen dimensions are:

$$W_{\text{p}} = 3\lambda = 0.27\mu, \quad W_{\text{n}} = 3\lambda = 0.27\mu.$$

2) Setup and Hold Time: Setup and hold times are determined by analyzing signal alignment with clock transitions:

$$t_{\text{su}} = 17.61\,\text{ps}, \quad t_{\text{h}} = 87.35\,\text{ps}.$$

B. MAGIC Layout

The layout of the D flip-flop, implemented in MAGIC, is shown in Figure 3. This circuit occupies a total area of $107\,\lambda \times 59\,\lambda$.

Post-layout simulations yield the following delays:

$$t_{\text{pcq}} = 168.03\,\text{ps}, \quad t_{\text{su}} = 29.25\,\text{ps}, \quad t_{\text{h}} = 97.86\,\text{ps}.$$

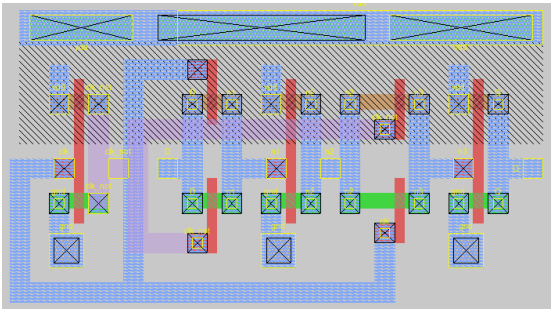

Fig. 3: MAGIC layout of the D flip-flop.

```
tpd_rise            =  1.495891e-10 targ=  5.299589e-09 trig=  5.150000e-09
tpd_fall            =  1.682114e-10 targ=  1.531821e-08 trig=  1.515000e-08
total_prop_delay    =  1.58900e-10
```

Fig. 4: Clock to Q delay obtained from MAGIC

III. CLA Adder

Proposed Structure: In order to enhance the speed of our addition module, we utilize a Carry Look-Ahead Adder (CLA). A traditional CLA operates by generating two key signals: propagate (P_i) and generate (G_i) for each bit, which help in reducing the overall delay.

$$P_{\text{i}} = a_i \oplus b_i$$

$$G_{\text{i}} = a_{\text{i}}.b_{\text{i}}$$

These signals allow the carry outputs to be calculated independently of the carry input from the previous stage, thus eliminating the sequential delay inherent in a cascade of full adders. This inherent parallelism is what makes the CLA faster than traditional ripple-carry adders.

In our design, we aim to accelerate the CLA further for even higher speeds. To achieve this, we introduce an additional signal, the "kill" signal (K_i).

$$K_{\text{i}} = \overline{a_i + b_i}$$

Moreover, we modify the logic of the generate signals.

$$G_{\text{i}} = \overline{a_i \oplus b_i}$$

In our CLA adder, we will be generating the kill, propagate and generate signals. From this, kill and generate will be exclusively used for the generation of the carry output at the end of all the adders, while propagate will be used along with the carry outputs to generate the sum bits.

$$C_1 = \overline{G_1} \tag{1}$$

$$C_2 = \overline{G_2 \cdot (K_2 + G_1)} \tag{2}$$

$$C_3 = \overline{G_3 \cdot (K_3 + G_2) \cdot (K_3 + K_2 + G_1)} \tag{3}$$

$$C_{\text{out}} = \overline{G_4 \cdot (K_4 + G_3) \cdot (K_4 + K_3 + G_2 \cdot (K_2 + G_1))} \tag{4}$$

$$S_{\text{i}} = P_i \oplus C_{i-1}$$

To generate the carry outputs (C_i), we use **static CMOS logic** as they provide stable and robust outputs. Hence, we create multi-input gates as shown. To perform the XOR operations to generate (P_i) and (S_i), we have chosen **Pass Transistor Logic**. Pass transistor logic offer faster switching compared to a traditional static CMOS. Moreover, they also consume less power and area as compared to static CMOS logic.

To generate the carry outputs (C_i), **Static CMOS logic** is utilized due to its stability and robustness in providing reliable outputs. Multi-input gates are employed as illustrated in the design. Apart from the carry outputs, every logic, including propagate, generate and kill signals in the adder has been implemented using **Pass Transistor Logic (PTL)**. Pass Transistor Logic offers faster switching speeds compared to traditional static CMOS logic, while also delivering advantages in terms of reduced power consumption and smaller area requirements. Pass Transistor Logic (PTL) further contributes to reducing the overall transistor count, as it eliminates the need to normalize the voltages of the carry outputs, which are subsequently reused to generate the sum bits. Once the sum bits are generated using PTL, CMOS inverters are employed to normalize the high and low voltage levels, ensuring proper logic voltage levels for the final outputs. This combination of PTL for efficiency and CMOS inverters for voltage normalization helps optimize both power consumption and transistor count. The proposed structure utilizes a total of 120 transistors, comprising 43 PMOS and 77 NMOS transistors.

Now, coming to the transistor sizing, for the static CMOS inverter use we use -

$$W_{\text{p}} = 11\lambda = 0.99\mu, \quad W_{\text{n}} = 3\lambda = 0.27\mu.$$

For implementing the carry outputs, the widths of the PMOS and NMOS are -

$$W_{\text{p}} = 4\lambda = 0.36\mu, \quad W_{\text{n}} = 3\lambda = 0.27\mu.$$

The sizing of the pass transistors is given as -

$$W_{\text{n}} = 6\lambda = 0.54\mu.$$

A. NGSpice Simulations

Upon simulating the proposed adder in NGSpice for various input cases of A and B, the output sequence observed is typically as follows: the output carry (C_{out}) is generated first, followed by S_1, S_2, S_3, and finally S_4.

This behavior can be attributed to the relative complexity of the carry propagation path. The output carry often involves a greater number of transistors along its path compared to the intermediate carry bits within the circuit. Consequently, the generation of S_4 is delayed, as the sum bits (S_i) are dependent on the corresponding carry bits. This sequential dependency aligns with the design and logic flow of the circuit.

Hence, it can be concluded that the worst-case delay of the adder is determined by the S_4 bit. This delay is significantly higher compared to the delays observed in other configurations, as it depends on the cumulative propagation of carry bits through the circuit. The sequential dependency of sum bits on carry bits contributes to this elevated delay for S_4. The specific addition scenario that results in the maximum delay can be intuitively identified. The worst-case delay occurs when C_4 is generated through the path containing the maximum number of transistors. In this design, that path corresponds to the inputs $A = 1001$ and $B = 1111$.

Although A and B are ideally interchangeable, the use of Pass Transistor Logic (PTL), where one input serves as the select line and the other as the data input, introduces asymmetry. This asymmetry causes $A = 1001$ and $B = 1111$ to produce the worst-case delay due to the specific configuration of signal propagation through the pass transistors.

```
tpd_rise           =  1.495891e-10 targ=  5.299589e-09 trig=  5.150000e-09
tpd_fall           =  1.682114e-10 targ=  1.531821e-08 trig=  1.515000e-08
total_prop_delay   =  1.58900e-10
```

Fig. 5: Total propagation delay obtained from NGSpice

$$t_{\text{pd}} = 792.16\,\text{ps}$$

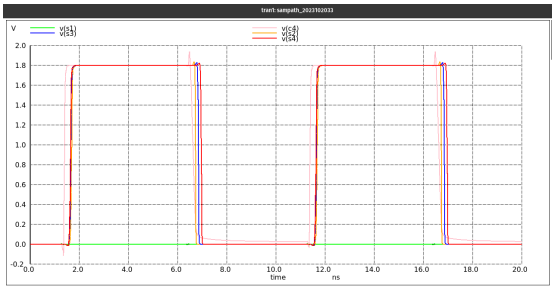

Fig. 6: NGSpice output when A = 1111 and B = 1111

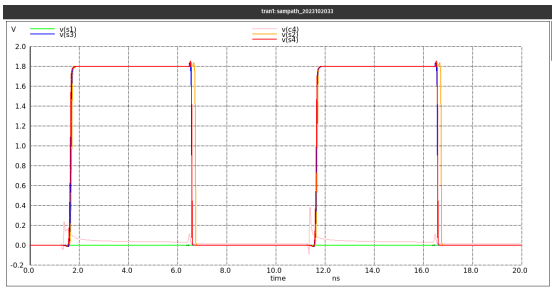

Fig. 7: NGSpice output when A=1001 B=0101

B. MAGIC Layout

For the layout of the CLA adder, the inputs are provided and immediately followed by inverters to generate their complements. The inputs and their complements are passed through pass transistors to generate the *generate*, *kill*, and *propagate* signals.

The overall design of the adder is divided into four stages, with each stage responsible for generating a corresponding carry output. These carry outputs are then routed through pass transistors to compute the sum bits. The complete layout occupies an area of $348\,\lambda \times 365\,\lambda$, which translates to $31.32\,\mu\text{m} \times 32.85\,\mu\text{m} = 1028.862\,\mu\text{m}^2$ in absolute dimensions.

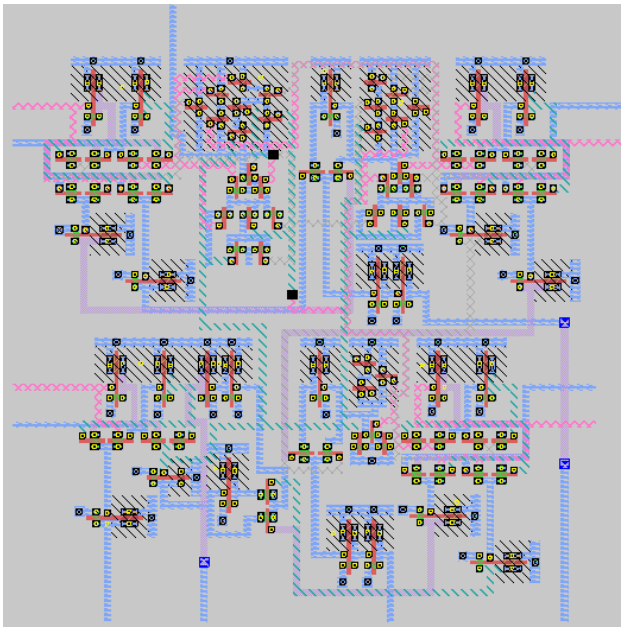

Fig. 8: MAGIC Layout

Upon running the above circuit in NGSpice from the extracted netlist, we get the following outputs.

In a similar fashion, we find the worst delay of the extracted circuit as -

$$t_{\text{pd}} = 1.096\,\text{ns}$$

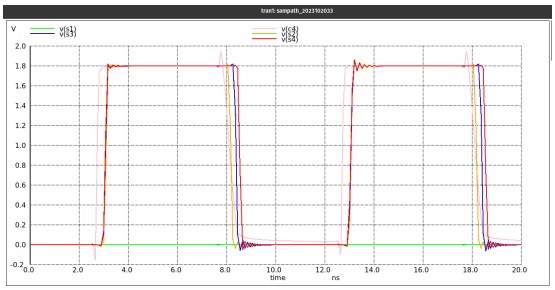

Fig. 9: MAGIC output when A = 1111 and B = 1111

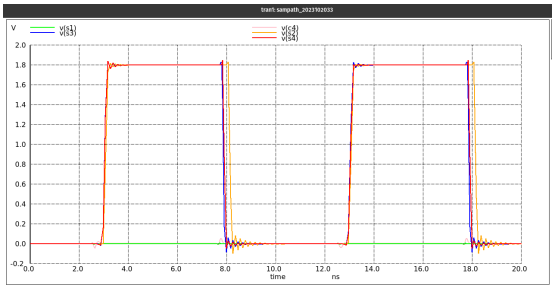

Fig. 10: MAGIC output when A=1001 B=0101

IV. Addition Module

To create a 4-bit synchronous addition module, both circuit topologies are integrated. The CLA adder receives its inputs from the outputs of the D flip-flops, and its outputs are fed into another set of flip-flops. This configuration synchronizes the adder with the clock, ensuring that the output is available at the positive edge of the immediate next clock cycle. The proposed structure utilizes a total of 250 transistors, comprising of 108 PMOS and 142 NMOS transistors.

A. NGSpice Simulations

For simplicity, ideal pulse inputs identical to each other are provided to the addition module. Along with the inputs A and B, a clock signal is supplied, operating at twice the frequency of the input bits.

From static timing analysis, we know that the minimum clock period is given by:

$$T_{\text{clk}} \geq T_{\text{pd}} + T_{\text{pcq}} + T_{\text{su}}$$

where T_{pd} is the propagation delay, T_{pcq} is the clock-to-q delay, and T_{su} is the setup time. Hence, we get the minimum clock period as

$$T_{\text{clk\_min}} = 947.26\,\text{ps} \quad F_{\text{max}} = 1.0556\,\text{GHz}$$

```
tpd_rise           =  1.173133e-09 targ=  3.723133e-09 trig=  2.550000e-09
tpd_fall           =  1.018083e-09 targ=  1.866808e-08 trig=  1.765000e-08
total_prop_delay   =  1.09561e-09
```

Fig. 11: Total propagation delay obtained from MAGIC

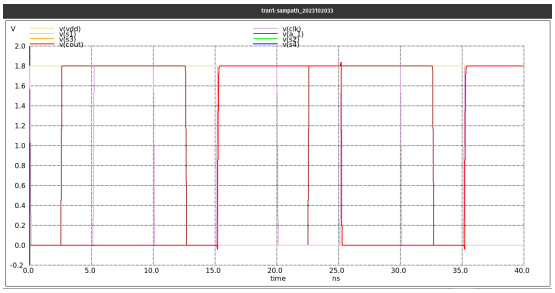

Fig. 12: NGSpice simulations when A = 1111 and B = 1111

B. MAGIC Layout

For the layout of the addition module, the inputs are provided to the D flip-flops and the outputs are obtained after the D flip-flops. This circuit occupies a total area of $605\,\lambda \times 686\,\lambda$.

From static timing analysis, we know that the minimum clock period is given by:

$$T_{\text{clk}} \geq T_{\text{pd}} + T_{\text{pcq}} + T_{\text{su}}$$

where T_{pd} is the propagation delay, T_{pcq} is the clock-to-q delay, and T_{su} is the setup time. Hence, we get the minimum clock period as

$$T_{\text{clk\_min}} = 1.293\,\text{ns} \quad F_{\text{max}} = 0.773\,\text{GHz}$$

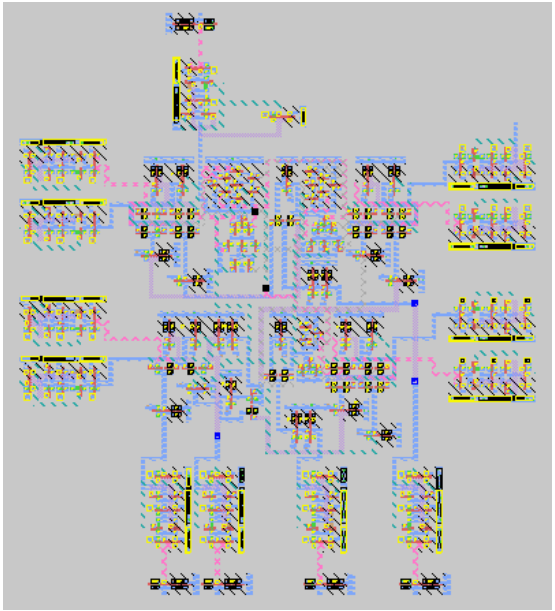

Fig. 14: MAGIC Layout of the addition module

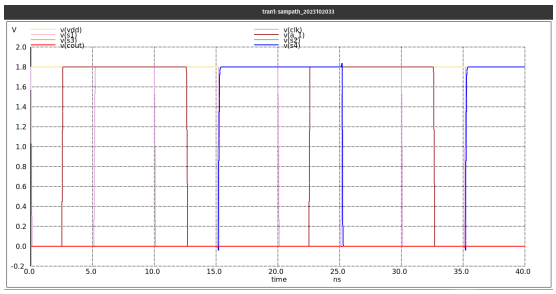

Fig. 13: NGSpice simulations when A = 1001 and B = 0101

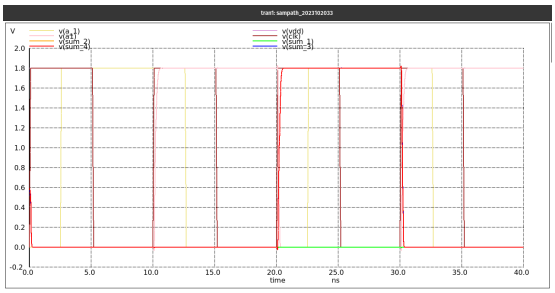

Fig. 15: NGSpice output when A=1001 B=0101

Now, coming to the transistor sizing, for the static CMOS inverter use we use -

$$W_{\mathrm{p}} = 11\lambda = 0.99\mu, \quad W_{\mathrm{n}} = 3\lambda = 0.27\mu.$$

V. Pre and Post Layout Results

Parameter	Pre-Layout	Post-Layout
t_{pcq}	137.49 ps	168.03 ps
t_{su}	17.61 ps	29.25 ps
t_{h}	87.35 ps	97.86 ps
t_{pd}	792.16 ps	1.096 ns
$T_{\mathrm{clk\_min}}$	947.26 ps	1.293 ns

TABLE I: Comparison of Pre-Layout and Post-Layout Timing Parameters.

VI. Verilog HDL and FPGA

The Verilog HDL code was further synthesized and implemented on an FPGA using Vivado. Inputs $A = 1111$ and $B = 1111$ were fed to the FPGA via input pins. The corresponding output waveforms were observed on an oscilloscope, as shown in Figs. 16–20a.

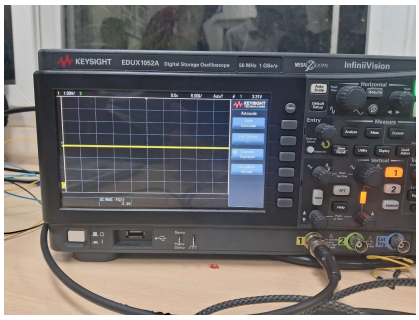

Fig. 16: C_{out}

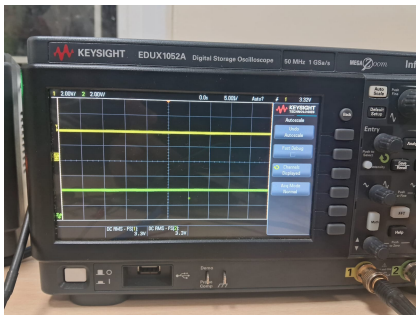

Fig. 17: S4 (Yellow) and S3 (Green)

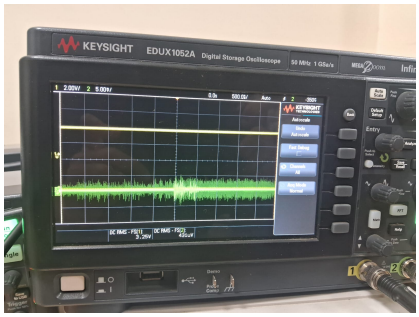

Fig. 18: S2 (Yellow) and S1 (Green)

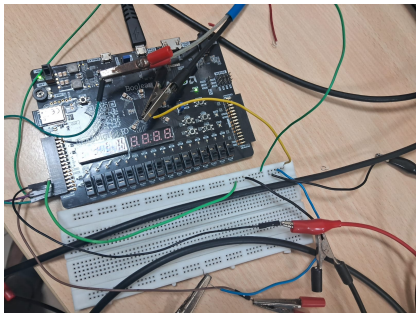

Fig. 19: FPGA board and hardware connections

(a) The entire hardware used

REFERENCES

[1] J. Miao and S. Li, "A Novel Implementation of 4-bit Carry Look-ahead Adder," *IEEE Transactions on Circuits and Systems II: Express Briefs*.