# CPE 133 Lab 2:

# Reduced Full Adder & Implementing/Simulating Function Forms

Prof. John Callenes-Sloan

Luis Gomez

## Summary:

**Full-Adder:** Digital circuit which takes three 1-bit inputs (OP_A, OP_B, Cin) and outputs their sum and carryover using the Sum of Products & Product of Sums approach. For this lab, we considered sum only.

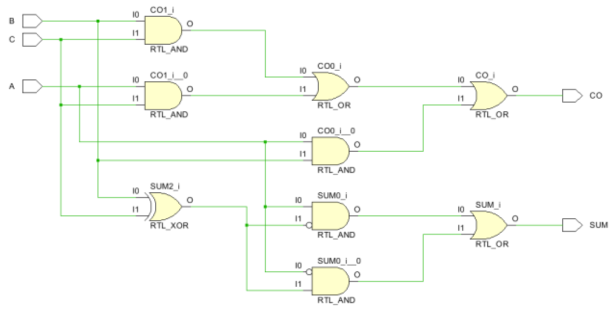
*SOP (w/ abbreviated terms): F = ~ABC + A~BC + AB~C + ABC*

*Reduced SOP (w/ abbreviated terms): F = (A & (~(B^C))) | ((~A) & (B^C))*

**Kavanaugh Map:** A (height) X BC (width); **Prime Essential Implicant** (Light red/blue)

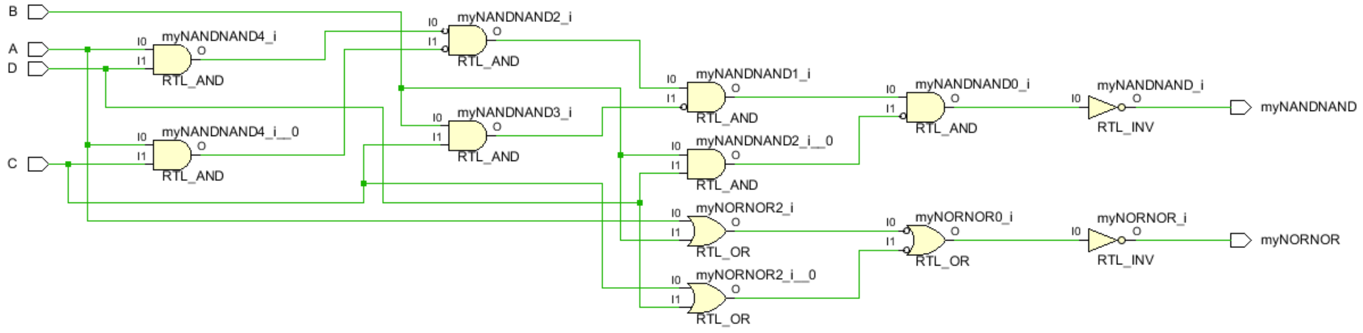
**Black Cell Borders indicate looping!** Four **1X1**loops for POS, Four **1X1** loops for SOP

| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 1 |
| **1** | 1 | 0 | 1 | 0 |



**Mystery-Circuit:** Digital circuit which takes four 1-bit inputs (A, B, C, & D) and outputs a function. According to K-Map & circuit implementation below:

**NAND/NAND** *F = ~( ~(AD) ~(AC)~(BC)~(BD) )* **NOR/NOR** *F = ~( ~(A + B) + ~(C + D) )*



**K-Map:** AB (height) X CD (width); **Prime Essential Implicant** (Light red/blue) **Implicant** (Deep red/blue)

**Black Cell Borders indicate looping!** Two **1X4** loops for POS, Four **2X2** loops for SOP

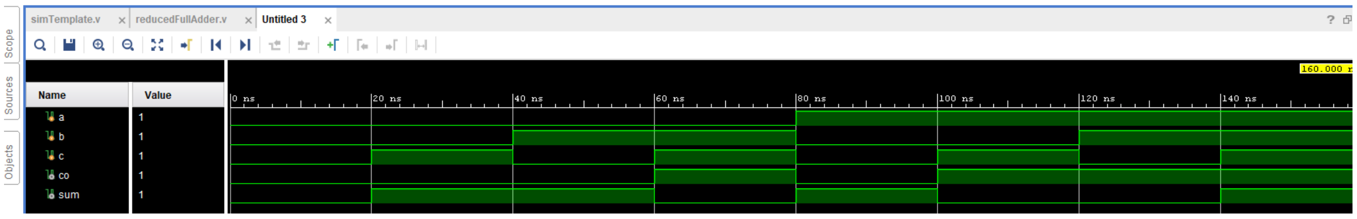
| A | B | C | D | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

|  | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 1 | 1 |
| 10 | 0 | 1 | 1 | 1 |

## Verification:

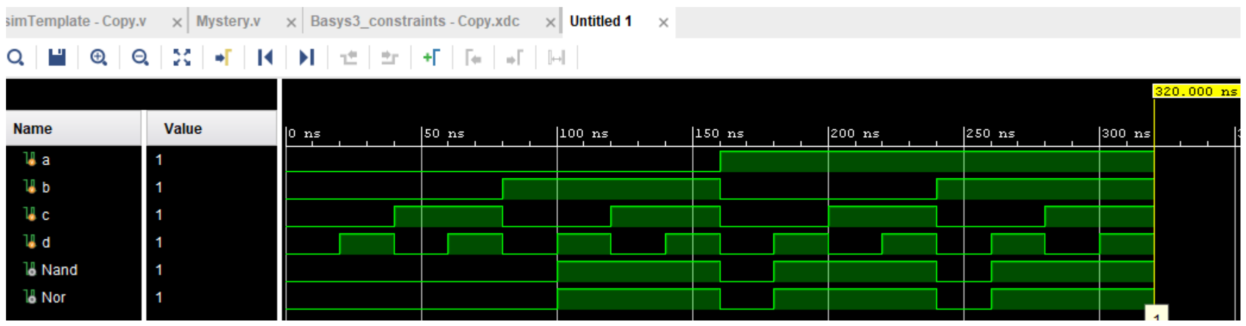
For both reduced FA and the Mystery circuit, we conducted behavioral simulations & board-tests to exhaustively verify the predicted behavior of the digital circuits. On the next page you will find screenshots of the resulting behavioral simulations and photos from the board-tests.

**Figure 1 reduced Full-Adder tests:**



| Name | Value |
|---|---|
| a | 1 |
| b | 1 |
| c | 1 |
| co | 1 |
| sum | 1 |

**Figure 2 Mystery circuit tests:**



| Name | Value |
|---|---|
| a | 1 |
| b | 1 |
| c | 1 |
| d | 1 |
| Nand | 1 |
| Nor | 1 |

## Questions:

Exp 2:

1. *Was there any advantage to using the reduced form of the FA equation when implementing your equations? Briefly explain.*

It was easier to implement out equations because they were shorter and more concise. Other than reducing the typing required, the reduced form did not provide an advantage.

2. *Since a FA does everything a HA does and more, briefly describe why you feel that HAs are still hanging around in digital design land.*

Half adders are still used in digital design because they are a basic building block for many components, such as a full adder and many other arithmetic operations.

Exp 3:

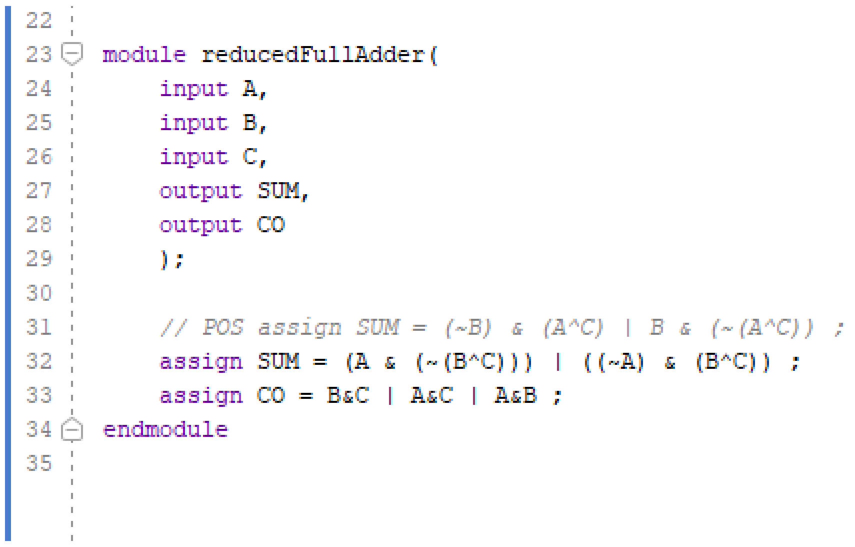
1. *Write down the reduced SOP, reduced POS, NAND/NAND and NOR/NOR forms of the circuit designed in this experiment. How many logic gates are used in each form? For this question, assume that the inputs to the system are all non-inverted; therefore, include the number of inverters required in your gate count. Assume you have the ability to use any type of gate with any number of inputs (such as 2,3,4-input NANDs, NORs, ANDs, ORs, etc.).*

   **5 gates, 8 inverters … NAND/NAND** *F = ~( ~(AD) ~(AC)~(BC)~(BD) )*
   **3 gates, 4 inverters … NOR/NOR** *F = ~( ~(A + B) + ~(C + D) )*
   **5 gates … Reduced SOP** F = (BD) + (BC) + (AD) + (AC)
   **3 gates … Reduced POS** F = (A + B)(C + D)

2. *Speculate on why integrated circuit manufacturers don't generally make items such as 7-input NAND gates.*

Manufacturers don't make items such as a 7-input NAND gates because they might be used for niche applications, everything could possibly be built with 2 input gates, or perhaps they might be too large/have too many transistors.

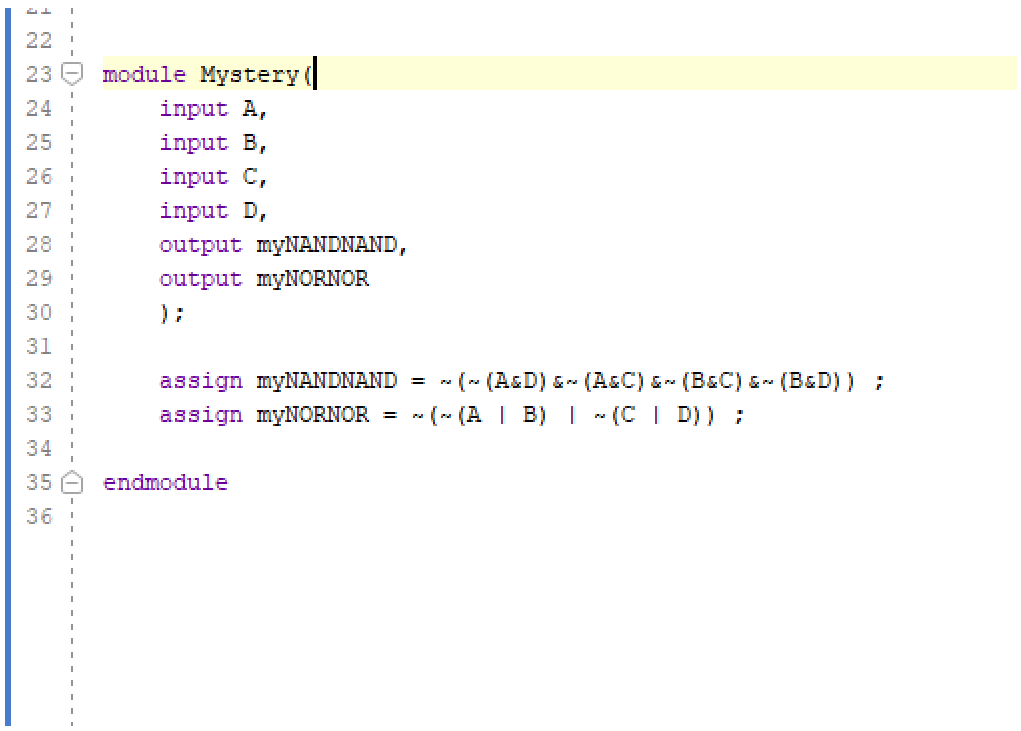
Reduced FA Design Code:



```
module reducedFullAdder(
    input A,
    input B,
    input C,
    output SUM,
    output CO
    );

    // POS assign SUM = (~B) & (A^C) | B & (~(A^C)) ;
    assign SUM = (A & (~(B^C))) | ((~A) & (B^C)) ;
    assign CO = B&C | A&C | A&B ;
endmodule
```

Mystery circuit code:



```
module Mystery(
    input A,
    input B,
    input C,
    input D,
    output myNANDNAND,
    output myNORNOR
    );

    assign myNANDNAND = ~(~(A&D)&~(A&C)&~(B&C)&~(B&D)) ;
    assign myNORNOR = ~(~(A | B) | ~(C | D)) ;

endmodule
```