Lab 12: VHDL

Arturo Salinas-Aguayo
CSE 2301: Principles and Practice of Digital Logic Design
Dr. Mohammad Khan, Section 003L-1248
Electrical and Computer Engineering Department

College of Engineering, University of Connecticut
Coded in LaTeX

Discussion

This lab introduced the concept of a Hardware Description Language in order to synthesize logic. VHDL stands for *Very High Speed Hardware Description Language* and was standardized by the IEEE for digital circuit production.

Example 1 The Hamming Parity Error Check in VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;


entity ParityGen is

 port(
  D : in std_logic_vector(3 downto 0);
  Q : out std_logic_vector(6 downto 0)
 );

end ParityGen;

architecture arch1 of ParityGen is
signal P1 : std_logic;
signal P2 : std_logic;
signal P3 : std_logic;
begin
 Q(6) <= P1;
 Q(5) <= P2;
 Q(4) <= D(3);
 Q(3) <= P3;
 Q(2) <= D(2);
 Q(1) <= D(1);
 Q(0) <= D(0);

 P1 <= Q(0) XOR Q(2) XOR Q(4);
 P2 <= Q(4) XOR Q(1) XOR Q(0);
 P3 <= Q(2) XOR Q(1) XOR Q(0);

end arch1;
```

Example 2 The ALU Reprise

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity ALU2ElectricBoog is

 port(
  B : in std_logic_vector(1 downto 0);
  A : in std_logic_vector(1 downto 0);
  Q : out std_logic_vector(3 downto 0);
  Op : in std_logic_vector(1 downto 0)
 );

end ALU2ElectricBoog;

architecture arch1 of ALU2ElectricBoog is
begin
 process (A, B, Op)
    begin
     if Op = "00" then
      Q <= "00" & A and "00" & B;
     elsif Op = "01" then
                Q <= "00" & A or "00" & B;
     elsif Op = "10" then
      Q <= unsigned("00" & A) + unsigned("00" & B);
     elsif Op = "11" then
                Q <= unsigned(A) * unsigned(B);
     end if;
    end process;
end arch1;
```

Example 3 The Use of `unsigned()` Cast

The `unsigned()` function in VHDL is used to cast a binary vector to an `UNSIGNED` type, treating the vector as a positive integer. This explicit casting allows VHDL to interpret binary data without ambiguity, particularly during arithmetic operations, ensuring that the

result is processed as an unsigned value. In VHDL, operations between `SIGNED` and `UNSIGNED` types require conversions to maintain compatibility, as VHDL does not implicitly mix these types.

In the context of your ALU code:

- `unsigned()` ensures that `A` and `B` are interpreted as positive integers, allowing for correct behavior in operations like addition (`+`) without treating them as potentially negative numbers.

Effect of Using `signed()` Instead of `unsigned()`

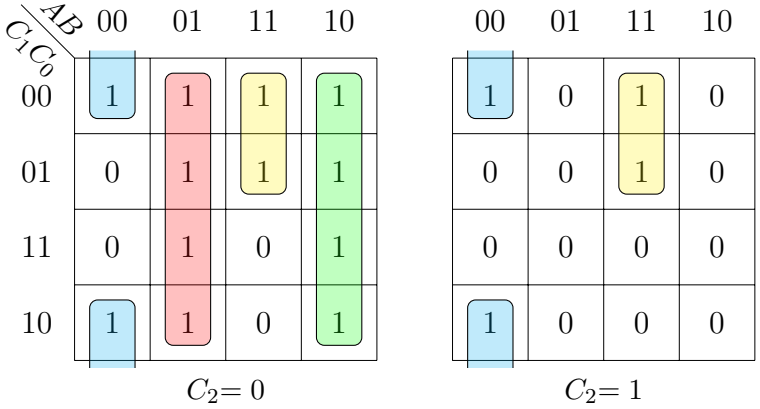
If you were to replace `unsigned()` with `signed()` in the ALU, the output could change significantly, particularly in cases where `A` or `B` have high bits set to `1` (interpreted as negative in 2's complement). For example:

- With `unsigned`, a binary vector such as `"1100"` would be interpreted as 12.
- With `signed`, the same vector `"1100"` would be interpreted as -4 in 2's complement form.

This change would result in different outputs for operations that involve addition or multiplication, where the result might switch from a positive to a negative value, depending on the high bits of the input.

Circuit Element Emulated by `if` and `elsif` Statements

In the ALU, the `if` and `elsif` statements emulate the behavior of a **multiplexer (MUX)**. The `Op` input acts as a selector that determines which operation (AND, OR, ADD, or MULTIPLY) the ALU will perform based on its value. Each condition (e.g., `Op = "00"`, `Op = "01"`, etc.) corresponds to a specific operation, similar to how a multiplexer selects an output based on control signals.



$$F = \overline{C_0}\overline{A}\overline{B} + \overline{C_1}AB + \overline{C_2}\overline{A}B + \overline{C_2}A\overline{B}$$

With the PLA implementation, each minterm for each output bit is placed and routed by the software and abstracted away from the wires. For the $C_2 = 0$ block, notice how there are potential hazards at bay. These are kept seperate in order to best maximize the effect of the 3-Dimensional 5 Variable Karnaugh Map. In this case, the blue and yellow squares "wrap around" the corresponding C_2 block.

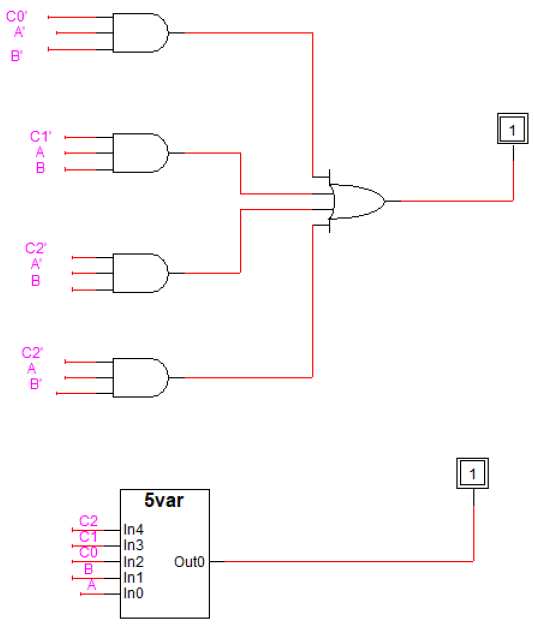


Figure 1: Traditional Gate Implementation and Black Box Implementation

The PLA form greatly reduces the complexity of visual schematic design by abstracting away the gate logic by programming it directly.

Example 4 A Continuation of the Triangle Parity

In this lab, we examined a 15-bit data stream configured with EVEN parity using a triangular code. The output identifies error positions by mapping the detected location to a four-bit decimal representation. An error is flagged whenever a mismatch in parity is detected across specific bit groupings, similar to Hamming code principles. Recall that in a Hamming code, the minimum number of parity bits p needed for m data bits satisfies $2^p \geq m + p + 1$. This ensures unique identification of single-bit errors. The triangular parity code, however, uses overlapping parity checks to pinpoint errors more precisely by leveraging redundancy within intersecting bit groups.

A	B	C	D	E	F	G	H	I	Error Position
0	0	0	0	0	0	0	0	0	None (No Error)
1	0	0	0	0	0	1	0	1	Parity Bit 5
1	1	0	0	0	0	1	0	0	Data Bit 4
1	0	1	0	0	0	0	1	1	Data Bit 3
1	0	0	1	0	0	0	1	0	Data Bit 2
1	0	0	0	1	0	0	0	1	Data Bit 1
0	1	0	0	0	1	0	0	1	Parity Bit 9
0	1	1	0	0	1	0	0	0	Data Bit 8
0	1	0	1	0	0	1	1	1	Data Bit 7
0	1	0	0	1	0	1	1	0	Data Bit 6
0	0	1	0	0	1	1	0	0	Parity Bit 12
0	0	1	1	0	1	0	1	1	Data Bit 11
0	0	1	0	1	1	0	1	0	Data Bit 10
0	0	0	1	0	1	1	1	0	Parity Bit 14
0	0	0	1	1	1	1	0	1	Data Bit 13
0	0	0	0	1	1	1	1	1	Parity Bit 15

Table 1: Truth Table for Error Detection (5 Inputs, 4 Outputs)

Example 5 The Overflow of a 3 bit by 3 bit full Adder

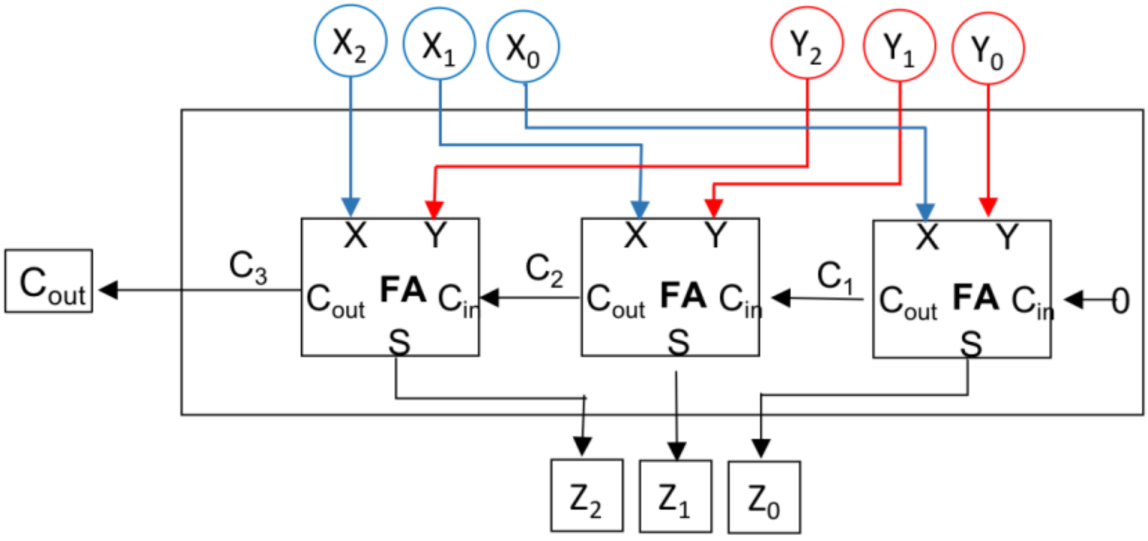


Figure 2: The 3x3 bit Full Adder Circuit

The output **OVERFLOW** bit, which can be used as a signal for the logic to indicate a potentially erroneous operation, can be determined quickly by comparing the MSB of Y and X. The equation can simply be put as:

$$\textbf{OVERFLOW} = \overline{X_2}\,\overline{Y_2}Z_2 + X_2Y_2\overline{Z_2}$$

Example 6 Programmable Array Logic
For this, the task is to draw a PLA for the equation:

$$F = \overline{C}A + BC$$

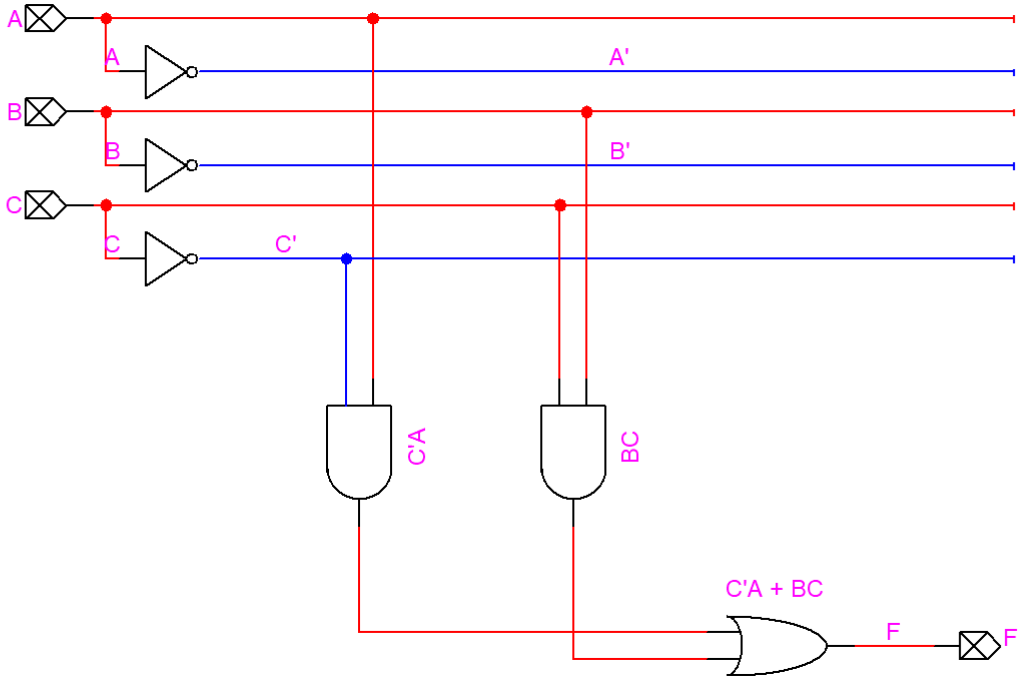


Figure 3: $F = \overline{C}A + BC$

This is done similarly to creating combinational logic at the begining of the course. Every wire being drawn out and extended to the combination of AND and OR gates is the characteristic of the PLA.