SPICE MODELING OF ALU

Saeid Moslehpour, Ph.D., Srikrishna Karatalpu
Department of Electrical & Computer Engineering
University of Hartford

Abstract

The microprocessor, also known as the Central Processing Unit (CPU), is the most essential part of a computer, and is a complete computation engine that is fabricated on a single chip. An Arithmetic and Logic Unit (ALU) is the heart of all microprocessors. The ALU is a combinational circuit that performs a number of different arithmetic and logical operations. In this changing world of electronics, the demand for faster and more compact ALUs is growing by the minute. The complexity and compact size of the ALU make it even more critical from the design point of view to test it in conjunction with pre-design parts before manufacturing. This paper is an attempt to achieve the above objective using CAD and SPICE simulation software. The main purpose of the project is to realize a method for importing a layout drawn in Tanner L-edit, simulated in T-Spice into PSpice referred, to as "*software talking*". To do so, an eight-instruction set CMOS ALU is laid out in Tanner L-edit and the extracted net-list is then simulated in T-Spice. An ALU equivalent design is then modeled in PSpice for further testing with pre-manufactured parts of the PSpice library. This project helps students understand the construction of an ALU.

Introduction

Students have difficulty in understanding and comprehending the construction of an ALU. We have designed a project to help students comprehend the construction of an ALU from gate down to the semiconductor level.

In this ALU design a top-down approach has been used and is clearly explained with the help of a block diagram in Figure 1. The top-level module consists of a four-bit ALU essentially performing eight important functions. The eight functions performed are *ADDITION, SUBTRACTION, AND, NAND, OR, NOR, XOR* and *XNOR.* Each of the above functions is performed on two four-bit inputs. The functions performed are bitwise operations. The bitwise output is then multiplexed out using an 8X1 multiplexer. Each of the single-bit building blocks are cascaded together to form a four-bit ALU [1].

The eight-instruction set of the ALU is briefly described below.

1. ***ADDITION:*** This function is performed by a Ripple-Adder. The output of the adder consists of four sum bits and a single carry-out bit.

2. ***SUBTRACTION:*** This function is performed by a Ripple-Adder. The second input is complimented using two's complement. The output of the adder consists of four sum bits and a single carry-out bit.

3. ***NAND***: The NAND gate is a universal gate. The output of a NAND gate is high if any one of the inputs is low. The NAND gate can be used to construct a number of logic operations.

4. ***AND*****:** The AND gate is a basic gate. The output of a AND gate is high only if both the inputs are high (i.e. the output of a AND gate is high when both the inputs are high). The AND gate can be constructed in CMOS by inverting the NAND gate.

5. ***NOR:*** The NOR gate is a universal gate. The output of a NOR gate is low if any one of the inputs is high (i.e. the output of a NOR gate is high when both the inputs

are low). The NOR gate can be used to construct various logic operations.

6. ***OR:*** The OR gate is a universal gate. The output of an OR gate is low. Only if both of the inputs are low (i.e. the output of an OR gate is high any one of the inputs are high). The OR gate can be constructed in CMOS by inverting the NOR gate.

7. **XOR:** The XOR gate can be constructed a combination of NAND and NOR gates. The output of the XOR gate is low when inputs are the same (0,0) or (1,1).

8. **XNOR:** This is constructed by inverting the XOR gate.

Each of the functions is performed on single-bit inputs, i.e., the functions performed are bitwise operations. The design is laid out in Tanner L-edit using **0.6 micron technology**. The layout is then extracted using an **ML12_5 model file**. The extracted net-list is then simulated using T-Spice.

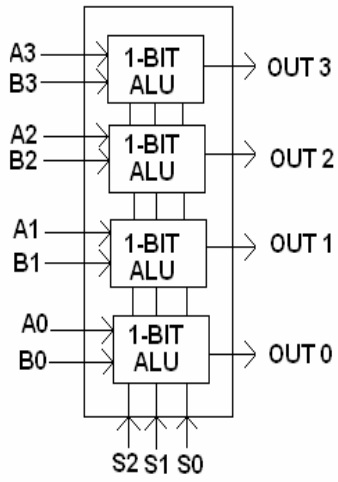

Figure 1: Four-Bit ALU Block Diagram.

The device is then modeled in Orcad PSpice. Spice A/D is a simulation program that models the behavior of a circuit containing any mix of analog and digital devices. Used with Orcad Capture for design entry, PSpice A/D can be imagined as a software-based breadboard circuit that can be used to test and refine design before ever touching a piece of hardware[2]. The PSpice modeling involves writing an equivalent code in Verilog used by the PSpice editor in modeling different electronic devices. The resultant propagation delay of the ALU has been modeled.

CMOS Design Methodology

Inverter

The inverter consists of an NMOS and a PMOS switch connected in series. The PSWITCH is connected from the VDD to the output and input for conduction when the input voltage is low. The NSWITCH is connected from the GND to the output and the input for conduction when the input voltage is high.

NAND Gate

The CMOS NAND gate is derived examining the K-MAP. The "0" dictates the AND structure, constructed using two NMOS in parallel. The "1" dictates the OR structure, constructed using two P-MOS connected in Series.

AND Gate

The CMOS AND gate is designed by inverting the CMOS NAND gate. The output of AND is high only when both the inputs are high.

NOR Gate

The CMOS NOR gate is derived examining the K-MAP. The "0" dictates the AND structure, constructed using two NMOS in parallel. The "1" dictates the OR structure, constructed using two P-MOS connected in series.

OR Gate

The CMOS OR gate has been designed by inverting the CMOS NOR gate. The output of

OR is high only when both or any one of the inputs is high.

XOR Gate

The Exclusive-OR, or XOR function can be described verbally as, "Either A or B, but not both." The output of an XOR gate is high only when any one of the inputs is low. A XOR gate can be designed by a combination of NAND and OR gates.

XNOR Gate

The gate is designed by inverting the XOR gate .The output of the XNOR gate is high when both the inputs are the same (0,0) or (1,1).

Full-Adder

The full-adder circuit adds three one-bit binary numbers (C, A and B) and outputs two one-bit binary numbers, a sum (S) and a carry (C).The full-adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. binary numbers. The output of XOR gate is called SUM, while the output of the AND gate is the CARRY. The AND gate produces a high output only when both inputs are high. The XOR gate produces a high output if either input, but not both, is high. The ‘”C” input for an ADDER is always made low.

Subtractor

Binary subtraction is performed by adding the two’s complement of the number to be subtracted. Two’s complement of a number can be achieved by inverting the number and adding one to it. This is achieved by inverting each bit of the number to be subtracted and adding “1” by means of the carry-in. The carry-in of a Subtractor must be “1.”

Multiplexer

A multiplexer is a combinatorial circuit that is given a certain number (usually a power of two) of *data inputs*; let us say 2^n, and n *address inputs* used as a binary number to select one of the data inputs. The multiplexer has a single output, which has the same value as the selected data input. The present design is an 8x1. The Multiplexer has been used for each single bit out depending on which input is needed to be sent out. This also ensures a faster ALU as the functions are performed as soon as the input is fed to the ALU.

The 3 control signals (S2, S1, S0) determine the desired output as shown below:

000→ADDITION
001→AND
010→NAND
011→OR
100→NOR
101→XOR
110→XNOR
111→SUBTRACTION

Four-Bit ALU

The single-bit ALUs are cascaded together to form a four-bit ALU shown in Figure 2.

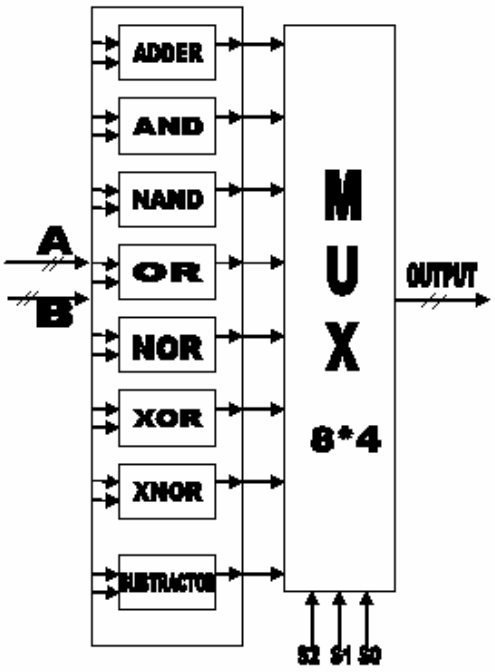

Figure 2. Four-bit ALU block diagram.

PSpice Design Methodology

OrCAD personal productivity tools have a long history of addressing demands. The powerful, tightly integrated PCB design suites include design capture, librarian tools, a PCB editor, an auto/interactive router, and an optional analog and mixed-signal simulator. The affordable, high-performance OrCAD product line is easily scalable with the full complement of Cadence Allegro PCB solutions. The OrCAD product line is owned by Cadence Design Systems, Inc.

Modeling Primitives

Primitives are primarily used in sub-circuits to model complete devices. Stimulus devices are used in the circuit to provide input for other digital devices during the simulation. Digital primitives are low-level devices whose main use is modeling off-the-shelf parts, often in combination with each other [3].

Behavioral primitives

The model simulator offers three primitives to aid in the modeling of complex digital devices:

- The Logic Expression
- The Pin-to-Pin Delay
- The Constraint Checker primitives

These devices are distinct from other primitives in that they allow data-sheet descriptions to be specified more directly, allowing a one-to-one correspondence using the function diagrams and timing specifications.

The Logic Expression primitive, LOGICEXP, uses free-format logic expressions to describe the behavior of the functional device.

The Pin-To-Pin Delay primitive, PINDLY, describes propagation delays using sets of rules based on the activity on the device inputs. Each of the stimulus behavior parts is described in detail below.

- **Device format :**

U<name> LOGICEXP (<no. of inputs>, <no .of outputs>)
+<digital power node> <digital ground node>
+<input node1><input node n>
+<output node 1><output node n>
+<timing model name>
+<I/O model name>
+ [IO_LEVEL=<value>]
+ [MNTYMXDLY=<value>]
+LOGIC:
+ <logic assignment>*

- **Timing Device Format :**

MODEL <timing model name>
UGATE [model parameters

- **Arguments and options :**

LOGIC→Marks the beginning of a sequence of one or more logic assignments. A logic assignment can have one of the two following forms:

<Output node>= {<logic expression>}
<Temporary value>= {<logic expression>}

An assignment to an output node causes the result of the *logic expression* to be scheduled on that output pin. Each *output node* must have exactly one assignment.[7] Any target of an assignment which is not specified as one of the nodes attached to the device defines a temporary variable. Once assigned, the *temporary values* can be used inside a subsequent *logic expression.* They are provided to reduce the complexity and improve the readability of the model. The rules for node names apply to *temporary value* names.

Logic Expression Operators

A "C-like" infix-notation expression returns one of the five digital logic levels. Like all other expressions, *logic expressions* within must be surrounded by curly braces "{ }." They can span

one or more lines using the "+" for continuation character in the first column position [4].

The logic operators are listed below from highest-to-lowest precedence

~ Unary not
& and
^ Exclusive OR
| OR

PSpice Model Editor

The Model Editor is used either to generate a new model or edit an existing model to create a new model. To generate a new model the method below has to be followed. For better understanding each is further reinforced with the help of a screen shot.

Using the File menu
From the File menu in the Model Editor, choose **New**.
Using the model menu from the model menu chose **copy from**.

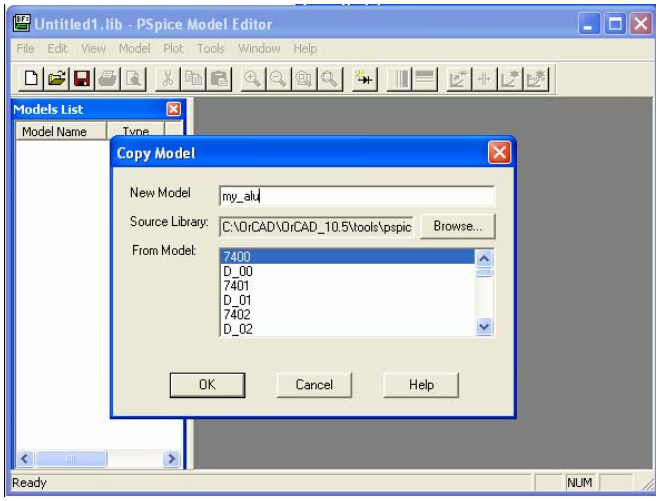

Figure 3. Screen shot of model editor when copy from is selected.

Select the **any Model** from the source library.
Click OK. A screen shot of this step has been shown in Figure 3.
Manually type in the behavioral or digital primitives of the device to be modeled.
Save the file as ".lib." A screen shot of this step is shown in Figure 4.

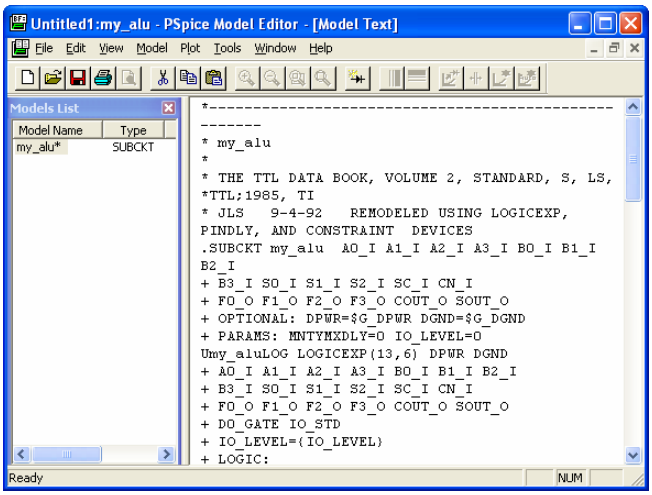

Figure 4. Screen shot of model editor with new model creation.

Exporting the Model to Capture Library

The Model Editor is used to import the model into Capture. To generate a new model in *Capture* the method below has to be followed.

- Using the File Menu

  a. From the File menu in Model Editor, choose *export* to capture part library.
  b. Show the path for importing from PSpice to *Capture* library by selecting the *browse* tab under title Enter output part library. A screen shot of this step is shown in Figure 5.

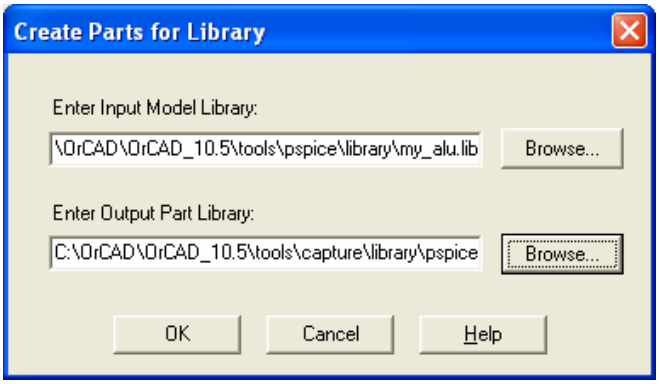

Figure 5. Screen shot of import /export tool bar in model editor.

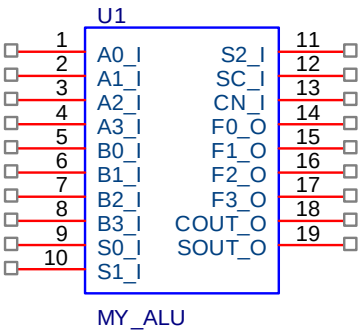

Figure 6. MY_alu pin layout.

Figure 6. shows the schematic view of ALU in *Capture* after importation.

Configuring New Model Library

After the part has been generated for a new/customized model library, the model library must be made available to the design. The model library containing custom simulation models is added to the project simulation profile.

1. In Capture, open Analog or Mixed-Circuit project.
2. From the *PSpice* menu choose *Edit Simulation Profile*.
3. Select the *Configuration Files* tab.
4. In the *Category* list box, select *Library*.
5. In the Filename text box, specify the location of the model library.
6. To make the library available to all designs, click *Add as Global*. If you want the library to be used only in the current design, select *Add to Design* and close the *Simulation Settings* dialog box. A screen shot of this step has been shown in Figure 7.

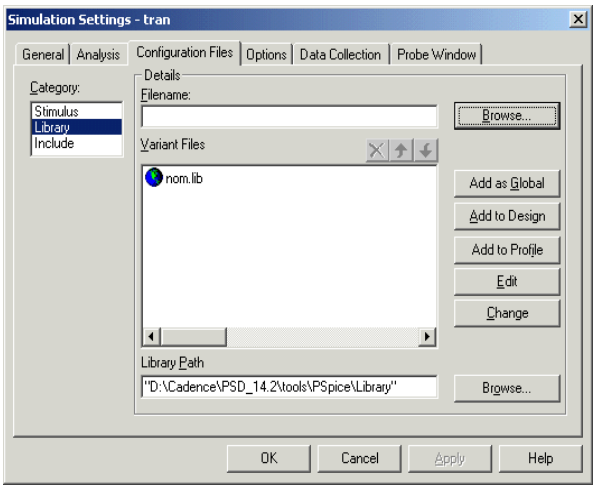

Figure 7. Screen shot of model wizard new simulation profile.

Note: Instead of editing a simulation profile, you can also create a new simulation profile. To do so, choose *New Simulation Profile* from the *PSpice* menu in Capture.

The above technique can be further explained by modeling an ALU in PSpice. The ALU designed below has been modeled keeping in mind the design structure of the "74181"[5].

The top module consists of a four-bit ALU. The four bit ALU designed consists of eight instruction sets. The eight functions performed are *ADDITION, SUBTRACTION, AND, NAND, OR, NOR, XOR and XNOR [6]*. Each of the above function is performed on two four-bit inputs. The functions performed are bit-wise operations as shown in Figure 8.

Each of the bitwise outputs is multiplexed out using 8X1 multiplexer shown in figure 8.Four single-bit building blocks are cascaded together to form a four-bit ALU.

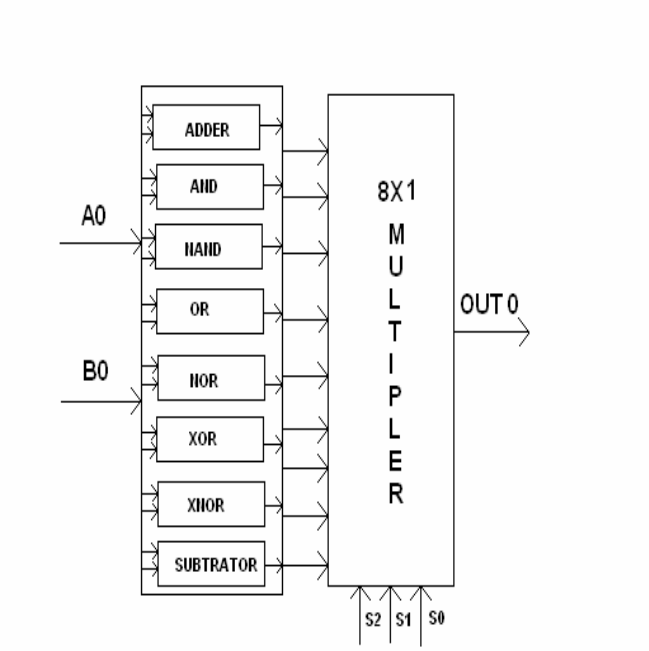

Figure 8. Single- Bit ALU Block Diagram.

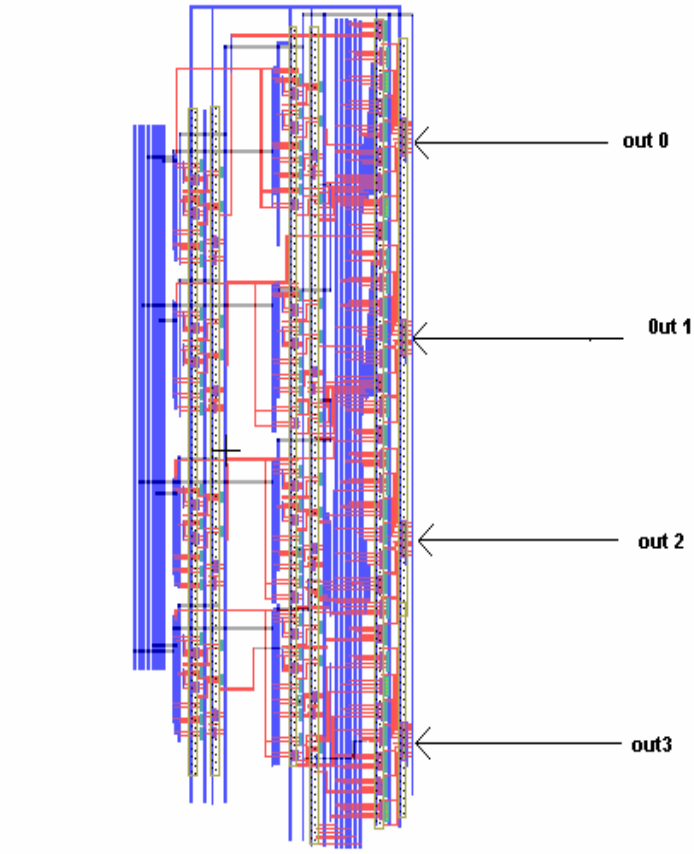

Figure 9: CMOS Four-Bit ALU Layout.

Figure 9. shows the CMOS ALU Layout and Results.

ALU Measurements

1. TOTAL NO. OF DEVICES : 982 MOS
2. APPROXIMATE AREA OF THE ALU 1431385.2 square microns
3. RISE TIME :10.13 nanoseconds
4. FALLTIME : 20.08 nanoseconds
5. TPLH : 11.53 nanoseconds
6. TPHL : 13.37 nanoseconds
7. PROPAGATION DELAY: 12.45 nanoseconds
8. SKEW: 48.

Figure 10. shows the inputs of Table 1 in T-Spice.

Figure 11. shows the outputs of Table 1 in T-Spice.

Figure 12. shows the ALU in PSpice Capture and Figure 13. shows the outputs of Table 1. in PSpice

Table 1. Instruction Set Functional.

A INPUT	B INPUT	FUNCTION (CONTROL SIGNAL)	OUTPUT
1010	0100	ADD(000)	1110
1010	0100	AND(001)	0000
1010	0100	NAND(010)	1111
1010	0100	OR(011)	1110
1010	0100	NOR(100)	0111
1010	0100	XOR(101)	1110
1010	0100	XNOR(110)	0001
1010	0100	SUB(111)	0110

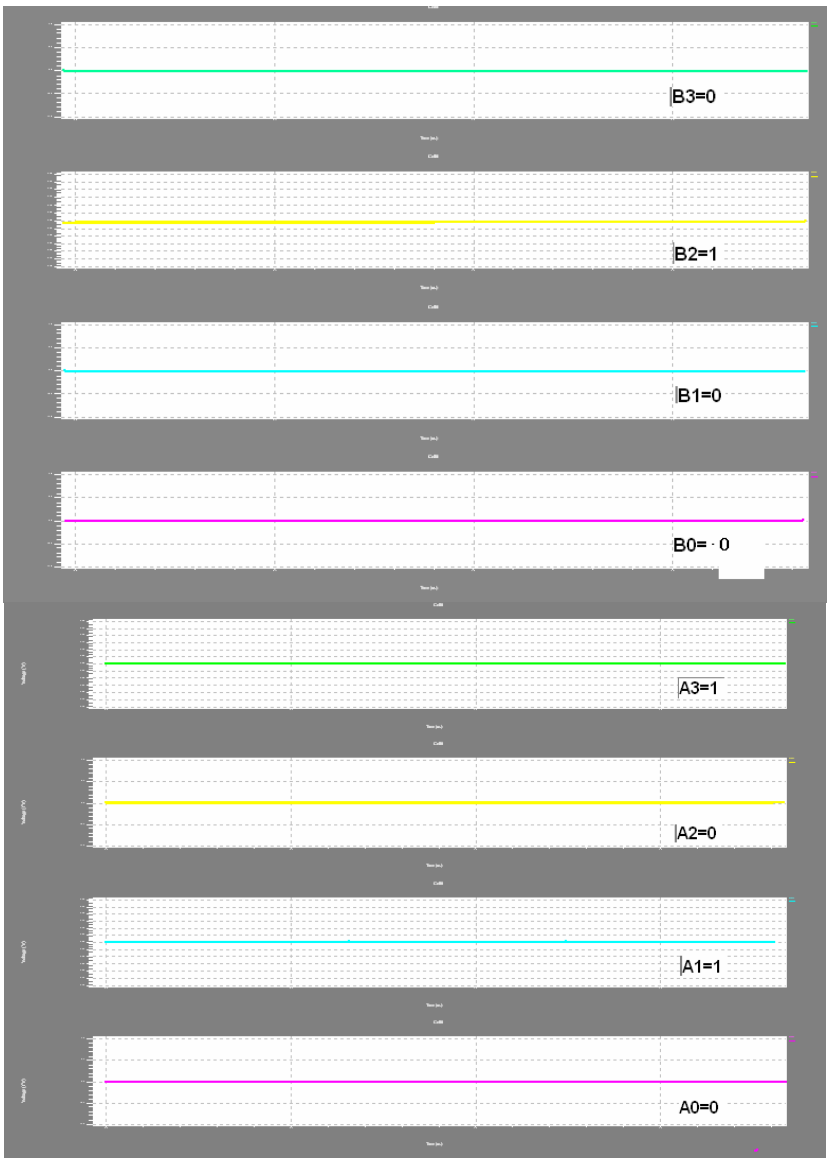

Figure 10. T-Spice Four-bit input.

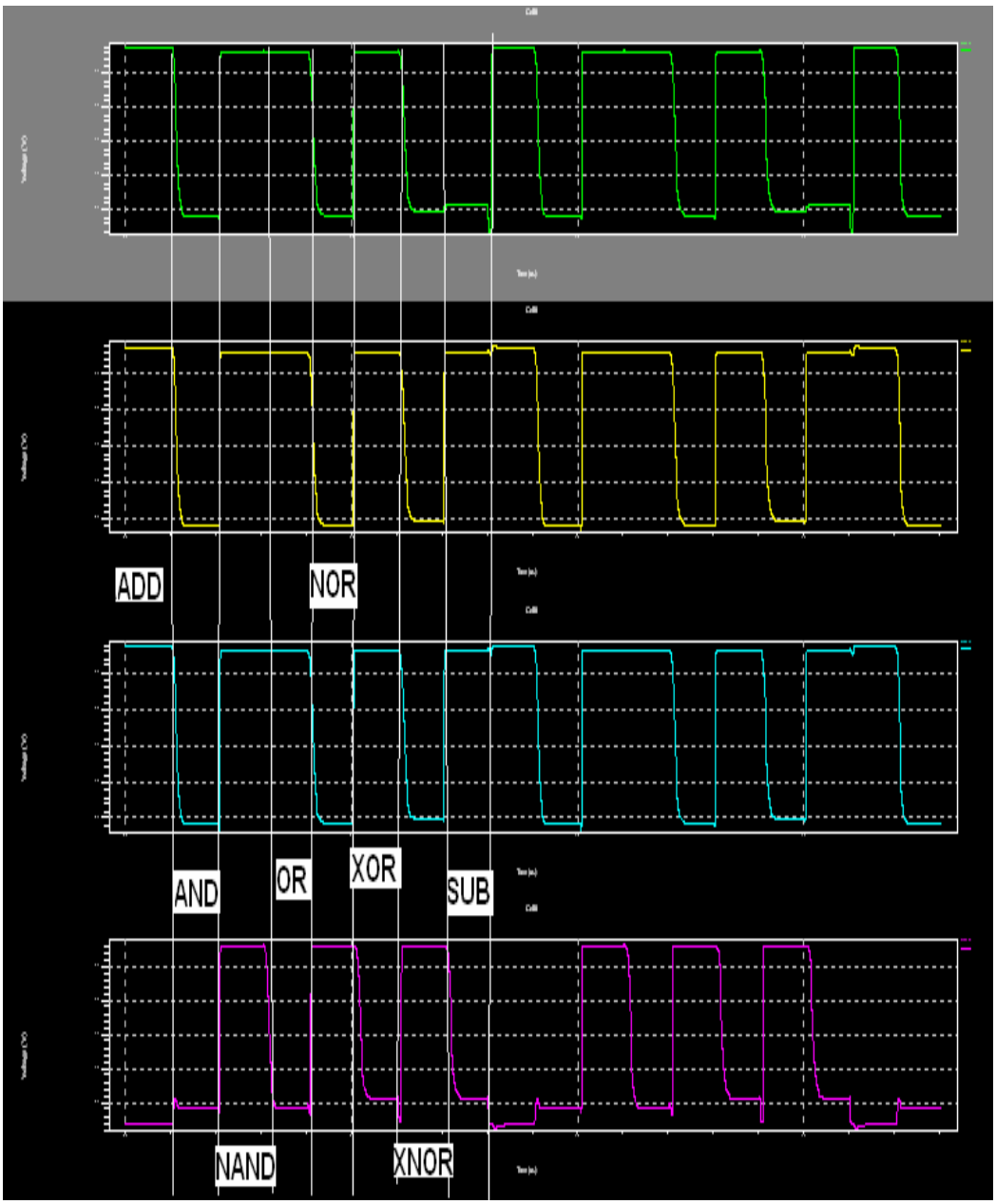

Figure 11. T-Spice Four-Bit Output.

PSpice Model Editor Code

```
.SUBCKT my_alu A0_I A1_I A2_I A3_I B0_I B1_I B2_I
+ B3_I S0_I S1_I S2_I SC_I CN_I
+ F0_O F1_O F2_O F3_O COUT_O SOUT_O
+ OPTIONAL: DPWR=$G_DPWR DGND=$G_DGND
+ PARAMS: MNTYMXDLY=0 IO_LEVEL=0
Umy_aluLOG LOGICEXP (13, 6) DPWR DGND
+ A0_I A1_I A2_I A3_I B0_I B1_I B2_I
+ B3_I S0_I S1_I S2_I SC_I CN_I
+ F0_O F1_O F2_O F3_O COUT_O SOUT_O
+ D0_GATE IO_STD
+ IO_LEVEL= {IO_LEVEL}
+ LOGIC:
+   A0   = {A0_I}
+   A1   = {A1_I}
+   A2   = {A2_I}
+   A3   = {A3_I}
+   B0   = {B0_I}
+   B1   = {B1_I}
+   B2   = {B2_I}
+   B3   = {B3_I}
+   S0   = {S0_I}
+   S1   = {S1_I}
+   S2   = {S2_I}
+   CN   = {CN_I}
+   SC   = {SC_I}
*
* Intermediate terms:
*
*LOGIC OUTPUT OF F0
+
I01={((~S2&~S1&~S0)&((A0^B0)^CN))|((~S2&~S1&S0)&(A0&B0))|((~S2&S1&~S0)&~(A0&B0))|(
(~S2&S1&S0)&(A0|B0))}
  +
I02={((S2&~S1&~S0)&~(A0|B0))|((S2&~S1&S0)&(A0^B0))|((S2&S1&~S0)&~(A0^B0))|((S2&S1&S
0)&(A0^~B0^SC))}
  + F0_O= {(I02|I01)}
  *LOGIC OUTPUT OF F1
  + I11 = {(A0&B0)| (B0&CN)| (CN&A0)}
  + I12 = {(A0&~B0)| (~B0&SC)| (SC&A0)}
  +I13={((~S2&~S1&~S0)&((A1^B1)^I11))|((~S2&~S1&S0)&(A1&B1))|((~S2&S1&~S0)&~(A1&B1
))|((~S2&S1&S0)&(A1|B1))}
  +I14={(S2&~S1&~S0&~(A1|B1))|((S2&~S1&S0)&(A1^B1))|((S2&S1&~S0)&~(A1^B1))|((S2&S1&
S0)&(A1^~B1^I12))}
  + F1_O = {I14|I13}
  *LOGIC OUTPUT OF F2
  + I21 = {(A1&B1)| (B1&I11)| (I11&A1)}
  + I22 = {(A1&~B1)| (~B1&I12)| (I12&A1)}
```

+I23={((~S2&~S1&~S0)&(A2^B2^I21))|((~S2&~S1&S0)&(A2&B2))|((~S2&S1&~S0)&~(A2&B2))|((~S2&S1&S0)&(A2|B2))}

+I24={((S2&~S1&~S0)&~(A2|B2))|((S2&~S1&S0)&(A2^B2))|((S2&S1&~S0)&~(A2^B2))|((S2&S1&S0)&(A2^~B2^I22))}

+ F2_O = {I24|I23}

*LOGIC OUTPUT OF F3

+ I31 = {(A2&B2)| (B2&I21)| (I21&A2)}

+ I32 = {(A2&~B2)| (~B2&I22)| (I22&A2)}

+I33=((~S2&~S1&~S0)&(A3^B3^I31))|((~S2&~S1&S0)&(A3&B3))|((~S2&S1&~S0)&~(A3&B3))|((~S2&S1&S0)&(A3|B3))}

+I34={((S2&~S1&~S0)&~(A3|B3))|((S2&~S1&S0)&(A3^B3))|((S2&S1&~S0)&~(A3^B3))|((S2&S1&S0)&(A3^~B3^I32))}

PSpice ALU Wiring

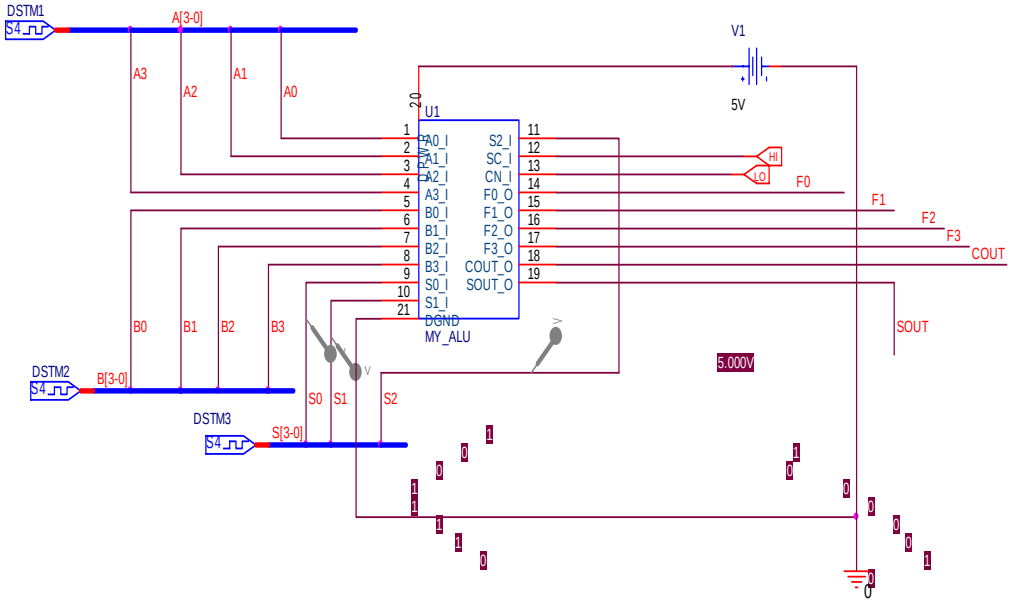

Figure 12. ALU in PSpice Capture.

Figure 13. ALU simulations in PSpice.

Conclusion

To stay competitive in today's market engineers must take a design from engineering through the manufacturing process with shorter design cycles and faster time to market. To be successful, you need a set of powerful, intuitive, and integrated tools that work seamlessly across the entire design flow [6].

OrCAD personal productivity tools have a long history of addressing these demands and more. And with the technique described above they make Digital Designs mere child's play. They help designers create and test different designs of their choice without even touching a piece of hardware. The present paper outlines a simple but an important method in designing digital devices in OrCAD Pspice. The method is better explained with the help of an eight-instruction set four-input ALU.

The issue of teaching digital computing is as old as computers; this project helps students to comprehend the construction of an ALU from gate down to semiconductor level. Electronic design automation (PSpice & TSpice) has enabled us to design an ALU that meets an industrial model.

References

1. Behrooz, Vahidiand and Jamal, Beiza, Using PSpice in Teaching Impulse Voltage Testing of Power Transformers to Senior Undergraduate Students in IEEE TRANSACTIONS ON EDUCATION, VOL. 48, NO. 2, MAY 2005.

2. L. Puglisi, P. Ferrari, P. Tenca, and A. Rebora, An advanced application of PSpice modeling and simulation for design optimization of push-pull dc/dc converter, in *Proc. 7th Inst. Elect. Eng. Int. Conf. Power Electronics Variable Speed Drives*, Genoa, Italy, Sep. 1998, pp. 117–120.

3. William Gerard Hurley and Chi Kwan Lee Development, Implementation, and Assessment of a Web-Based Power Electronics Laboratory in IEEE TRANSACTIONS ON EDUCATION, VOL. 48, NO. 4, NOVEMBER 2005.

4. Stephen Prigozgy, senior IEEE member. Novel application of Spice in engineering education in IEEE vol for education, vol 32, No 1 Feb. 1989

5. J. M. Deskur, PSpice simulation of power electronic and motion control systems, *Proc. IEEE Int. Symp. Industrial Electronics*, pp. 195–200, Jul. 1997.

6. K. T. Chau and C. C. Chan, "A Spice compatible model of permanent magnet dc motor drives," in *Proc. 1995 IEEE Int. Conf. Power Electronics and Drive Systems*, vol. 1, Kowloon, Singapore, Feb. 1995, pp.477–482.

7. OrCAD PSpice help manual.

Biographical Information

Saeid Moslehpour is an Assistant Professor in the Electrical and Computer Engineering Department in the College of Engineering, Technology, and Architecture at the University of Hartford. He holds PhD (1993) from Iowa State University and Bachelor of Science, Master of Science (1990), and Education Specialist (1992) degrees from University of Central Missouri. His research interests include logic design, CPLDs, FPGAs and distance learning. Email: moslehpou@hartford.edu

Srikrishna Karatalapu received his Bachelor of Science degree in electrical engineering in Hyderabad, India and Master of Engineering degree from the University of Hartford in 2006.