MEng VLSI - Static Timing Analysis + Pipelining

Prof. Dr.-Ing. Friedrich Beckmann

Hochschule Augsburg

How fast can we compute?

Timing Analysis - Latency and Throughput - Example ADC

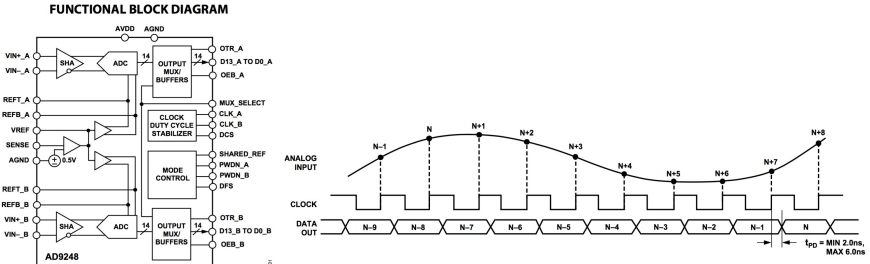


Figure 2. Timing Diagram

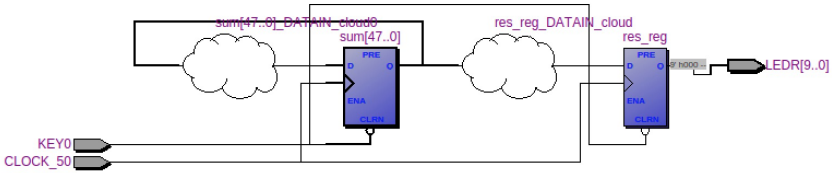
- Maximum Conversion Rate 65 MSamples/s $\Rightarrow$ Throughput ist 65 MHz
- I takes 7 clock cylces from analog in to digital out $\Rightarrow$ Latency is 7 clock cylces

Example circuit: Counter + Majority VHDL Code

```
1  architecture rtl of de1_sta is
2  function cnt_ones (x : unsigned) return integer is
3    variable cnt : integer range 0 to x'length;
4  begin
5    cnt := 0;
6    for i in x'range loop
7      if x(i) = '1' then
8              cnt := cnt + 1;
9          end if;
10   end loop;
11   return cnt;
12 end function;
13 signal sum : unsigned(47 downto 0);
14 signal res, res_reg : std_ulogic;
15 signal no_of_ones : integer range 0 to sum'length;
16 begin
17 res <= '1' when no_of_ones > sum'length/2 else '0';
18 res_reg <= '0' when rst_n = '0' else res when rising_edge(clk);
19 sum <= (others => '0') when rst_n = '0' else sum + 1 when rising_edge(clk);
20 no_of_ones <= cnt_ones(sum);
21 end architecture;
```

https://gitlab.elektrotechnik.hs-augsburg.de/haf/2023-vlsi-lab-problems/-/blob/main/src/de1_sta.vhd

Output is '1', if more than 1/2 of counter bits are '1'



What limits the maximum clock frequency?

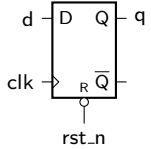
D-Flipflop Timings

Setup- and Holdtime

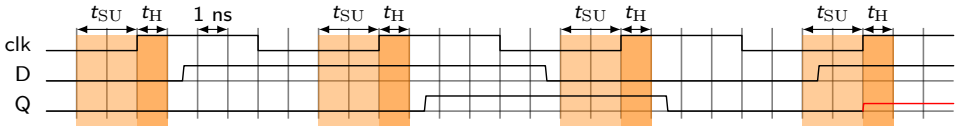
The D-Flipflop stores the data at the D-Input at a rising edge at the clock input. The signal at the D input must be stable during an interval around the clock edge. The interval is defined by the setuptime t_{SU} and the holdtime t_{H}. The setuptime is the time before the clock edge and the holdtime is the time after the edge.

Clock to Output Delay

The state of the flipflop will be visible after the Clock-to-Output Delay t_{CQ} after the rising edge of the clock signal at the output Q.

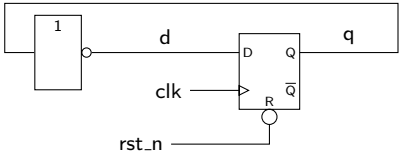


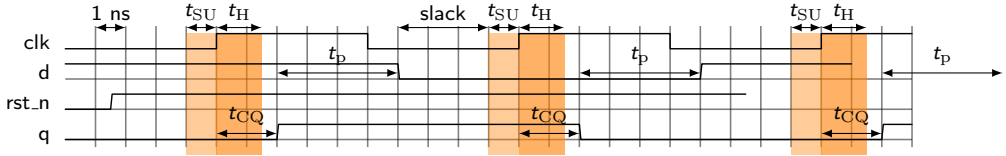
Parameter	Abb.	t / ns
Setuptime	t_{SU}	2
Holdtime	t_{H}	1
Clock to Output Delay	t_{CQ}	1,5



Simple Circuit - slack and maximum clock frequency

$t_{p,Inverter} = 4ns, t_{CQ} = 2ns, t_{setup} = 1ns, t_{hold} = 1.5ns, f_{\mathrm{clk}} = 100\mathrm{MHz}$

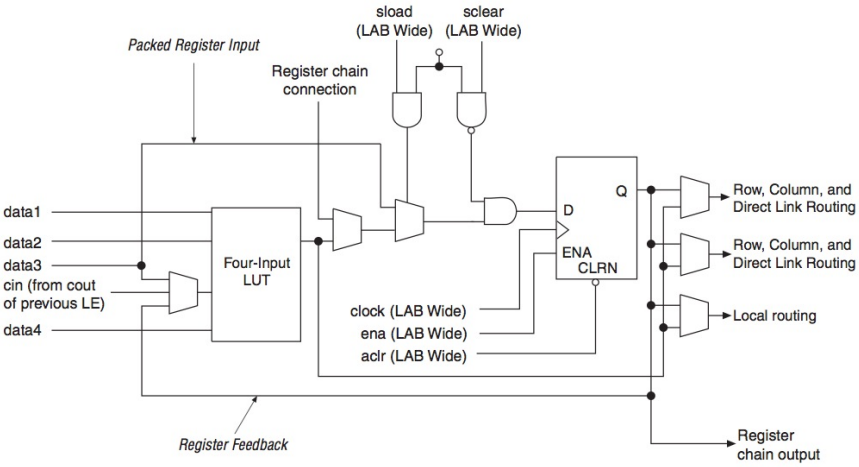




- Maximum clock speed depends on combinational delay of the logic, Clock to output delay and Setup time of FF.
- The signal at the d input of a FF has to be stable for the setup time before the next clock edge.
- The slack is the margin until a timingviolation will happen. Here the slack is 3ns.
- Minimum possible clock period 7ns (slack = 0ns). Maximum clock frequency: 142 MHz.

FPGA - Logic Elements

Figure 2–3. LE in Normal Mode

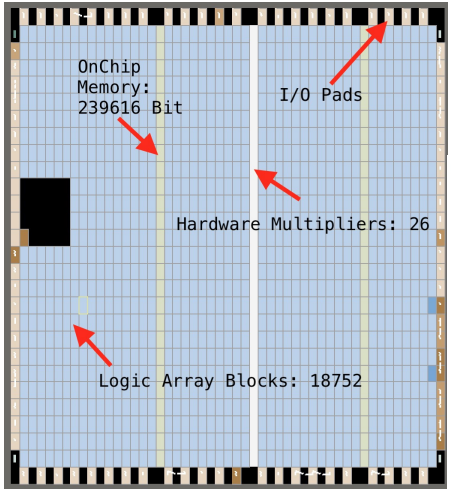


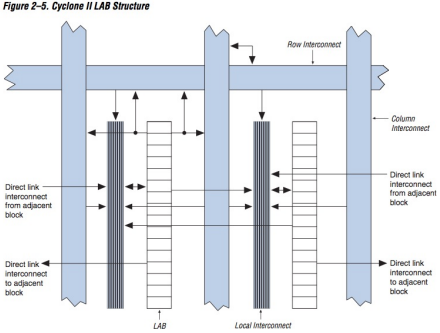
Timings of the Cyclone II FPGA

Table 5–16. LE_FF Internal Timing Microparameters

Symbol	Parameter	-6 Speed Grade		-7 Speed Grade		-8 Speed Grade		Unit
		Min	Max	Min	Max	Min	Max	
t_{SU}	LE register setup time before clock	35		40		46		ps
t_{H}	LE register hold time after clock	170		190		211		ps
t_{CO}	LE register clock-to-output delay		255		282		310	ps
t_{CLR}	Minimum clear pulse width	191		217		244		ps
t_{PRE}	Minimum preset pulse width	191		217		244		ps
t_{CLKL}	Minimum clock low time	252		306		362		ps
t_{CLKH}	Minimum clock high time	249		304		359		ps
t_{LUT}	LE combinational LUT delay for data-in to data-out		447		556		664	ps

FPGA - Logic Elements + Routing



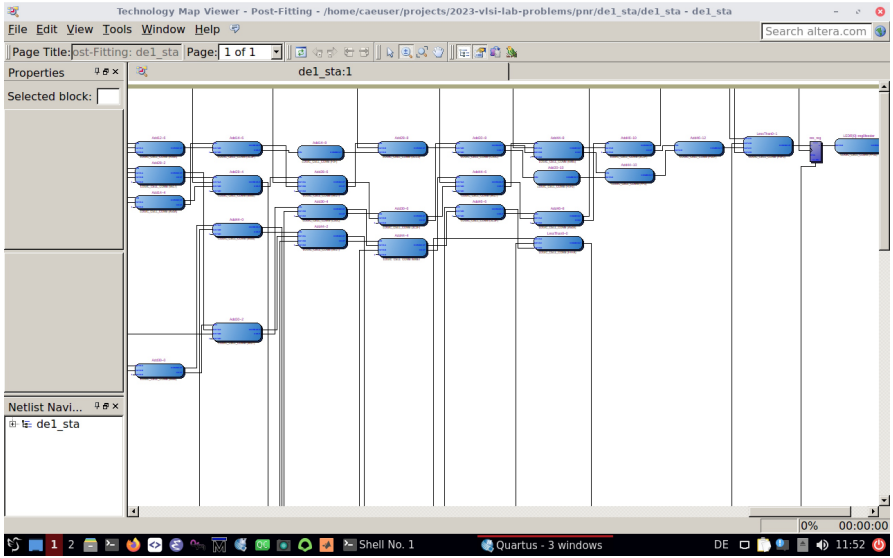


Timings of the Cyclone II FPGA

Table 5–15. Cyclone II Performance Notes (Part 1 of 2) *Notes (1), (2)*

Applications		Resources Used			Performance			
		LEs	M4K Memory Blocks	DSP Blocks	-6 Speed Grade	-7 Speed Grade	-8 Speed Grade	Units
LE	16-to-1 multiplexer *(3)*	11	0	0	382.99	302.84	259.87	MHz
	32-to-1 multiplexer *(3)*	24	0	0	292.74	237.98	204.49	MHz
	16-bit counter	16	0	0	445.23	387.74	341.64	MHz
	64-bit counter	64	0	0	188.82	168.37	150.92	MHz
Memory M4K block	Simple dual-port RAM 128 × 36 bit	0	1	0	260.01	216.73	180.57	MHz
	True dual-port RAM 128 × 18 bit	0	1	0	260.01	216.73	180.57	MHz
	FIFO 128 x 36 bit	24	1	0	260.01	216.73	180.57	MHz

Technology Map Viewer after synthesis

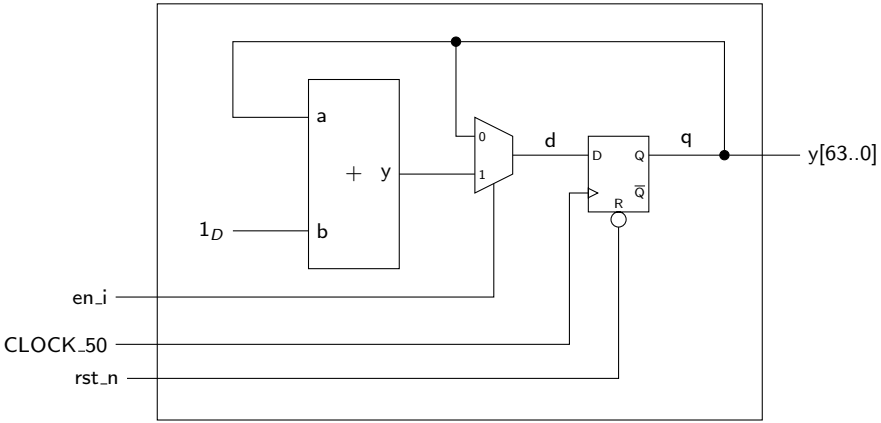


Synopsys Design Constraints file *.sdc

```
1 create_clock -period 20.0 -name CLOCK_50 [get_ports CLOCK_50]
2 set_input_delay -clock CLOCK_50 2 [all_inputs]
3 set_output_delay -clock CLOCK_50 2 [all_outputs]
```

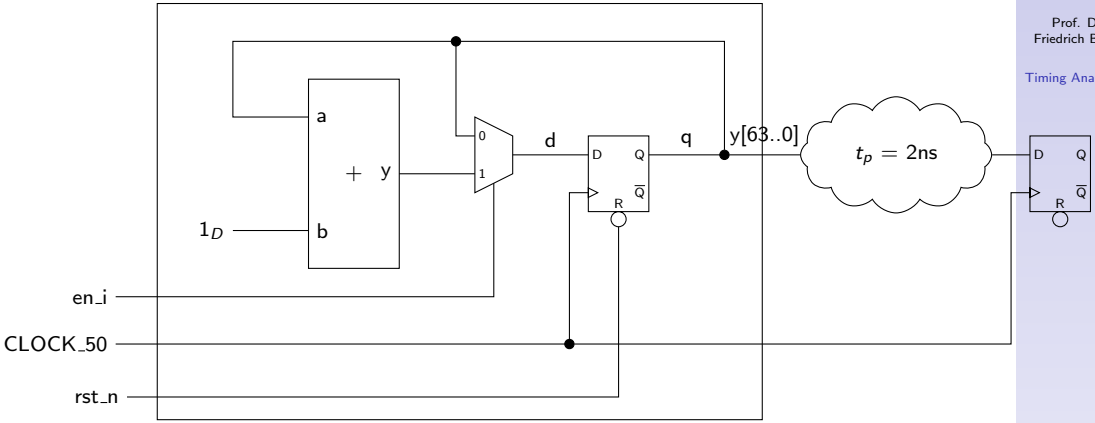
- Define 20 ns clock period = 50 MHz for CLOCK_50 port
- Constrain inputs and outputs with "virtual" registers clocked with CLOCK_50

SDC file - Primary Inputs and Outputs Example



- Almost all paths are constrained by defining the clock period of CLOCK_50
- The paths from the primary inputs (here en_i) ending at the D inputs of the flipflops are missing.
- The paths from the Q outputs of the flipflops to the primary outputs (y[63..0]) are missing
- Idea: "virtual" flipflops clocked also with CLOCK_50 and input/output logic delay

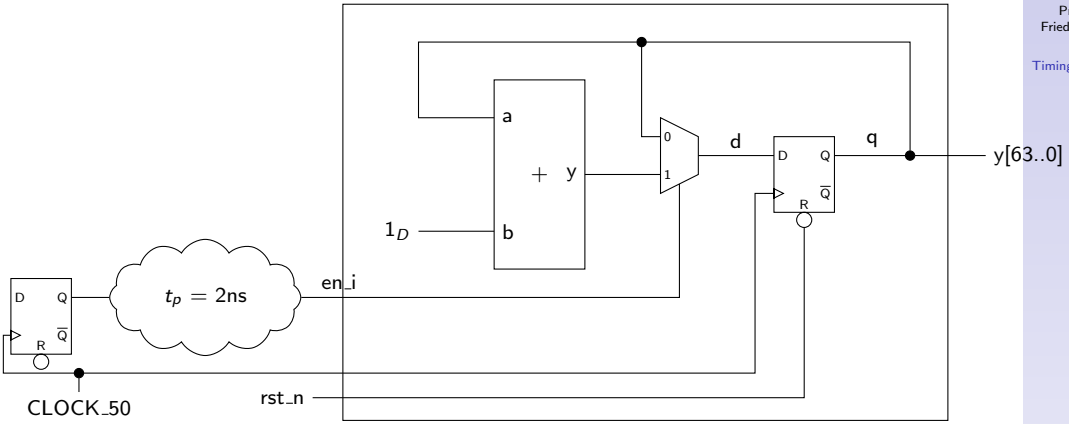
SDC file - Constraining primary outputs




```
1 set_output_delay -clock CLOCK_50 2 [all_outputs]
```

- set_output_delay defines "virtual" flipflops clocked with CLOCK_50 and a "virtual" logic with 2ns propagation delay
- The flipflops are virtually connected at all outputs.

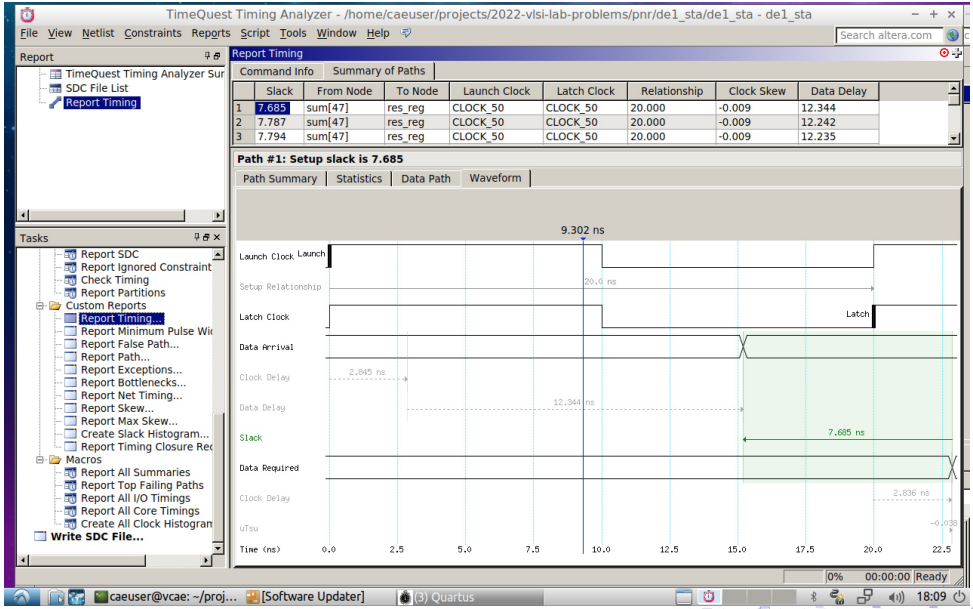
SDC file - Constraining primary inputs



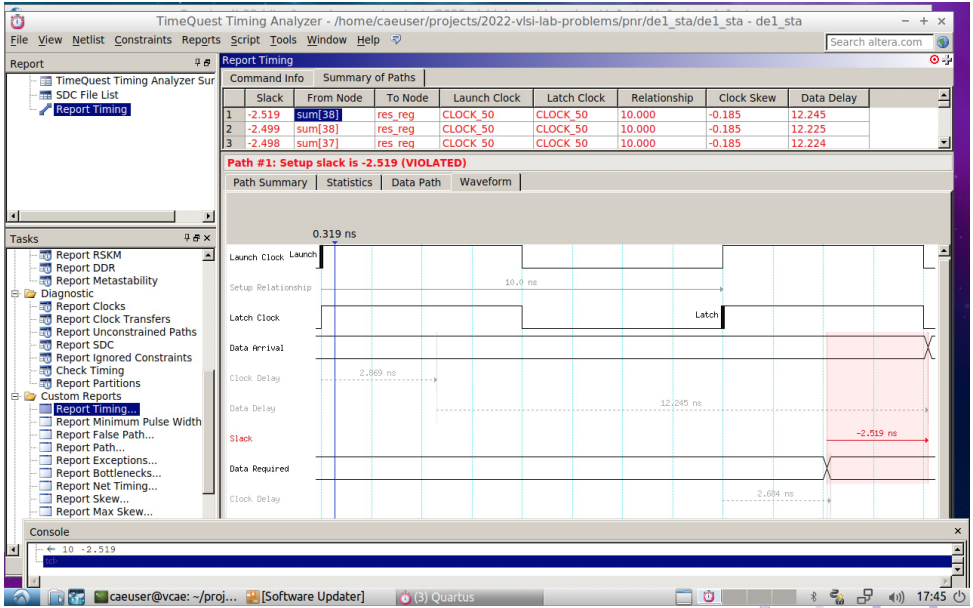

```
1 set_input_delay -clock CLOCK_50 2 [all_inputs]
```

- set_input_delay defines a "virtual" flipflop clocked with CLOCK_50 and a "virtual" logic with 2ns propagation delay

Quartus Static Timing Analysis for 50 MHz clock frequency - o.k.



Quartus Static Timing Analysis for 100 MHz clock frequency - violated



Increasing clock frequency by pipelining

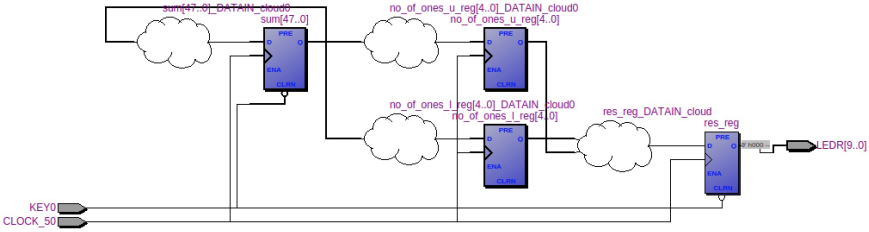
Clock frequency limitations

Idea: Reduce logic depth by introducing a register in the middle

Introduce Pipeline Register

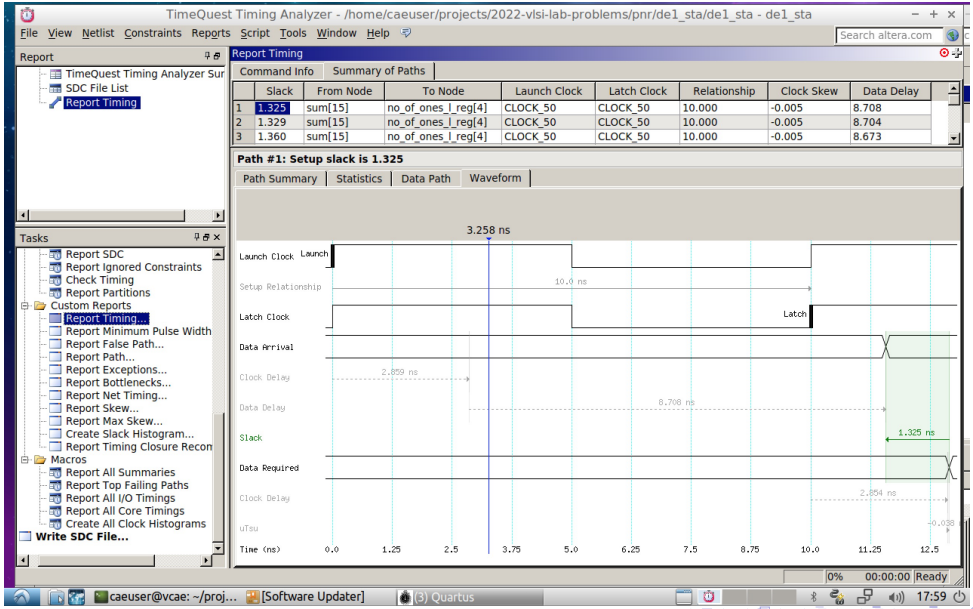
```
1  architecture rtl of de1_sta is
2  function cnt_ones (x : unsigned) return integer is
3    variable cnt : integer range 0 to x'length;
4  begin
5    cnt := 0;
6    for i in x'range loop
7      if x(i) = '1' then
8            cnt := cnt + 1;
9          end if;
10   end loop;
11   return cnt;
12 end function;
13 signal sum : unsigned(47 downto 0);
14 signal clk, rst_n : std_ulogic;
15 signal res, res_reg : std_ulogic;
16 signal no_of_ones : integer range 0 to sum'length;
17 signal no_of_ones_l, no_of_ones_u,
18        no_of_ones_l_reg, no_of_ones_u_reg : integer range 0 to sum'length/2;
19 begin
20 res <= '1' when no_of_ones > sum'length/2 else '0';
21 res_reg <= '0' when rst_n = '0' else res when rising_edge(clk);
22 sum <= (others => '0') when rst_n = '0' else sum + 1 when rising_edge(clk);
23 no_of_ones <= no_of_ones_l_reg + no_of_ones_u_reg;
24 no_of_ones_u <= cnt_ones(sum(sum'high downto sum'length/2));
25 no_of_ones_u_reg <= no_of_ones_u when rising_edge(clk);
26 no_of_ones_l <= cnt_ones(sum(sum'length/2-1 downto 0));
27 no_of_ones_l_reg <= no_of_ones_l when rising_edge(clk);
28 end architecture;
```

Count Ones with Pipelining

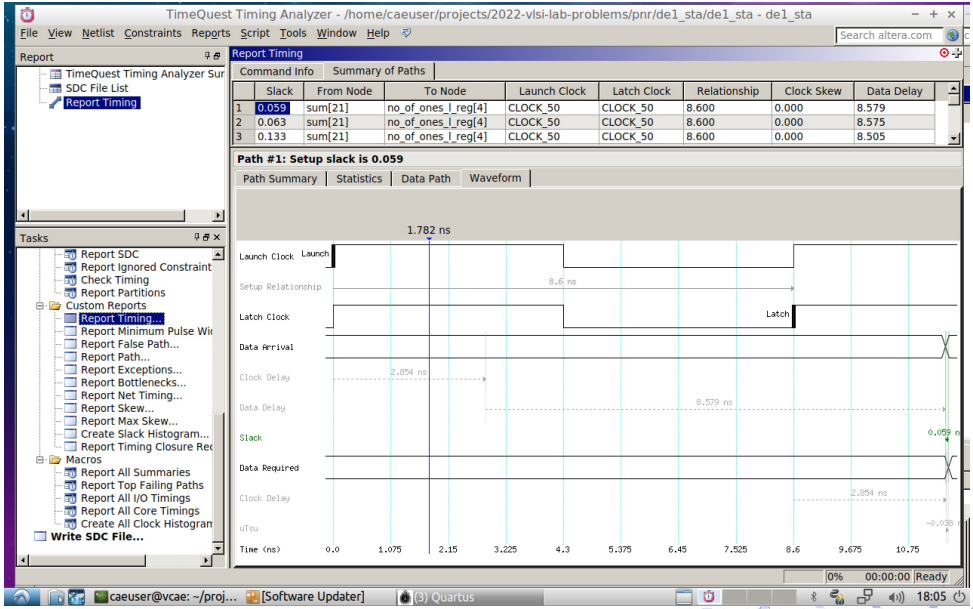


- Result is still computed for every clock cycle ⇒ Throughput identical
- Result is delayed by one clock cycle ⇒ Latency increased by one clock cycle

Timing Analysis with Pipeline Registers at 100 MHz - Slack 1.35 ns



Timing Analysis with Pipeline Registers at 116 MHz - Slack 59 ps



Timing Analysis 48 Bit Counter at 166 MHz - Slack 37 ps