DESIGN AND SIMULATION OF 1 BIT ARITHMETIC LOGIC UNIT DESIGN USING PASS-TRANSISTOR LOGIC FAMILIES

V. Venkata Rao¹, T. Indira², Ch. Krishna Priya³, P. Vasanthi⁴
¹Professor and Head of the Department of ECE
²,³,⁴Assistant Professor,
Narasaraopeta Engineering College
Narasaraopet, Guntur (Dist.), A.P. India
¹vvenkatarao2k9tec@gmail.com

June 21, 2018

Abstract

In this paper a 1 bit Arithmetic Logic Unit has been designed and implemented with a combinational logic circuits containing a number of functional components for different arithmetic and logic operations using Transmission Gate Logic (TGL) and Dual Value Logic (DVL). The performance of developed Arithmetic Logic Unit Design has been analyzed and compared in terms of area and power using TSMC CMOS 90nm, 70nm and 50nm technology. The schematic circuit of 1 bit Arithmetic Logic Unit has been designed using DSCH tool and its equivalent layout has been created using Microwind tool. It can be observed from simulation results that 90nm, 70nm and 50nm technology based Arithmetic Logic Unit Design using Dual Value Logic has shown reduced transistor count, area and Power compare to the Arithmetic Logic Unit Design using CPL, DPL & TGL.
1 INTRODUCTION

Digital circuits design combinational circuits that implements a function of its inputs based on either arithmetic or logic functions consists of logic gates implemented in the Complementary Metal Oxide Semiconductor (CMOS) technology. In Digital circuits design mainly three parameters power, delay and area are considered. The dynamic power is consumed only when the circuit performs a function and signals change zero to one and one to zero. The circuit Leakage or static power is consumed all the time, i.e., even when the circuit is idle. The dynamic power cannot be eliminated completely because it is caused by the computing activity. It can, however, be reduced by circuit design techniques. Whenever a logic gate changes state, power is consumed. The solution is then realized at the transistor level designs. In this work, a 1-bit ALU is designed at transistor level for low power and minimum area. The rest of the paper is organized as follows; Section II will discuss about the Pass-Transistor Logic Families. Section III will discuss on the overview of Arithmetic Logic Unit. Section IV will discuss on Arithmetic Logic Unit Using TGL and DVL. Section V conducts a comparative simulation study among the Arithmetic Logic Unit Using TGL and DVL, with a detailed discussion on the derived results. Section VI provides the summary and final conclusions of the work presented.

2 PASS-TRANSISTOR LOGIC FAMILIES

In Digital electronics circuits design using the Pass Transistor Logic (PTL) describes several logic families used in the design of integrated circuits. By using Pass Transistor Logic design the transistors count can be reduced compared to different logic gates, by eliminating redundant transistors. There are mainly two...
design types of pass-transistor circuit styles: One is the pass-transistor circuits design which uses only NMOS pass-transistor called Complementary Pass-transistor Logic (CPL) and second one is pass-transistor circuits design which uses both NMOS and PMOS transistors which can be used as Double Pass-transistor Logic (DPL), Transmission Gate Logic (TGL) and Dual value logic (DVL).

A. Complementary pass-transistor logic
Complementary pass-transistor logic circuit (CPL) [1] consists of complementary inputs/outputs, a NMOS pass-transistor network, and it can be used to connect inverter on output side using CMOS as shown in Fig 2.1(a). The circuit function is implemented as a tree consisting of pull-down and pull-up branches. Since the threshold voltage drop of NMOS transistor degrades the high level of pass-transistor output nodes, the output signals are restored by CMOS inverters. CPL has traditionally been applied to the arithmetic building blocks and has been shown to result in high-speed operation due to its low input capacitance and reduced transistor count. Design of two input AND/NAND logic using CPL and OR/ NOR logic using CPL as shown in Fig 2.1(b) and (c).

![Diagram](image)

Figure 2.1: (a) Complementary pass-transistor logic circuit. (b) AND/NAND logic using CPL (c) OR/ NOR logic using CPL

B. Double pass-transistor logic
To avoid problems of reduced noise margins in CPL, twin PMOS transistor branches are added to N-tree in DPL [2]. This addition results in increased input capacitances. However its symmetrical arrangement and double-transmission characteristics compensate
for the speed degradation arising from increased loading. The full swing operation improves circuit performance at reduced supply voltage with limited threshold voltage scaling. Design of two input AND & NAND logic using DPL as shown in Fig 2.2(a) and (b).

Figure 2.2: (a) AND logic using DPL. (b) NAND logic using CPL

C. Transmission Gate Logic
The Transmission Gate Logic design [3] using CMOS gate to appreciate advanced logic functions employing a PMOS and NMOS that is called complementary transistors. Transmission gate has a switch with low resistance and capacitance having ratio less logic. Also, DC characteristic of this gate is independent of input levels. It is designed by NMOS and PMOS transistors connecting the source to source and drain to drain terminals. Because the NMOS transistor is switching strong signal 1 and PMOS transistors switching strong signal 0 each transistor is flipped on-off by the enable signals, then input pass towards the output, the symbols shown in Fig 2.3(a) and truth table of Transmission Gate Logic as shown in Truth Table 1. The below Transmission Gate Logic voltage is applied on X node is a HIGH, the complementary Logic zero is applied to LOW on . The two transistors (NMOS and PMOS) conduct and pass the signal from INPUT to OUTPUT. The AND gate and OR gate using TGL are shown in Fig. 2.3(b) and (c). The 2-input TGL AND/OR gates are full-swinging, but not restoring for all input combinations.

D. Dual value logic
The main drawback of DPL is its redundancy, i.e. it requires more transistors than actually needed for the realization of a function. To overcome the problem of redundancy, a new logic family, DVL [4], is derived from DPL. It preserves the full swing operation of DPL with reduced transistor count. As introduced in DVL circuit
Figure 2.3: (a) Transmission Gate (b) TGL AND gate (c) TGL OR gate

Table 1: TGL Table

<table>
<thead>
<tr>
<th></th>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>X (don’t care)</td>
<td>Z (high impedance)</td>
</tr>
</tbody>
</table>
can be derived from DPL and TGL circuits in three steps, consisting of: Elimination of redundant branches, Signal rearrangement (resize), Selection of the faster halves.

The AND gate and OR gate using DVL as shown in Fig. 2.4(a) and (b). The 2-input DVL AND/OR gates are full-swinging but non-restoring, as well. The style we consider in this work is DVL, which preserves the full swing operation of DPL and TGL with reduced transistor count.

Figure 2.4: (a) DVL AND gate. (b) DVL OR gate.

3 OVERVIEW OF ARITHMETIC LOGIC UNIT

An arithmetic-logic unit (ALU) is the part of a computer processor (CPU) that carries out arithmetic and logic operations on the operands in computer instruction words. In some processors, the ALU is divided into two units, an arithmetic unit (AU) and a logic unit (LU). In an arithmetic-logic unit (ALU) design different blockssuch as decoder, full adder, logical circuits and summing circuit are combined as shown in Fig. 3.1. In the arithmetic unit, multiplication and division are done by a series of adding or subtracting and shifting operations. In the logic unit, OR, AND, XOR and XNOR operations are designed.
4 1 Bit ARITHMETIC LOGIC UNIT using TGL and DVL

A. 1 Bit Arithmetic Logic Unit using Transmission gate logic
In this method 1 bit Arithmetic-Logic Unit is designed using Transmission Gate Logic as shown in Fig.4.1.

B. Arithmetic Logic Unit using Dual value logic
In this method 1 bit Arithmetic-Logic Unit is designed using Dual value logic as shown in Fig.4.2.

1 Bit Arithmetic Logic Unit using Transmission Gate Logic and Dual Value Logic that generates output depending on three control inputs are \( F_0 \), \( F_1 \) & \( F_2 \) and 2 input variables \( A \) and \( B \) & carry in. Its operation is summarized in Table II.

- If control inputs are \( F_0 \), \( F_1 \) & \( F_2 \) is “000” condition, when \( A \) & \( B \) inputs receive binary numbers and Cin is zero, then OR logic operation is performed.

- If control inputs are \( F_0 \), \( F_1 \) & \( F_2 \) is “001” condition, when \( A \) & \( B \) inputs receive binary numbers and Cin is zero, then B
Figure 4.1: 1 Bit Arithmetic Logic Unit using Transmission gate logic.
Figure 4.2: 1 Bit Arithmetic Logic Unit using Dual value logic
input are complemented operation is performed.

- If control inputs are F0, F1 & F2 is “010” condition, when A & B inputs receive binary numbers and Cin is zero, then multiplication operation is performed.

- If control inputs are F0, F1 & F2 is “011” condition, when A & B inputs receive binary numbers and Cin is zero, an addition with carry operation is performed.

- If control inputs are F0, F1 & F2 is “100” condition, when A & B inputs receive binary numbers and Cin is zero, then XOR logic operation is performed.

- If control inputs are F0, F1 & F2 is “101” condition, when A & B inputs receive binary numbers and Cin is zero, then XNOR logic operation is performed.

- If control inputs are F0, F1 & F2 is “110” condition, when A & B inputs receive binary numbers and Cin is zero, then Logic ‘0’ operation is performed.

- If control inputs are F0, F1 & F2 is “111” condition, when A & B inputs receive binary numbers and Cin is zero, then Logic ‘1’ operation is performed.

<table>
<thead>
<tr>
<th>CONTROL INPUTS</th>
<th>ALU FUNCTION</th>
<th>CARRY OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2 F1 F0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td>A OR B</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>B'</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>AB</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>A+B</td>
<td>Carry from A+B</td>
</tr>
<tr>
<td>1 0 0</td>
<td>A XOR B</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>A XNOR B</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Logic '0'</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Logic '1'</td>
<td>0</td>
</tr>
</tbody>
</table>
5 RESULTS AND DISCUSSIONS

In this paper, the 1 Bit ALU design using TGL and DVL schematic circuits are drawn and simulated output waveforms are generated. The power results of 1 Bit ALU design using TGL and DVL are tabulated. We perform the simulations using Microwind tool in 90nm, 70nm & 50nm technology and observed that in 1 Bit ALU design using DVL circuit the power and area are reduced compared to the 1 Bit ALU design using TGL in different technologies. The corresponding waveforms are shown below.

Figure 5.1: Output wave form of 1 Bit ALU using TGL using 90 nm CMOS Technology.

6 CONCLUSIONS

As per the analysis the power consumption is minimum with the channel length of 50 nm for the TSMC model compared to that of TSMC model with channel length 90 nm and 70nm. 1 Bit ALU design using DVL circuit reduced the power and area compared to the 1 Bit ALU design using TGL in different technologies.
Figure 5.2: Output wave form of 1 Bit ALU using TGL using 70 nm CMOS Technology.

Figure 5.3: Output wave form of 1 Bit ALU using TGL using 50 nm CMOS Technology.
Figure 5.4: Output wave form of 1 Bit ALU using DVL using 90 nm CMOS Technology.

Figure 5.5: Output wave form of 1 Bit ALU using DVL using 70 nm CMOS Technology.
Figure 5.6: Output wave form of 1 Bit ALU using DVL using 50 nm CMOS Technology.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>90nm technology</th>
<th>70nm technology</th>
<th>50nm technology</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALU design using TGL</td>
<td>ALU design using TGL</td>
<td>ALU design using DVL</td>
</tr>
<tr>
<td>Area</td>
<td>7371.1μm²</td>
<td>5949.2μm²</td>
<td>4717.5μm²</td>
</tr>
<tr>
<td>Power</td>
<td>0.284mW</td>
<td>0.218mW</td>
<td>26.2μW</td>
</tr>
</tbody>
</table>
References


