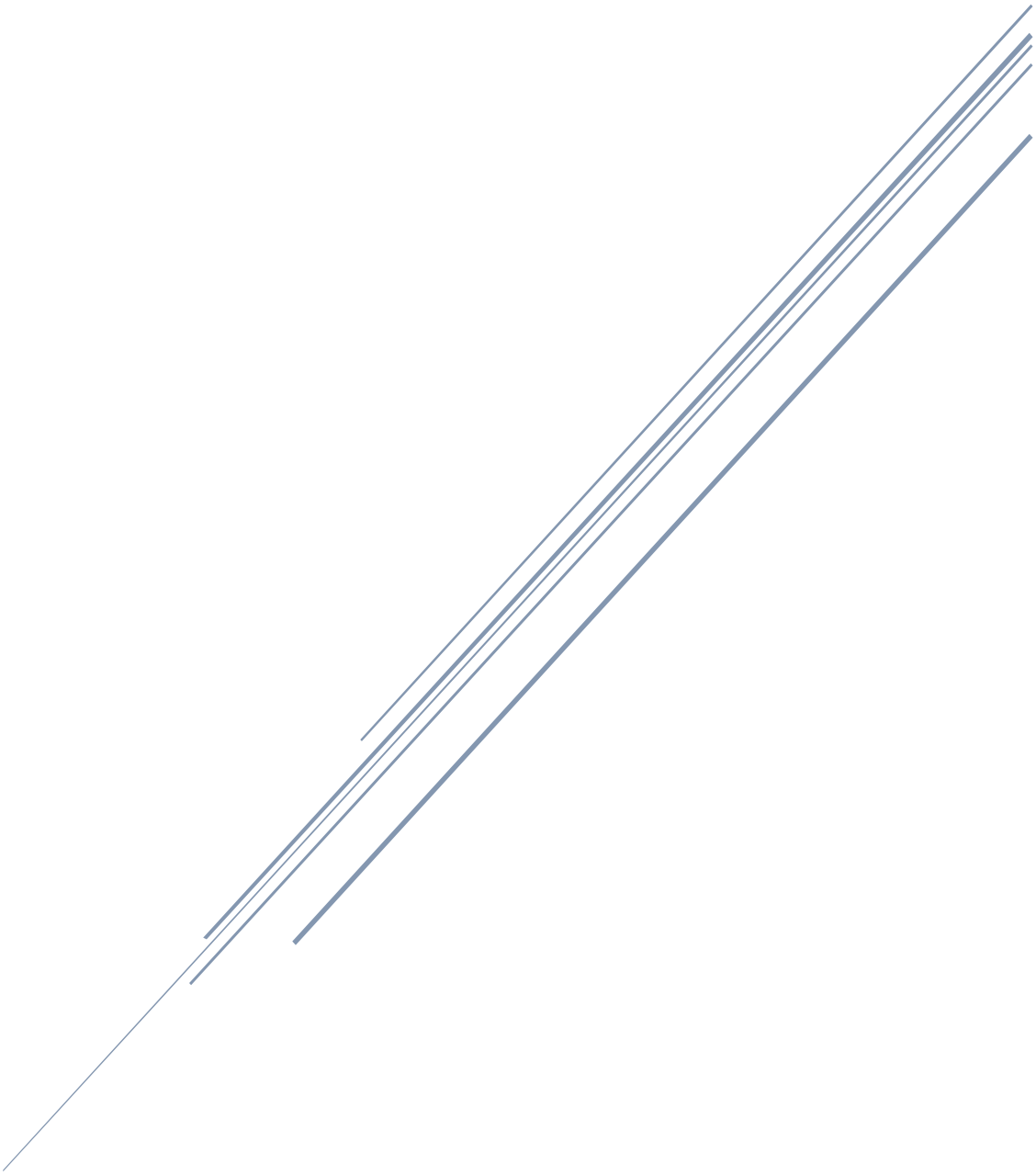
VERILOG ASSIGNMENT 2

21604552 – Emre Hancı



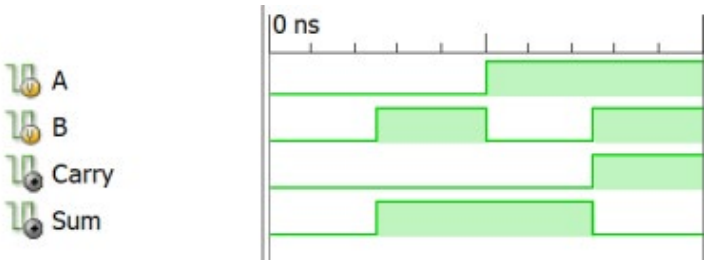
Hacettepe University - Department of Computer Engineering

BBM233 Logic Design Lab.

-Half Adder Verilog Codes

```
module Assign2(Carry,Sum,A,B);
        input A,B;
        output Carry,Sum;
        assign Sum = A ^ B;
        assign Carry = A & B;
endmodule
```

-Half Adder Waveform



-Half Adder Testbench

```
module Assign2_test;
        // Inputs
        reg A;
        reg B;
        // Outputs
        wire Carry;
        wire Sum;
        Assign2 uut (
                .Carry(Carry),
                .Sum(Sum),
                .A(A),
                .B(B)
        );
        initial begin
                // Initialize Inputs
                A = 0;
                B = 0;
                #50;
                A = 0;
                B = 1;
                #50;
                A = 1;
                B = 0;
                #50;
                A = 1;
                B = 1;
                #50;
                // Add stimulus here
        end
endmodule
```

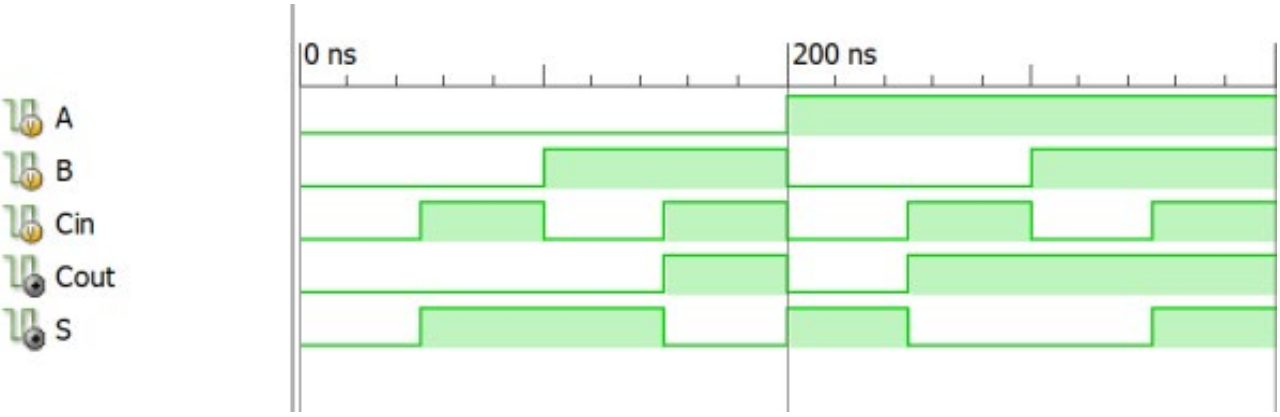
- 1-Bit Full Adder Verilog Codes

```
module Assign2(A,B,Cin,Cout,S);
	input A,B,Cin;
	output Cout,S;
	wire xor_AB,and_AB,CinXorAB;
	xor(xor_AB,A,B);
	xor(S,Cin,xor_AB);
	and(and_AB,A,B);
	and(CinXorAB,Cin,xor_AB);
	or(Cout,and_AB,CinXorAB);
endmodule
```

- 1-Bit Full Adder Testbench

```
module Assign2_test2;
	reg A;
	reg B;
	reg Cin;
	wire Cout;
	wire S;
	Assign2 uut (
		.A(A),
		.B(B),
		.Cin(Cin),
		.Cout(Cout),
		.S(S)
	);
	initial begin
		// Initialize Inputs
		A = 0; B = 0; Cin = 0; #50;
		A = 0; B = 0; Cin = 1; #50;
		A = 0; B = 1; Cin = 0; #50;
		A = 0; B = 1; Cin = 1; #50;
		A = 1; B = 0; Cin = 0; #50;
		A = 1; B = 0; Cin = 1; #50;
		A = 1; B = 1; Cin = 0; #50;
		A = 1; B = 1; Cin = 1; #50;
	end
endmodule
```

- 1-Bit Full Adder Waveform



- 4-Bit Full Adder Verilog Codes

```
module FullAdder(A,B,Cin,Cout,S);
        input A,B,Cin;
        output Cout,S;
        wire xor_AB,and_AB,CinXorAB;
        xor(xor_AB,A,B);
        xor(S,Cin,xor_AB);
        and(and_AB,A,B);
        and(CinXorAB,Cin,xor_AB);
        or(Cout,and_AB,CinXorAB);
endmodule
module FourBitAdder(A,B,Cin,Cout,S);
        input Cin;
        input [3:0] A,B;
        output Cout;
        output [3:0] S;
        wire Crry0,Crry1,Crry2;
        FullAdder
op0(A[0],B[0],Cin,Crry0,S[0]);
        FullAdder
op1(A[1],B[1],Crry0,Crry1,S[1]);
        FullAdder
op2(A[2],B[2],Crry1,Crry2,S[2]);
        FullAdder
op3(A[3],B[3],Crry2,Cout,S[3]);
Endmodule
```

- 4-Bit Full Adder Verilog Codes

```
module cases;
        reg [3:0] A;
        reg [3:0] B;
        reg Cin;
        wire Cout;
        wire [3:0] S;
        FourBitAdder uut (
                .A(A),
                .B(B),
                .Cin(Cin),
                .Cout(Cout),
                .S(S)
        );
        initial begin
                A = 0000; B = 0000; Cin = 0;
                #50
                A = 0100; B = 0011; Cin = 1;
                #50
                A = 0011; B = 0111; Cin = 1;
                #50
                A = 1000; B = 0100; Cin = 0;
                #50
                A = 0101; B = 0101; Cin = 0;
                #50
                A = 1101; B = 0101; Cin = 1;
                #50
                A = 0111; B = 1101; Cin = 0;
                #50
                A = 0111; B = 1101; Cin = 0;
        end
endmodule
```

- 4-Bit Full Adder Waveform