NMK20603

Computer Architecture

MY1

Verilog Basics

MY1

😎 Note:

- mainly, Verilog95!
- some 2K1 (will state)

MY1

Describe:

A logic block

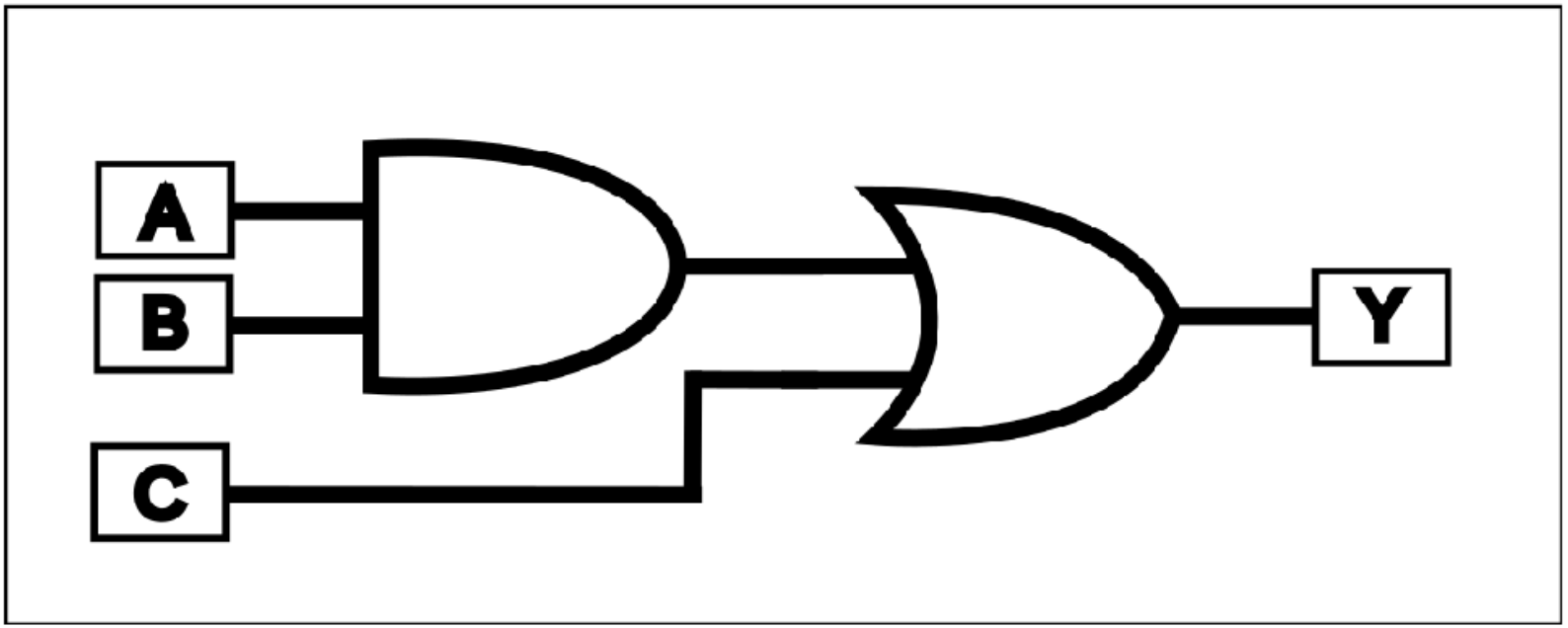
3 Input 1 Output

MY1

Schematic!

MY1

mylogic



MY1

Basic Construct: Module!

MY1

```
module mylogic ();
endmodule
```

Parenthesis holds port list (csv)

Describe:

Ports!

MY1

```
module mylogic (a,b,c,y);
endmodule
```

Ports can be:

- input
- output
- inout

Use input &

output ONLY!

MY1

Describe:

Port Types!

MY1

```
module mylogic (a,b,c,y);
input a,b,c;
output y;
endmodule
```

MY1

Describe:

Logic Elements!

MY1

Data types

- net (keyword:wire)
- variable (keyword: reg)

MY1

Keywords

- assign (wire)
- always/initial (reg)

MY1

RULE:

comb. logic

use assign/wire only!

MY1

always/reg

only for seq. logic

(and tb)

MY1

RULE:

allowed operators

AND, OR, INV

MY1

```
module mylogic (a,b,c,y);
input a,b,c;
output y;
wire y;
assign y = (a & b) | c;
endmodule
```

MY1

RULE:

2-input logic gates ONLY

MY1

Need temporary signal ⇒ wire!

```
module mylogic (a,b,c,y);
input a,b,c;
output y;
wire y, t;
assign t = a & b;
assign y = t | c;
endmodule
```

MY1

RULE:

All signals @wire must be declared!

Done...

⇒ save file!

RULE:

1 file 1 module

RULE:

file name ===

module name (.v)

How to test?

MY1

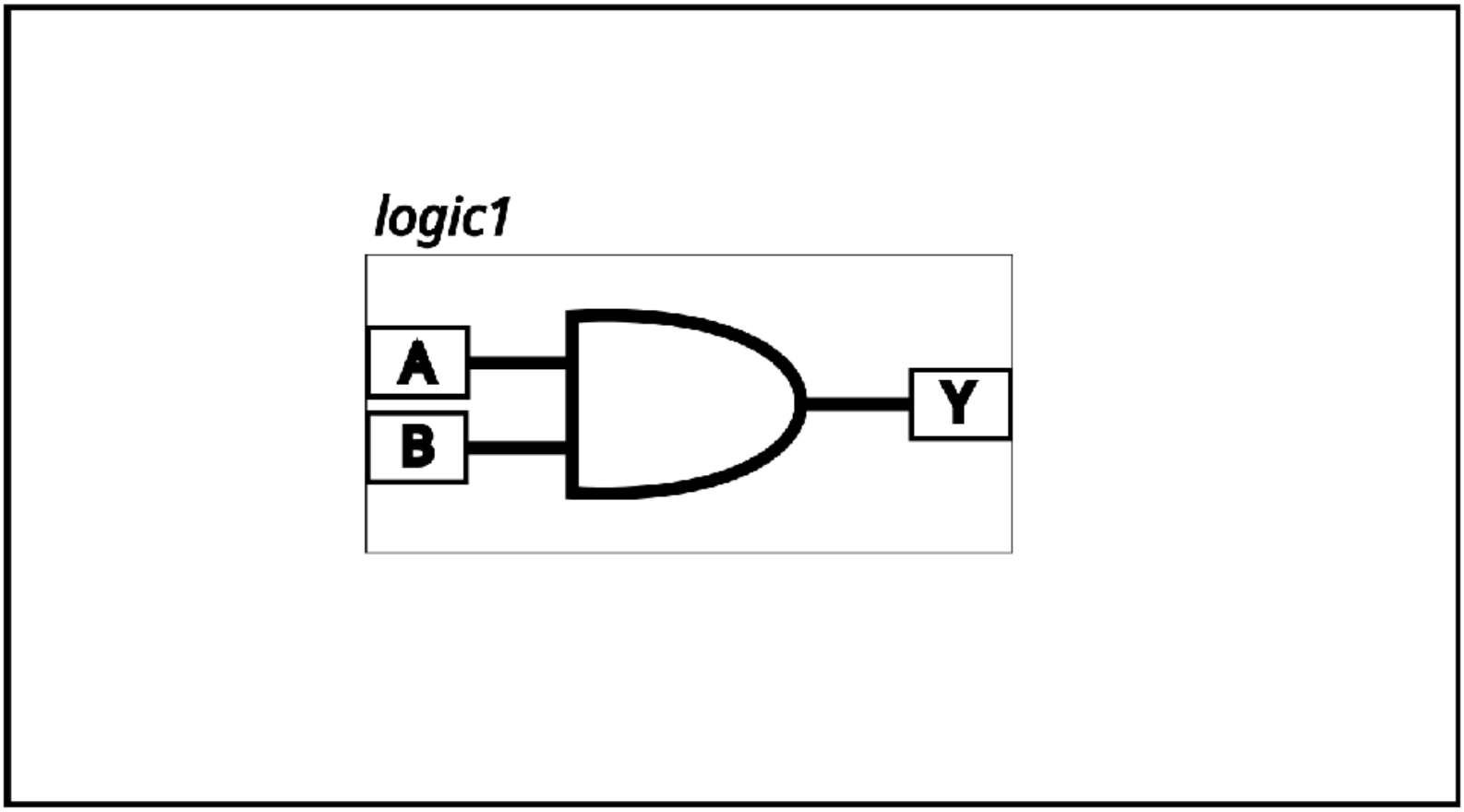
Imagine:

Breadboard Testing

logic gate (IC)

MY1

test?



MY1

...

⇒ generate stimuli

⇒ monitor reponse

MY1

RULE:

All module MUST have a testbench

A testbench (tb)

is also a module!

RULE:

Tb name ===

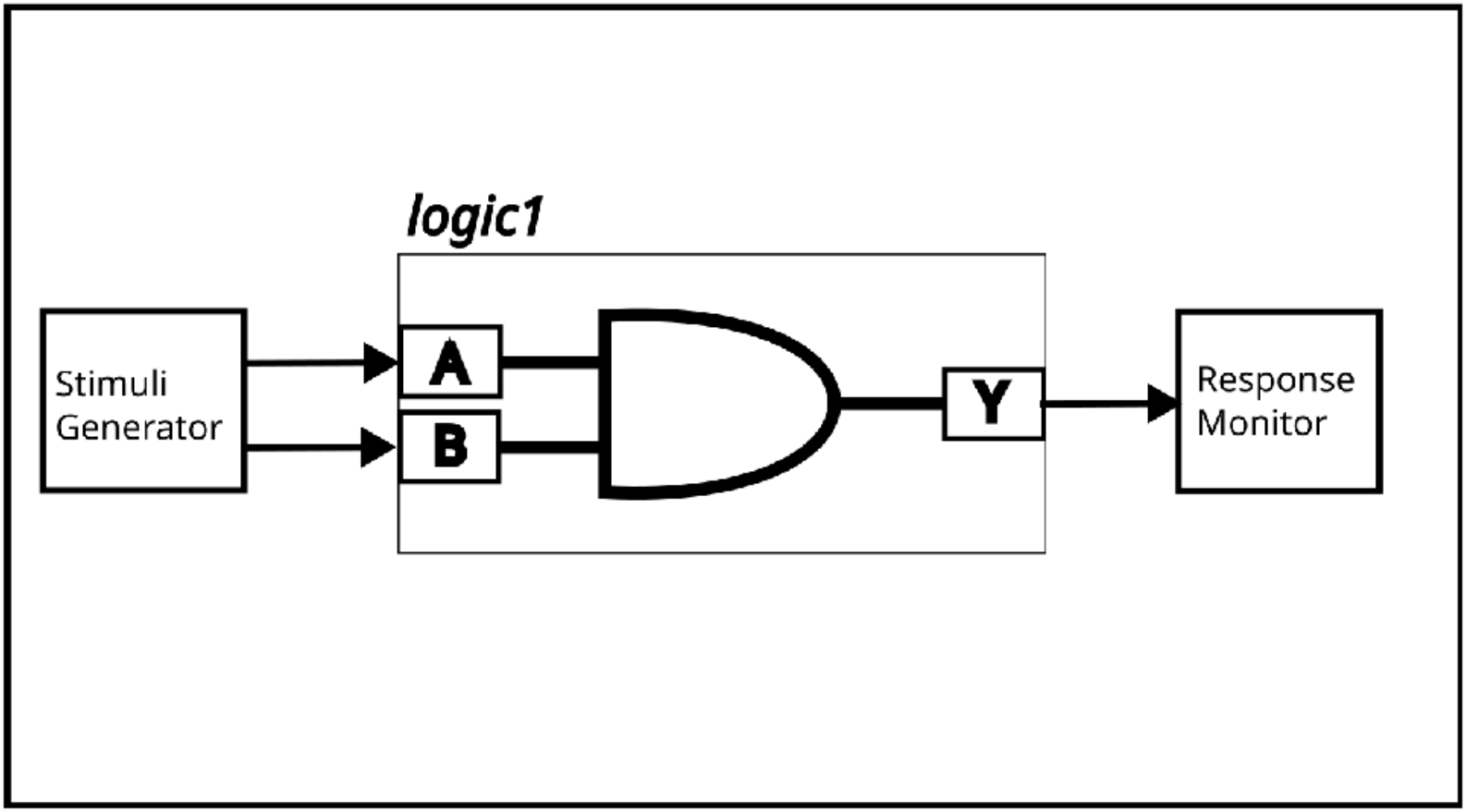
module name + _tb

MY1

Schematic!

MY1

logic1_tb



MY1

Describe:

Testbench

MY1

```
module mylogic_tb ();
endmodule
```

Tb needs

NO PORT!

Design Hierarchy:

Instantiate a module

MY1

```
module mylogic_tb ();
mylogic dut (P,Q,R,Z);
endmodule
```

MY1

RULE:

Port connection(s)

by ordered list ONLY!

MY1

- dut === design under test

Specify

Tb nets

@ 'variables'

```
module mylogic_tb ();
reg P,Q,R; // dut inputs
wire Z; // dut output
mylogic dut (P,Q,R,Z);
endmodule
```

MY1

Describe:

Stimuli 'Generator'

MY1

Keyword

- initial
  - ⇒ like always, but once
  - → block: begin & end

```
module mylogic_tb ();
reg P,Q,R;
wire Z;
initial
begin
end
mylogic dut (P,Q,R,Z);
endmodule
```

MY1

Input Sequence Generator?

MY1

Needs:

- loop counter

Keyword

- integer (not int!)

```
module mylogic_tb ();
reg P,Q,R;
wire Z;
integer loop;
initial
begin
  for (loop=0;loop<8;loop=loop+1)
  begin
  end
end
end
mylogic dut (P,Q,R,Z);
endmodule
```

MY1

Connect input

3x1-bit as 3-bit?

MY1

Syntax

- {} pair

⇒ concatenation op

```
module mylogic_tb ();
reg P,Q,R;
wire Z;
integer loop;
initial
begin
  for (loop=0;loop<8;loop=loop+1)
  begin
    {P,Q,R} = loop;
    #10; // delay to 'see' change! TU!
  end
end
mylogic dut (P,Q,R,Z);
endmodule
```

MY1

delay (#) is in

variable TU (time-unit)

MY1

Done...

⇒ save file!

Homework:
Verilog code for given logic

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0