EE 421 / CS 425 Digital System Design Laboratory 4

Fall 2023

Shahid Masud

A Not-for-Profit University

Today's Topics

- Dataflow modeling in Verilog
- Behavior Description of Digital Systems
- High level constructs help in testing and simulation
- Modeling Delays in wires, gates and functional blocks

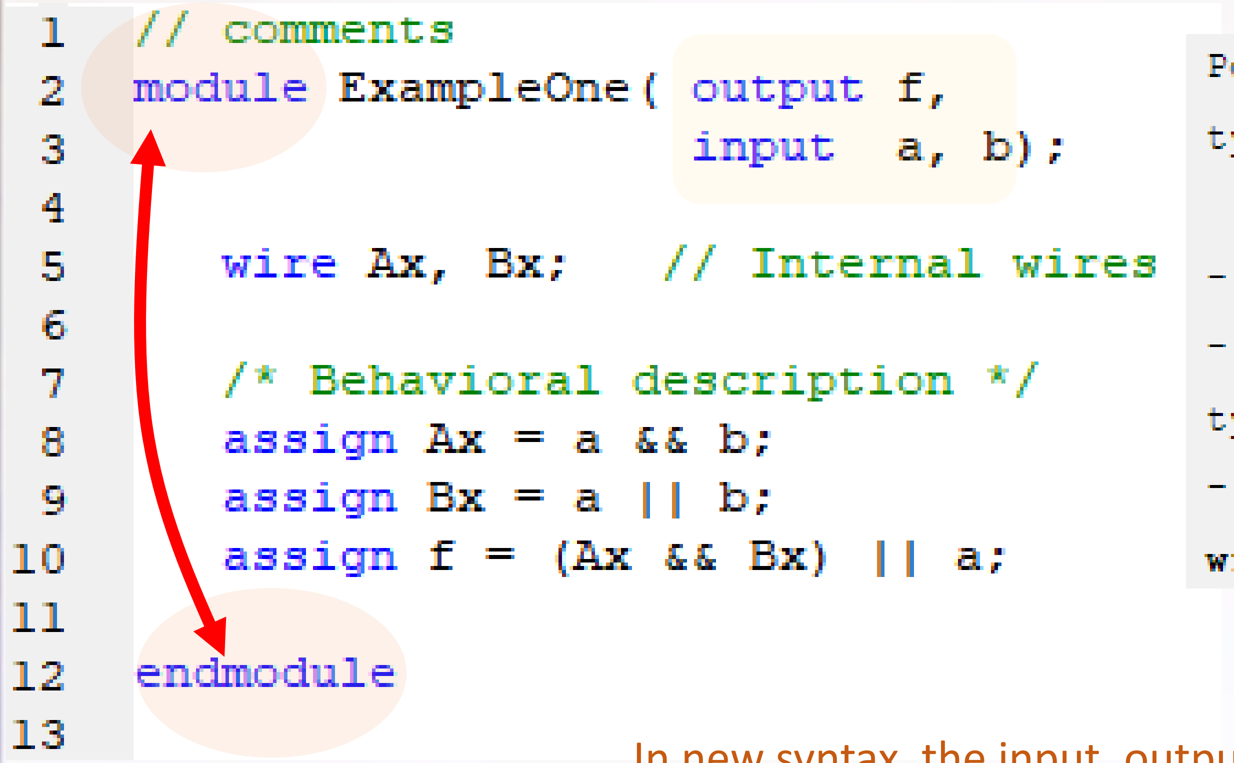
Design Capture in Verilog HDL

Verilog Allows Design Capture at Various Hierarchy Levels:

1. Switch Level (NMOS and PMOS transistors)
2. Gate Level (Describing Circuit as Logic Gates)
    a) Data Flow (RTL Sequential) Level
3. Dataflow Level ✓
4. Behavior Level ✓

The Design is Captured in a text file or obtained from schematic diagram

Verilog Syntax



```
// comments
module ExampleOne( output f,
                   input  a, b);

   wire Ax, Bx;    // Internal wires

   /* Behavioral description */
   assign Ax = a && b;
   assign Bx = a || b;
   assign f = (Ax && Bx) || a;

endmodule
```

```
Post-2001 Verilog allows the port name, direction, and
type to be declared together.

- Each port needs to have a user-defined name.
- The port directions are declared to be one of the three
types: input, output, or inout.
- A port can take on any of the data types, but only
wires, registers, and integers are synthesizable.
```

In new syntax, the input, output is defined as above.
In old syntax, this definition was after the module (…);
input a, b, …..;
output c,d,….;

Examples of assign statements

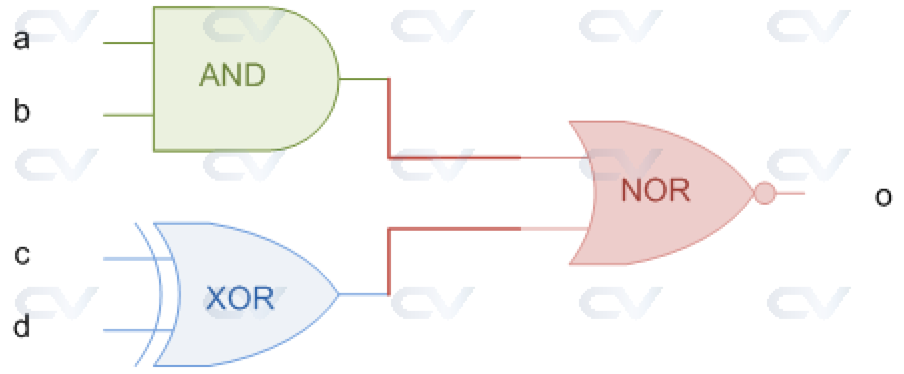
```
wire Out;
assign Out = A & B;
assign {COut, Sum} = A + B + CIn;
wire #50 Out = A & B;
```

```
module delay_test(  input a,  // Assume a=0 initialized at time '0'
  input b,  // Assume b=1 initialized at time ‘0’
  output reg c,   output reg d );
initial
begin
  #20 c = (a|b); //a|b gets evaluated after 20ns and gets assigned to ‘c’ immediately
  d = #50 (a&b); //a&b gets evaluated at time 20ns but gets assigned to ‘c’ after 70ns (20ns+50ns)
end
endmodule
```

LUMS
A Not-for-Profit University

Combinational Logic Design

Consider the following digital circuit made from combinational gates and the corresponding Verilog code.



Combinational logic requires the inputs to be continuously driven to maintain the output unlike sequential elements like flip flops where the value is captured and stored at the edge of a clock. So an `assign` statement fits the purpose the well because the output `o` is updated whenever any of the inputs on the right hand side change.

```verilog
 // This module takes four inputs and performs a boolean
// operation and assigns output to o. The combinational
// logic is realized using assign statement.

module combo (  input   a, b, c, d,
                                 output  o);

  assign o = ~((a & b) | c ^ d);

endmodule
```

4 Bit and 32 Bit Full Adders using assign

```
module adder (sum_out, carry_out, carry_in, ina, inb) ;
output [3:0] sum_out;
input [3:0] ina, inb;
output carry_out;
input carry_in;
wire carry_out, carry_in;
wire[3:0] sum_out, ina, inb;
  assign {carry_out, sum_out} = ina + inb + carry_in;
endmodule
```

- A 32-bit adder with carry-in and carry-out:

```
module add32_carry
    (input[31:0] a,b,
    input cin,
    output[31:0] sum,
    output cout);

  assign {cout, sum} = a + b + cin;
endmodule
```

concatenation

Using a ternary ?: operator

```
output = <expression> ? <value if true> : <value if false>;
```

```
//you can use an assign statement with ternary operator ?:,
//if you want it to remain as a wire type:

assign x = (val==0) ? a : (val==1) ? b : 'bx ;
```

```
module tristatebuffer( );

reg data_in, enable;

wire pad;

assign pad = (enable) ? data_in : 1'bz;

//When enable is 1, the pad is assigned the value of data_in, and when enable is 0, the pad is tristated.
```

LUMS
A Not-for-Profit University

Mux using assign

```
assign q = addr ? b : a;

assign q = addr[1] ? (addr[0] ? d : c) : (addr[0] ? b : a);
```

LUMS
A Not-for-Profit University

ALU Modules using assign

2-to-1 MUX

```
module mux32two
      (input [31:0] i0,i1,
       input sel,
       output [31:0] out);

   assign out = sel ? i1 : i0;
endmodule
```

32-bit Adder

```
module add32
      (input [31:0] i0,i1,
       output [31:0] sum);

   assign sum = i0 + i1;
endmodule
```

32-bit Subtracter

```
module sub32
      (input [31:0] i0,i1,
      output [31:0] diff);

   assign diff = i0 - i1;
endmodule
```

3-to-1 MUX

```
module mux32three
       (input [31:0] i0,i1,i2,
        input [1:0] sel,
        output reg [31:0] out);

always @ (i0 or i1 or i2 or sel)
begin
  case (sel)
    2'b00: out = i0;
    2'b01: out = i1;
    2'b10: out = i2;
    default: out = 32'bx;
  endcase
end
endmodule
```

16-bit Multiplier

```
module mul16
      (input [15:0] i0,i1,
       output [31:0] prod);

// this is a magnitude multiplier
// signed arithmetic later
assign prod = i0 * i1;

endmodule
```

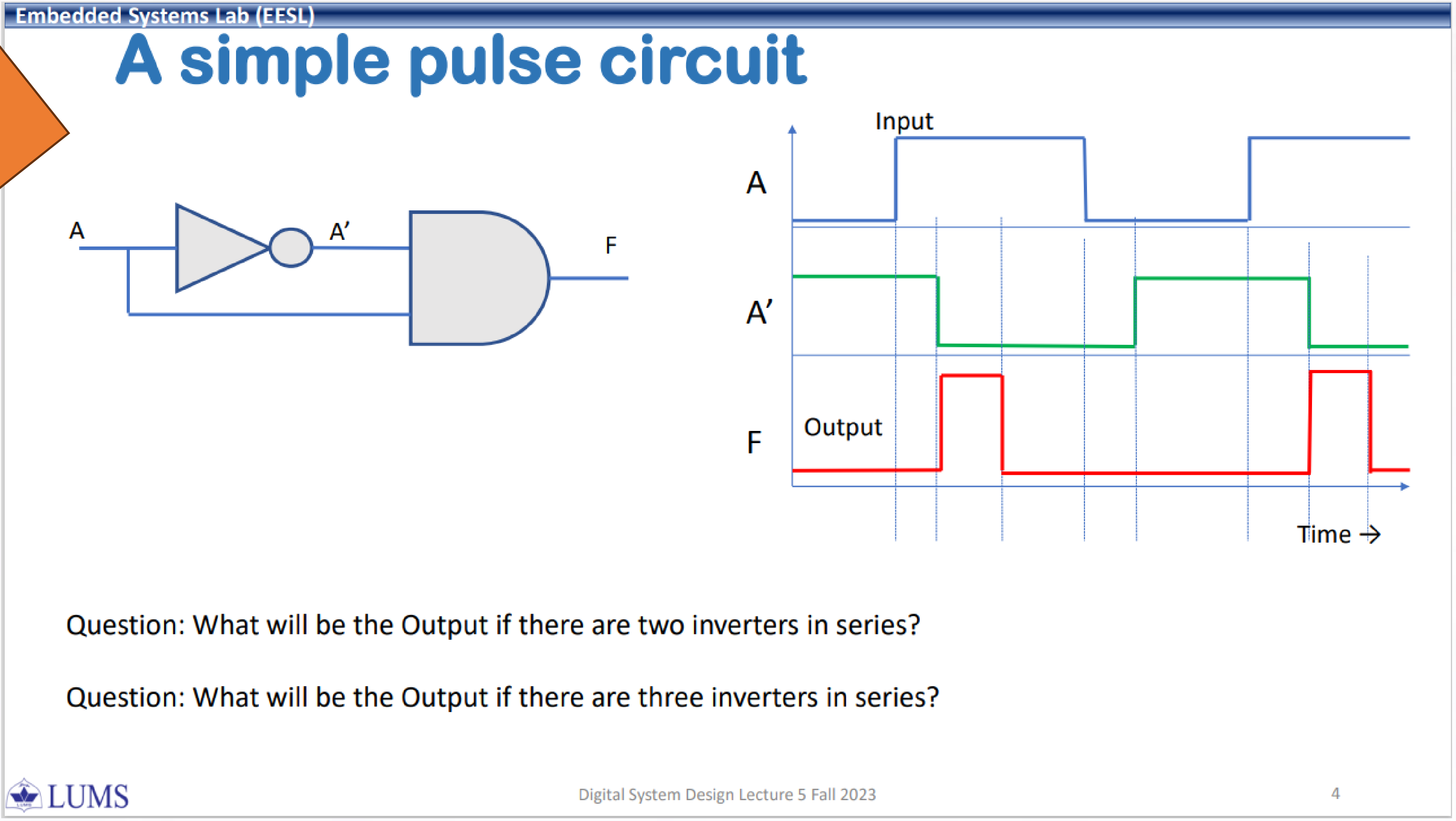
Possible operators in assign

Operator Type	Operator Symbol	Operation Performed	Number of Operands
Arithmetic	*	Multiply	Two
	/	Divide	Two
	+	Add	Two
	-	Subtract	Two
	%	Modulus	Two
Logical	!	Logical negation	One
	&&	Logical and	Two
	||	Logical or	two
Relational	>	Greater than	Two
	<	Less than	Two
	>=	Greater than or equal	Two
	<=	Less than or equal	Two
Equality	==	Equality	Two
	!=	Inequality	Two
	===	Case equality	Two
	!==	Case inequality	Two

Operator Type	Operator Symbol	Operation Performed	Number of Operands
Equality	==	Equality	Two
	!=	Inequality	Two
	===	Case equality	Two
	!==	Case inequality	Two
Bitwise	~	Bitwise NOT	One
	&	Bitwise AND	Two
	|	Bitwise OR	Two
	^	Bitwise XOR	Two
	^~ or ~^	Bitwise XNOR	Two
Reduction	&	Reduction AND	One
	~&	Reduction NAND	One
	|	Reduction OR	One
	~|	Reduction NOR	One
	^	Reduction XOR	One
	~^ or ^~	Reduction XNOR	One
Shift	<<	Left shift	Two
	>>	Right Shift	Two
Concatenation	{ }	Concatenation	Any Number
Replication	{ { } }	Replication	Any Number
Conditional	? :	Condition	Three

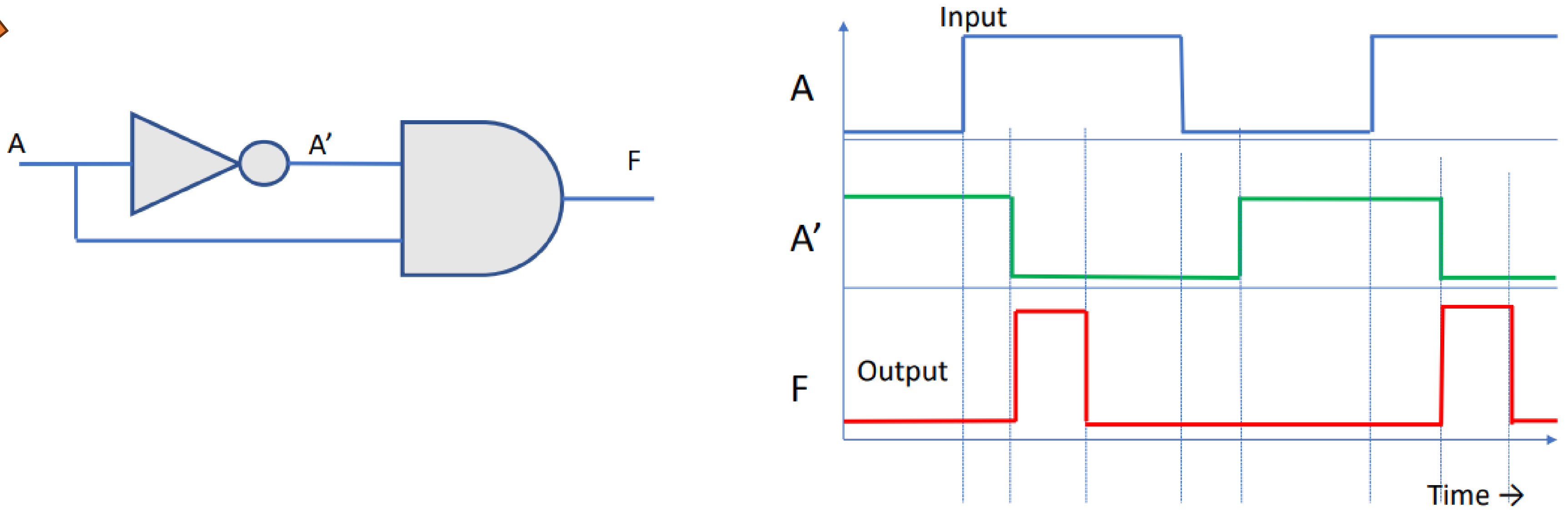
From lecture 5 in class

How will you model this with delays?



Embedded Systems Lab (EESL)

A simple pulse circuit



Question: What will be the Output if there are two inverters in series?

Question: What will be the Output if there are three inverters in series?

LUMS Digital System Design Lecture 5 Fall 2023 4

LUMS A Not-for-Profit University