Design of Efficient Pass-transistor Binary ALU

Ziyan Qi, Xingguo Xiong, Linfeng Zhang, Md Tauseef Riasat
Department of Electrical Engineering
University of Bridgeport
Bridgeport, USA
ZiyanQi1985@gmail.com

Abstract—Pass-transistor logic (PTL) uses pass transistors to directly pass logic levels from inputs to outputs to implement different logic functions. Compared to static CMOS design, pass transistor logic implements the same logic level with much less transistor count. Thus it is more area efficient and leads to less power consumption. Pass-transistor logic has been widely used in low power VLSI design, quantum computing, nano-electronics and optical computing. Arithmetic Logic Unit (ALU) is one of key components in modern microprocessor design. In this paper, an 8-bit ALU based on pass-transistor logic is implemented in PSpice. The ALU is designed using PSpice hierarchy. The design of each individual component (such as NAND gate, full adder, multiplier, etc.) is discussed in detail. PSpice power simulation is used to verify the power consumption of the ALU design.

Keywords—Pass-transistor Logic, Low Power VLSI, Arithmetic Logic Unit (ALU), PSpice, Power Simulation.

I. INTRODUCTION

With the scaling down of transistor size, a modern VLSI circuit may integrate millions or billions of transistors in a single chip. Such high level of integration leads to increased power consumption and circuit area. Increased power consumption will reduce the battery life for portable electronics. Increased circuit area also leads to decreased fabrication yield. For more power and area efficient design, we need to exploit different logic families. Static CMOS has been the most dominant design in VLSI industry during recent decades [1]. However, it may not be the most power and area efficient design. A standard N-input CMOS circuit generally requires 2N MOS transistors (N NMOS transistors in pull-down network and N PMOS transistors in pull-up network). Early pseudo-NMOS design requires only N+1 transistors (N NMOS transistors plus one PMOS transistor which is constantly ON). However, it suffers from static leakage current when output is expected to be logic low. Furthermore, pseudo-NMOS logic is ratioed logic, i.e., the correct function relies on the proper sizing of the transistors. These limit the application of pseudo-NMOS logic. Dynamic logic (e.g. Φ-n network, Φ-p network, or domino logic) requires less number of transistors, thus leading to more area-efficient design [2]. However, the outputs of dynamic circuits are not always available. They work in pre-charge and evaluation phases, and outputs are valid only in evaluation phase. Furthermore, dynamic circuits suffer from charge leakage effect. When output is expected to be “1”, it relies on the charge previously stored in the output node capacitance to maintain its logic level to be “1”. The output node is not connected to power rail Vdd to obtain logic “1”, as CMOS circuit does. Due to the charge leakage in the output nodes, this voltage level gradually decreases and eventually cannot be correctly recognized as logic “1”. This eventually leads to logic error of the circuit. Pass-transistor logic offers the advantages of reduced transistors count, smaller circuit area and lower energy consumption [3]. It does not suffer from charge leakage problem as in dynamic logic. The outputs of PTL are valid all the time as in static CMOS. Thus pass-transistor logic can be a good choice for low power VLSI design or if area is an important concern. Pass-transistor logic has been widely used in low power VLSI design, quantum computing, nano-electronics and optical computing [4][5].

Various pass transistor logic circuits have been reported. In [6], an efficient pass-transistor logic synthesis method for sequential circuits is proposed. The proposed logic synthesis can generate both combinational and sequence circuits based on pass-transistor logic. Compared to logic synthesis using traditional CMOS cell library which typically contains hundreds of different types of cells, the proposed pass-transistor logic synthesis uses only three types of cells: 2-to-1 multiplexer (MUX), inverter, and D flip-flop (DFF). Post-layout simulation shows the proposed PTL synthesis generates results with better area, faster speed and lower power consumption compared to traditional CMOS design. In [7], a low power and high speed 8×8 bit multiplier using non-clocked pass transistor logic is reported. It is based on carry-save (CSA) multiplier architecture. Its adder is designed using NMOS pass transistors with PMOS transistors as cross-coupled devices. A combination of NMOS and PMOS transistors is sued on the mirror logic and inverters of full adder circuit. The designed multiplier demonstrates lower power consumption and faster speed. In [8], the design of a low power 54×54 bit multiplier based on pass-transistor logic is reported. Compared to traditional pass-transistor logic, it utilizes regenerative feedback with PMOS switches to further reduce the power consumption and propagation delay. The multiplier uses high compression-rate compressors without Booth Encoding, as well as a 108-bit conditional summer adder with separated carry generation block. In [9], a low-cost design of serial-parallel multipliers using hybrid pass-transistor logic and CMOS logic is reported. In it, pass-transistor logic (PTL)-
based D flip-flop and T flip-flop are used in finite field multiplication. It compares the power consumption of serial-parallel finite field multiplication using pure CMOS, pure PTL, and hybrid PTL/CMOS logic.

Arithmetic Logic Unit (ALU) performs arithmetic and logical operations. The ALU is a fundamental building block of the CPU (central processing unit) of a computer [10]. Various ALU designs based on pass-transistor logic have been reported. In [11], the design of a pass-transistor logic 4-bit ALU is proposed. Its schematic and layout are designed and simulated in Tanner EDA tool. The Elmore delay model is used to estimate the delay of the ALU, and its value is compared to the simulation result. In [12], the design of a low power ALU using 8-transistor full adder and PTL-based multiplexer (MUX) circuits is reported. Simulation result shows that the power and area of the designed ALU are greatly reduced to more than 70% compared to the existing methods. In [13], a super low power 8-bit CPU with pass-transistor logic is reported. It is based on an original pass-transistor logic family with single-rail pass-transistor logic (SPL) and Single-rail Pass-transistors and Holders logic (SPHL). It is shown to save 74% power of that of the commercial CMOS Z80 CPU cores using the same design rules. In this paper, an 8-bit ALU based on pass-transistor logic is implemented in PSPICE. The design of each individual component is discussed in detail. In order to overcome the threshold drop of pass transistors, CMOS inverter is used as a buffer to recover the degraded logic levels to full power rails. PSPICE simulation is used to verify the correct function of the ALU design. The power consumption of the designed ALU is extracted with SPICE power simulation.

II. DESIGN OF INDIVIDUAL COMPONENTS

Considering the complexity of the ALU, PSPICE hierarchy design is used to design the complete 8-bit ALU based on pass-transistor logic. We first design individual components (e.g. NAND gate, full adder, multiplier, etc.) of the ALU using pass transistor logic. Once the correct function of the component is verified with PSPICE simulation, the component is defined as a block. In higher hierarchy level, the component can be directly recalled as a block without redesigning the component circuitry again. This saves time and effort of designers. It leads to improved efficiency in large circuit design with repeating components. In the PSPICE design of the ALU components, by default, the size of NMOS transistors is set to be $W_n = 3 \mu m$, $L_n = 2 \mu m$. For symmetrical timing response (equal rising and falling delay), the size of PMOS transistors is set to be $W_p = 9 \mu m$, $L_p = 2 \mu m$. We use MbreakN4 model (4-terminal device) for NMOS transistors, and use MbreakP4 (4-terminal device) model for all PMOS transistors. The bulk of all PMOS transistors is connected to $V_{dd}$, and the bulk of all NMOS transistors is connected to Gnd to avoid leakage current between device and substrate.

2.1. PTL OR Gate

PSPICE design of a PTL (pass-transistor logic) OR gate is shown in Figure 1. It consists of one PMOS transistor and one NMOS transistor to pass either input A or B to output. The gate input of both transistors is connected to input B. The output can be derived as

\[
\text{Out} = A \cdot \bar{B} + B \cdot \bar{A} = A \cdot \bar{B} + B
\]

\[
= A \cdot \bar{B} + B \cdot (1 + A) = A \cdot (\bar{B} + B) + B
\]

\[
= A \cdot 1 + B = A + B
\]

This proves the output does implement the function of an OR gate. To verify the function of the designed OR gate, piecewise-linear voltage ($V_{PWL}$) sources are applied to inputs. The simulated waveforms are shown in Figure 2. As we can see from Figure 1, when input $AB=00$, output is “0” (logic low). When input $AB=01$, 10, or 11, output becomes “1” (logic high). This verifies the correct function of the designed PTL OR gate.

![Fig. 1. PTL OR gate circuit](image1)

![Fig. 2. PSPICE simulation of PTL OR gate](image2)

However, as shown in the waveform, when output is expected to be “1”, it is not exactly 5V (i.e. $V_{dd}$). Instead, it is around 4.2V. This is due to the threshold drop of pass transistors. As we know, MOS transistor needs to maintain its $V_{GS}$ Voltage drop to be larger than threshold voltage in order for the transistor to be turned ON. In order to maintain the threshold drop, PMOS transistor can pass a strong “1” but a weak “0”; NMOS transistor can pass a strong “0” but a weak “1”. This indicates that voltage passing through pass transistors will have level degradation issue. If it is only a single PTL gate, this may not be a big problem. However, if multiple PTL logic gates are cascaded to each other, this level degradation may become even worse and eventually cause logic error. Thus certain action needs to be taken to overcome this level degradation problem of PTL logic. Several solutions can be used to overcome this issue. For example, one can insert level restorer circuit to restore the logic level whenever the voltage drops too much. Level restorer uses a PMOS or NMOS transistor to connect the node to either Vdd or Gnd. When the voltage level is degraded, the corresponding PMOS or NMOS switch will be turned ON, hence connecting the node to either Vdd or Gnd power source to obtain a strong logic level. Alternately, one can use CMOS transmission gate (parallel connection of NMOS and PMOS transistors) to replace pass transistor. But this doubles the transistor count, so it’s not very area efficient. Another solution is to insert CMOS buffers...
because CMOS logic can recover logic level to full power rails. When output of CMOS logic is logic “1” or “0”, it is connected to either Vdd or Gnd, hence achieving strong rail-to-rail voltage levels. Adding CMOS buffers will introduce extra gates, hence leading to power and area overhead. However, this ensures the correct function of the PTL circuit. Thus it is still worthy. In this research, we add CMOS inverters as buffer to recover the logic levels of PTL logic. It effectively overcomes the level degradation problem of PTL circuit.

2.2 PTL NOR Gate

In order to overcome the level degradation issue of PTL OR gate, we add a CMOS inverter as a buffer to the output of PTL OR gate. This makes it into a PTL NOR gate. The design is shown in Fig. 3. As shown in Fig. 3, a CMOS inverter is added to the output of previously designed PTL OR gate. This changes the function into a PTL NOR gate. The transistor count of the PTL NOR gate is 4, which is the same as a CMOS NOR gate. PSPICE simulation is used to verify the correct function of the PTL NOR gate, as shown in Fig. 4. As we can see in Fig. 4, when both inputs AB=00, output is “1”. When inputs are AB=01, 10, 11, the output is “0”. This proves the correct function of NOR gate. Furthermore, from the PSPICE simulation we can see that the output of PTL NOR gate achieve full power rails (strong “1” and strong “0”). No level degradation due to threshold drop of pass transistor is observed. This verifies that the CMOS inverter introduced has well recovered the logic levels of pass-transistor logic.

2.3. PTL NAND Gate

Similarly a PTL NAND gate can be designed. It is simply a PTL AND gate connected to a CMOS inverter. The PSPICE design of the PTL NAND gate is shown in Fig. 5. The PSPICE simulation of the input/output waveforms of the designed PTL NAND gate are shown in Fig. 6. As shown in the figure, when both inputs are AB=00 or 11, output is “0”. However, if inputs are AB=01 or 10, output is “1”. This verifies the correct function of a NAND gate. Furthermore, the output waveform does not show any level degradation. The logic “1” is full Vdd (5V) and logic “0” is exactly 0V. This verifies that by adding the CMOS inverter as a buffer, the level degradation issue due to the threshold drop of pass transistors is overcome. The designed NAND gate can be used for the PTL ALU design. If we need to design a PTL AND gate, we simply add another CMOS inverter to the PTL NAND gate. Based on NOR/OR, NAND/AND and inverters, any complex logic function can be implemented.

2.4. PTL XOR Gate:

XOR gate is wildly used in half adder, subtractor and multiplier. The design of a PTL XOR gate is shown in Fig. 7. It consists of only four transistors: two NMOS transistors and two PMOS transistors. This is much simpler than the CMOS XOR gate design. PSPICE simulation of the PTL XOR gate is shown in Fig. 8. As seen from the figure, when both inputs are AB=00 or 11, output is “0”. However, if inputs are AB=01 or 10, output is “1”. This verifies the correct function of the XOR gate. A slight level degradation is also observed for the third pattern. However, this level degradation is very small and it won’t cause serious problem for the circuit. If we want to recover the logic level, we can again use a CMOS inverter to connect to the output of PTL XOR gate. This will convert the PTL XOR gate into a PTL XNOR gate.

Fig. 3. PSPICE design of PTL NOR gate

Fig. 4. PSPICE simulation waveform for PTL NOR gate

Fig. 5. PSPICE design of PTL NAND gate

Fig. 6. PSPICE simulation of PTL NAND gate

Fig. 7. PTL XOR gate circuit
2.5. PTL Full Adder

As important part to establish adder/subtractor and multiplier, full adder is also constructed using pass-transistor logic. Fig. 9 shows the design of pass-transistor Full Adder circuit. As shown in the figure, two CMOS inverters are connected to Sum and Cout outputs of the PTL full adder. They are used as buffers to recover the logic levels of PTL full adder. This ensures that there is no voltage threshold drop for the PTL full adder. This is important because for 16-bit adder design in the ALU circuit, we need to cascade 16 full adder circuits in series. If not corrected, voltage degradation in PTL full adders can be added up and eventually causes logic error of the circuit. The PTL full adder is also simulated with PSPICE to verify the correct function of the design. Simulation result shows that the threshold drop of pass transistor logic has been corrected by the extra four CMOS inverters added to the circuit. The overall PTL full adder design contains 18 transistors.

2.6. PTL MUX

MUX is an important component to signal flow of ALU. The PSPICE design of a PTL 2-to-1 MUX is shown in Fig. 11. The PSPICE simulation waveforms of the 2-to-1 MUX is shown in Fig. 12. From the simulation, we can see that input A=5V, input B=0V. When selection signal S=”0”, input A=5V is passed to output. When selection signal S=”1”, input B=0V is passed to output. This verifies the correct function of the PTL MUX design.

III. ALU DESIGN

Once the individual components of PTL logic circuits have been designed, they are defined as blocks. In higher hierarchy level, these blocks can be directly recalled when needed without being re-implemented again. This greatly improve the efficiency of the 8-bit ALU design.

3.1. PTL 8-bit Adder/Subtractor Design

There are many discussions about the theory of 8-bit Adder. Eight-bit full adder can be designed as ripple-carry full adder, i.e., the serial connection of 8 1-bit full adders. The carry-out from previous adder is used as carry-in for the next adder. This is the slowest but the basic design of 8-bit full adder. For binary subtract operation, it can be implemented with adder function because subtracting a number is equivalent to adding its negative value. So we can combine Adder with Subtractor together. To produce negative counterpart, we need to invert each bit of subtrahend and add it together. A negative number is expressed as its two’s complement. To achieved the inversion of bits we use XOR gates with one fixed logic-low input. With one normally-low input, XOR gate acts as an inverter. If we couple this inversion with a bit sent to the carry-in of the full adder, a subtraction operation is achieved. When we control input to low, the entire diagram will work as an Adder. The design of 8-bit Adder/Subtractor is shown in Fig. 12.
3.2 PTL 8-bit Multiplier Design

Normally multiplier works in shift-add theory. Array multiplier is used in this design. The entire design is shown in Fig. 13.

3.3 PTL 8-bit ALU Design

Arithmetic Logic Unit (ALU) is an important part of computer processor (CPU). The function of ALU is to carry out logic and arithmetic operations on the operands in computer instruction words. In this design, ALU carries operations of addition, subtraction, multiplication of integers and bitwise AND, NAND, OR, NOR and XOR functions.

In this work, we implement eight function modes for the ALU. They are AND, NAND, OR, NOR, XOR, Addition, Subtraction and Multiplication. Among these functions, addition and subtraction can be achieved by one block—Adder/Subtractor. So we just need a 16-bit 7-to-1 MUX to select the outputs. It will select one out of seven 16-bit data to be passed to the outputs depending on the working mode. The PSPICE design of the 16-bit 7-to-1 MUX circuit is shown in Fig. 14.

IV. RESULTS AND DISCUSSION

After the complete PTL-based 8-bit ALU is designed in PSPICE, we used some random patterns to test the function of the ALU circuit. We verified all the 8 functions: AND, NAND, OR, NOR, XOR, Addition, Subtraction and Multiplication. Simulation results prove the correct function of the ALU design. As the goal of this research is to design a low power ALU based on pass transistor logic, we are interested in the power consumption of the ALU circuit. In order to find out the power consumption of the ALU design, PSPICE power simulation [14] is used to extract its power. We designed some random pattern sequence for the ALU circuit. The average power of the ALU circuit for the given input pattern sequence is extracted by SPICE power simulation.

In SPICE power simulation [14], a power measurement circuit which consists of a current controlled current source, a capacitor and a resistor connected in parallel. The total current flowing into the ALU circuit is used to control the current controlled current source. The current from current-controlled current source is used to charge the capacitor. By reading the voltage across the capacitor, it is numerically equal to the average power consumption of the circuit-under-test (CUT).
The simulated power curve (V(Pav), which is the voltage across the capacitor) is shown in Fig. 16.

![Fig. 16. PSPICE power simulation of PTL-based 8-bit ALU](image)

The average power consumption $P_{avg}(0\rightarrow t)$ during any time period $0\rightarrow t$ can be calculated from following equation (in value, not in unit):

$$P_{avg}(0 \rightarrow t) = \frac{V(Pav)}{t} \times T$$

where $T$ is the time period for which the measured voltage across capacitor numerically equals to the average power of the CUT. It is also used to set the coefficient of current-controlled current source $K = \frac{V_{dd} \cdot C}{T}$

$$K = \frac{V_{dd} \cdot C}{T}$$

In it, $C$ is the capacitor in power measurement circuitry. The total energy $E(0\rightarrow t)$ consumed during time period $0\rightarrow t$ can be calculated as

$$E(0 \rightarrow t) = \frac{V(Pav)}{t} \times T$$

In our case, we set $T=200ns$. From the PSPICE power simulation waveform in Fig. 16, we can read that for $t=200ns$, $V(Pav)=52.82mV$. Thus the average power of the 8-bit PTL ALU for $t=0\rightarrow 200ns$ is found to be $P_{avg}=52.282mW$. From Equation (4), the total energy consumption during $t=0\rightarrow 200ns$ is found to be

$$E(t=0 \rightarrow 200ns) = 52.82 \times 10^{-3} \times 200 \times 10^{-9} = 1.046 \times 10^{-3}J=10.46nJ$$

V. CONCLUSION AND FUTURE WORK

In this paper, the PSpice design and simulation of a PTL (pass transistor logic) 8-bit ALU circuit is reported. CMOS buffers are used to overcome the voltage level degradation problem in pass transistor logic. PSPICE power simulation is used to extract the power consumption of the PTL 8-bit ALU circuit in response to given test pattern sequence. The average power is found to be 52.282mW for the given input pattern sequence. From the result we can see that the power consumption of the ALU circuit is greatly reduced due to the pass-transistor logic design. The major reason that PTL ALU design can save power is because PTL design generally can lead to much less transistor count compared to traditional CMOS design. The total transistor count of the pass-transistor logic-based ALU is 1640. This includes 32 transistors for 8-bit NAND gate, 16 transistors for 8-bit AND gate, 16 transistors for 8-bits OR gate, 16 transistors for 8-bits NOR gate, 32 transistors for XOR gate, 180 transistors for 8-bit Adder/Subtractor, 1156 transistors for 8-bit Multiplier and 192 transistors for 16-bits 7-to-1 MUX. Compared with conventional ALU, which need 2522 transistors, PTL ALU can save almost 35% transistor count. Less transistor count means smaller circuit area and lower power consumption. On the other hand, using pass-transistor logic can reduce the power consumption for a large extent, when MOS transistor switches their states. When output is expected to be logic “1” or “0”, it is not connected to Vdd or Gnd. Instead, the output node is connected to one of the inputs to get logic “1” or “0”. This also leads to reduced power consumption.

In the future, we will further improve the performance of the PTL ALU. For example, we will build more functions into the ALU circuit and improve its speed. Furthermore, currently we insert CMOS buffers to overcome the level degradation issue. But this leads to extra overhead in circuit area and delay. We will try to find out some other more efficient ways to recover logic levels without inducing large area and delay overhead.

REFERENCES