بسم الله الرحمن الرحیم

پروژه طراحی سیستم های دیجیتال

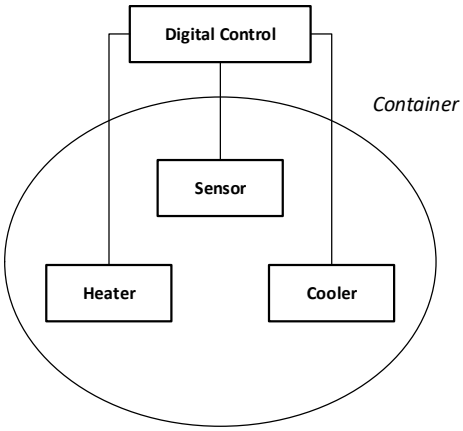
استاد : دکتر رضایی

پدرام جبارزاده – ۹۶۲۱۳۰۱۰

تابستان ۱۴۰۰

طراحی و پیاده‌سازی واحد کنترل دیجیتال یک سیستم انکوباتور (Incubator)

هدف این پروژه طراحی و پیاده‌سازی واحد کنترل دیجیتال یک سیستم انکوباتور (Incubator) به منظور کاربرد صنعتی است. در این سیستم مطابق شکل زیر، یک حسگر دما، یک واحد خنک‌کننده (Cooler) مجهز به پنکه و یک واحد گرم‌کننده (Heater) وجود دارد.



شکل (۱): شکل کلی سیستم انکوباتور

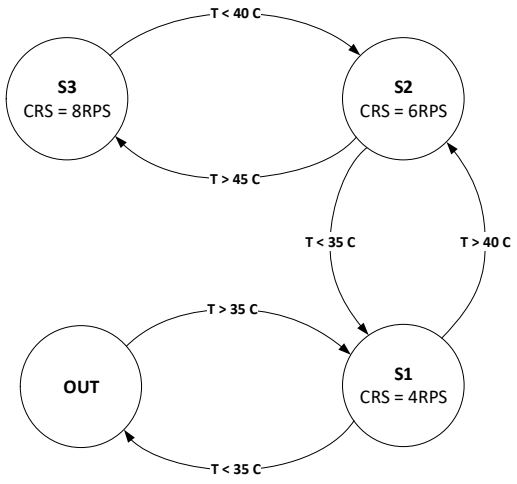
در این سیستم یک حسگر دما وجود دارد که دمای محفظه را که میان ۱۰- تا ٦٠+ درجه سانتیگراد متغیر است می‌خواند و در قالب یک عدد ۸ بیتی به سیستم شما تحویل می‌دهد. دما هر دقیقه یک بار از حسگر دریافت می‌شود و بر اساس آن واحد کنترل دیجیتال تصمیم می‌گیرد که:

۱- چگونه واحدهای گرم‌کننده و سردکننده را فعال و غیر فعال کند.

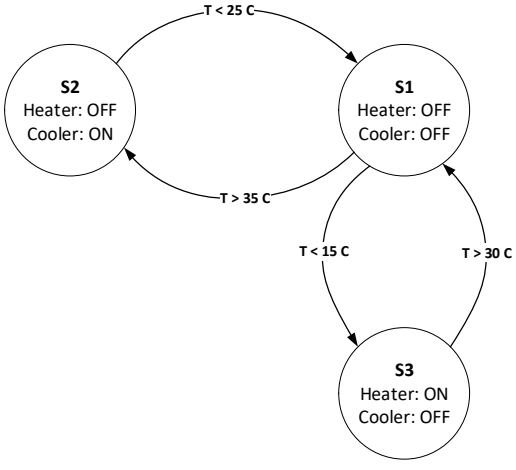
۲- چگونه در صورت فعال بودن واحد سردکننده دور پنکهٔ آن را تنظیم کند.

روش کار واحد کنترل دیجیتال توسط دو نمودار حالت توصیف شده است؛

در این نمودار حالت برچسب‌های مربوط به یال‌ها نشان‌دهنده تغییرات دمایی هستند که باعث تغییر حالت در واحد کنترل دیجیتال شده و باعث واکنش سیستم به شکل روشن و خاموش شدن خنک‌کننده و یا تغییر دور پنکه می‌شوند.



شکل (۲): الف) ماشین حالت خنک کننده



شکل(۲): ب) ماشین حالت سیستم Incubator

دقت کنید که در نمودارهای فوق نمودار حالت سمت راست زمانی فعال می‌شود (از حالت out خود خارج می‌شود) که نمودار حالت سمت چپ در حالت S۲ که در آن خنک‌کننده روشن است قرار گرفته باشد و اگر نمودار سمت چپ در حالت S۲ خود نباشد نمودار حالت سمت راست غیر فعال می‌شود (یعنی وارد حالت out خود می‌شود) چون معنی ندارد که وقتی خنک‌کننده خاموش است دور آن تنظیم گردد.

در این پروژه هدف طراحی واحد کنترل دیجیتال است و قسمت‌های حسگر، گرم‌کننده و خنک‌کننده وجود خارجی ندارند؛ لذا دانشجویان باید عددی که مثلاً توسط حسگر دما خوانده می‌شود خود با استفاده از امکانات بورد FPGA به مدار اعمال کنند و واکنش سیستم (مانند دور پنکه) را به جای آنکه گرم‌کننده و سردکننده واقعی روشن و خاموش شوند با روشن و خاموش شدن LED نمایش دهند.

پیاده سازی

برای این پروژه ما سه موجودیت زیر را درنظر گرفته ایم:

- **Digital_Control**: واحد کنترل دیجیتال سیستم انکوباتور
- **Cooler**: واحد خنک کننده برای کنترل دور پنکه
- **Incubator**: یک موجودیت براي اينكه بتوانيم اجزاي سيستم انكوباتور خود را در آن به هم وصل كنيم.

این سیستم میتواند با دو موجودیت نیز پیاده سازي شود. اما دلیل استفاده از سه موجودیت از این نظر است که در آینده براي توسعه این سیستم به مشكلي بر نخوريم با طراحي يک بخش جدا مثل Heater آن را در موجودیت Incubator به سیستم کلی متصل نماییم.

در شرح پروژه طبق شکل(۲) الف) و ب) دو ماشین حالت برای سیستم خنک کننده و کنترل دیجیتال در اختیار ما قرار داده شده است.

در ماشین حالت واحد کنترل دیجیتال طبق شکل (۲) ب، میتوانیم برای هر حالت خروجی و نام های زیر را در نظر بگیریم:

- S۱ [constant S۱ : STATE_TYPE := "۰۰";] → ۰۰ (Heater: Off, Cooler: Off)
- S۲ [constant S۲ : STATE_TYPE := "۰۱";] → ۰۱ (Heater: Off, Cooler: On)
- S۳ [constant S۳ : STATE_TYPE := "۱۰";] → ۱۰ (Heater: On, Cooler: Off)

با توجه به حالت و خروجی هایی که معین کردیم میتوان گفت که برای سیستم کنترل دیجیتال انکوباتور میتوانیم از ماشین حالت Medvedev استفاده کنیم چراکه خروجی ها با حالت(State)ها یکی هستند یا به عبارتی خروجی های ما برابر با همان حالت(State)های ما هستند.

در ماشین حالت سیستم خنک کننده طبق شکل(۲) الف، میتوانیم برای هر حالت خروجی و نام های زیر را درنظر بگیریم:

- S۱ [constant S۱ : STATE_TYPE := "۰۰";] → ۰۱۰۰ (CRS: ٤RPS)
- S۲ [constant S۲ : STATE_TYPE := "۰۱";] → ۰۱۱۰ (CRS: ٦RPS)
- S۳ [constant S۳ : STATE_TYPE := "۱۰";] → ۱۰۰۰ (CRS: ۸RPS)
- OUT [constant OUT_STATE : STATE_TYPE := "۱۱";] → ۰۰۰۰ (CRS: ۰RPS)

با توجه به حالت(State)ها و خروجی هایی که طبق نمودار حالت تعریف کرده ایم، مشاهده میشود که خروجی با خود حالات مفاوت است. در اینجا مناسب است که از ماشین حالت مور(Moore) استفاده نماییم.

موجودیت DIGITAL_CONTROL که FSM خود را با روش Medvedev پیاده سازی کرده ایم بصورت زیر است: (۲ Proccess)

```
                    NEXTSTATE <= S۱;
                else
                    NEXTSTATE <= S۲;
                end if;

            when S۳ =>
                if TEMPRATURE > "۰۰۰۱۱۱۱۰" then
                    NEXTSTATE <= S۱;
                else
                    NEXTSTATE <= S۳;
                end if;

            when others =>
                NEXTSTATE <= S۱;
        end case;
    end process CMB;

    (HEATER_STATUS, COOLER_STATUS) <= STATE;
end DATAPATH;
```

```
    REG:process(CLK, RESET)
        begin
            if RESET = '۱' then
                STATE <= S۱;
            elsif CLK'event and CLK='۱' then
                STATE <= NEXTSTATE;
            end if;
        end process REG;

     CMB:process(STATE, NEXTSTATE, TEMPRATURE)
         begin
         case STATE is
            when S۱ =>
                if TEMPRATURE < "۰۰۰۰۱۱۱۱" then
                    NEXTSTATE <= S۳;
                elsif TEMPRATURE > "۰۰۱۰۰۰۱۱" then
                    NEXTSTATE <= S۲;
                else
                    NEXTSTATE <= S۱;
                end if;

            when S۲ =>
                if TEMPRATURE < "۰۰۰۱۱۰۰۱" then
```

همانطور که مشاهده میشود در قطعه کد بالا از others استفاده شده. در برای سنتزپذیری این موضوع از روش Hand Coding استفاده شده است. به این صورت که با تعریف ثوابت زیر کدینگ های خود را کنترل نموده ایم:

```
subtype STATE_TYPE is std_ulogic_vector (۱ downto ۰) ;
constant S۱ : STATE_TYPE := "۰۰";
constant S۲ : STATE_TYPE := "۰۱";
constant S۳ : STATE_TYPE := "۱۰";
```

که در این حالت استفاده از others معنی پیدا میکند.

موجودیت COOLER که FSM خود را با روش Moore پیاده سازی کرده ایم بصورت زیر است: (۳ Proccess)

```
library IEEE;
use IEEE.STD_LOGIC_١١٦٤.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

entity COOLER is
    port(TEMPRATURE : in std_logic_vector(٧ downto ٠);
         CLK, RESET : in std_logic;
         ENABLE : in std_logic;
         CRS :out std_logic_vector(٣ downto ٠));
end COOLER;

architecture DATAPATH of COOLER is
subtype STATE_TYPE is std_ulogic_vector (١ downto ٠);
constant S١ : STATE_TYPE := "٠٠";
constant S٢ : STATE_TYPE := "٠١";
constant S٣ : STATE_TYPE := "١٠";
constant OUT_STATE : STATE_TYPE := "١١";

signal NEXT_CRS: std_logic_vector(٣ downto ٠);

signal STATE, NEXTSTATE : STATE_TYPE := OUT_STATE;
begin
    REG:process (CLK, RESET)
    begin
        if RESET='١' then
            STATE <= OUT_STATE;
            CRS <= "٠٠٠٠";
        elsif CLK'event and CLK='١' then
            STATE <= NEXTSTATE;
            CRS <= NEXT_CRS;
        end if;
    end process REG;

    CMB:process(ENABLE, STATE, NEXTSTATE, TEMPRATURE) is
    begin
        if ENABLE='١' then
            case STATE is
                when OUT_STATE =>
                    if(TEMPRATURE > "٠٠١٠٠٠١١") then
                        NEXTSTATE <= S١;
                    else
                        NEXTSTATE <= OUT_STATE;
                    end if;
                when S١ =>
                    if(TEMPRATURE > "٠٠١٠١٠٠٠") then
```

```
                    NEXTSTATE <= S٢;
                elsif(TEMPRATURE<"٠٠١١٠٠١") then
                    NEXTSTATE <= OUT_STATE;
                else
                    NEXTSTATE <= S١;
                end if;

            when S٢ =>
                if(TEMPRATURE > "٠٠١٠١١٠١") then
                    NEXTSTATE <= S٣;
                elsif(TEMPRATURE < "٠٠١٠٠٠١١") then
                    NEXTSTATE <= S١;
                else
                    NEXTSTATE <= S٢;
                end if;

            when S٣ =>
                if(TEMPRATURE < "٠٠١٠١٠٠٠") then
                    NEXTSTATE <= S٢;
                else
                    NEXTSTATE <= S٣;
                end if;

            when others =>
                NEXTSTATE <= OUT_STATE;
        end case;
    else
        NEXTSTATE <= OUT_STATE;
    end if;
end process CMB;

OUTPUT: process (STATE, TEMPRATURE)
begin
    case STATE is
        when OUT_STATE =>
             NEXT_CRS <= "٠٠٠٠";
        when S١ =>
             NEXT_CRS <= "٠١٠٠";
        when S٢ =>
             NEXT_CRS <= "٠١١٠";
        when S٣ =>
             NEXT_CRS <= "١٠٠٠";
        when others =>
            NEXT_CRS <= "٠٠٠٠";
    end case;
end process OUTPUT;
end DATAPATH;
```

در موجودیت COOLER میتوانیم از روش اضافه کردن وضعیت(state) Dummy نیز استفاده کنیم. اگر بخواهیم از این روش استفاده کنیم کافیست قطعه کد زیر را:

```
subtype STATE_TYPE is std_ulogic_vector (۱ downto ۰);
constant S۱ : STATE_TYPE := "۰۰";
constant S۲ : STATE_TYPE := "۰۱";
constant S۳ : STATE_TYPE := "۱۰";
constant OUT_STATE : STATE_TYPE := "۱۱";
```

با قطعه کد زیر جابجا کنیم:

```
type STATE_TYPE is (S۱, S۲, S۳, OUT_STATE, DUMMY_STATE);
```

و حالت های others در دو روند CMB و OUTPUT را حذف و بجای آن DUMMY_STATE را قرار دهیم.

پس قطعه کد زیر در روند CMB را:

```
when others =>
      NEXTSTATE <= OUT_STATE;
```

را با کد زیر جایگزین میکنیم.

```
when DUMMY_STATE =>
    NEXTSTATE <= OUT_STATE;
```

همچنین در روند OUTPUT نیز قطعه کد زیر را:

```
when others =>
    NEXT_CRS <= "۰۰۰۰";
```

با کد زیر جایگزین میکنیم.

```
when DUMMY_STATE =>
    NEXT_CRS <= "۰۰۰۰";
```

توجه: ما روش Hand Coding را پیشنهاد میکنیم زیرا در روش دوم درواقع ما Coding را به عهده نرم افزار گذاشته ایم در حالی که در روش دوم خودمان بصورت دستی اینکار را انجام داده ایم.

و در پایان موجودیت Incubator که در آن واحدای های مختلف سیستم را به هم وصل میکنیم بصورت زیر است:

```
library IEEE;
use IEEE.STD_LOGIC_١١٦٤.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

entity INCUBATOR is
    port (
        TEMPRATURE : in  STD_LOGIC_vector(٧ downto ٠);
        CLK, RESET : in  STD_LOGIC;
        HEATER_STATUS, COOLER_STATUS  : out  STD_LOGIC;
        CRS : out STD_LOGIC_vector(٣ downto ٠)
    );
end INCUBATOR;

architecture INCUBATOR_ARCH of INCUBATOR is

signal ENABLE_COOLER: STD_LOGIC := '٠';
begin

    DIGITAL_CONTROL_MODULE: DIGITAL_CONTROL port map(
            TEMPRATURE => TEMPRATURE,
            CLK => CLK,
            RESET => RESET,
            HEATER_STATUS => HEATER_STATUS,
            COOLER_STATUS => ENABLE_COOLER
        );
    COOLER_STATUS <= ENABLE_COOLER;

    COOLER_STATUS_MODULE: COOLER port map(
            TEMPRATURE => TEMPRATURE,
            CLK => CLK,
            RESET => RESET,
            ENABLE => ENABLE_COOLER,
            CRS => CRS
        );

end INCUBATOR_ARCH;
```

در صفحه بعد نیز قطعه کد بالا را بصورتی دیگر نوشته ایم.

همچنین میتوانیم پیکربندی را بصورت مجزا بنویسیم اما قطعه کد بالا تمیز تر و خواناتر میباشد. اگر بخواهیم پیکربندی ما جدا نوشته شود بایستی موجودیت Incubator را بصورت زیر تعریف کنیم:

```
library IEEE;
use IEEE.STD_LOGIC_١١٦٤.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;

entity INCUBATOR is
    port (
        TEMPRATURE : in  STD_LOGIC_vector(٧ downto ٠);
        CLK, RESET : in  STD_LOGIC;
        HEATER_STATUS, COOLER_STATUS  : out  STD_LOGIC;
        CRS : out STD_LOGIC_vector(٣ downto ٠)
    );
end INCUBATOR;

architecture INCUBATOR_ARCH of INCUBATOR is

component DIGITAL_CONTROL is
    port (
        TEMPRATURE : in STD_LOGIC_vector( ٧ downto ٠);
        CLK, RESET : in STD_LOGIC;
        HEATER_STATUS, COOLER_STATUS : out STD_LOGIC
    );
end component;

component COOLER is
    port (
        TEMPRATURE : in STD_LOGIC_vector( ٧ downto ٠);
        CLK, RESET : in STD_LOGIC;
        ENABLE : in STD_LOGIC;
        CRS :out STD_LOGIC_vector( ٣ downto ٠)
    );
end component;

signal ENABLE_COOLER: STD_LOGIC := '٠';
begin

    DIGITAL_CONTROL_MODULE: DIGITAL_CONTROL port map(
            TEMPRATURE => TEMPRATURE,
            CLK => CLK,
            RESET => RESET,
            HEATER_STATUS => HEATER_STATUS,
            COOLER_STATUS => ENABLE_COOLER
        );
    COOLER_STATUS <= ENABLE_COOLER;
```

```
    COOLER_STATUS_MODULE: COOLER port map(
            TEMPRATURE => TEMPRATURE,
            CLK => CLK,
            RESET => RESET,
            ENABLE => ENABLE_COOLER,
            CRS => CRS
        );

end INCUBATOR_ARCH;

configuration INCUBATOR_CFG of INCUBATOR is
  for INCUBATOR_ARCH
    for DIGITAL_CONTROL_MODULE : DIGITAL_CONTROL
      use entity work.DIGITAL_CONTROL(DATAPATH);
    end for;

    for COOLER_STATUS_MODULE : COOLER
      use entity work.COOLER(DATAPATH);
    end for;
  end for;
end INCUBATOR_CFG;
```

همانطور که مشاهده میکنیم در حالت اول کد تمیز تر و خواناتری را خواهیم داشت.

شبیه سازی

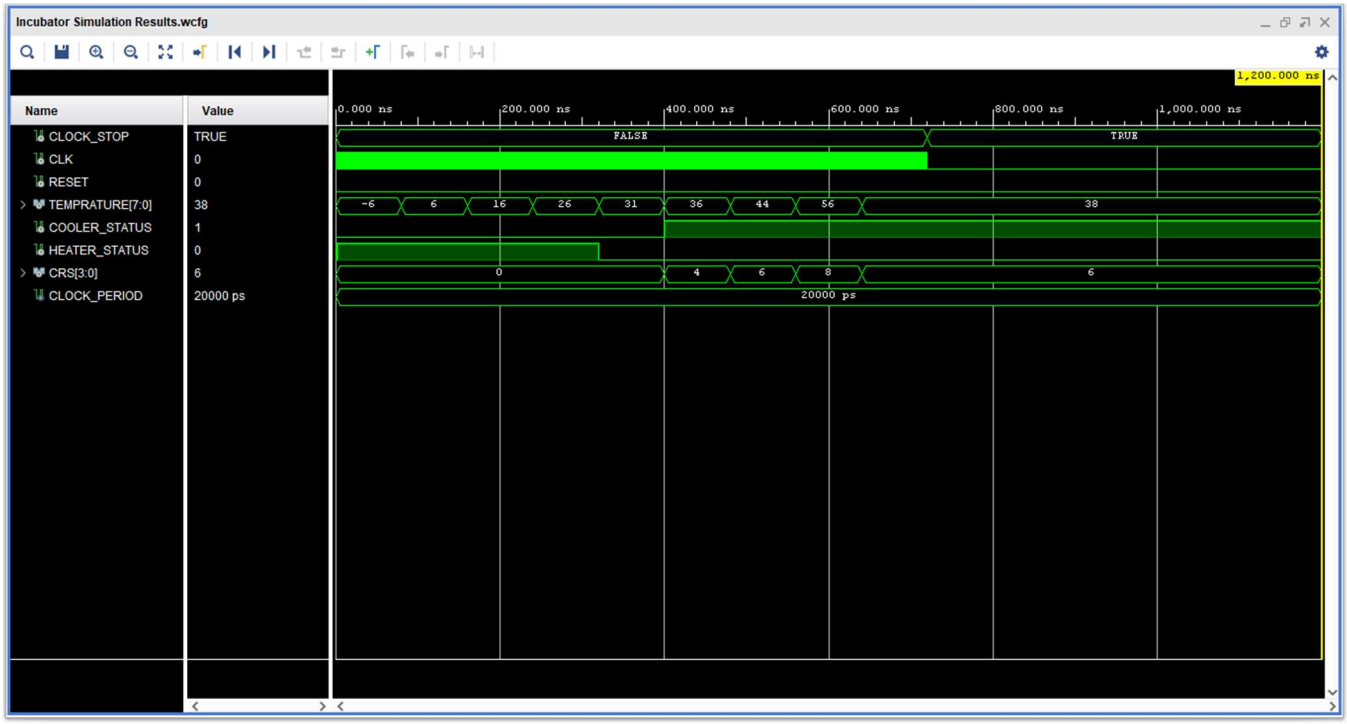
در شبیه سازی این پروژه همانطور که از پیاده سازی موجودیت ها و طراحی ما مشخص بود بایستی از یک کلاک در testbench خود استفاده کنیم. که در نهایت testbench خود را بصورت زیر پیاده سازی نموده ایم:

```
library IEEE;
use IEEE.STD_LOGIC_١١٦٤.ALL;
use ieee.numeric_std.all;
use ieee.math_real.all;

library FINAL;
use FINAL.FINAL_COMPONENTS.all;

entity incubator_tb is
end incubator_tb;

architecture Behavioral of incubator_tb is
    -- Initializing clock signals
    constant CLOCK_PERIOD: time := ٢٠ ns;
    signal CLOCK_STOP: boolean := false;

    -- Initializing incubator signals
    signal CLK, RESET: std_logic := '١';
    signal TEMPRATURE: std_logic_vector(٧ downto ٠);
    signal COOLER_STATUS, HEATER_STATUS: std_logic := '٠';
    signal CRS: std_logic_vector(٣ downto ٠) := (others => '٠');

begin

    CLOCK_GENERATOR: process
    begin
        while not CLOCK_STOP loop
            wait for CLOCK_PERIOD / ١٠٠٠;
            CLK <= not CLK;
        end loop;
        wait;
    end process CLOCK_GENERATOR;

    INCUBATOR_SIM: process
    begin
        RESET <= '٠';
        TEMPRATURE <= std_logic_vector(to_signed(-٦,  TEMPRATURE'length));
        wait for CLOCK_PERIOD * ٤;
```

```
--        RESET <= '1';
--        wait for CLOCK_PERIOD * 4;

--        RESET <= '0';
--        wait for CLOCK_PERIOD * 4;

        TEMPRATURE <= std_logic_vector(to_signed(7,  TEMPRATURE'length));
        wait for CLOCK_PERIOD * 4;

        TEMPRATURE <= std_logic_vector(to_signed(17,  TEMPRATURE'length));
        wait for CLOCK_PERIOD * 4;

        TEMPRATURE <= std_logic_vector(to_signed(27,  TEMPRATURE'length));
        wait for CLOCK_PERIOD * 4;

        TEMPRATURE <= std_logic_vector(to_signed(31,  TEMPRATURE'length));
        wait for CLOCK_PERIOD * 4;

        TEMPRATURE <= std_logic_vector(to_signed(37,  TEMPRATURE'length));
        wait for CLOCK_PERIOD * 4;

        TEMPRATURE <= std_logic_vector(to_signed(44,  TEMPRATURE'length));
        wait for CLOCK_PERIOD * 4;

        TEMPRATURE <= std_logic_vector(to_signed(57,  TEMPRATURE'length));
        wait for CLOCK_PERIOD * 4;

        TEMPRATURE <= std_logic_vector(to_signed(38,  TEMPRATURE'length));
        wait for CLOCK_PERIOD * 4;
        CLOCK_STOP <= true;
        wait;
    end process;

    INC: INCUBATOR
        port map(
            TEMPRATURE =>  TEMPRATURE,
            CLK => CLK,
            RESET => RESET,
            COOLER_STATUS => COOLER_STATUS,
            HEATER_STATUS => HEATER_STATUS,
            CRS => CRS
        );
end Behavioral;
```

در testbench خود برای خوانایی بهتر از تابع to_signed از کتابخانه numeric_std استفاده کرده ایم که با دریافت دو عدد ورودی، عدد باینری علامتدار با طول ورودی دوم و مقدار ورودی اول برمیگرداند.

در نهایت نتایج شبیه سازی بصورت زیر خواهد بود:



شکل (۳): نتایج شبیه سازی testbench انکوباتور

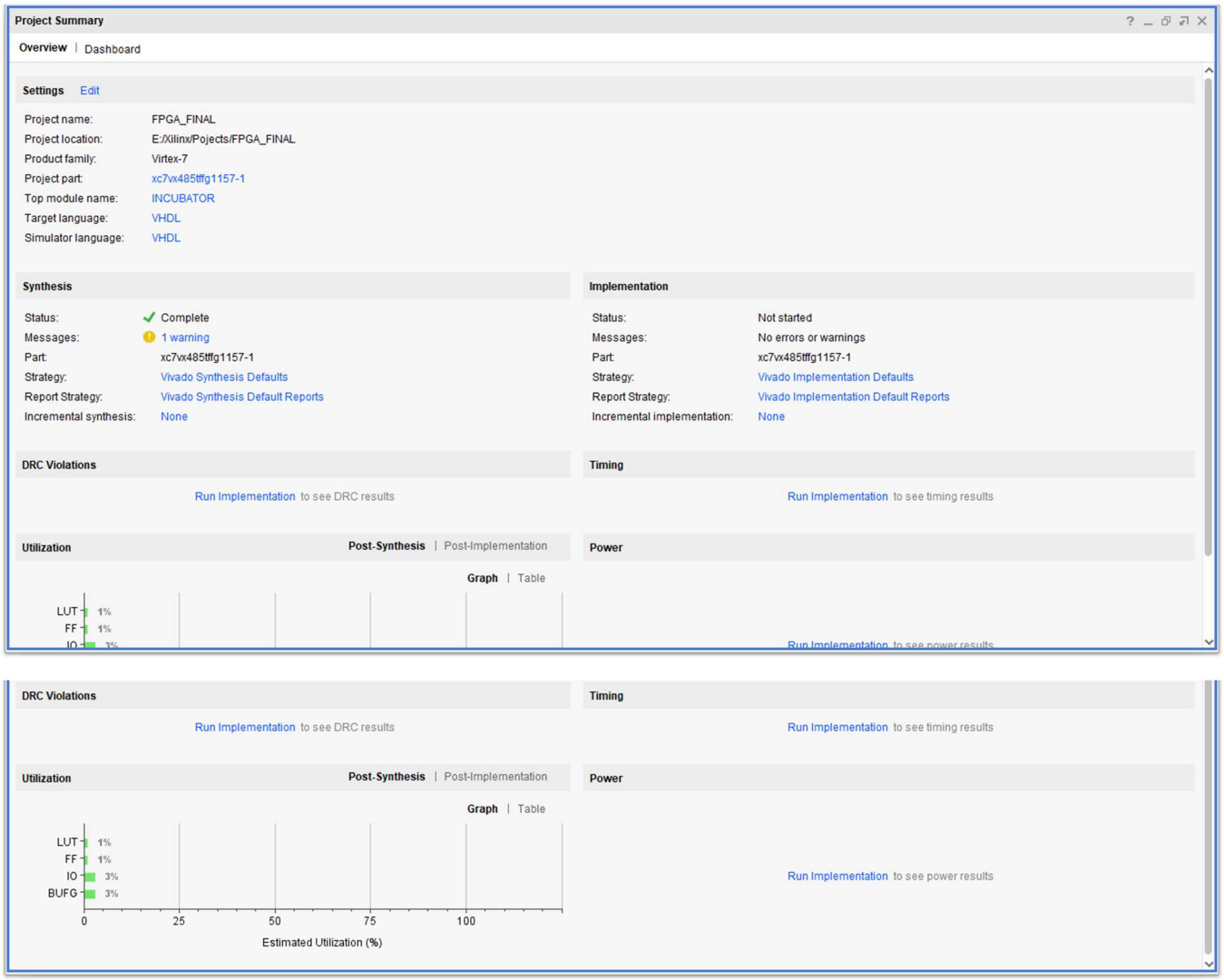
کد های پیکیج FINAL_COMPONENTS که در کتابخانه FINAL تعریف شده اند (بصورت مشابه با میانترم):

```
library IEEE;
use IEEE.STD_LOGIC_١١٦٤.ALL;

package FINAL_COMPONENTS is

    component INCUBATOR is
        port(
            TEMPRATURE: in std_logic_vector(٧ downto ٠);
            CLK, RESET: in std_logic;
            COOLER_STATUS, HEATER_STATUS: out std_logic;
            CRS: out std_logic_vector(٣ downto ٠)
        );
    end component;

end package;

package body FINAL_COMPONENTS is
end package body ;
```

نتایج سنتزپذیری

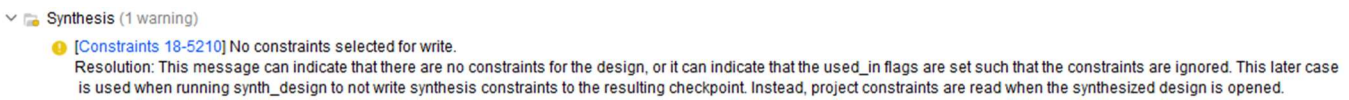
برای بررسی سنتزپذیری از دو نرم افزار Vivado و ISE استفاده نمودیم. از Vivado برای بررسی کلی سنتر پذیری استفاده نمودیم و از ISE آز آنجا که گزارش های کاملتری را ایجاد میکند، برای گزارش گیری استفاده کردیم.

خلاصه نتایج سنتز با نرم افزار Vivado:



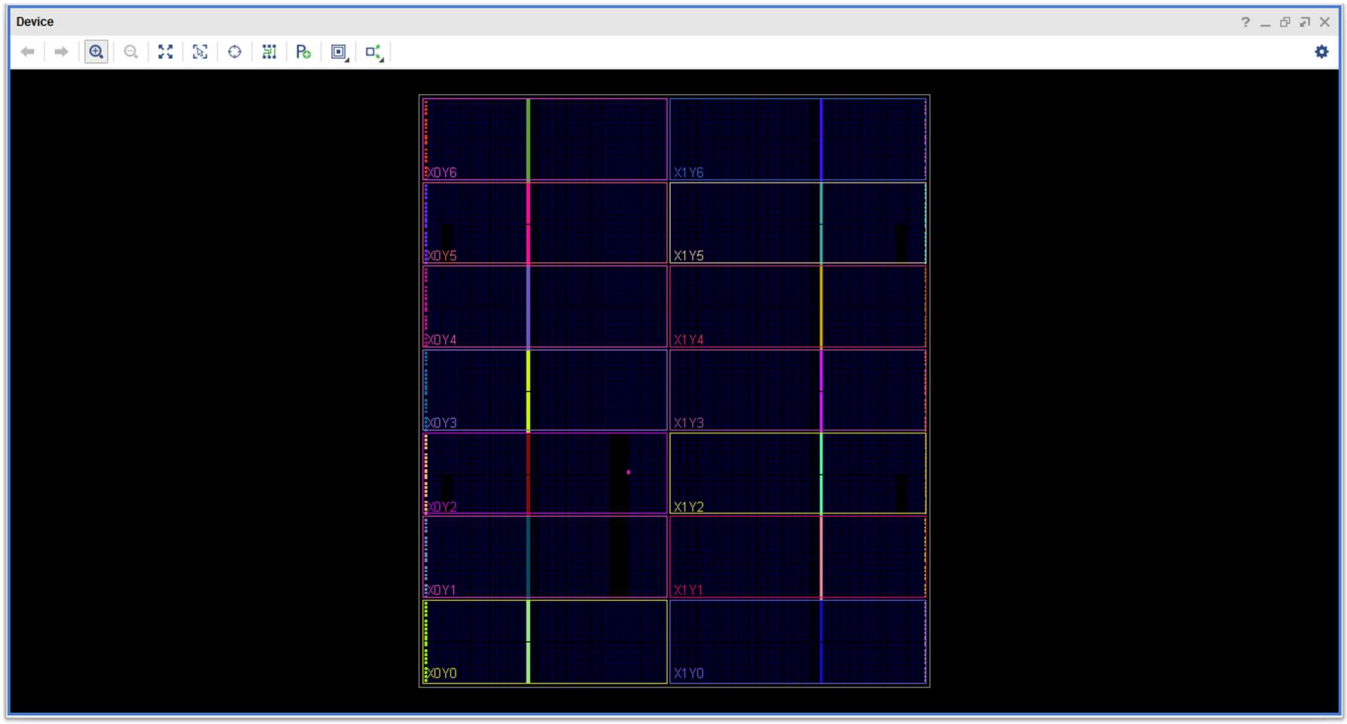
شکل (٤): خلاصه نتایج سنتزپذیری در نرم افزار Vivado

یک Warning مشاهده میشود که بصورت زیر است:



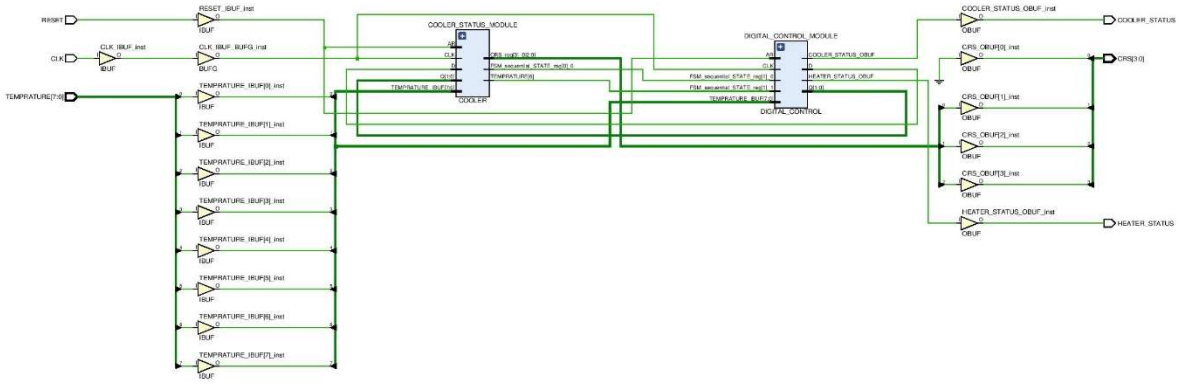
این Warning مربوط به عدم وجود فایل Constraints برای پیکربندی پین ها میباشد که تعریف فایل Constraint خارج از محدوده پروژه میباشد.

نتایج Device:



شکل (۵): خروجی Device در Vivado

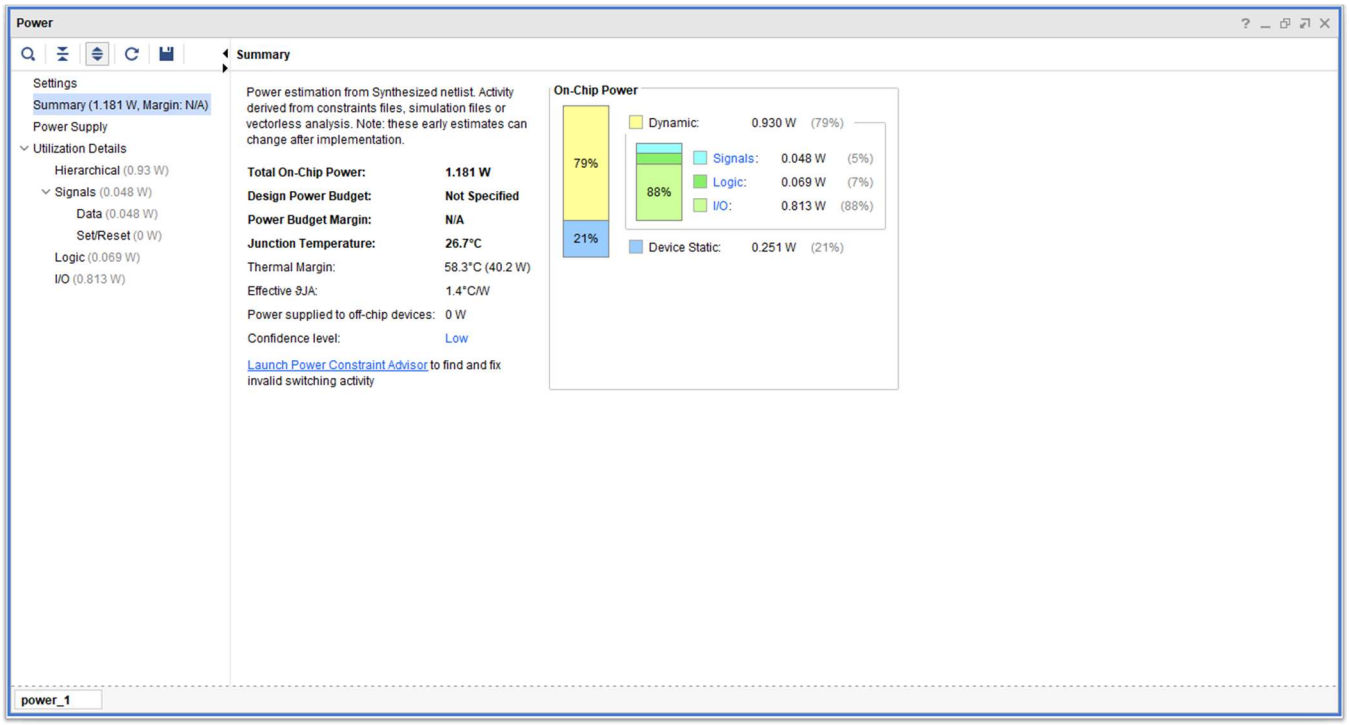
خروجی شماتیک:



شکل (٦): خروجی شماتیک (موجودیت ها بصورت بلوکی) در Vivado

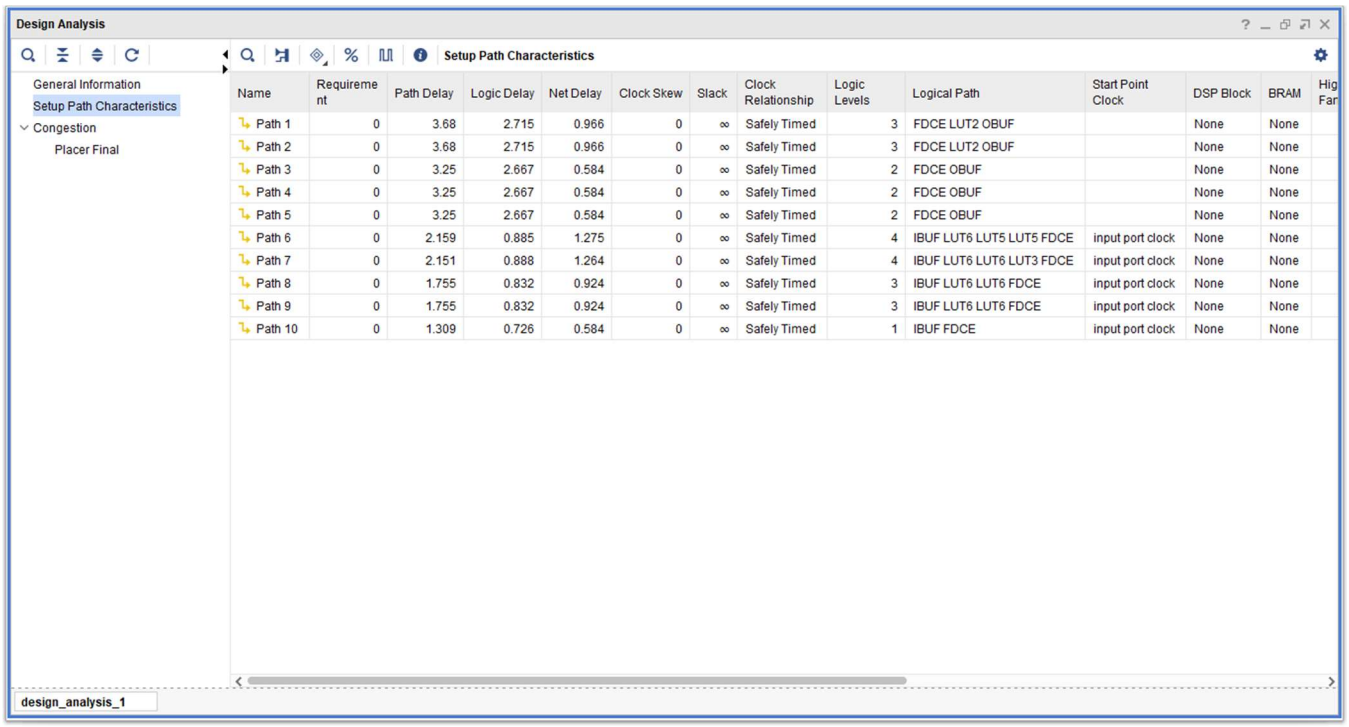
در شماتیک بالا دو موجودیت دیجیتال کنترل و خنک کننده بصورت بلوکی است. برای مشاهده شماتیک کامل فایل Schematic.pdf در فایل های پروژه را بررسی نمایید.

خروجی Power در Vivado:



شکل (۷): خروجی Power در Vivado

خروجی گزارش Design در Vivado:



Name	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Slack	Clock Relationship	Logic Levels	Logical Path	Start Point Clock	DSP Block	BRAM	Hig Far
Path 1	0	3.68	2.715	0.966	0	∞	Safely Timed	3	FDCE LUT2 OBUF		None	None	
Path 2	0	3.68	2.715	0.966	0	∞	Safely Timed	3	FDCE LUT2 OBUF		None	None	
Path 3	0	3.25	2.667	0.584	0	∞	Safely Timed	2	FDCE OBUF		None	None	
Path 4	0	3.25	2.667	0.584	0	∞	Safely Timed	2	FDCE OBUF		None	None	
Path 5	0	3.25	2.667	0.584	0	∞	Safely Timed	2	FDCE OBUF		None	None	
Path 6	0	2.159	0.885	1.275	0	∞	Safely Timed	4	IBUF LUT6 LUT5 LUT5 FDCE	input port clock	None	None	
Path 7	0	2.151	0.888	1.264	0	∞	Safely Timed	4	IBUF LUT6 LUT6 LUT3 FDCE	input port clock	None	None	
Path 8	0	1.755	0.832	0.924	0	∞	Safely Timed	3	IBUF LUT6 LUT6 FDCE	input port clock	None	None	
Path 9	0	1.755	0.832	0.924	0	∞	Safely Timed	3	IBUF LUT6 LUT6 FDCE	input port clock	None	None	
Path 10	0	1.309	0.726	0.584	0	∞	Safely Timed	1	IBUF FDCE	input port clock	None	None	

شکل (۸): خروجی گزارش Design در Vivado

در جهت مرتب بودن این فایل گزارش های خروجی نرم افزار ISE با فرمت Html در پوشه ISE Reports در فایل های پروژه قرار دارد.
درصورت مبهم بودن نام گذاری های فایل پروژه فایل README.htm را مشاهده بفرمایید.

منابع (Sources)

Moore architecture and VHDL templates

FSM: Moore - Fundamentals

FSM: Medvedev – Fundamentals

Meaning of synthesis warning: [Constraints ۱۸-۵۲۱۰] No constraint will be written out.

Synthesize VHDL package or library