Digital System Design (ELCT501)

Practical Lab Report

ELEVATOR

Team 3

Student's Name	ID	Group
Ali Ayman Elbadawy	49-33967	T-27
Arwa Hesham Ghoneim	52-12468	T-30
Heba Ashraf Ahmed	52-22680	T-30
Ibrahim Emad Elsayed	52-2884	T-27
Mohamed Ahmed Saeed	52-10194	T-30
Rowan Bahaa Hegazy	52-20033	T-29
Shaheer Sherif	52-0433	T-27

1. Design and Methodology

The elevator mechanism we decided to use consists of the elevator cabin (25cm x 15cm x 15cm), a timing belt which passes over two pulleys, one fixed on the motor carrying the cabin and the other fixed on an 8mm pin used to carry the counterweight. Both pulleys are on the same vertical height in order to be balanced.

We used counterweight so that the load on the motor is minimized and the torque required for the motion is reduced. To decide the weight of the counterweight (0.75Kg), it was found that it should be equal to the weight of the cabin (0.5Kg) + half the operating load of the elevator (0.25Kg).

We used a stepper motor (Nema 17) which has a rated torque of 0.16 N.m and needs rated current 1.2A supplied from a 12V power supply. Rotation of the motor is transmitted to the timing belt via a 5 mm pulley. In order to control the stepper motor, we found out that it is required to generate step signal at a certain desired frequency (500 microseconds). To be able to do this, we implemented a clock Divider which uses the clock of the fpga (pin W5 at frequency 100MHz) to generate the required steps of the motor. Drv8825 motor driver was used to control the step and direction inputs needed for the coils to rotate the shaft of the motor only when desired. The driver was supplied with 3.3V from the fpga. We controlled the enable pin of the driver using the fpga so that the driver only runs during transition states and not while the elevator is not being requested at any level. We calculated that if the motor runs at full steps, 1 second is needed to go from one level to another so a signal called temp was specified to count N clock cycles (N = 10,000,000).

Concerning the control for the motion of the elevator, we have a total of 7 buttons (4 inside the cabin and 1 outside the cabin at each level of the three). The buttons on the inside the cabin includes a buzzer in case of emergency. Furthermore, we used an LCD to show the level at which the elevator currently is.

The materials used for the structure were: Acrylic (5 mm) and MDF (5 mm).

For our methodology and code, we used Moore FSM with 7 states which will be described further down the report. Our design accepts inputs from the user at any time and a flag (updown) was used to handle the priority of requests (E.g.: If the elevator was coming up from the ground floor to the first floor and buttons 0 and 2 were pressed, the elevator continues to level 2 first then go back to level 0). We have outputs called move and atlevel. Move is 0 at all transition states (meaning that the motor is moving) and 1 at all levels. Atlevel outputs were used to represent the floor

that the elevator is currently in. Finally, we used a flag called waitlevel which is used to wait at a certain level for a specified time (1 sec) before going to any other requested level. From the timing report, Worst negative slack is 5.218ns so 10ns – 5.218 ns = 4.79ns which means Max Operating frequency equals 1/4.79ns = 209 MHz.

2. States

We used Moore FSM with seven states

State s0: ground floor

State s1: first floor

State s2: second floor

State upA: when the elevator is between level 0 and 1 and is going up

State upB: when the elevator is between level 1 and 2 and is going up

State downA: when the elevator is between level 0 and 1 and is going down

State downB: when the elevator is between level 1 and 2 and is going down

We have three flags to know when the elevator is called to a floor f0, f1, f2

Also, there are other important conditions

When Temp = N: to decide if the elevator finished moving between any two states, or to decide how much time the motor must rotate for elevator to reach a new level (S0, S1, S2).

When waitlevel = N: which indicates the amount of time the elevator waits at level before transitioning to a new level

Move: This signal comes from the FSM (final file) and goes into clockDiv file and its use is to output the motorclock signal at frequency 1kHz only at transition states

There is another flag which is updown: to know the previous state of the elevator and decide where to go for the next request (handling priority based on if the elevator was going up or down previously)

3. VHDL Code

For our VHDL code there are 3 main files:

- Final : this file contains all the logic and flags needed for operation; it is basically the finite state machine (FSM)
- ClkDiv : this is used to generate the required step frequency for optimum operation of the stepper motor; this frequency is 1kHz
- Finalelevator: this file contains final and clkDiv as components and represents everything put together to be able to fully run the elevator. The relationship between ClkDiv and final is the move signal which indicates when (during transition states) to output the motorclock signal which is a square wave of frequency 1kHz

Inputs:

1. lvl0in, lvl1in, lvl2in represent the buttons inside the elevator
2. lvl0out, lvl1out, lvl2out represent the buttons outside the elevator
3. clk is the clock signal from fpga (100MHz) and reset is reset (mainly used for simulation)

Outputs:

1. motorDir: direction of motor
2. motorclock: provides square wave frequency 1KHz to motor as required by driver
3. enable: enable signal for the motor (when 0 the motor is rotating)
4. atlvl0, atlvl1, atlvl2, at0, at1, at2: signals which represent at which level the elevator is, two sets are done as they perform different functions; three are for the Arduino and the other three are used as LEDs on FPGA to make sure the signal is consistent.

- **This is the first file named (final)**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;
entity final is
   Port (lvl0out, lvl1out, lvl2out, lvl0in, lvl1in, lvl2in, clock, reset : in std_logic;
    motorDir, atlvl0, atlvl1, atlvl2, move, enable, at0, at1, at2 : out std_logic );
end final;

architecture Behavioral of final is
signal f0, f1, f2, updown: std_logic;
type state_type is (s0, upA, downA, s1, upB, downB, s2);
signal s: state_type;
signal N, temp, waitLevel: integer:=0;
signal nozero, noone, notwo:std_logic;

begin

transition: process(clock,reset,lvl0in,lvl0out,lvl1in,lvl1out,lvl2in,lvl2out) -- state transition
begin
if ( clock = '1' and clock'event) then
if (reset = '1') then
  s <= s0; f0 <= '0'; f1 <= '0'; f2 <= '0'; N <= 100000000;
else
      if ( (lvl0out = '1'  or lvl0in = '1' ) and nozero = '0' ) then
            f0 <= '1';
      end if;
      if ( (lvl1out = '1'  or lvl1in = '1' )and noone = '0' ) then
            f1 <= '1';
```

```
end if;
if ( (lvl2out = '1'  or lvl2in = '1' )and notwo ='0' ) then
      f2 <= '1';
end if;
case (s) is
      when s0 =>
            if ((f1 = '1' or f2 = '1')and waitLevel >= 100000000) then
              N <= 100000000; s <= upA ;
            end if;

      when upA =>
            if (f1 = '1' and temp=N) then
              s <= s1; N <= 100000000; f1<='0';
            elsif (f2 = '1' and (f1 = '0') and temp=N) then
              s <= upB; N <= 100000000;
            else
              s <= upA; N <= 100000000;
            end if;

      when downA =>
            if (temp=N and f0 = '1') then
              s<= s0; N <= 100000000; f0<='0';
            end if;

      when s1 =>
         if(waitLevel >= 100000000) then
            if ((f0 = '1' and f2 = '1' and updown = '0') or (f0 = '1' and f2 = '0')) then
              s <= downA; N <= 100000000;
            elsif (( f0 = '0' and f2 = '1' )or( f0 = '1' and f2 = '1' and updown = '1' )) then
```

```
                s <= upB; N <= 100000000;
              end if;
            end if;

        when upB =>
              if (temp=N and f2 = '1') then
                s <= s2; N <= 100000000; f2 <= '0';
              end if;

        when downB =>
              if (f0 = '1' and f1 = '0' and temp=N) then
                s <= downA; N <= 100000000;
              elsif (f1='1' and temp=N) then
                s <= s1; N <= 100000000; f1 <= '0';
              end if;

        when s2 =>
              if ((f1 = '1' or f0 = '1') and waitLevel >= 100000000) then
                s <= downB; N <= 100000000;
              end if;
        end case;
        end if;
    end if;
end process transition;


 stepper: process(s) ---stepper process
   begin
   case s is
```

```
when s0 =>   updown <= '1'; motorDir<='0'; move <= '1'; enable <= '1';
             atlvl0 <= '1'; atlvl1 <= '0'; atlvl2 <= '0';
             at0 <= '1'; at1 <= '0'; at2 <= '0';
             nozero <= '1'; noone <= '0'; notwo <= '0';
when s1 =>   updown <= updown; motorDir <= '0'; move <= '1'; enable <= '1';
             atlvl1 <= '1'; atlvl0 <= '0'; atlvl2 <= '0';
             at1 <= '1'; at0 <= '0'; at2 <= '0';
             nozero <= '0'; noone <= '1'; notwo <= '0';
when s2 =>   updown <= '0';  motorDir <= '0'; move <= '1'; enable <= '1';
             atlvl2 <= '1'; atlvl0 <= '0'; atlvl1 <= '0';
             at2 <= '1'; at0 <= '0'; at1 <= '0';
             nozero <= '0'; noone <= '0'; notwo <= '1';
when upA =>  updown <= '1'; motorDir <= '0'; move <= '0'; enable <= '0';
             atlvl0 <= '0'; atlvl1 <= '0'; atlvl2 <= '0';
             at0 <= '0'; at1 <= '0'; at2 <= '0';
             nozero <= '0'; noone <= '0'; notwo <= '0';
when upB=>   updown <= '1'; motorDir <= '0'; move <= '0'; enable <= '0';
             atlvl0 <= '0'; atlvl1 <= '0'; atlvl2 <= '0';
             at0 <= '0'; at1 <= '0'; at2 <= '0';
             nozero <= '0'; noone <='0'; notwo <= '0';
when downA=> updown <= '0'; motorDir <= '1'; move <= '0'; enable <= '0';
             atlvl0 <= '0'; atlvl1 <= '0'; atlvl2 <= '0';
             at0 <= '0'; at1 <= '0'; at2 <= '0';
             nozero <= '0'; noone <= '0'; notwo <= '0';
when downB=> updown <= '0'; motorDir <= '1';  move <= '0'; enable <= '0';
             atlvl0 <= '0'; atlvl1 <= '0'; atlvl2 <= '0';
             at0 <= '0'; at1 <= '0'; at2 <= '0';
             nozero <= '0'; noone <= '0'; notwo <= '0';
end case;
```

```
end process stepper ;


temproc: process(clock)
begin
   if(clock'event and clock='1') then
         if(s = upA or s = upB or s = downB or s = downA) then
            temp <= temp+1;
            if(temp = N)then
              temp <= 0;
            end if;
         elsif(s = s0 or s = s1 or s = s2) then
            temp <= 0;
         end if;
    end if;
end process;


waitLevelproc: process(clock)
begin
   if(clock'event and clock='1')then
      if((s = s0 or s = s1 or s = s2) and waitLevel <= 100000000) then
        waitLevel <= waitLevel + 1;
      elsif(s = upA or s = upB or s = downA or s = downB) then
        waitLevel <= 0;
      end if;
   end if;
end process;
end Behavioral;
```

- **This is the second file named (clockDiv)**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity clockDiv is
   Port ( clk,move:in std_logic;motorclock: out std_logic);
end clockDiv;

architecture Behavioral of clockDiv is
signal divider:integer:=0;
signal tmp:std_logic;
begin
process(clk)
begin
   if(clk='1' and clk'event) then
      divider<=divider+1;
      if(divider=50000) then tmp<='1';
      elsif(divider=100000) then tmp<='0'; divider<=1;
      end if;

   end if;
   if (move <= '0') then motorclock <= tmp;
   elsif (move <= '1') then motorclock <= '0';
   end if;
end process;
end Behavioral;
```

- **This is the last file named (finalelevator)**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity finalelevator is
Port ( lvl0out, lvl1out, lvl2out, lvl0in, lvl1in, lvl2in, clk, reset : in std_logic;
motorDir, atlvl0, atlvl1, atlvl2, motorclock, enable, at0, at1, at2 : out std_logic);
end finalelevator;

architecture Behavioral of finalelevator is
signal move:std_logic;
component clockDiv is
Port ( clk, move: in std_logic; motorclock: out std_logic);
end component;
component final is
Port (lvl0out, lvl1out, lvl2out, lvl0in, lvl1in, lvl2in, clock, reset : in std_logic;
   motorDir, atlvl0, atlvl1, atlvl2, move, enable, at0, at1, at2: out std_logic );
end component;
begin
unit1: clockDiv port map(clk,move,motorclock);
unit2: final port map(lvl0out, lvl1out, lvl2out, lvl0in, lvl1in, lvl2in, clk, reset,
   motorDir, atlvl0, atlvl1, atlvl2, move, enable, at0, at1, at2);

end Behavioral;
```

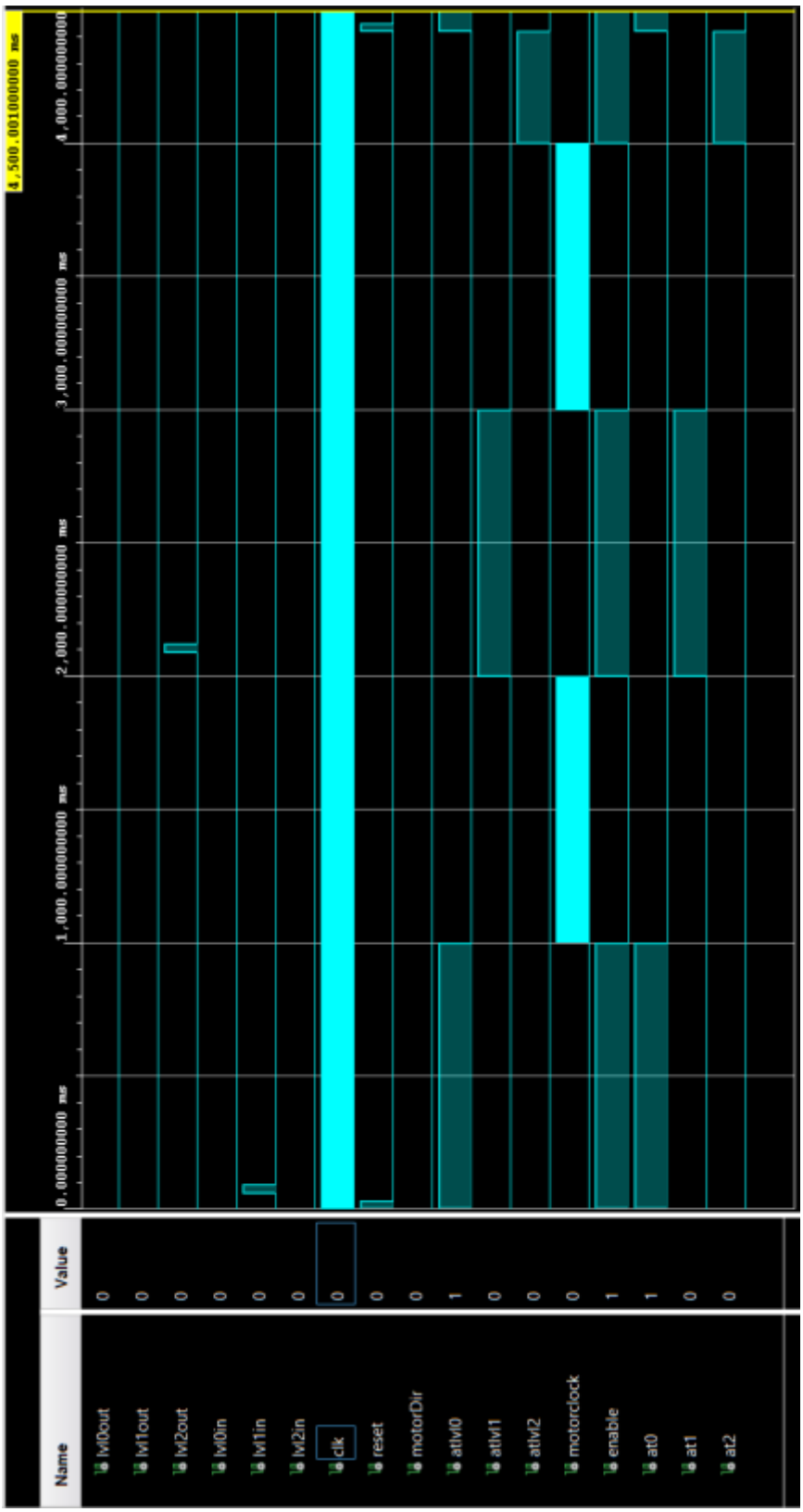
4. Testing and Simulation

- **Test bench**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.Numeric_Std.all;

entity finalelevator_tb is
end finalelevator_tb;

architecture behavior of finalelevator_tb is
component finalelevator
    Port (lvl0out, lvl1out, lvl2out, lvl0in, lvl1in, lvl2in, clk, reset : in std_logic;
     motorDir, atlvl0, atlvl1, atlvl2, motorclock, enable, at0, at1, at2 : out std_logic );
 end component;

 signal lvl0out, lvl1out, lvl2out, lvl0in, lvl1in, lvl2in, clk, reset: std_logic;
 signal motorDir, atlvl0, atlvl1, atlvl2, motorclock, enable, at0, at1, at2: std_logic ;

begin
 uut: finalelevator port map ( lvl0out  => lvl0out,
                lvl1out  => lvl1out,
                lvl2out  => lvl2out,
                lvl0in   => lvl0in,
                lvl1in   => lvl1in,
                lvl2in   => lvl2in,
                clk    => clk,
                reset    => reset,
                motorDir => motorDir,
                atlvl0   => atlvl0,
```

```
                atlvl1   => atlvl1,
                atlvl2   => atlvl2,
                motorclock => motorclock,
                enable   => enable,
                at0      => at0,
                at1      => at1,
                at2      => at2 );

  stimulus: process
  begin
    lvl0in <= '0'; lvl1in <= '0'; lvl2in <= '0';
    lvl0out <= '0'; lvl1out <= '0'; lvl2out <= '0';
    reset <= '1'; wait for 30 ms;
    reset <= '0'; wait for 30 ms;
    lvl1in <= '1'; wait for 30 ms;
    lvl1in <= '0'; wait for 2000 ms;
    lvl2out <= '1'; wait for 30 ms;
    lvl2out <= '0'; wait for 2300 ms;
    reset <= '1'; wait for 30 ms;
    reset <= '0'; wait;
  end process;

  clock: process
  begin
     clk <= '0', '1' after 5 ns; -- clock_period / 2
     wait for 10 ns; -- clock_period
  end process;
end;
```

- **Simulation results**



In the simulation results, a signal was sent from lvl1in at 60 ms. We can see that at 1 sec when the elevator stayed at ground floor for waitlevel (at0 and atlvl0 changes from 1 to 0), the enable changed to 0 and the motorclock started a waveform (since the simulation is zoomed out, it appears as one block at HIGH however in reality the motorclock keeps oscillating between 0 and 1 for a period of 1 ms, frequency 10 KHz; the same goes for the clock except that its period is 10 ns). We left state s0 and entered transition state upA.

At 2 sec, the motor finished rotating (enable = 1 and motorclock = 0) and we can see that atlvl1 and at1 are now HIGH which signifies that the motor arrived at the first floor.

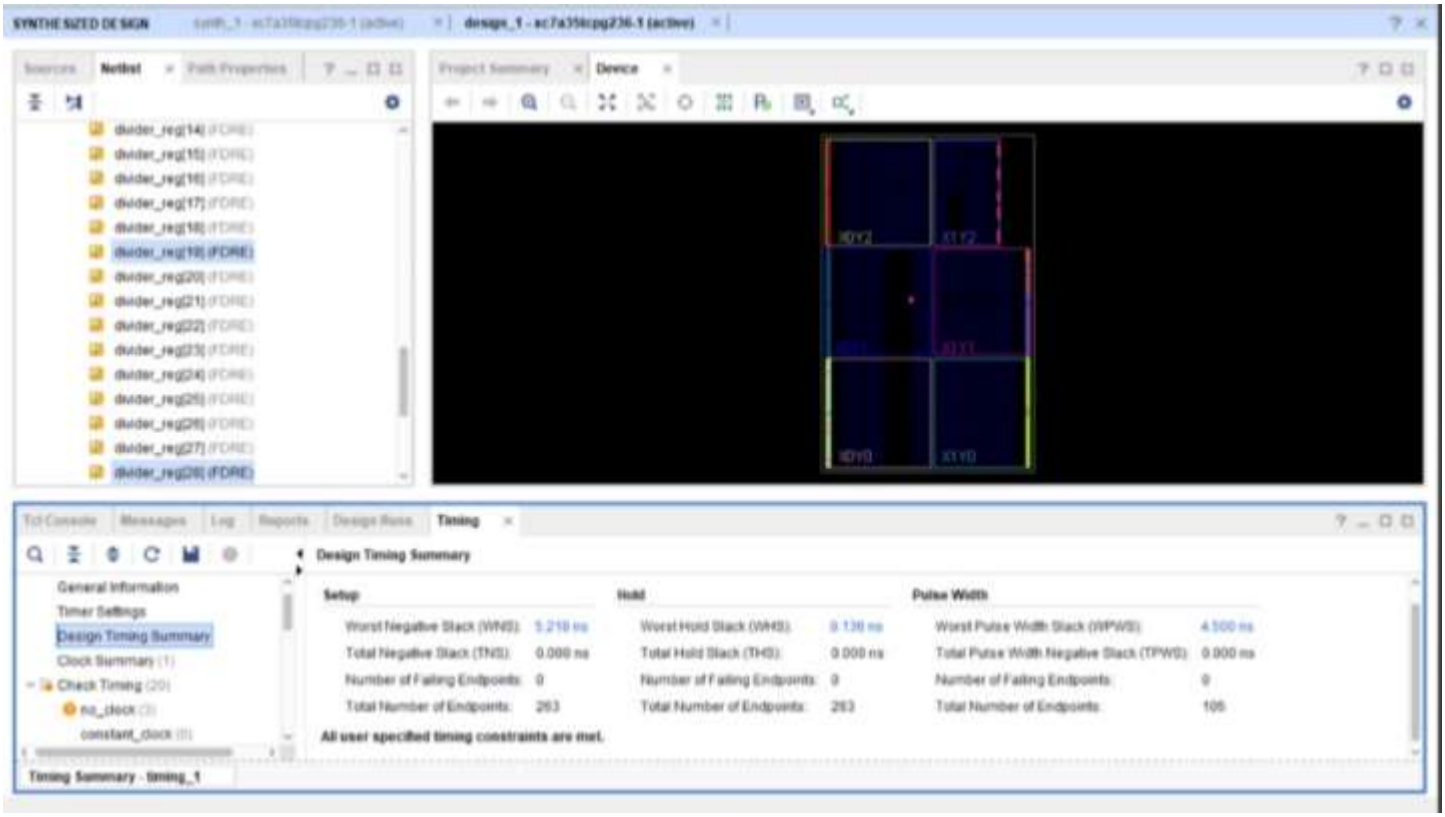
After arriving by few milliseconds, a signal was sent from lvl2out. Again after waiting 1 sec (which corresponds to waitLevel; now we are at t = 3 sec) the elevator starts moving (enable = 0 and motorclock starts oscillating).

Finally at t = 4 sec, the elevator arrives at second floor (at2 and atlvl2 = 1).

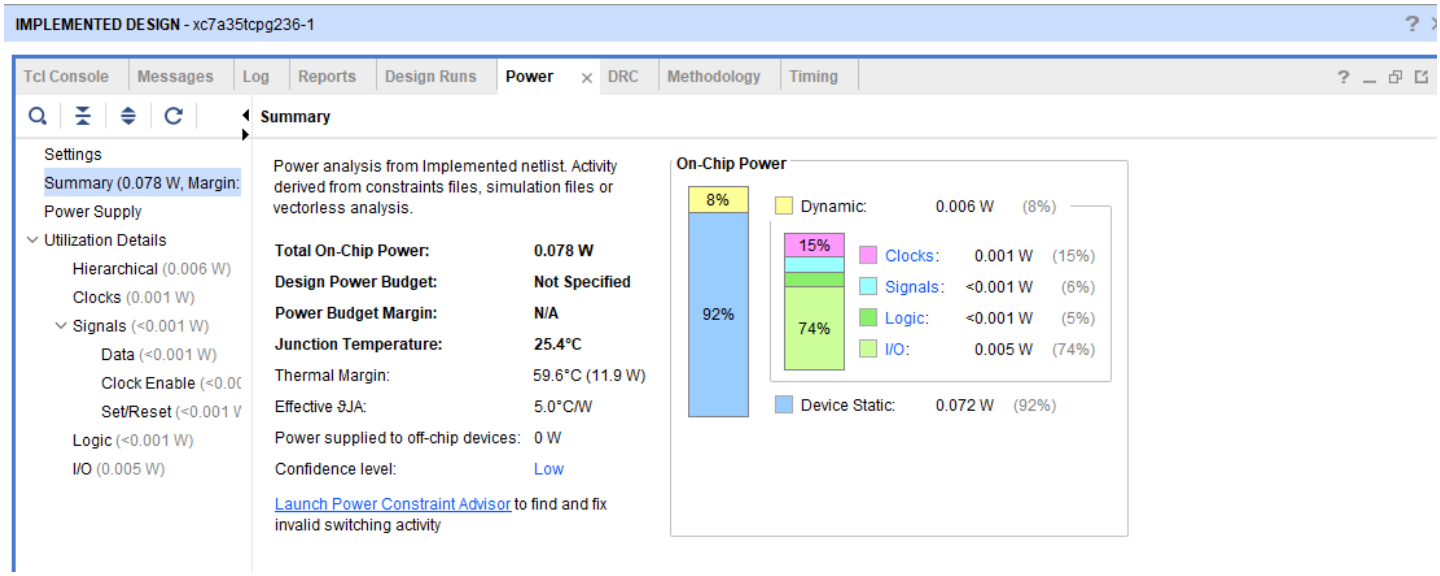
After 300 ms, a reset signal was sent, and we can see that the code is stopped and brought back to initial state (atlvl0 and at0 = 1).

5. Generated Reports (Vivado)

The synthesis implementation device and timing



The power details screen



- The synthesis report

| Tool Version : Vivado v.2020.1 (win64) Build 2902540 Wed May 27 19:54:49 MDT 2020

| Date : Sun Dec 25 17:35:29 2022

| Host : LAPTOP-7U1ULFBR running 64-bit major release (build 9200)

| Command : report_utilization -file finalelevator_utilization_synth.rpt -pb finalelevator_utilization_synth.pb

| Design : finalelevator

| Device : 7a35tcpg236-1

| Design State : Synthesized

Utilization Design Information

Table of Contents

---

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	81	0	20800	0.39
LUT as Logic	81	0	20800	0.39
LUT as Memory	0	0	9600	0.00
Slice Registers	105	0	41600	0.25
Register as Flip Flop	104	0	41600	0.25
Register as Latch	1	0	41600	<0.01
F7 Muxes	1	0	16300	<0.01
F8 Muxes	0	0	8150	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed, for a more realistic count.

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-

| 0 | Yes | - | - |

| 0 | Yes | - | Set |

| 1 | Yes | - | Reset |

| 3 | Yes | Set | - |

| 101 | Yes | Reset | - |

+-------+--------------+-------------+--------------+

2. Memory

+----------------+------+-------+-----------+-------+

| Site Type | Used | Fixed | Available | Util% |

+----------------+------+-------+-----------+-------+

| Block RAM Tile | 0 | 0 | 50 | 0.00 |

| RAMB36/FIFO* | 0 | 0 | 50 | 0.00 |

| RAMB18 | 0 | 0 | 100 | 0.00 |

+----------------+------+-------+-----------+-------+

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

3. DSP

+-----------+------+-------+-----------+-------+

| Site Type | Used | Fixed | Available | Util% |

+-----------+------+-------+-----------+-------+

| DSPs | 0 | 0 | 90 | 0.00 |

+-----------+------+-------+-----------+-------+

4. IO and GT Specific

+----------------------------+------+-------+-----------+-------+

| Site Type | Used | Fixed | Available | Util% |

+----------------------------+------+-------+-----------+-------+

| Bonded IOB | 17 | 0 | 106 | 16.04 |

| Bonded IPADs | 0 | 0 | 10 | 0.00 |

| Bonded OPADs | 0 | 0 | 4 | 0.00 |

| PHY_CONTROL | 0 | 0 | 5 | 0.00 |

| PHASER_REF | 0 | 0 | 5 | 0.00 |

| OUT_FIFO | 0 | 0 | 20 | 0.00 |

| IN_FIFO | 0 | 0 | 20 | 0.00 |

| IDELAYCTRL | 0 | 0 | 5 | 0.00 |

| IBUFDS | 0 | 0 | 104 | 0.00 |

| GTPE2_CHANNEL | 0 | 0 | 2 | 0.00 |

| PHASER_OUT/PHASER_OUT_PHY | 0 | 0 | 20 | 0.00 |

| PHASER_IN/PHASER_IN_PHY | 0 | 0 | 20 | 0.00 |

| IDELAYE2/IDELAYE2_FINEDELAY | 0 | 0 | 250 | 0.00 |

| IBUFDS_GTE2 | 0 | 0 | 2 | 0.00 |

| ILOGIC | 0 | 0 | 106 | 0.00 |

| OLOGIC | 0 | 0 | 106 | 0.00 |

+----------------------------+------+-------+-----------+-------+

5. Clocking

Site Type	Used	Fixed	Available	Util%
BUFGCTRL	1	0	32	3.13
BUFIO	0	0	20	0.00
MMCME2_ADV	0	0	5	0.00
PLLE2_ADV	0	0	5	0.00
BUFMRCE	0	0	10	0.00
BUFHCE	0	0	72	0.00
BUFR	0	0	20	0.00

6. Specific Feature

Site Type	Used	Fixed	Available	Util%
BSCANE2	0	0	4	0.00
CAPTUREE2	0	0	1	0.00
DNA_PORT	0	0	1	0.00
EFUSE_USR	0	0	1	0.00
FRAME_ECCE2	0	0	1	0.00
ICAPE2	0	0	2	0.00
PCIE_2_1	0	0	1	0.00
STARTUPE2	0	0	1	0.00

| XADC | 0 | 0 | 1 | 0.00 |

+-------------+------+-------+-----------+-------+

7. Primitives

+----------+------+---------------------+

| Ref Name | Used | Functional Category |

+----------+------+---------------------+

| FDRE | 101 | Flop & Latch |

| LUT2 | 53 | LUT |

| CARRY4 | 35 | CarryLogic |

| LUT3 | 16 | LUT |

| LUT6 | 13 | LUT |

| LUT4 | 11 | LUT |

| OBUF | 9 | IO |

| IBUF | 8 | IO |

| LUT5 | 7 | LUT |

| LUT1 | 3 | LUT |

| FDSE | 3 | Flop & Latch |

| MUXF7 | 1 | MuxFx |

| LDCE | 1 | Flop & Latch |

| BUFG | 1 | Clock |

+----------+------+---------------------+

8. Black Boxes

+----------+------+

| Ref Name | Used |

+----------+------+

9. Instantiated Netlists

+----------+------+

| Ref Name | Used |

+----------+------+

- The implementation report

| Tool Version : Vivado v.2020.1 (win64) Build 2902540 Wed May 27 19:54:49 MDT 2020

| Date : Sun Dec 25 17:37:55 2022

| Host : LAPTOP-7U1ULFBR running 64-bit major release (build 9200)

| Command : report_drc -file finalelevator_drc_opted.rpt -pb finalelevator_drc_opted.pb -rpx finalelevator_drc_opted.rpx

| Design : finalelevator

| Device : xc7a35tcpg236-1

| Speed File : -1

| Design State : Synthesized

Report DRC

Table of Contents

---

1. REPORT SUMMARY

Netlist: netlist

Floorplan: design_1

Design limits: <entire design considered>

Ruledeck: default

Max violations: <unlimited>

Violations found: 0

+------+----------+-------------+------------+

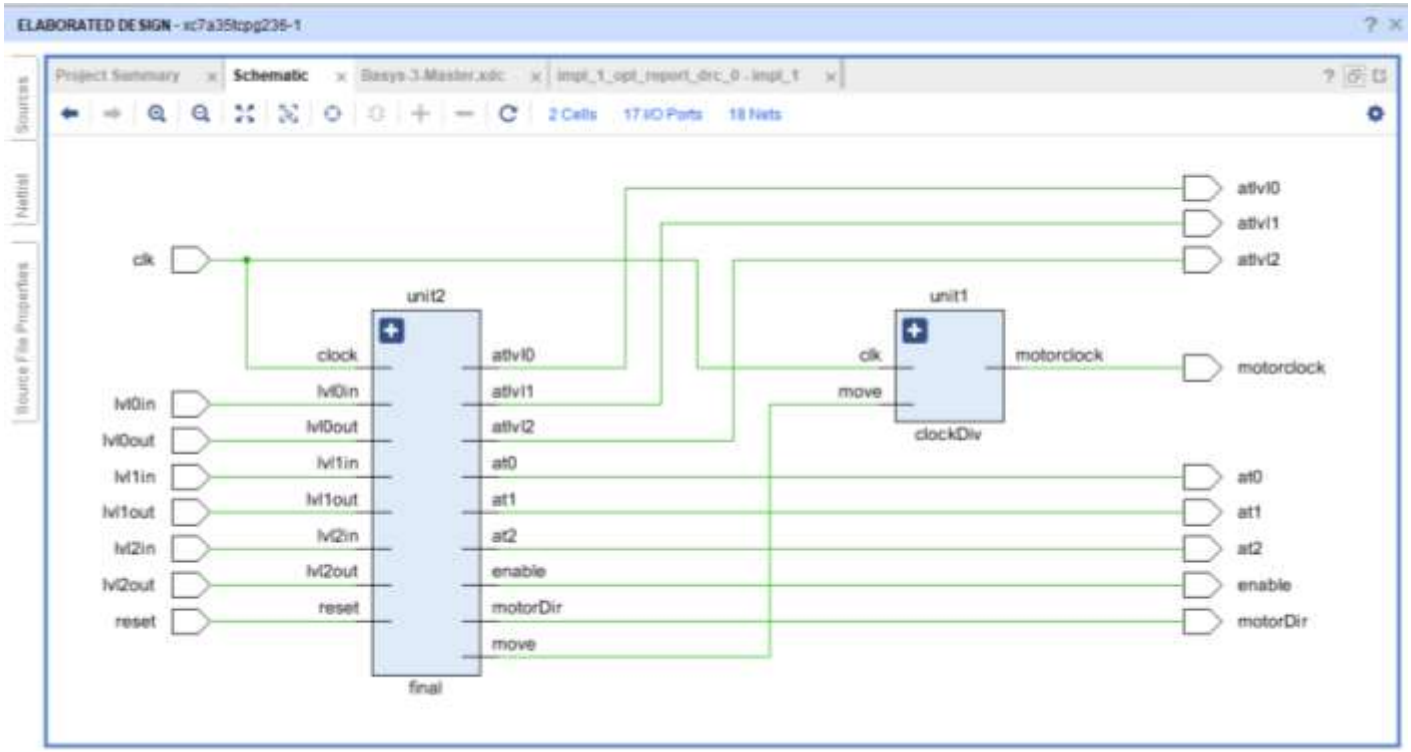
| Rule | Severity | Description | Violations |

+------+----------+-------------+------------+

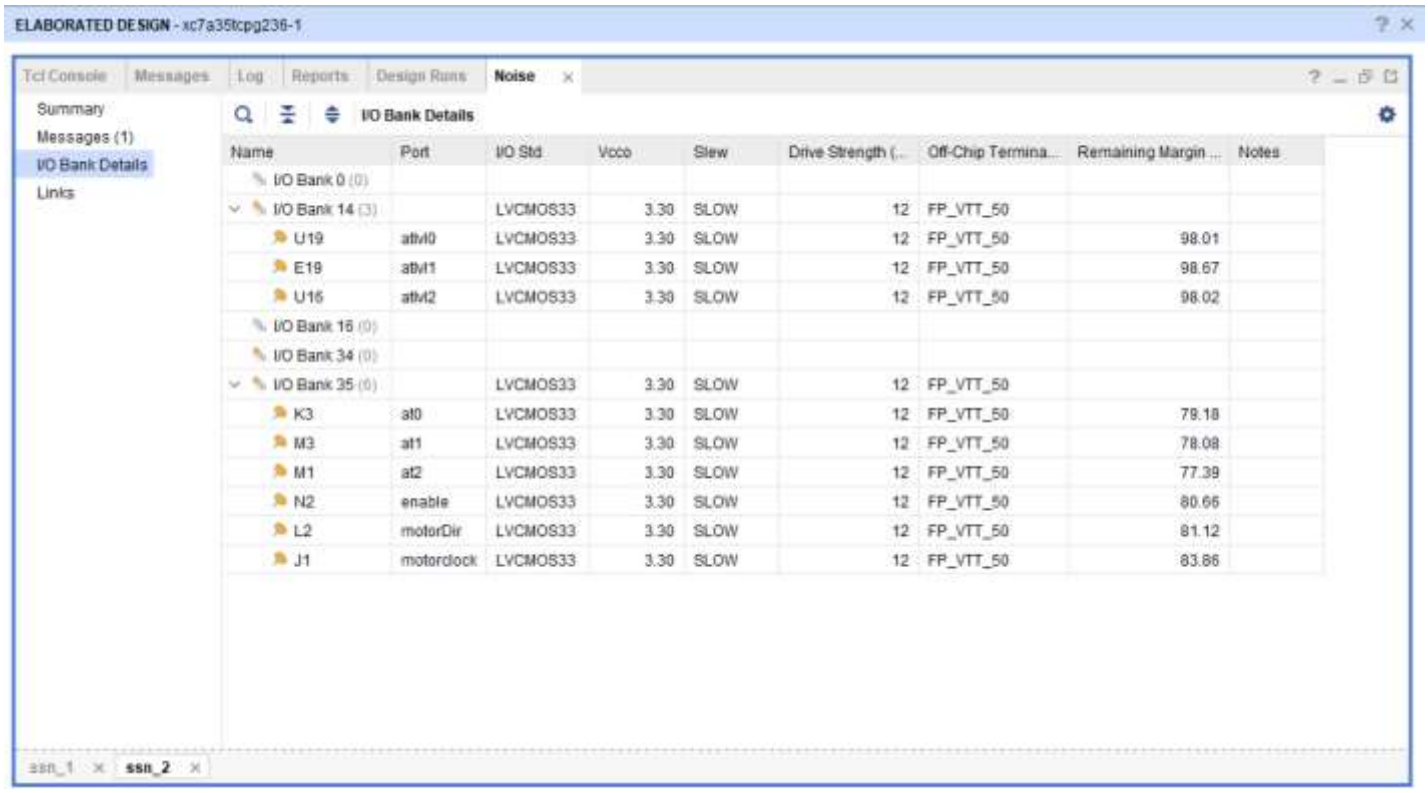
+------+----------+-------------+------------+

2. REPORT DETAILS

- The schematic



- The report noise



- Clock Interaction Summary