

Learning Outcomes

1. Learn to program Logic gates in Dataflow Modeling Verilog HDL codes.
2. Construct a circuit diagram in a Logisim software and refer the result by using Dataflow Verilog program.

Review:

- *Gate level Modeling*

The module implemented in terms of logic gates and interconnections between these gates. This is similar to drawing the actual circuit of the system but only on the code form. Verilog supports basic logic gates as predefined primitives. All logic circuits can be designed by using basic gates.

Module 2: Dataflow Modelling

At this level the module is designed by specifying the data flow. The designer is aware of how data flows between hardware registers and how the data is processed in the design. Dataflow modeling makes use of the functions that define the working of the circuit instead of its gate structure

A combinational circuit describing the by *Boolean expression* is ease to program in a dataflow modeling. This dataflow modeling style gives more complex than primitive gates (gate level).

The following are the syntax in dataflow modeling:

~ NOT	& AND	OR	~ & NAND
~ NOR	^ EX-OR		~ ^ EX-NOR

```
assign out1 = in1 & in2;           // perform and function on in1 and in2 and assign the result to out1
assign out2 = ~ in1;
```

Example 1: Half Adder in Digital Logic

The addition of 2 bits is done using a combination circuit called **Half adder**. The *input variables* are augend and addend bits, and output variables are sum & carry bits. A and B are the two input bits.

Truth Table:

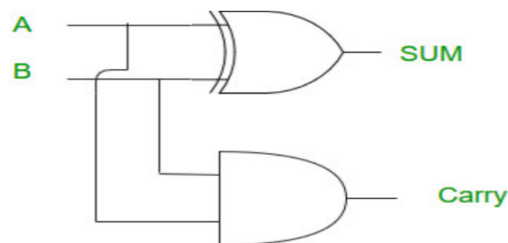
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logical Expression:

Sum = A XOR B

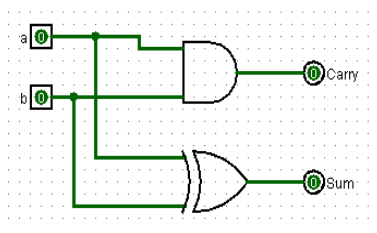
Carry = A AND B

Implementation:



References: <https://www.geeksforgeeks.org/half-adder-in-digital-logic/>

Logic Gates Verilog Source Code

Circuit Diagram (Half Adder)	Boolean Expression	Gate Level source code
	$\text{Carry} = ab$ $\text{Sum} = a \oplus b$	<pre> module HAdder (a, b, sum, ca); input a,b; // define input ports output sum,ca; // define output ports xor x1(sum,a,b); // sum = a ⊕ b and a1(ca,a,b); // carry = ab end module </pre>

```

module Hadder (a,b, sum,ca);
input a,b;
output sum,ca;
assign sum = a ^ b;    // Sum = a ⊕ b
assign ca = a & b;     // Carry = ab
endmodule

```

```

module TestBench;
reg a,b;                // define input ports
wire sum,ca;           // output of the circuit
initial
begin                  // (in C language) open bracket
$display ("a[1] a[0] Carry Sum"); // print
    a = 1'b0;          // apply input where a = 0 (binary number)
    b = 1'b0;          // apply input where b = 0 (binary number)
    #4 $finish;        // time from 0 to 3
end                    // (in C language) close bracket

always #2 a = ~a;      // time = 2 where: ~ = not / inverter
always #1 b = ~b;      // time = 1 where: ~ = not / inverter

HAdder U1(a,b,sum,ca); // call the module HAdder

initial                // setup monitoring
$monitor("%g  %b  %b  %b  %b", $time,a,b,ca,sum);
endmodule

```

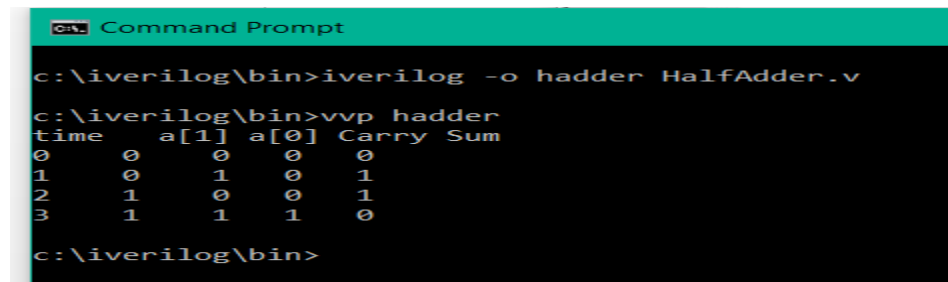
Table for variable input :

```

a b
0 0 → 0
0 1 → 1
1 0 → 2
1 1 → 3

```

Type **iverilog -o hadder HalfAdder.v** and hit the enter key



```

c:\iverilog\bin>iverilog -o hadder HalfAdder.v
c:\iverilog\bin>vvp hadder
time  a[1] a[0] Carry Sum
0      0   0   0     0
1      0   1   0     1
2      1   0   0     1
3      1   1   1     0
c:\iverilog\bin>

```

Figure 1: Half Adder Simulation Result using Gate Level Code

where:

Dash o (-o) indicates that you are trying to get the output and save the compiled code to the variable **sample_code** right after the -o

Class Participation:

To learn the program, convert the given program in Dataflow Modeling code.

```

module samplegate(a,b,c,d,x);
input a,b,c,d;
output x;
and o1(g1,a,b);
nor o2(g2,c,g1);
xor o3 (x,g1,g2,d);
endmodule

```

Task Assessment:

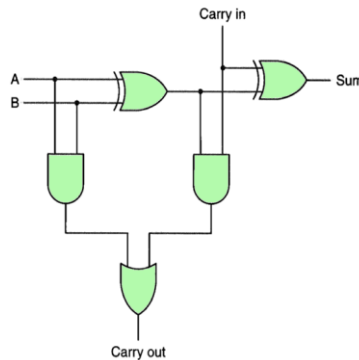
Part 1: Programming Exercises: Create a Dataflow Modeling Verilog HDL program for the following statement:

1. Listed down the Basic codes in Verilog program
 - a. 2 input - OR gate
 - b. 2 input - NAND gate
 - c. 3 input - NOR gate
 - d. 3 input - XOR gate
 - e. 3 input - XNOR gate

Part 2: Full Adder

1. Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as Carry in. The output carry is designated as Carry out and the normal output is designated as S which is SUM.
2. $S = A \oplus B \oplus \text{Cin}$ $C = AB + \text{Cin} (A \oplus B)$

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Assessment

- Hands-On Exercise – Laboratory

References:

- <https://www.javatpoint.com/verilog-data-flow-modeling>
- <https://www.geeksforgeeks.org/full-adder-in-digital-logic/#:~:text=Full%20Adder%20is%20the%20adder,as%20S%20which%20is%20SUM>
- <https://www.sciencedirect.com/topics/computer-science/full-adder>