Computer Architecture

Week 4: System Verilog Tutorial I

Fenerbahçe University

Professor & TAs

Prof: Dr. Vecdi Emre Levent

Office: 311

Email: emre.levent@fbu.edu.tr

TA: Arş. Gör. Uğur Özbalkan

Office: 311

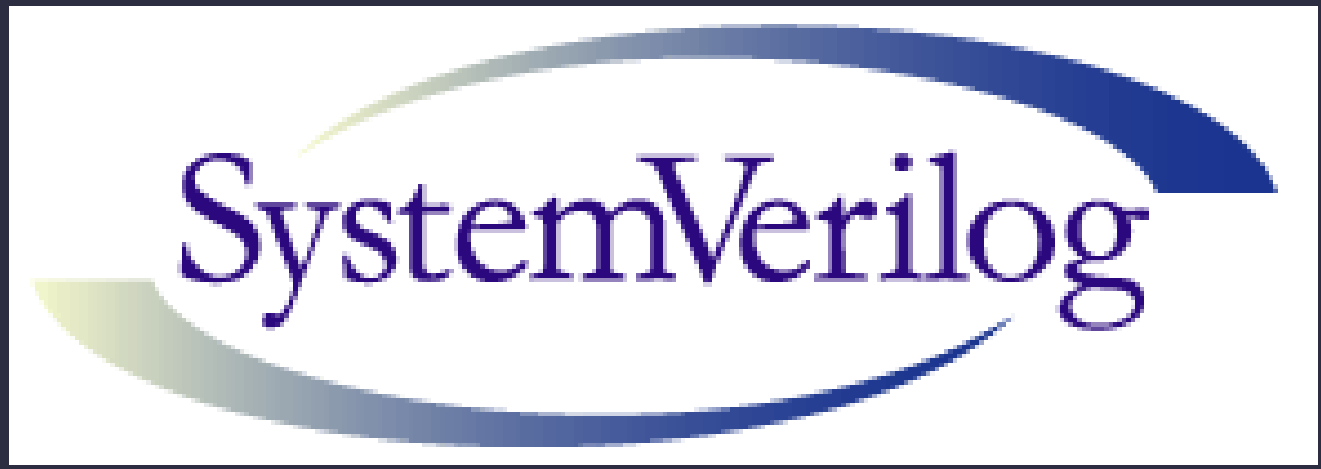
Email: ugur.ozbalkan@fbu.edu.tr

Course Plan

- System Verilog for Synthesis I

System Verilog

- Developed over Verilog Language
- Lots of new features



System Verilog

- Most of the additional features are for verification
- Some features can also be used for synthesis

- As of March 2021, the number of job positions containing System Verilog keyword on Linkedin is approximately **4000**.

System Verilog

- IEEE Standard, IEEE 1800
- First release; 2002

System Verilog

- In this course, the synthesis features of System Verilog will be covered.

Logic Data Type

- In Verilog Language we use;
  - Reg
  - Wire

Data types

Logic Data Type

- Reg data type are using at always blocks
- Wire data type are using at assign blocks

Logic Data Type

- Reg data type means Register, however it doesnt mean a register always
- For ex.
  - always@(*) block, synthesizer will synthesize combinational logic, so it will be wire

Logic Data Type

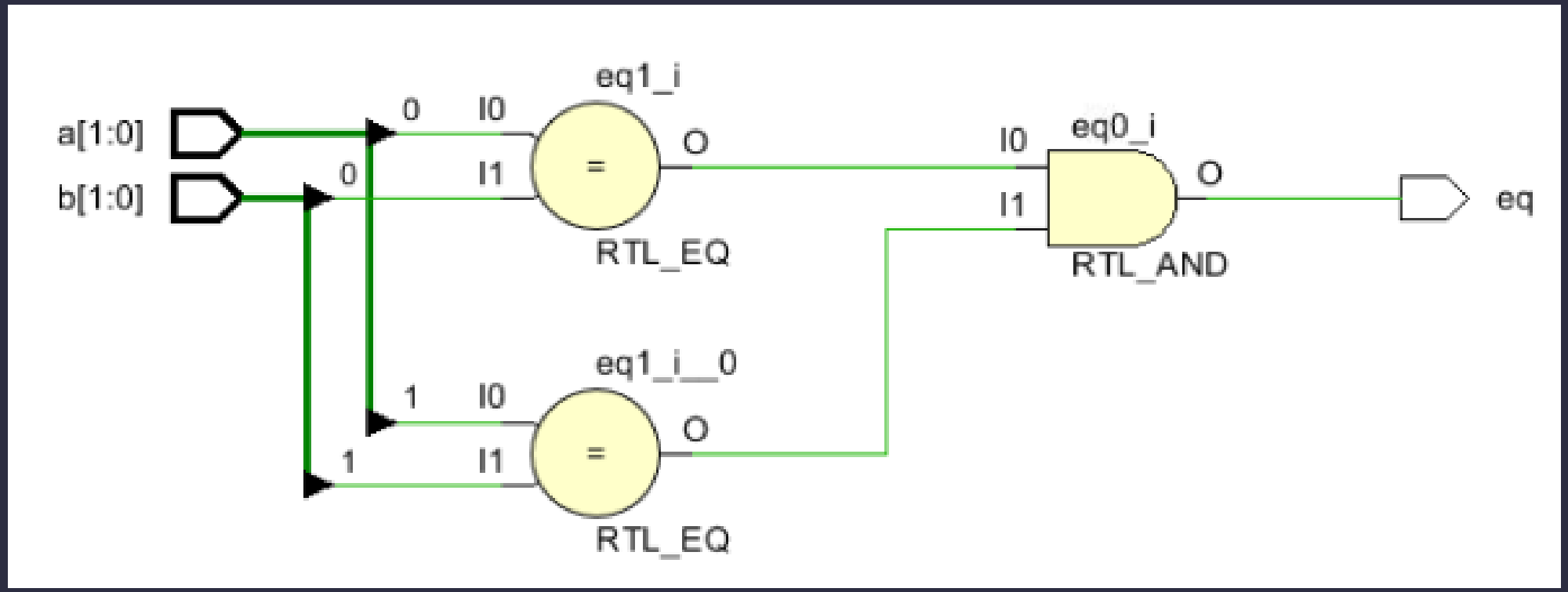
- To overcome this complexity, a new data type called "logic" has been created.
- Whether a logic data type variable will be a wire or a register is decided by the synthesizer at the end of the synthesis process.

Logic Data Type

```
`timescale 1ns / 1ps

module top
(   input logic[1:0] a, b,
    output logic eq
);

always @(*) begin
    if (a[0]==b[0] && a[1]==b[1])
        eq = 1;
    else
        eq = 0;
end

endmodule
```

Logic Data Type
Combinational Circuit
Example

Logic Data Type



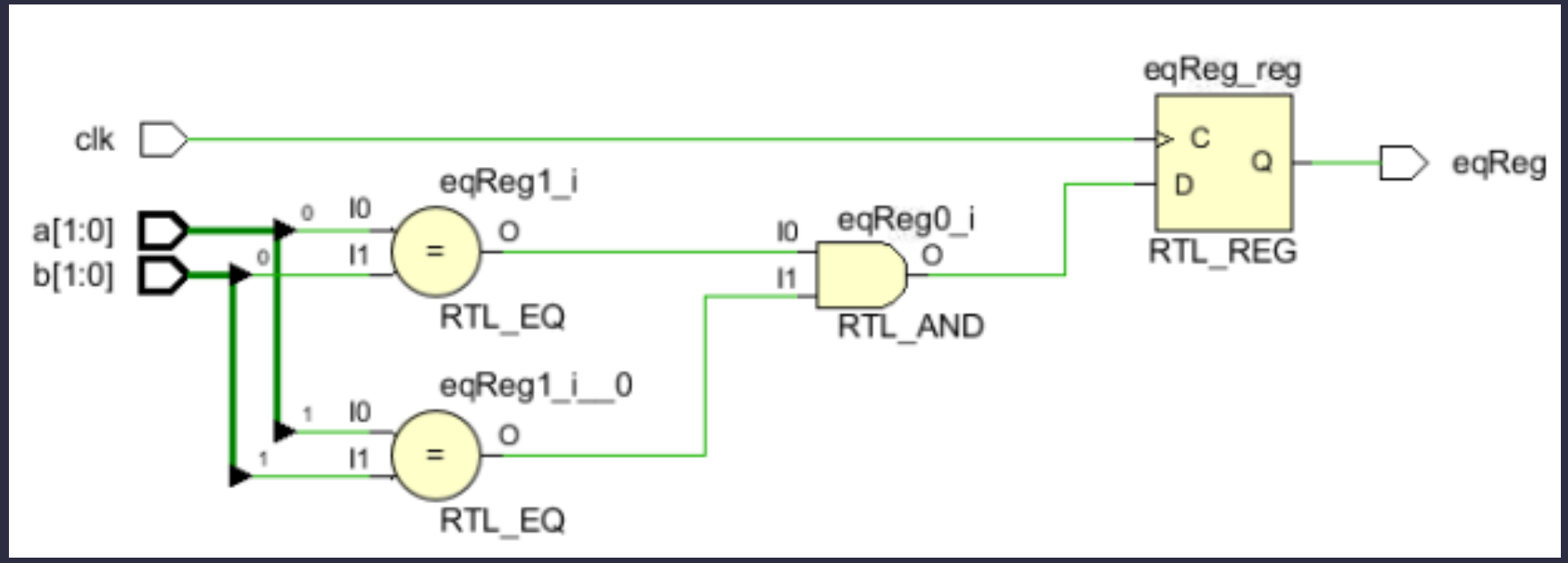
Synthesized Logic

Logic Data Type

```
`timescale 1ns / 1ps
module top
(
    input logic clk,
    input logic[1:0] a, b,
    output logic eqReg
);

logic eq;

always @(*) begin
    eq = eqReg;
    if (a[0]==b[0] && a[1]==b[1])
        eq = 1;
    else
        eq = 0;
end

always@(posedge clk) begin
    eqReg <= eq;
end

endmodule
```

Logic Data Type Sequential Circuit Example

Logic Data Type



Synthesized Logic

Always Blocks

- In Verilog language, logic designed in always blocks can be combinational or sequential circuit as a result of the synthesis.
- Before synthesizing, it is not known whether the logic in always blocks will be sequential or combinational circuitry.

Always Blocks

- Also, the synthesis tool takes time to make this inference (Combinational or Sequential Circuit).

Always Blocks

- When designing a combinational circuit in structures such as state machines, if the output of the combinational circuit is forgotten to be encoded in all cases, a latch will occur.
- Systemverilog language has 3 different always blocks to eliminate such problems.

Always Blocks

New blocks:

- always_comb
- always_latch
- always_ff

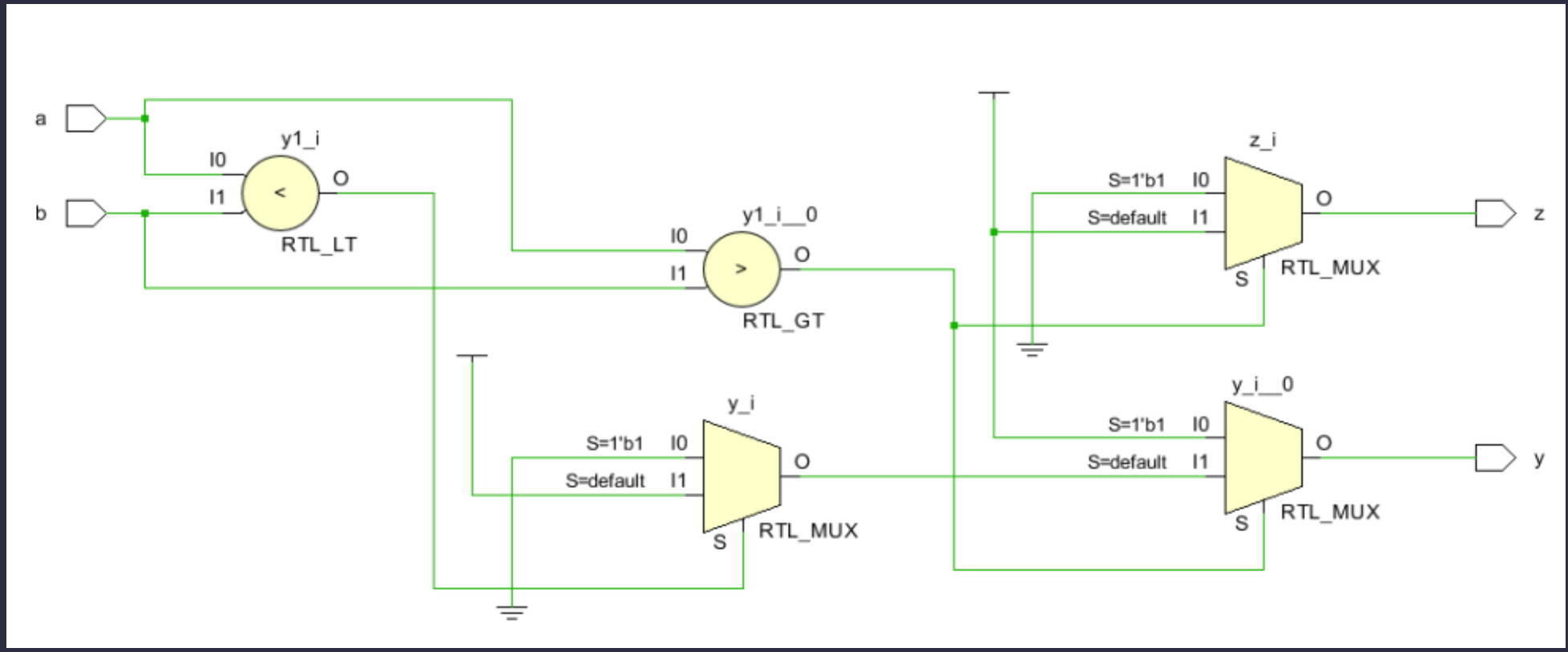
always_comb block:

- Combinational circuits are designed in the always comb block.
- There is no need to write a sensitivity list.

always_comb:

```
module always_comb_test
(
    input logic a, b,
    output logic y, z
);
always_comb begin
    if (a > b) begin
        y = 1;
        z = 0;
    end
    else if (a < b) begin
        y = 0;
        z = 1;
    end
    else begin
        y = 1;
        z = 1;
    end
end
endmodule
```

always_comb:



Synthesized Logic

always_comb block:

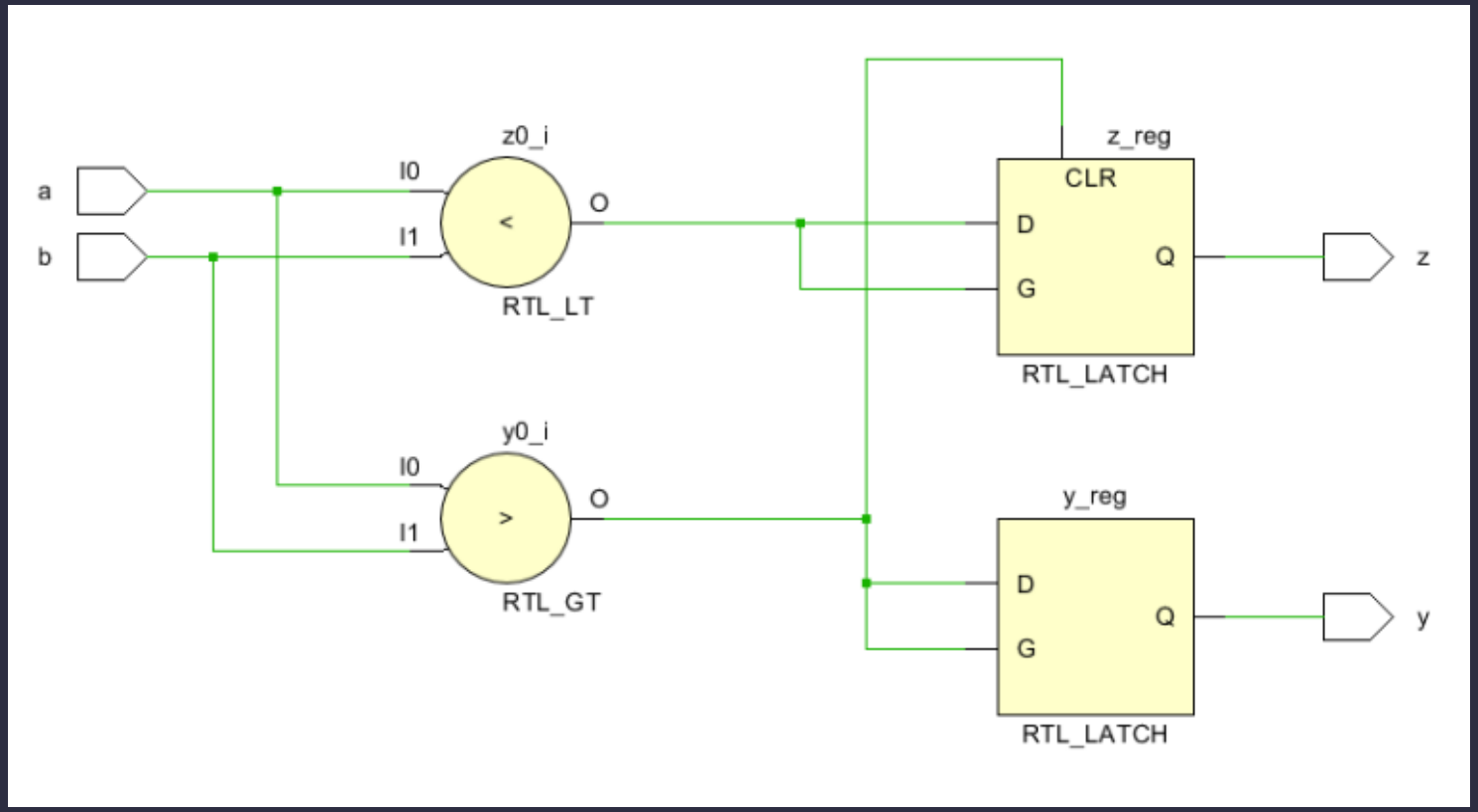
- If latch occurs in the design, the synthesizer will warn during synthesis.

always_comb block:

Latch example
at always comb block

```
module top
(
    input logic a, b,
    output logic y, z
);

always_comb begin
    if (a > b) begin
        y = 1;
        z = 0;
    end
    else if (a < b) begin
        z = 1;
    end
end

endmodule
```

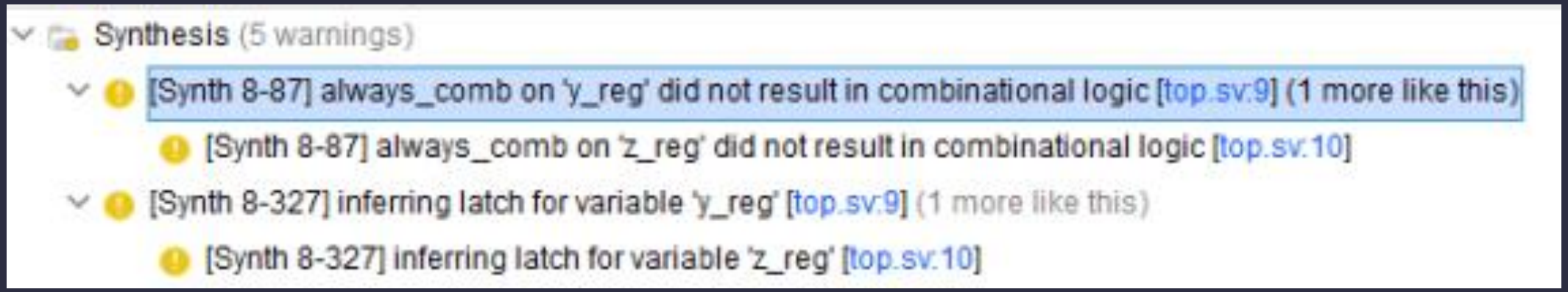
always_comb:



Synthesized Logic

always_comb block:

- During synthesizing the design, synthesis tool output a latch warning.



always_lat ch block:

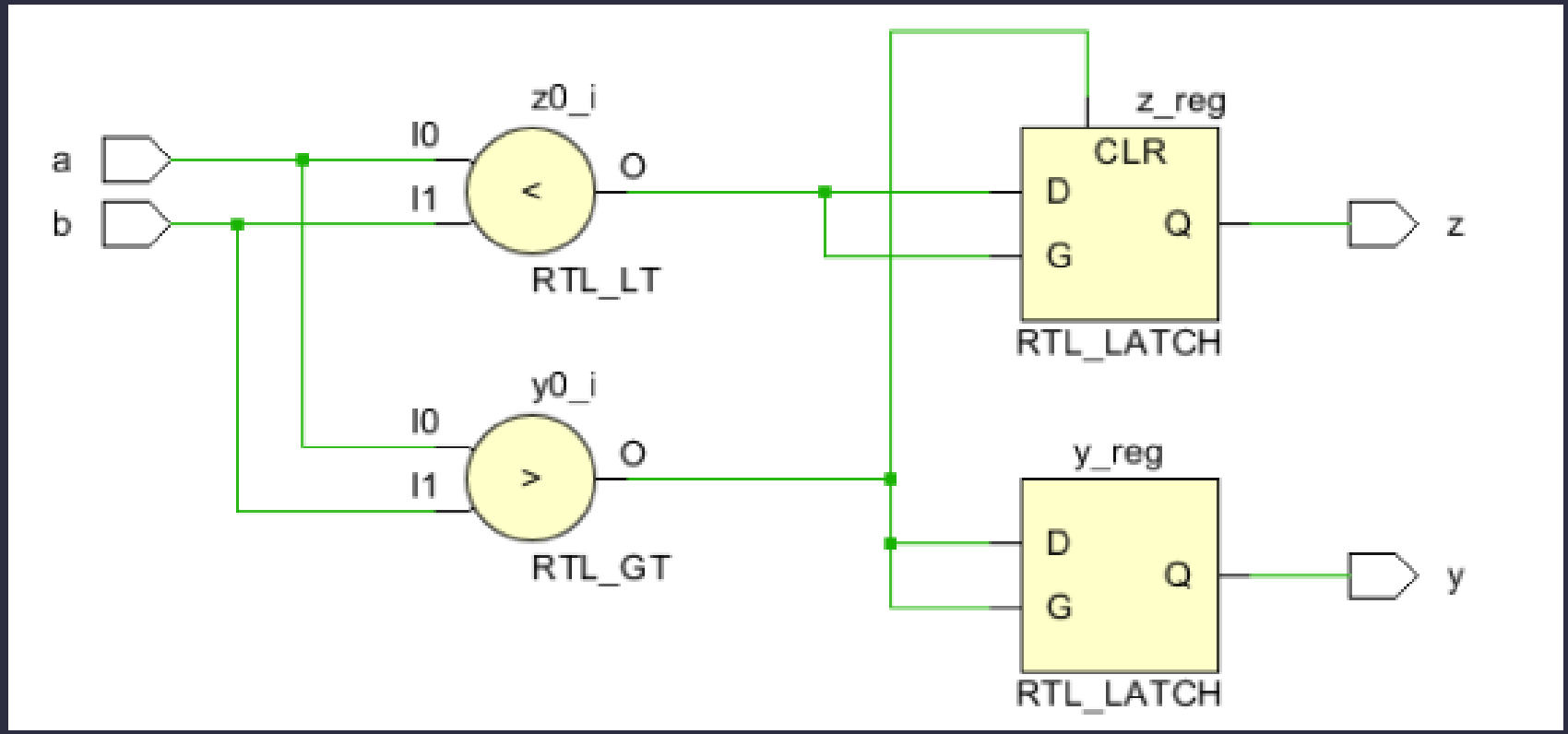
- The always_latch block can be used in designs that require the use of latch.
- Synthesis tool wont warn latch when the design contains latch.

always_lat ch block:

Latch example

```
module top
(
    input logic a, b,
    output logic y, z
);

always_latch begin
    if (a > b) begin
        y = 1;
        z = 0;
    end
    else if (a < b) begin
        z = 1;
    end
end

endmodule
```

always_lat ch block:



always_lat ch block:

- When the design is synthesized, it will not give any warning / error (Vivado outputs latch warning, not in Menthor Graphics Precision RTL synthesis tool).
- Likewise, when a design that does not contain a latch is placed in a latch block, synthesis tool will output a warning.

always_ff block:

- This block is used only to make sequential circuit definitions.
- The Sensitivity list is used.

System Verilog

always_ff block:

Flip-flop example

```
module top
(
    input logic clk, reset,
    output logic y, z
);
always_ff @(posedge clk, posedge reset) begin
    if (reset) begin
        y <= 0;
        z <= 0;
    end
    else begin
        y <= 1;
        z <= 1;
    end
end
endmodule
```

always_ff block: