Laboratory practice 3

4x4-bit multipliers

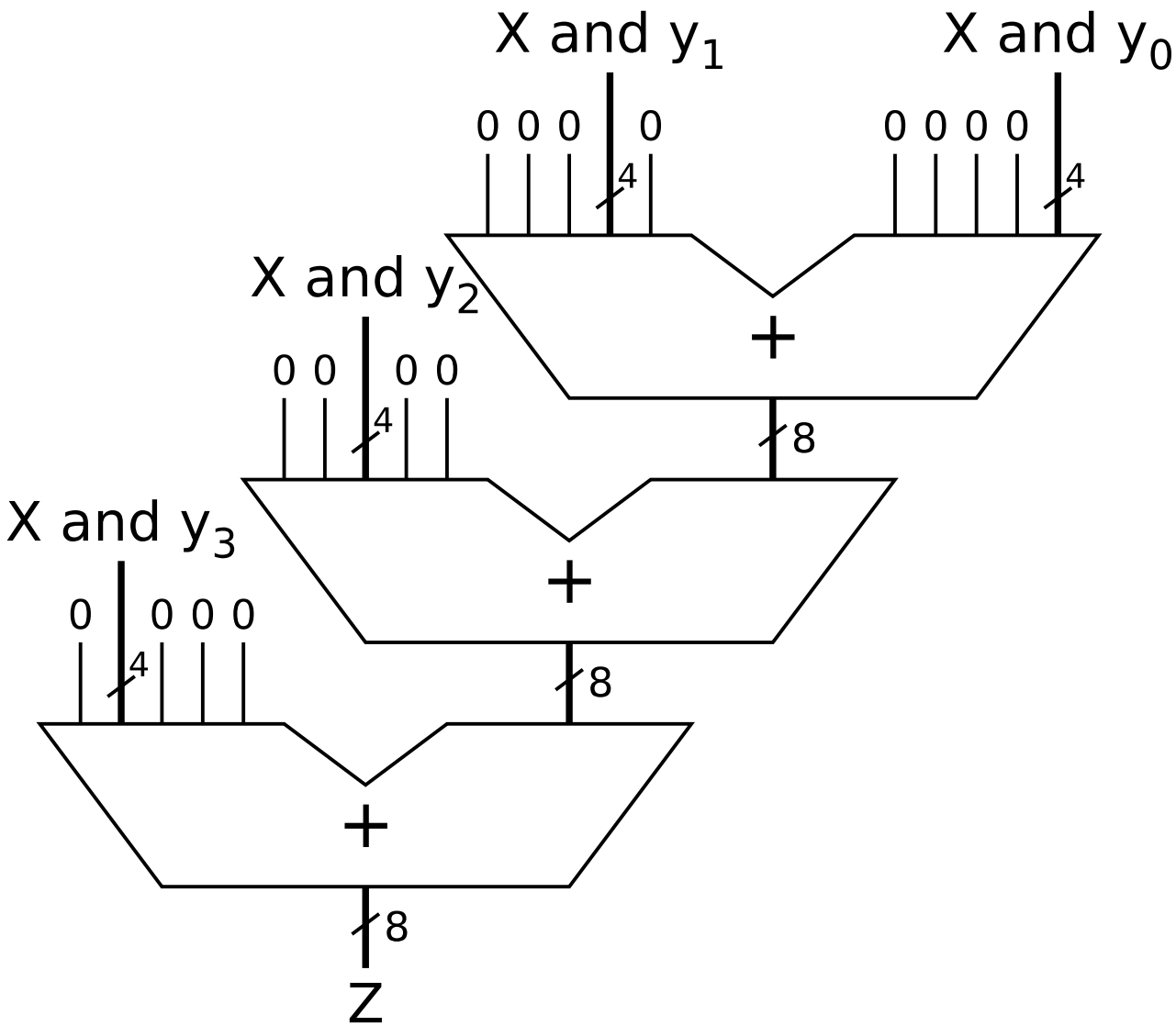
Objective

- Implement two different 4x4-bit multipliers:
  - Using the `numeric_std` library operator '*'
  - Using adders (from Lab 1)
- Study the Vivado reports to find:
  - The combinational elements that were used
  - Then maximum combinational path delay

8-bit adders

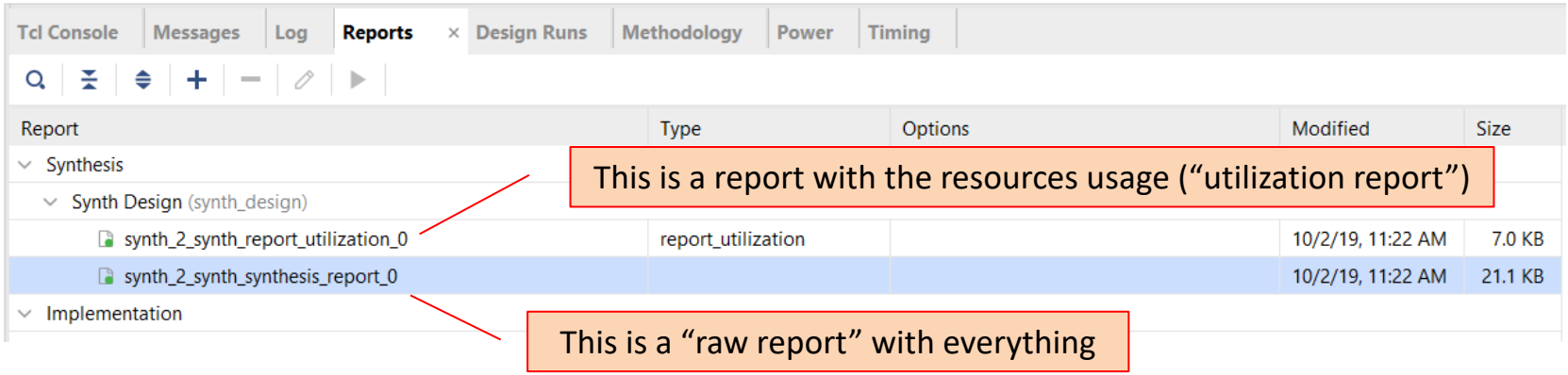
```
entity mult8b is
  port(
    X : in  std_logic_vector(3 downto 0);
    Y : in  std_logic_vector(3 downto 0);
    Z : out std_logic_vector(7 downto 0)
  );
end mult8b_std;
```

8-bit adders



Synthesis reports

- After the synthesis, in the Reports tab (down), you can see two synthesis reports:

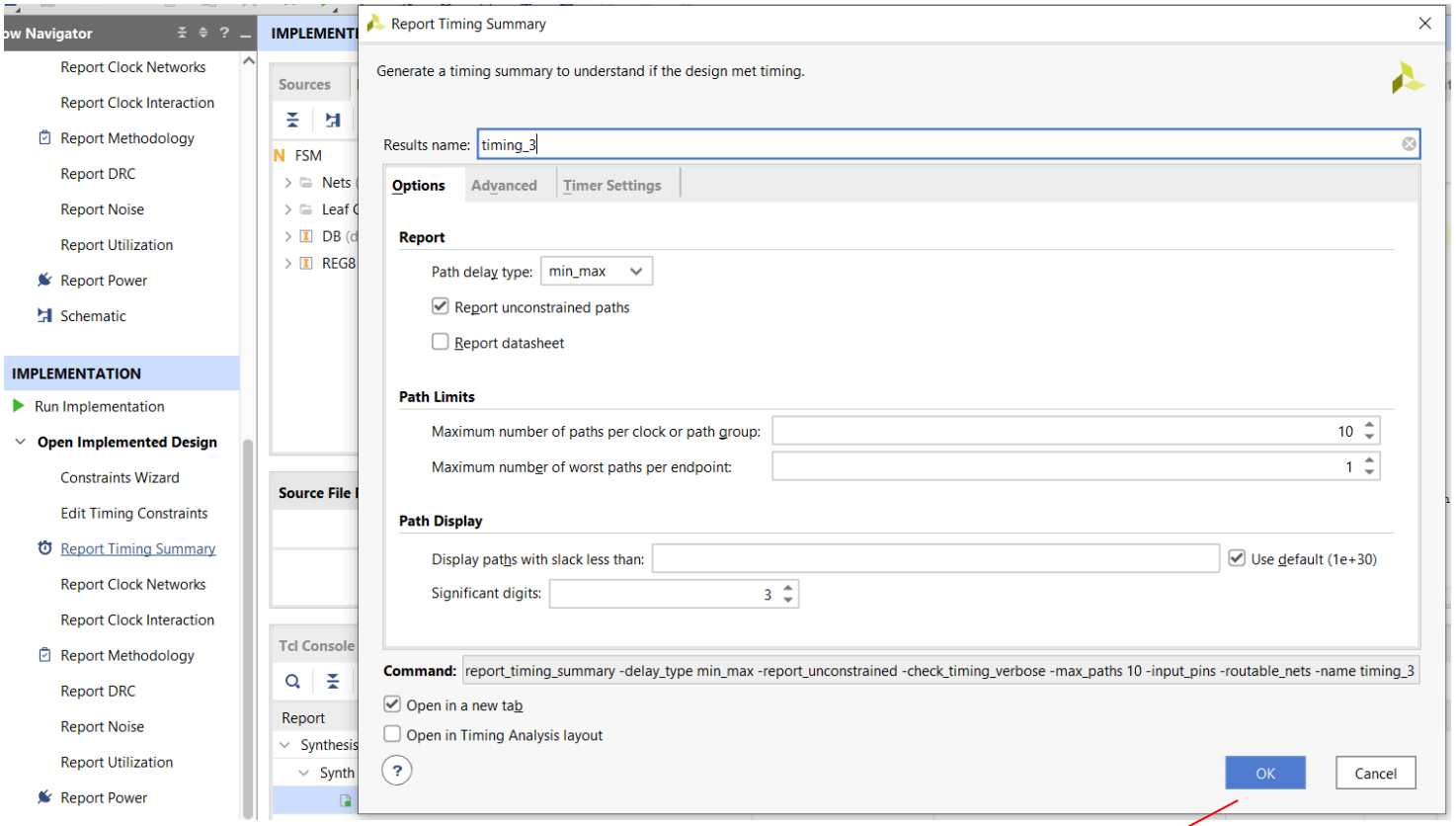


- For instance (utilization report, taken from Lab 2):

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	46	0	20800	0.22
LUT as Logic	46	0	20800	0.22
LUT as Memory	0	0	9600	0.00
Slice Registers	35	0	41600	0.08
Register as Flip Flop	35	0	41600	0.08
Register as Latch	0	0	41600	0.00
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

Timing reports

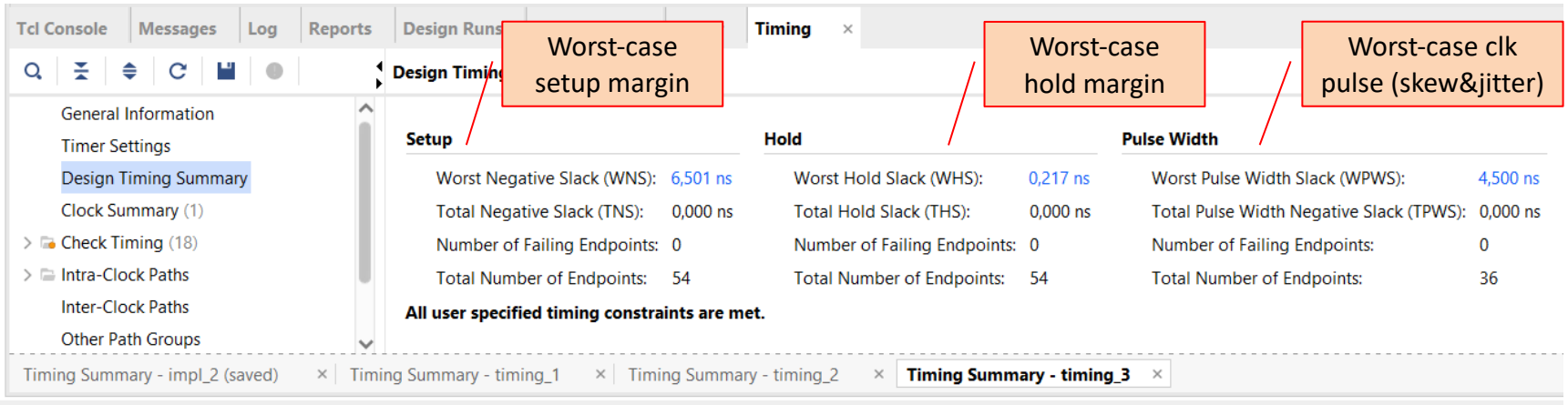
- Under “Open Implemented Design”, both in SYNTHESIS and in IMPLEMENTATION, you can click on “Report Timing Summary” (the timing summary after the IMPLEMENTATION is more accurate)



This window appears, when clicking on OK, the report is generated

Timing reports

- These reports are visible by clicking on the “Timing” tab (down in the Vivado GUI):



- **“All user specified timing constraints are met”** → How does the tool know which is the input clock period that is used in this design? **By means of the .xdc file**:

```
create_clock -add -name clk -period 10.00 -waveform {0 5} [get_ports clk]
```

- That line (which we already used in .xdc files in Labs 1&2) characterizes the clk signal, which is taken from the oscillator coming from pin W5 (in the Nexys4 board).
  - It defines a clock period (10 ns), when it becomes 0 (0 ns) and when it becomes to 1 (5 ns)
- No timing analysis is possible without this line, neither is the tool able to place&route the hardware according to the temporal constraints.
- This line is also visible when clicking on “Clock Summary” (one of the sections of the report – left)

Testbench

```
-- Stimulus process
p_stim : process
  variable v_i : natural := 0;
  variable v_j : natural := 0;
begin
  i_loop : for v_i in 0 to 15 loop
    j_loop : for v_j in 0 to 15 loop
      X      <= std_logic_vector(to_unsigned(v_i, 4));
      Y      <= std_logic_vector(to_unsigned(v_j, 4));
      Z_xpct <= std_logic_vector(to_unsigned(v_i * v_j, 8));
      wait for 5 ns;
      assert Z = Z_xpct
        report "Error multiplying, "&integer'image(v_i)& " * "
               &integer'image(v_j)& " = "&integer'image(v_i*v_j)&
               " not "&integer'image(to_integer(unsigned(Z)))
        severity error;
      wait for 5 ns;
    end loop j_loop;
  end loop i_loop;
  wait;
end process p_stim;
```