Embedded and Real-Time Systems (EENG 34030): Main Problem

Submission Group 04

QINGYU ZHANG and vn22984
SHURAN YANG and rw22242
HAIBO LIAN and tb22111

Goal & Overall Layout

The responsibility of our group is to design a bridge and integrate the Cortex M0 with the SoC.

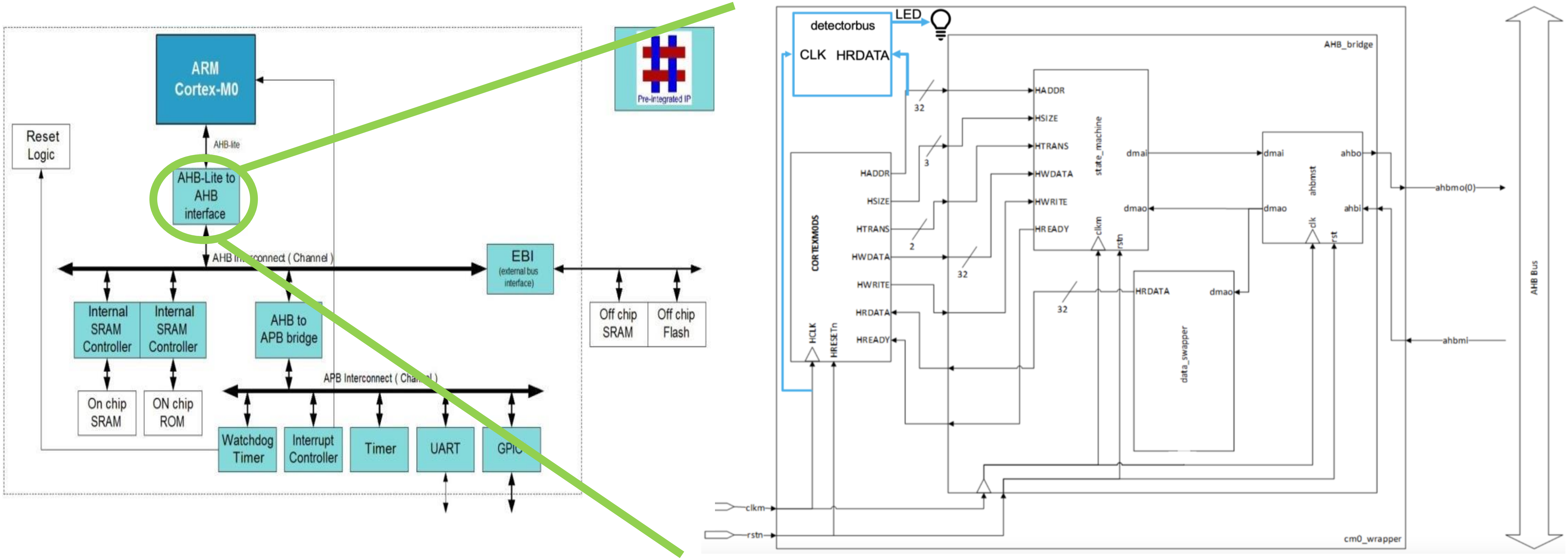


Figure 1.1 The goal and the connection of the detection module.

University of BRISTOL

State machine flowchart :

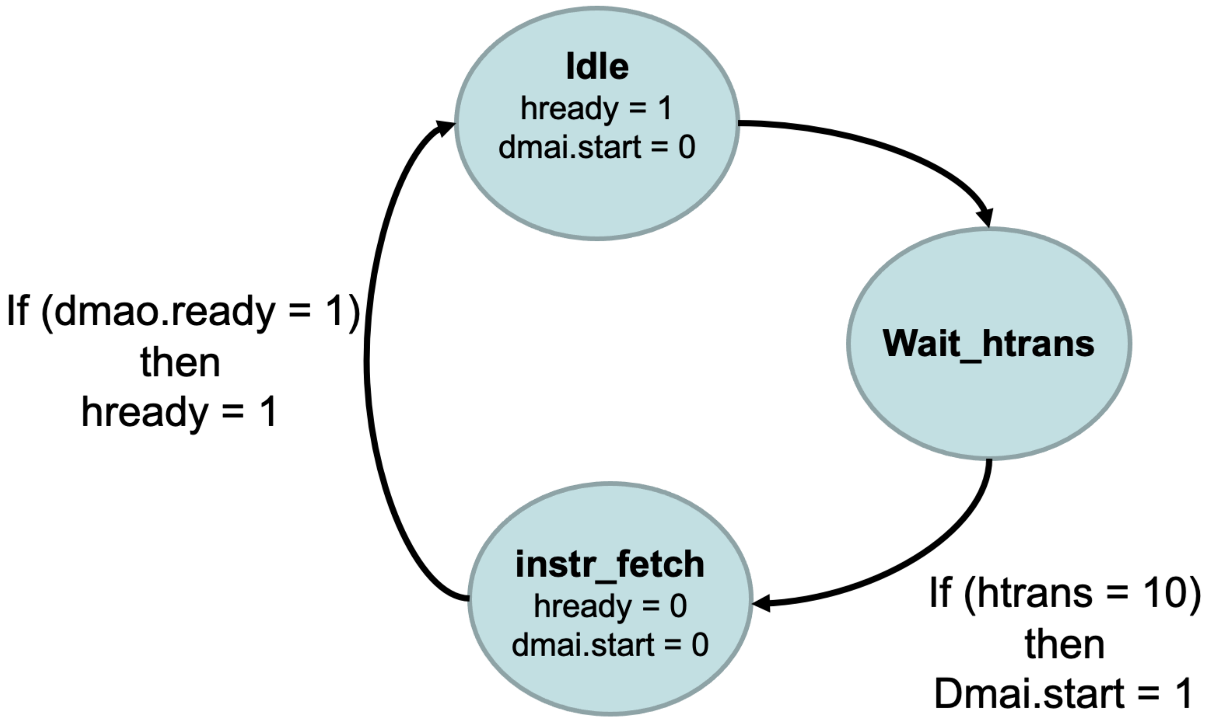


Figure 1.2 The State machine diagram.

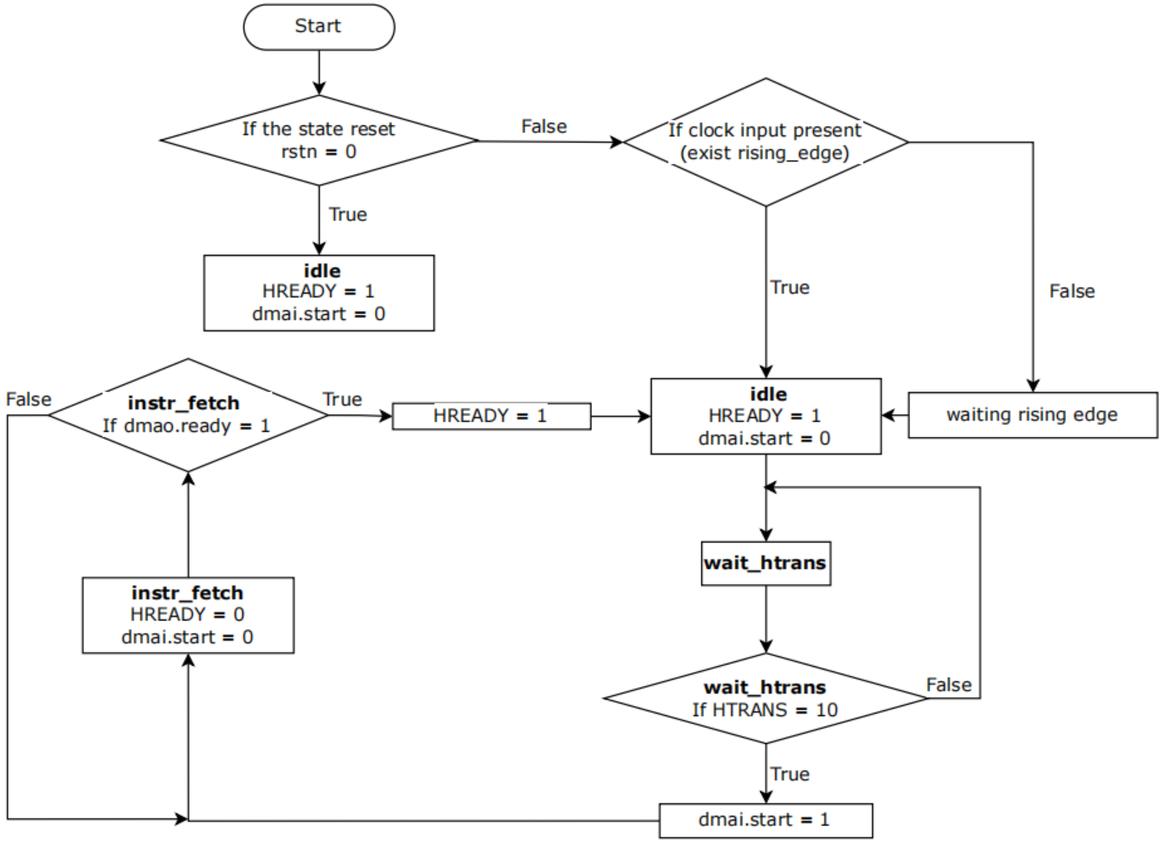


Figure 1.3 The State machine flowchart.

State machine flowchart :

In the design of the state machine we use three processes, namely reg_sate, com_state and output_state. The flow charts of three processes and a general flow chart are drawn respectively.

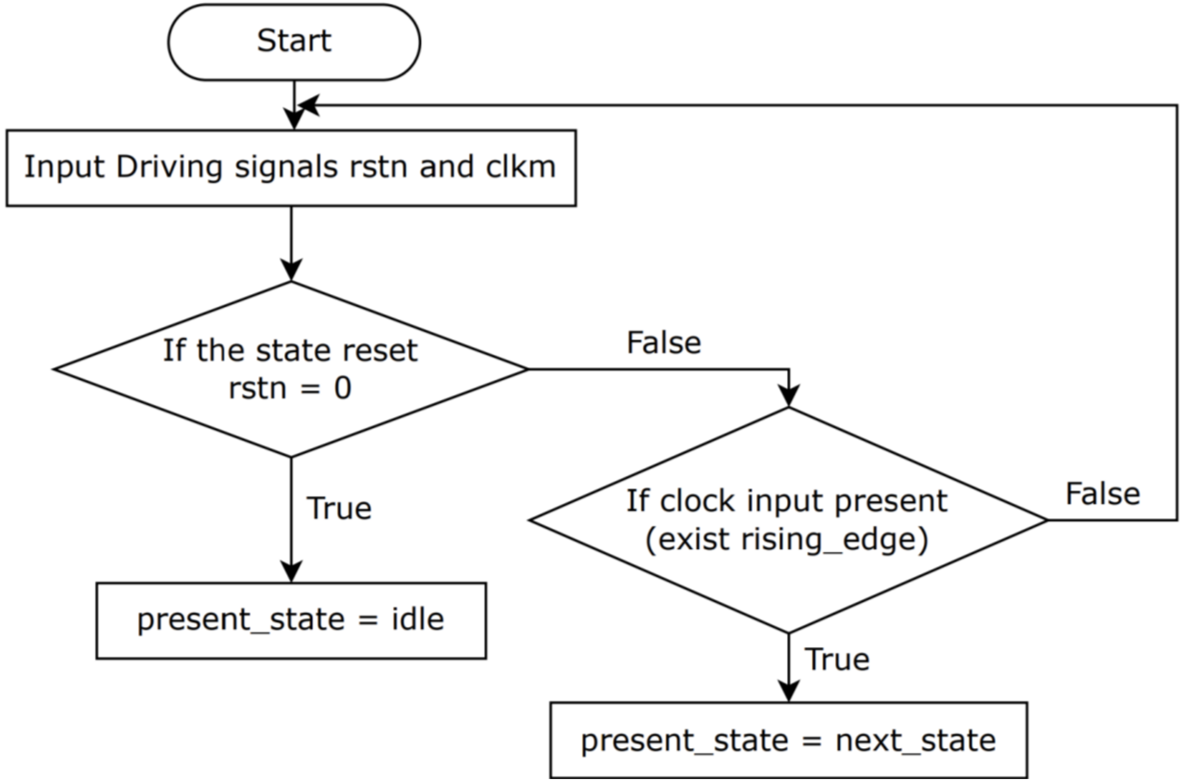


Figure 1.4 The reg_sate machine flowchart.

State machine flowchart :

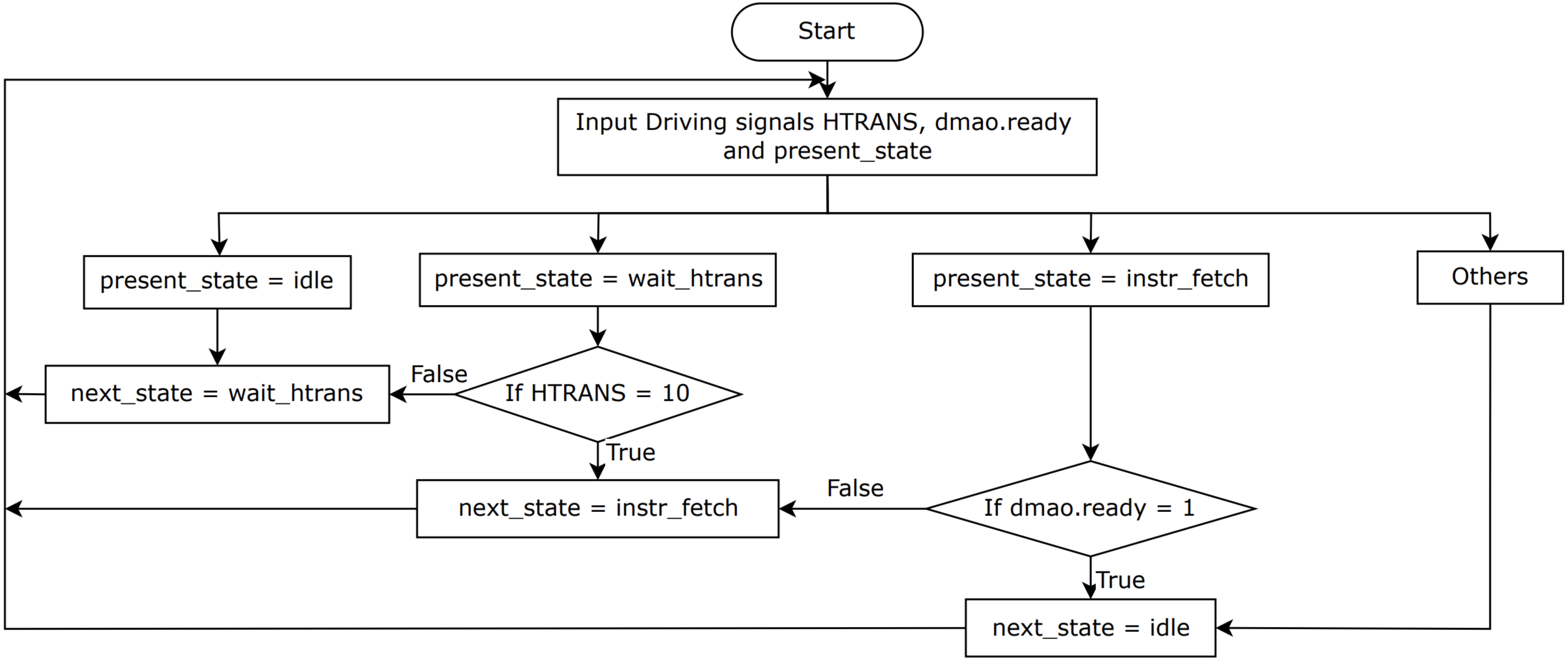


Figure 1.5 The com_sate machine flowchart.

State machine flowchart :

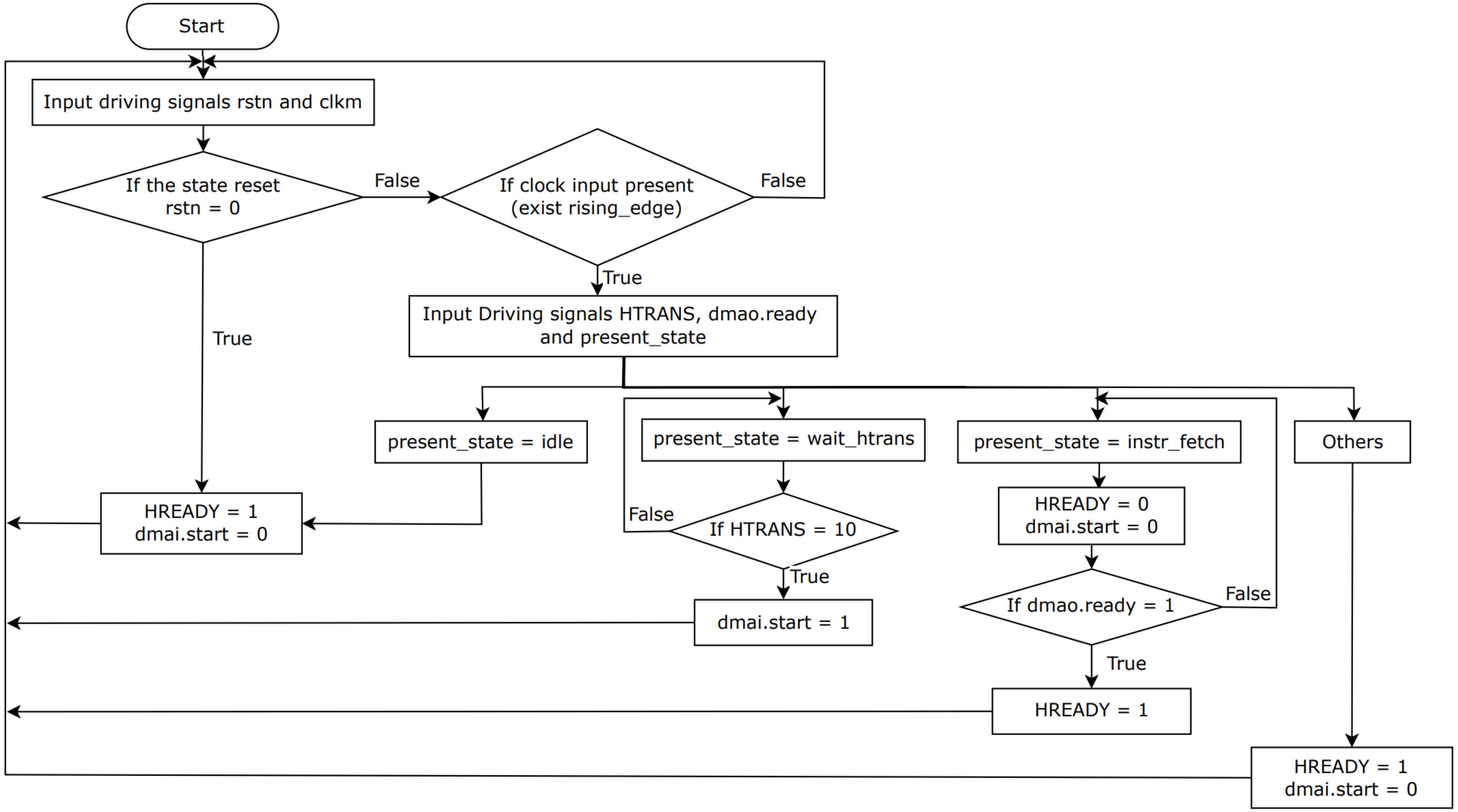
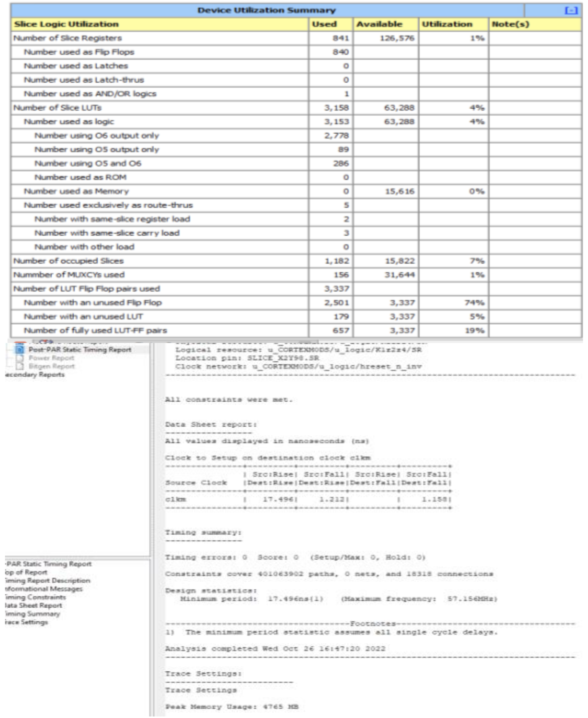


Figure 1.6 The output_sate machine flowchart.

FPGA Implementation Details Table:

FPGA device selected:	Spartan 6 xc6slx100-3fgg676
Total Number of occupied Slices:	1182
Total Number of Slice Flip Flops:	840
Total Number of Slice LUTs:	3518
Best possible clock frequency:	57.156MHz
Cortex-M0 Number of occupied Slices:	1176
Cortex-M0 Number of Slice Flip Flops:	830
Cortex-M0 Number of Slice LUTs:	3136
Bridge Number of occupied Slices:	3
Bridge Number of Slice Flip Flops:	7
Bridge Number of Slice LUTs:	6



Device Utilization Summary

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	841	126,576	1%	
Number used as Flip Flops	840			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	1			
Number of Slice LUTs	3,158	63,288	4%	
Number used as logic	3,153	63,288	4%	
Number using O6 output only	2,778			
Number using O5 output only	89			
Number using O5 and O6	286			
Number used as ROM	0			
Number used as Memory	0	15,616	0%	
Number used exclusively as route-thrus	5			
Number with same-slice register load	2			
Number with same-slice carry load	3			
Number with other load	0			
Number of occupied Slices	1,182	15,822	7%	
Nummber of MUXCYs used	156	31,644	1%	
Number of LUT Flip Flop pairs used	3,337			
Number with an unused Flip Flop	2,501	3,337	74%	
Number with an unused LUT	179	3,337	5%	
Number of fully used LUT-FF pairs	657	3,337	19%	

Post-PAR Static Timing Report
Power Report
Bitgen Report
econdary Reports

```
  Logical resource: u_CORTEXM0DS/u_logic/Klz2z4/SR
  Location pin: SLICE_X2Y98.SR
  Clock network: u_CORTEXM0DS/u_logic/hreset_n_inv
--------------------------------------------------------------------------------

All constraints were met.

Data Sheet report:
-----------------
All values displayed in nanoseconds (ns)

Clock to Setup on destination clock clkm
---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clkm           |   17.496|    1.212|         |    1.158|
---------------+---------+---------+---------+---------+

Timing summary:
---------------

Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)

Constraints cover 401063902 paths, 0 nets, and 18318 connections

Design statistics:
   Minimum period:  17.496ns(1)   (Maximum frequency:  57.156MHz)

------------------------------------Footnotes-----------------------------------
1)  The minimum period statistic assumes all single cycle delays.

Analysis completed Wed Oct 26 16:47:20 2022
--------------------------------------------------------------------------------

Trace Settings:
-------------------------
Trace Settings

Peak Memory Usage: 4765 MB
```

-PAR Static Timing Report
op of Report
iming Report Description
nformational Messages
iming Constraints
ata Sheet Report
iming Summary
race Settings

Table 1.1 The FPGA Implementation Details.

Design Checklist:

1. Check **swapper** and state machine VHDL code, check clock usage and correct combinational and sequential processes.

The figure 1.7 shows the Data_swapper model works in the right way.

Figure 1.7 Correct simulation of swapper.

Design Checklist:

1. Check swapper and **state machine** VHDL code, check clock usage and correct combinational and sequential processes.

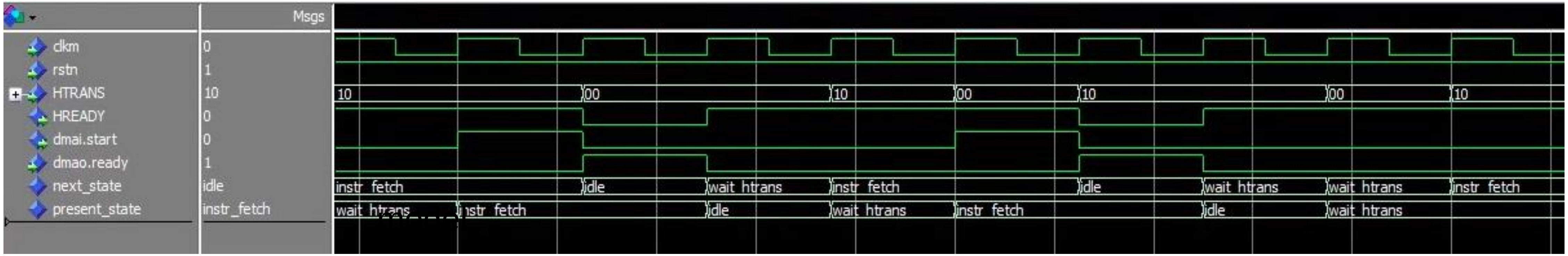


Figure 1.8 Correct simulation of state machine.

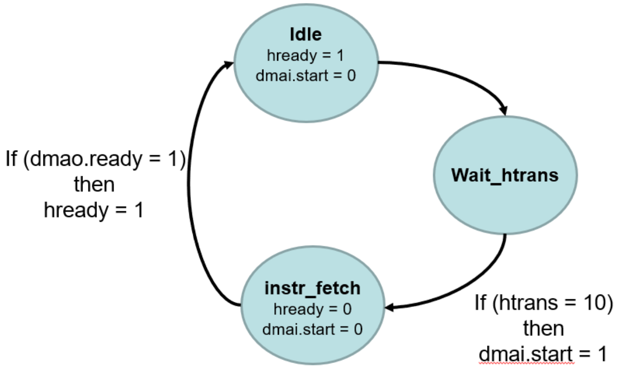


Figure 1.2 The State machine diagram.

The figure 1.8 shows the state machine in action. The figure 1.2 shows the state machine diagram.

1. **Idle** is the initial state. HREADY is 1 and dmai.start is 0.
2. Then state will set to **wait_htrans**.
3. When the HTRANS is set to ‘10’, at the next rising edge of clkm the dmai.start is 1 and the state is set to **instr_fetch**. At next rising edge of clkm, the HREADY and dmai.start are set to 0.
4. When dmao.ready is ‘1’, at next rising edge of clkm, the HREADY is set to 1 and the state will be set to **idle**. At next rising edge of clkm, the Hat next rising edge of clkm READY is 1 and dmai.start is 0.

The state machine works in the right way. ☺

Design Checklist:

2. Verify VHDL code for hierarchy that should include AHB bridge, processor etc. Check components inside AHB bridge are correctly connected and look complete.

```vhdl
entity ahb_bridge is
  port(
    -- Clock and Reset
    clkm : in std_logic;
    rstn : in std_logic;
    -- AHB Master records
    ahbmi : in ahb_mst_in_type;
    ahbmo : out ahb_mst_out_type;
    -- ARM Cortex-M0 AHB-Lite signals
    HADDR : in std_logic_vector (31 downto 0);
    HSIZE : in std_logic_vector (2 downto 0);
    HTRANS : in std_logic_vector (1 downto 0);
    HWDATA : in std_logic_vector (31 downto 0);
    HWRITE : in std_logic; -- AHB write control
    HRDATA : out std_logic_vector (31 downto 0);
    HREADY : out std_logic -- AHB stall signal
  );
```

```vhdl
component state_machine is
  port(
    clkm : in std_logic;
    rstn : in std_logic;
    HADDR : in std_logic_vector (31 downto 0);
    HSIZE : in std_logic_vector (2 downto 0);
    HTRANS : in std_logic_vector (1 downto 0);
    HWDATA : in std_logic_vector (31 downto 0);
    HWRITE : in std_logic;
    HREADY : out std_logic;
    dmai : out ahb_dma_in_type;
    dmao : in ahb_dma_out_type
  );
end component;

--declare a component for ahbmst
component ahbmst is
  generic (
    hindex  : integer := 0;
    hirq    : integer := 0;
    venid   : integer := VENDOR_GAISLER;
    devid   : integer := 0;
    version : integer := 0;
    chprot  : integer := 3;
    incaddr : integer := 0);
  port(
    rst  : in  std_ulogic;
    clk  : in  std_ulogic;
    dmai : in ahb_dma_in_type;
    dmao : out ahb_dma_out_type;
    ahbi : in  ahb_mst_in_type;
    ahbo : out ahb_mst_out_type
  );
end component;
```

```vhdl
--declare a component for data_swapper
component data_swapper is
  port(
    clkm : in std_logic;
    rstn : in std_logic;
    HRDATA : out std_logic_vector (31 downto 0);
    dmao : in ahb_dma_out_type
  );
end component;
```

```vhdl
-----------------------------
-- the define of signal
signal dmai : ahb_dma_in_type;
signal dmao : ahb_dma_out_type;
```

```vhdl
-----------------------------
-- the begin of action
begin

--instantiate state_machine componen
  u_state_machine : state_machine
  port map(
    clkm => clkm,
    rstn => rstn,
    HADDR => HADDR,
    HSIZE => HSIZE,
    HTRANS => HTRANS,
    HWDATA => HWDATA,
    HWRITE => HWRITE,
    HREADY => HREADY,
    dmai => dmai,
    dmao => dmao
  );

--instantiate the ahbmst component a
  u_ahbmst : ahbmst
  port map(
    clk => clkm,
    rst => rstn,
   dmai => dmai,
    dmao => dmao,
    ahbi => ahbmi,
    ahbo => ahbmo
  );

--instantiate the data_swapper compo
  u_data_swapper : data_swapper
  port map(
    clkm => clkm,
    rstn => rstn,
    HRDATA => HRDATA,
    dmao => dmao
  );

end structural;
```

```
# Compile of ahb_bridge.vhd was successful.

VSIM 21>
```

Figure 1.9 The main VHDL code.

Design Checklist:

3. Show that the simulator compiles the VHDL code without errors.

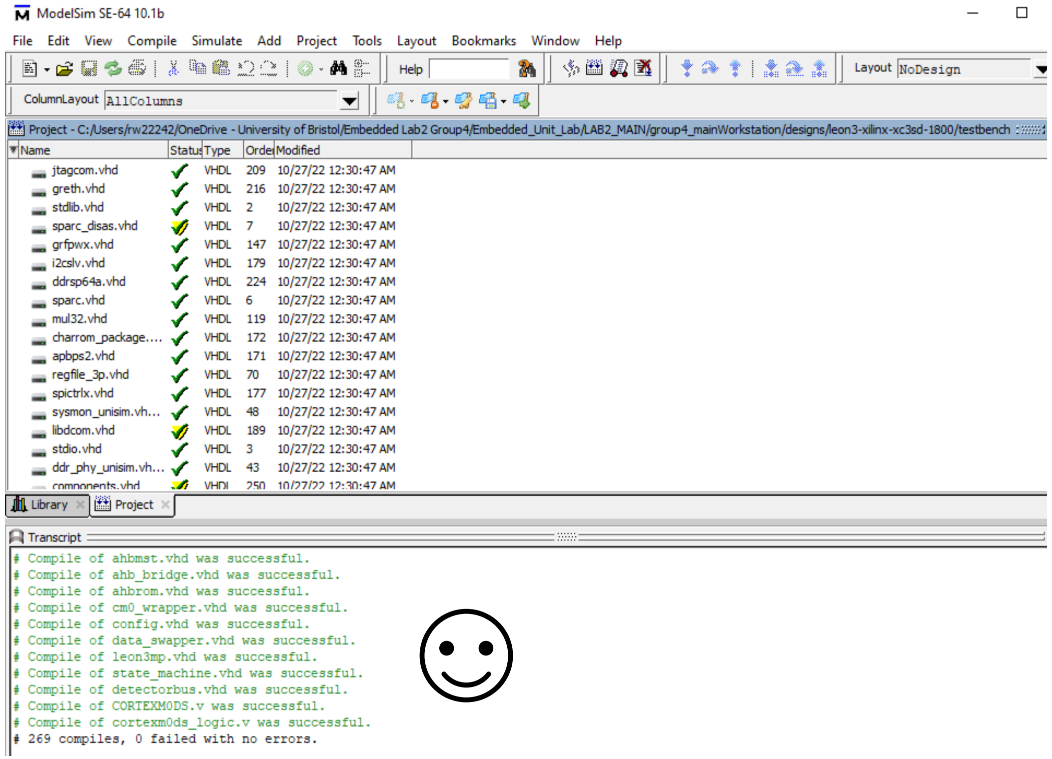
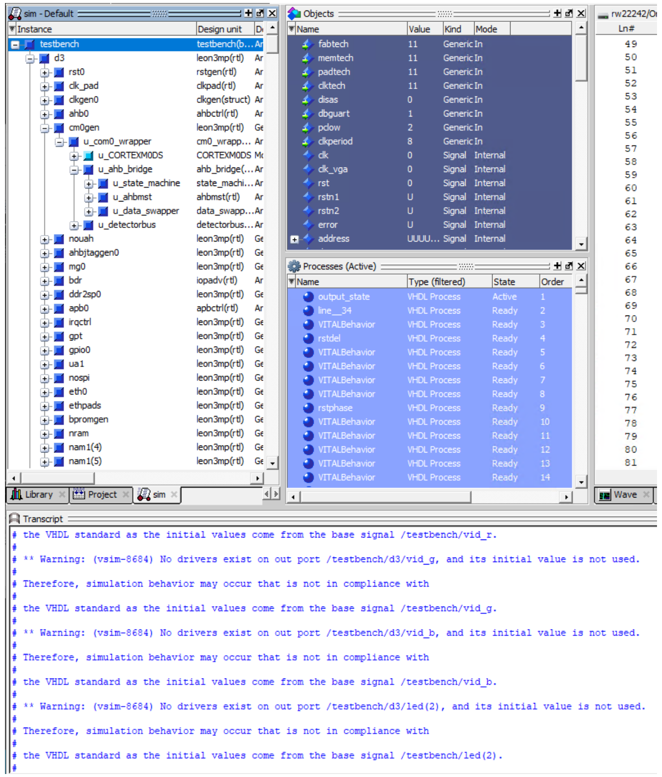


Figure 1.10 Compile VHDL code correctly.

University of BRISTOL

Design Checklist:

4. Show that the simulator loads the system VHDL without errors and testbench starting.



☺

Figure 1.11 Correct simulation of testbench.

Design Checklist:

5. Show that the Cortex-M0 processor is running without entering undefined states.

☺ The figure 1.12 shows the processor is running without entering undefined states.

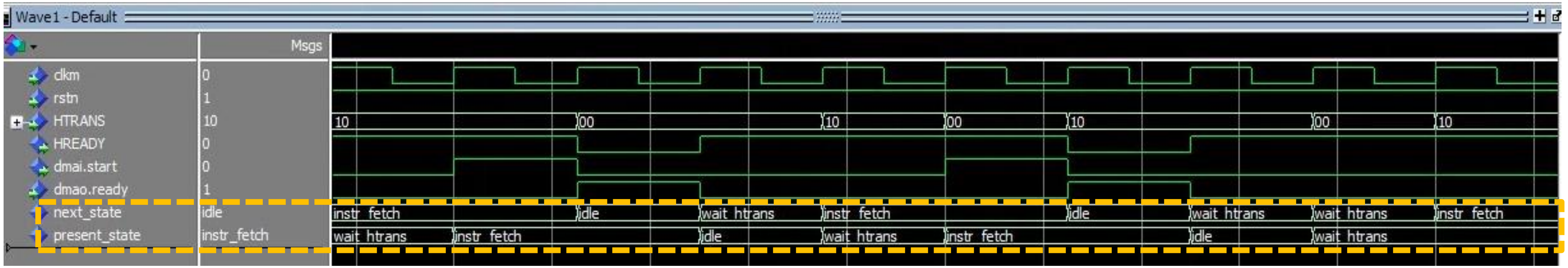


Figure 1.12 Normal operation of the Cortex-M0 processor.

Design Checklist:

6. Show that the processor is running continuously without halting after some time.

We simulated 22000 us. The figure 1.13 shows the processor is running continuously without halting after some time.

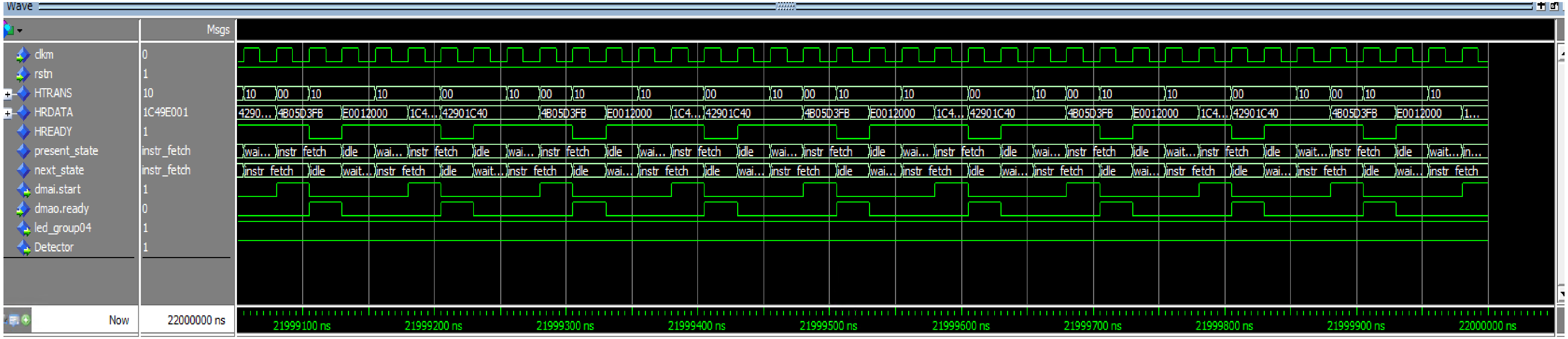


Figure 1.13 The processor runs normally without halting.

Design Checklist:

7. Show that the data patterns that trigger the led blinking appear in the HRDATA signal (your group number).

We are **group 4**. According to the figure 1.14, we found HRDATA signal has our group number to trigger the led_group4.

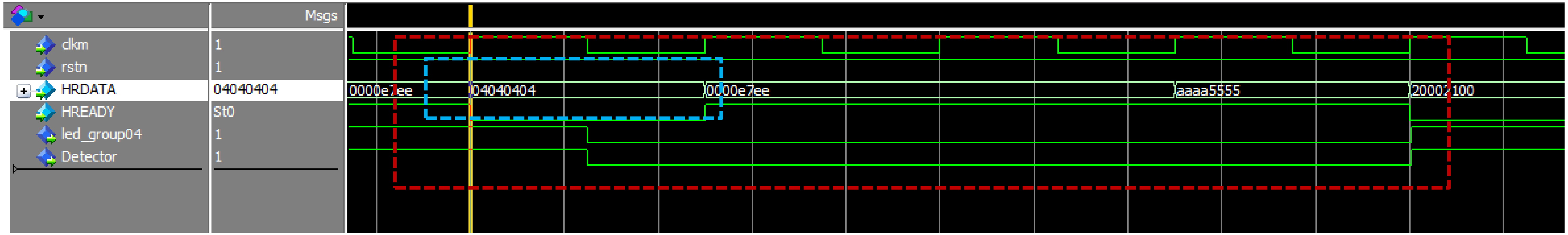
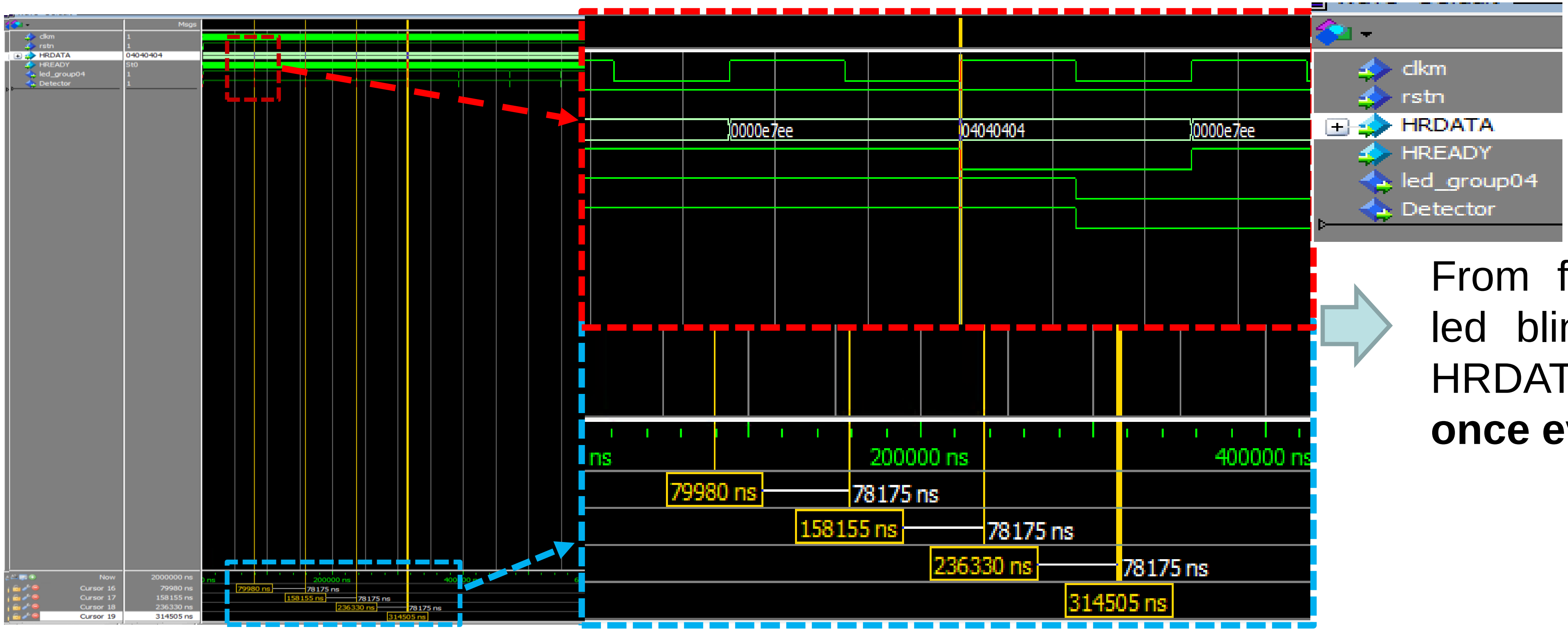


Figure 1.14 Blinking of LEDs.

Design Checklist:

8. Show that the data patterns in HRDATA signal that trigger the led blinking are repeating periodically. (It should repeat once every ~62000 ns)



From figure 1.15, the led blinking trigger in HRDATA signal **repeat once ever 78175ns**. ☺

Figure 1.15 Periodically repeat trigger LED flashing on HRDATA signal.

Design Checklist:

9. Show that the data patterns in HRDATA that trigger the led blinking repeating periodically are connected to a new user defined output in the testbench such as led.

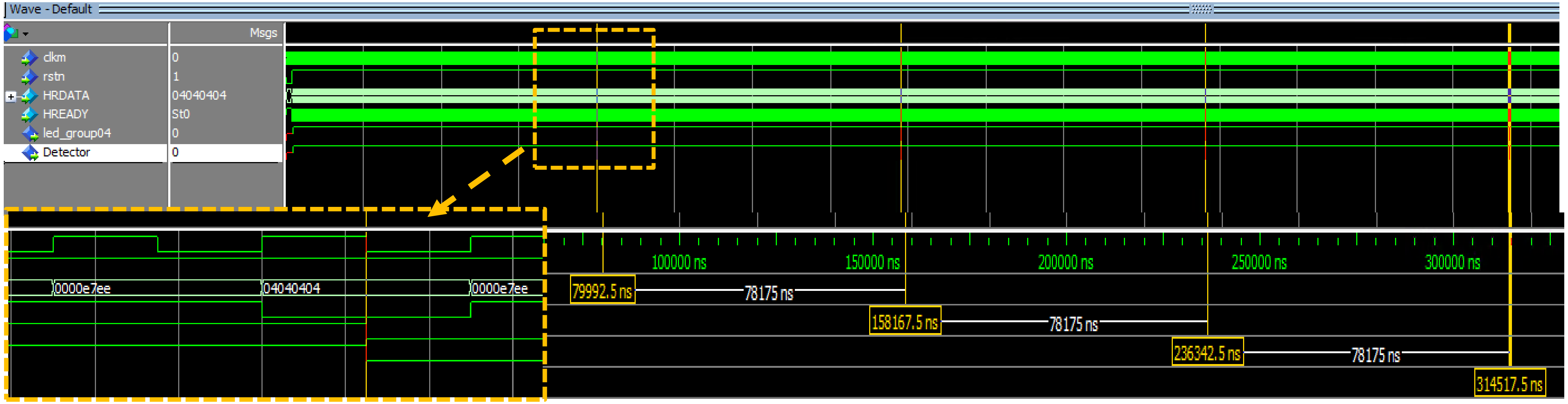


Figure 1.16 New user defined output in testbench is connected.

Design Checklist:

10. Check Vivado/ISE project for code completeness and correct Leon3 library set up. Does your design satisfy timing constraints?

State	TIMESPEC Name *	Clock Time Name	Clock Net *	Period	Duty Cycle	Edge	Reference TIMESPEC	Factor	Phase Shift
OK	TS_clkm	clkm	clkm	20 ns	50 %	HIGH			

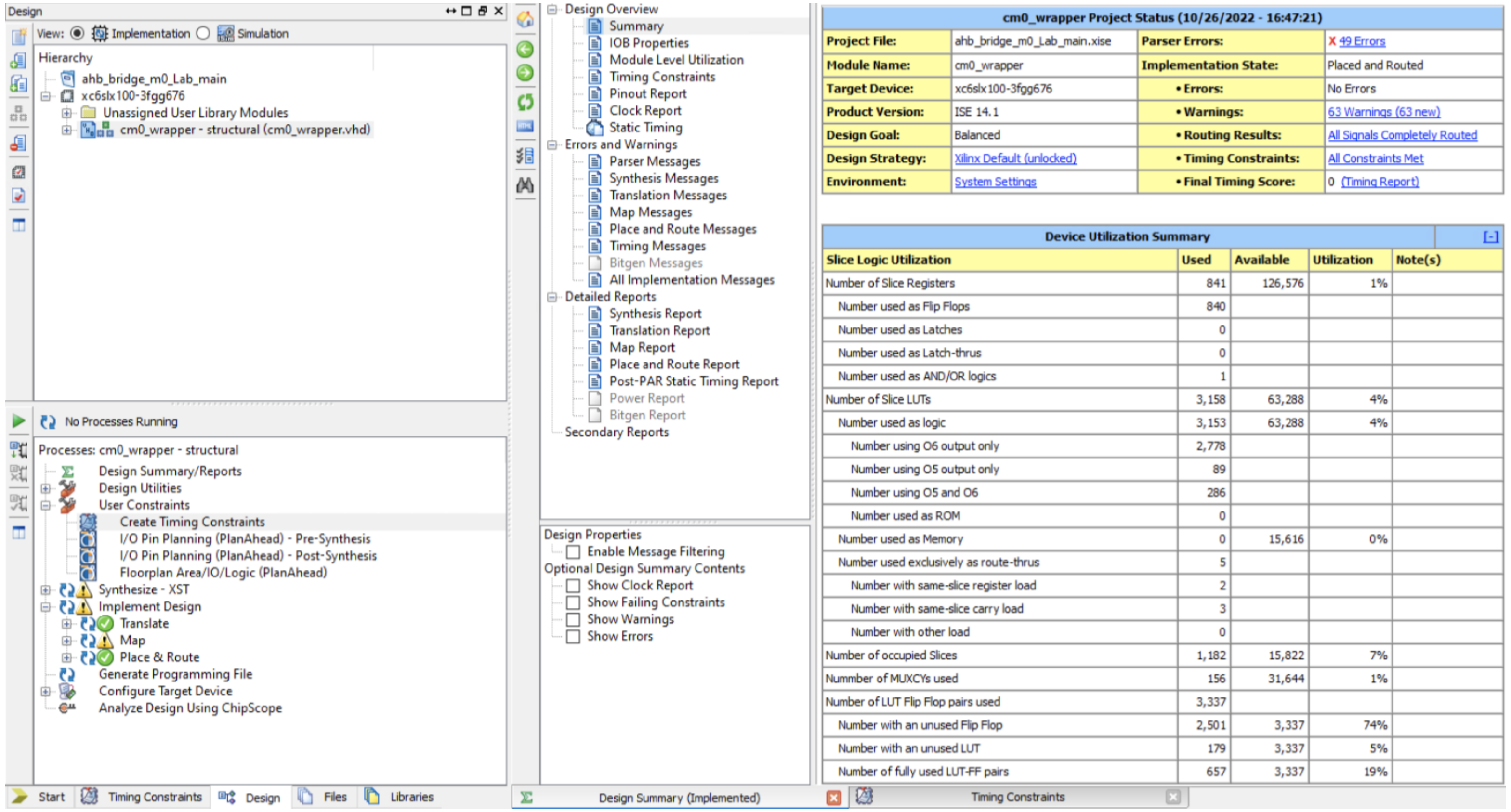
```
Timing summary:
---------------

Timing errors: 0  Score: 0  (Setup/Max: 0, Hold: 0)

Constraints cover 401063902 paths, 0 nets, and 18318 connections

Design statistics:
   Minimum period:  17.496ns{1}   (Maximum frequency:  57.156MHz)
```

$17.496ns < 20\ ns$

So our design satisfy timing constraints.

Design Checklist:

11. Check the implementation reports and FPGA Implementation Details Table.



☺

Figure 1.17 The summary of design overview.

Design Checklist:

12. Show the simulation of the netlist with additional timing data (SDF files) working correctly with led blinking repetitions.

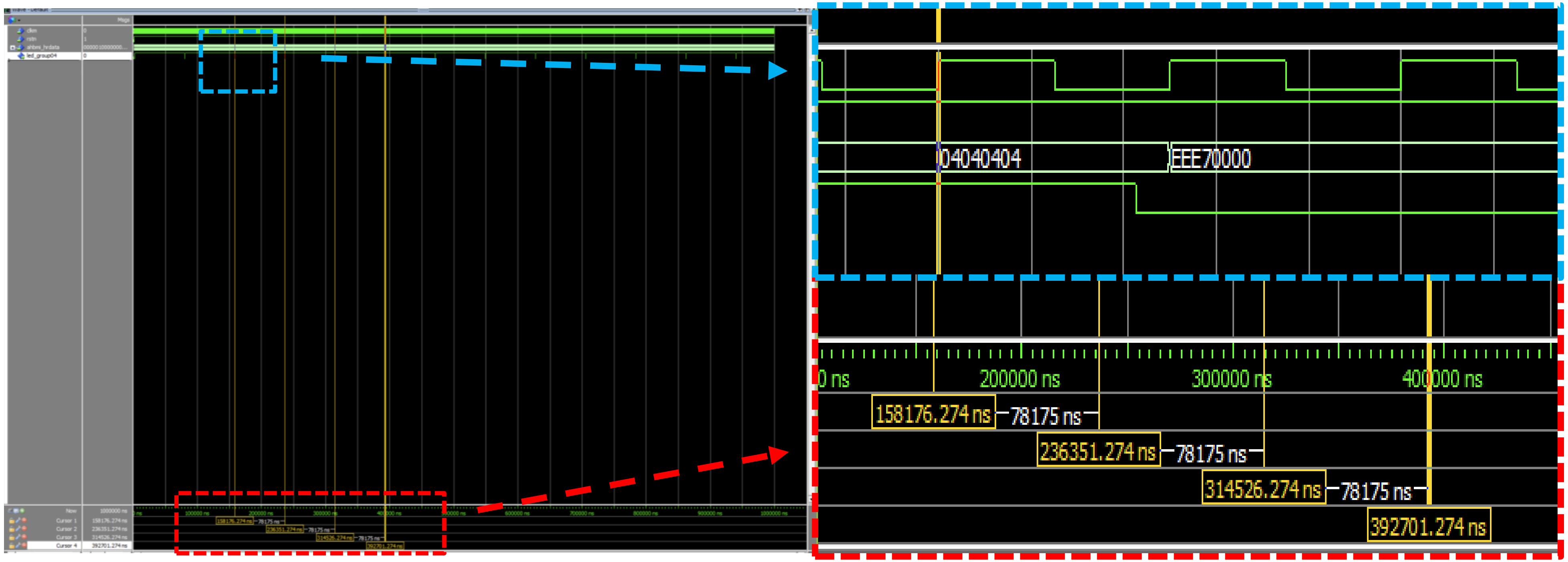


Figure 1.18 Netlist simulation about using SDF file.

☺

Design Checklist:

12.1 Compare the period(T) of **led_group04** blinking simulation between the netlist **with SDF files** and the netlist **without SDF files**.

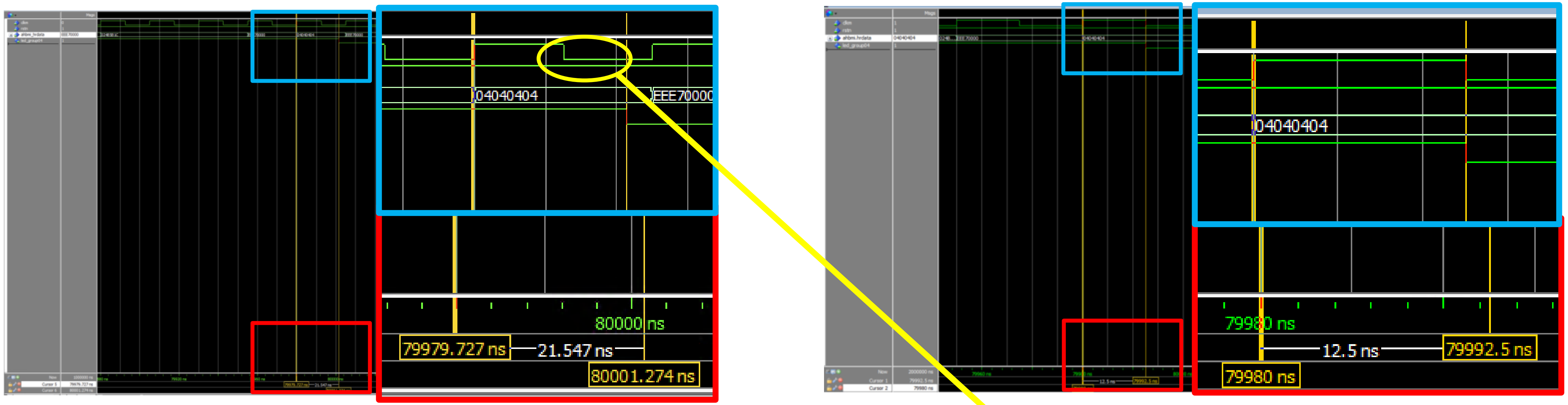


Figure 1.19 Compare after(left) and before(right) using SDF files.

$21.547ns - 12.5ns = 9.047ns$

1 period = 25ns

After add SDF the led blinking time delay = **0.362 period**

Thank You

Question?