دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده برق

عنوان

گزارش پروژه امتیازی معماری کامپیوتر و ریزپردازنده

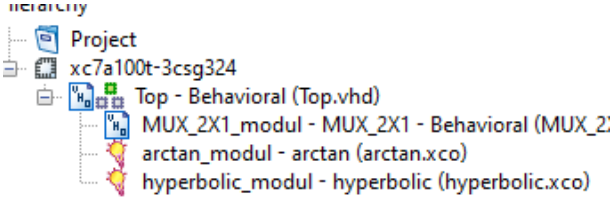
نگارش
پارسا محمّدی – ۹۹۲۳۱۲۱

استاد درس
دکتر شریعتمدار مرتضوی

خرداد ۱۴۰۲

فهرست صفحه

در این پروژه قرار است کار با الگوریتم CORDIC و پیاده‌سازی عملی آن آموخته شود. در این پروژه فایل های زیر قرار دارند.



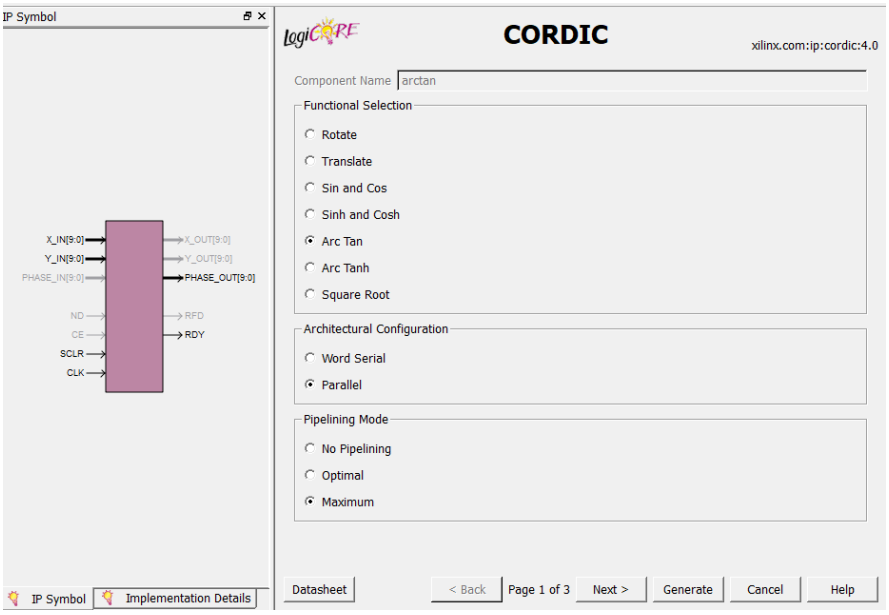
تصویر ۱ – فایل های موجود در پروژه

با توجه به اینکه در آموزش های ارسالی پیاده سازی توابع ریاضی با الگوریتم CORDIC با استفاده از IP Core آموزش داده شده است پس در این پروژه الگوریتم های CORDIC با طراحی دو ماژول arctan و sinh & cosh پیاده سازی شده است.

در این پروژه ابتدا پیاده سازی هرکدام از فایل ها گزارش داده می‌شود و بعد کد top بیان می‌شود و در نهایت خروجی کد تست بنچ نمایش داده می‌شود.

پیاده سازی arctan

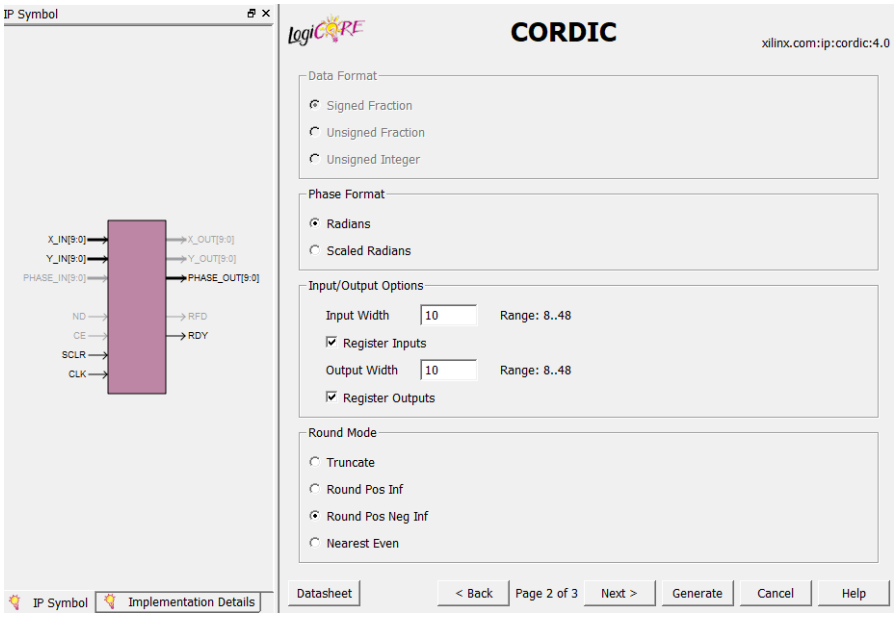
ابتدا در قسمت new source گزینه IP Core را انتخاب می‌کنیم. در قسمت های بعد CORDIC را انتخاب می‌کنم و پس اندکی زمان فایل xco ساخته می‌شود. با کلیک بر روی این فایل می توانیم IP Core خود را کانفیگ کنیم.



این قطعه می‌تواند تعدادی از توابع را با استفاده از الگوریتم CORDIC محاسبه کند. برای این پروژه گزینه arc tan را محاسبه می‌کنیم.

در قسمت بعد مشخص می‌شود که داده های ورودی به صورت موازی یا سریال وارد می‌شوند که در این پروژه داده ها را به صورت موازی وارد می‌کنیم.

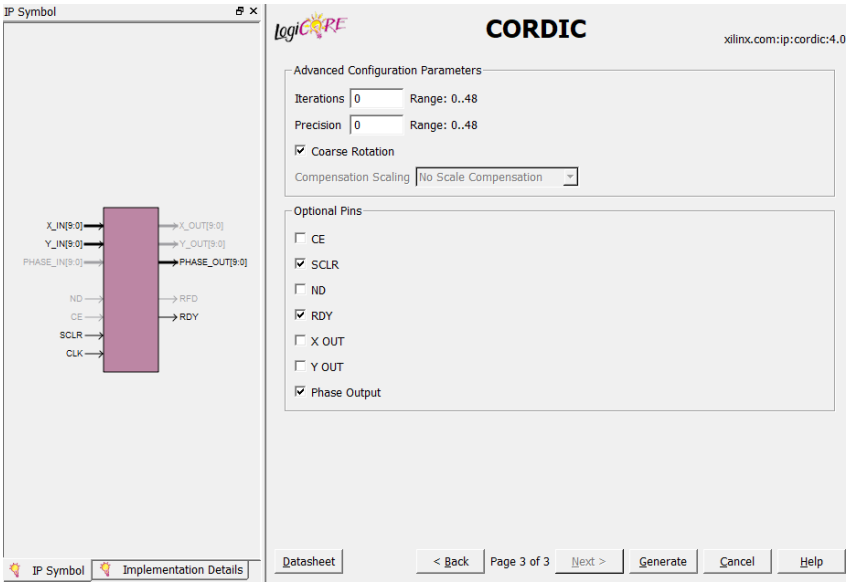
در قسمت pipelining mod با توجه به نوع پایپلاین در خواستی معماری درون قطعه را در تغییر می‌دهد. در این پروژه ما این بخش را maximum انتخاب می‌کنیم تا خروجی در سریع ترین زمان ممکن آماده شود.



در صفحه بعد میتوانیم نوع فاز ورودی را انتخاب کنیم که آن را بری روی رادیان عادی قرار می‌دهم.

در بخش بعدی تعداد بیت های وروی و خروجی تعیین می‌شود. که در این پروژه ۱۰ بیتی قرار داده می‌شود. بدیهی است که با توجه به این که الگوریتم CORDIC مقادیر را به صورت عددی محاسبه می‌کند افزارش تعداد بیت قدت را افزایش می‌دهد ولی از طرفی زمان محاسبه و گیت مصرفی نیز افزایش میابد.

در بخش آخر این صفحه نوع رند کردن عدد خروجی بیان شده است که آنرا روی منفی گذاشته ام تا به نزدیکترین عدد منفی به پایین رند کند.



در صفحه آخر کانفیگ CORDIC در بخش اول تعداد دفعات تکرار و همچنین دقت نهایی را میخواد. در داکیومت های این قطعه نوشته شده است زمانی که هر دو این ورودی ها بر روی صفر قرار گیرند و خود سیستم بهترین حالت ممکن را انتخاب می‌کند.

در قسما آخر پین های اختیاری قرار دارد که هرکدام از آنها مسئولیت انجام یک کار را دارند.

- CE : clock enable است که باعث رسیدن کلاک به مدار در صورت ۱ بودن می شود.
- SCLR : این پین دروافع ریست سنکرون مدار می‌باشد و با ۱ شدن آن مدار ریست میشود. این پایه در این پروژه استفاده شده است زیرا زمانی که به فعالیت این قطعه نیاز نداریم آن را در حالت ریست قرار دهیم.
- ND : new sample in input می‌باشد که باعث می‌شود قطعه متوجه شود که ورودی جدید آماده است.
- RDY : new output data is ready می باشد. چون محاسبات این قطعه سنگین است چندین کلاک زمان می‌برد تا جواب آماده شود. به این منظور زمانی که جواب آماده می‌شود این پایه ۱ می‌شود تا دیگر قطعات را آگاه کند که محاسبات تمام شده است و خروجی آماده شده است.
- X out & Y out همان ورودی ها به خروجی متقل میکنند
- Phase out : این پایه فاز محاسبه شده را خروجی می‌دهد.

در شکل سمت چپ شمای کلی از قطعه موجود است.

بعد از تعریف این قطعه می‌توان آن را مانند دیگر ماژول ها به صورت component استفاده کرد و با دستور port map به پورت های دسترسی داشت.

فرمت داده ها arctan

همان طور که قبلا بیان شد ماژول های CORDIC با فاز های رادیانی کار می‌کنند. فاز های ورودی و خروجی اعداد علامت دار اعشاری باینری می‌باشد. در داکیومت این قطعات اطلاعات مربوط به ورودی ها و خروجی ها آورده شده است.

Table 7: **ArcTan**

Signal	Range	Description
X_IN	-1 <= X_IN <=1	Input X Coordinate
Y_IN	-1 <= Y_IN <=1	Input Y Coordinate
PHASE_OUT	-Pi <= Phase Out <= Pi	Output Angle

Example 5: ArcTan

The input vector (Xin, Yin) is expressed as a pair of fixed-point 2's complement numbers with an integer width of 2 bits (1QN format). The output angle, Pout radians, is expressed as a fixed-point 2's complement number with an integer width of 3 bits (2QN format).

In this example, the input/output width is set to 10 bits.

Xin : "0010100000" => 00.10100000 => 0.625

Yin : "0010000000" => 00.10000000 => 0.500

Pout : "0001010110" => 000.1010110=> 0.672

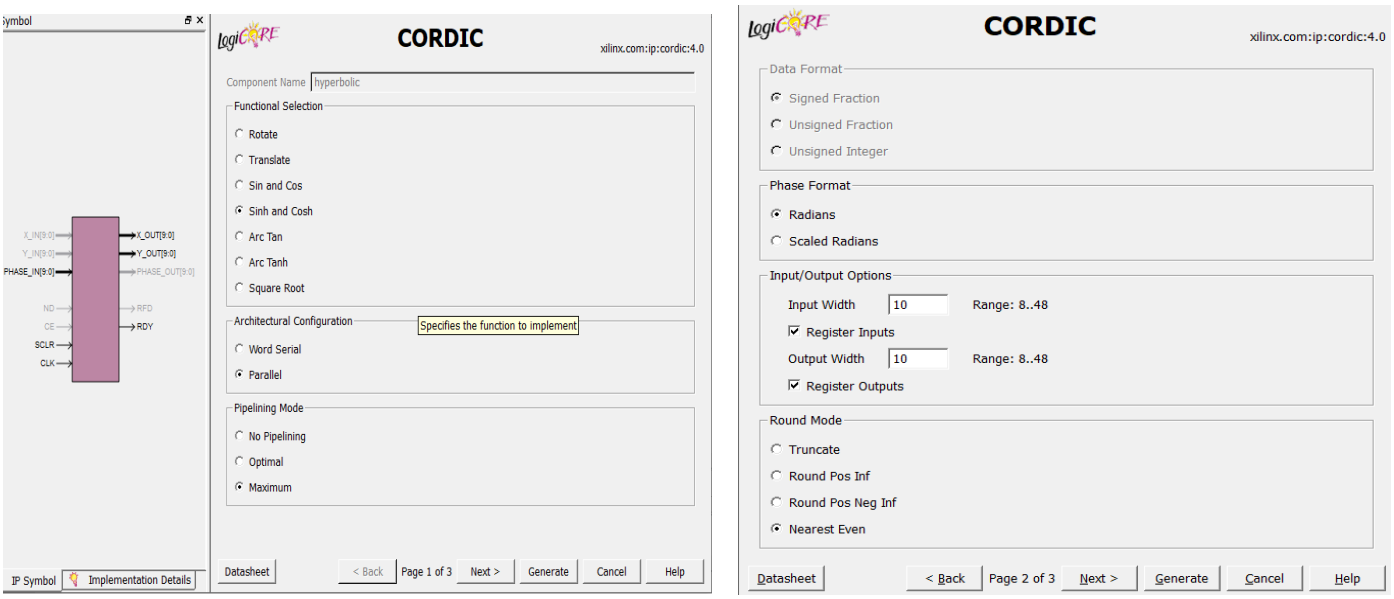
تصویر ۲ – بخشی از داکیومت مربوط به ماژول های CORDIC می باشد که نوع و بازه داده های ورودی و خروجی را مشخص کرده است.

- ماژول arctan دو ورودی x و y دارند که این دو ورودی مختصات داده را روی دایره مثلثاتی مشخص می‌کنند.
- از ۱۰ بیت این دو ورودی یکی علامت و یکی صحیح می‌باشد و بقیه اعشاری می‌باشند.
- از ۱۰ بیت خروجی ۱ بیت علامت که با ارزش ترین بیت می‌باشد و دو بیت بعد از آن که صحیح می باشند و بقیه بیت ها اعشاری اند.

پیاده‌سازی cosh & sinh

مراحل ساخت این ماژول هم مانند arctan می باشد فقط در بخش هایی تفاوت دارند که در ادامه بیان می‌شود.

علت طراحی این ماژول هم در ادامه بیان می‌شود.



در این ماژول فانکشن را روی sinh و cosh می گذاریم و رند کردن را روی نزدیک ترین میگذاریم تا نتایج دقیق تر شود.

فرمت داده ها

فرمت داده های این ماژول هم در داکیومنتیشن ماژول آمده است و به صورت زیر می‌باشد.

Table 6: **Sinh and Cosh**

Signal	Range	Description
PHASE_IN	-Pi/4 <= PHASE_IN <= Pi/4	Input Hyperbolic Angle
X_OUT	1 <= X_OUT < 2	Output Cosh
Y_OUT	-2 <= Y_OUT < 2	Output Sinh

Example 4: Sinh and Cosh

The input hyperbolic angle, Pin, is expressed as a fixed-point 2's complement number with an integer width of 3 bits (2QN format). The output vector, (Xout, Yout), is expressed as a pair of fixed-point 2's complement numbers with an integer width of 2 bits (1QN format).

In this example the input/output width is set to 10 bits.

Pin : "0001001110" => 000.1001110 => 0.781

Xout : "0100110001" => 01.00110001 => 1.191

تصویر ۳ – قسمتی از داکیومت CORDIC که فرمت داده های خروجی و ورودی را مشخص می‌کند.

پیاده‌سازی پروژه

در این پروژه از ما خواسته شده است که دو تابع arctan و e^x محاسبه شود. در ماژول CORDIC تمام توابع ریاضی وجود ندارند ولی می‌توان آنهایی که وجود ندارند را با استفاده آنهایی که کودیک آنها موجود است پیاده سازی کرد. برای این پروژه arctan وجود دارد ولی e^x وجود ندارد. در فایل آموزشی که ارسال شده است آموزش محاسبه اکسپوننشیال آورده شده است که به صورت زیر است.

- Although the CORDIC algorithms can only compute a limited numbe of functions directly, there are many more that can be achievec indirectly:

$$\tan z = \frac{\sin z}{\cos z} \qquad \tan^{-1} w = \tan^{-1} \frac{\sqrt{1-w^2}}{w}$$

$$\tanh z = \frac{\sinh z}{\cosh z} \qquad \sin^{-1} w = \tan^{-1} \frac{w}{\sqrt{1-w^2}}$$

$$\ln w = 2\tanh^{-1} \left| \frac{w-1}{w+1} \right| \qquad \cosh^{-1} w = \ln(w + \sqrt{1-w^2})$$

$$e^z = \sinh z + \cosh z \qquad \sinh^{-1} w = \ln(w + \sqrt{1+w^2})$$

$$w^t = e^{t \ln w} \qquad \sqrt{w} = \sqrt{(w+1/4)^2 - (w-1/4)^2}$$

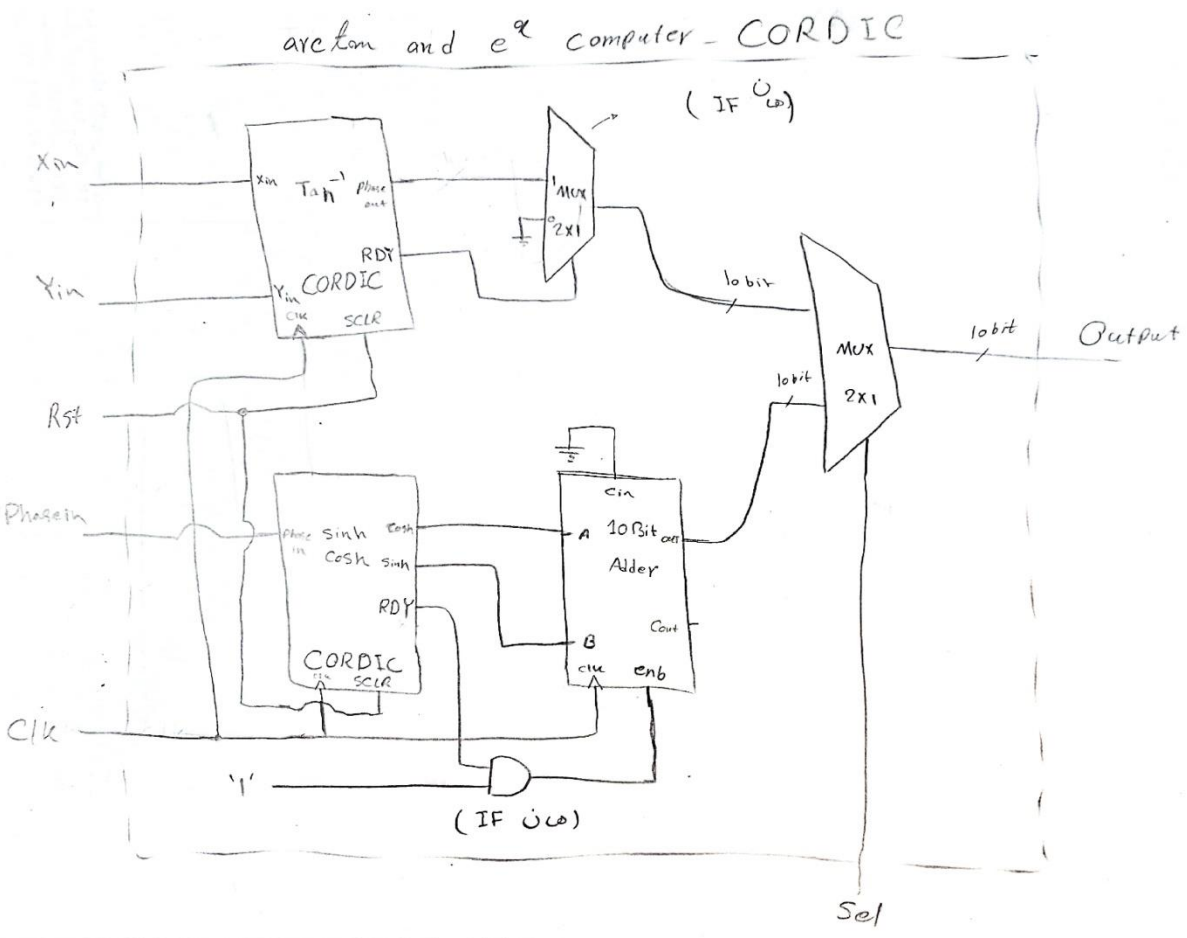
همان طور که مشخص است می‌توان اکسپوننشیال را به صورت زیر محاسبه کرد.

$$e^x = \sinh x + \cosh x$$

در مورد رابطه فوق دو نکته مهم وجود دارد.

- روابط بالا در ماژول کوردیک فقط برای x های رادینانی جواب می‌دهد.
- با توجه به این که ماژول کوردیک به صورت عدد محاسبات را انجام می‌دهد و ما تعداد بیت های کمی به صورت ورودی داده اینم این رابطه حول صفر و زوایه های کوچک دقت خوبی دارد و در زاویه های بزرگ دقت آن کاهش می‌یابد.

برای پیاده سازی پروژه معماری زیر در نظر گرفته شده است.



تصویر ۴ – معماری پروژه

توضیح کد های top module

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity Top is
7      Port (
8          input_phase : in  STD_LOGIC_VECTOR (9 downto 0);
9          x_in : in STD_LOGIC_VECTOR (9 downto 0);
10         y_in : in STD_LOGIC_VECTOR (9 downto 0);
11         clk : in  STD_LOGIC;
12         rst : in  STD_LOGIC;
13         sel : in  STD_LOGIC;
14         result : out  STD_LOGIC_VECTOR (9 downto 0)
15     );
16 end Top;
```

ابتدا کتاب خانه های مورد نیاز را اضافه کرده و سپس ورودی ها و خروجی ها را مشخص می‌کنیم.

ورودی input_phase برای فاز ماژول های هیپربولیک می‌باشد و دو ورودی x و y هم برای تانژانت معکوس می‌باشد که همان طور که قبلا گفته شد استفاده می‌شوند. ورودی rst برای ریست کردن ماژول ها و ورودی sel هم برای انتخاب ماژول ها می‌باشد. خروجی هم داده ها هم با resualt مشخص شده است.

```
19     COMPONENT MUX_2X1
20         PORT (
21             input1 : IN std_logic_vector(9 downto 0);
22             input2 : IN std_logic_vector(9 downto 0);
23             sel : IN std_logic;
24             clk : IN std_logic;
25             output : OUT std_logic_vector(9 downto 0)
26         );
27     END COMPONENT;
28
29     COMPONENT arctan
30         PORT (
31             x_in : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
32             y_in : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
33             phase_out : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
34             rdy : OUT STD_LOGIC;
35             clk : IN STD_LOGIC;
36             sclr : IN STD_LOGIC
37         );
38     END COMPONENT;
39
40     COMPONENT hyperbolic
41         PORT (
42             phase_in : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
43             x_out : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
44             y_out : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
45             rdy : OUT STD_LOGIC;
46             clk : IN STD_LOGIC;
47             sclr : IN STD_LOGIC
48         );
49     END COMPONENT;
50
```

بعد در ادامه ماژول های ساخته شده نیز به صورت کامپوننت اضافه شده‌اند.

برای بدست آوردن اطلاعات کامپوننت هایی که IP Core اند ما می‌توانیم از بخش view instantiation template اطلاعات آن را بدست آوریم.

```
56  begin
57
58      MUX_2X1_modul: MUX_2X1 PORT MAP(
59          input1 => arctan_out,
60          input2 => exp_out,
61          sel => sel,
62          clk => clk,
63          output => mux_out
64      );
65
66      arctan_modul : arctan
67          PORT MAP (
68              x_in => x_in,
69              y_in => y_in,
70              phase_out => arctan_out,
71              rdy => arctanRDY,
72              clk => clk,
73              sclr => rst
74          );
75
76      hyperbolic_modul : hyperbolic
77          PORT MAP (
78              phase_in => input_phase,
79              x_out => cosh_out,
80              y_out => sinh_out,
81              rdy => hyperbolicRDY,
82              clk => clk,
83              sclr => rst
84          );
85
```

در ادامه بعد از begin کد آغاز می‌شود و ابتدا portmap ها را با توجه به معماری که در بخش قبل نمایش داده شد انجام می‌دهیم.

توجه شود که هر دو ماژول یه یک ریست متصل اند تا باهم ریست شوند.

در ادامه کد با توجه به اینکه قطعات ما کلاک دارند و محاسبات مربوط به ماژول های arctan و sinh چندین کلاک زمان می‌برند پس ما نیاز داریم اقداماتمان را در داخل پرسس انجام دهیم.

```
    process(clk, hyperbolicRDY, arctanRDY)
    begin
        if rising_edge(clk) then
            if (sel = '0') then -- arctanh
                if arctanRDY = '1' then
                    result <= mux_out;
                else
                   exp_out <= "0000000000";
                end if;

            elsif (sel = '1') then -- E^x
                result <= mux_out;
                if hyperbolicRDY = '1' then
                    exp_out <= cosh_out + sinh_out;
                else
                  exp_out <= "0000000000";
                end if;
            end if;
        end if;
    end process;

end Behavioral;
```

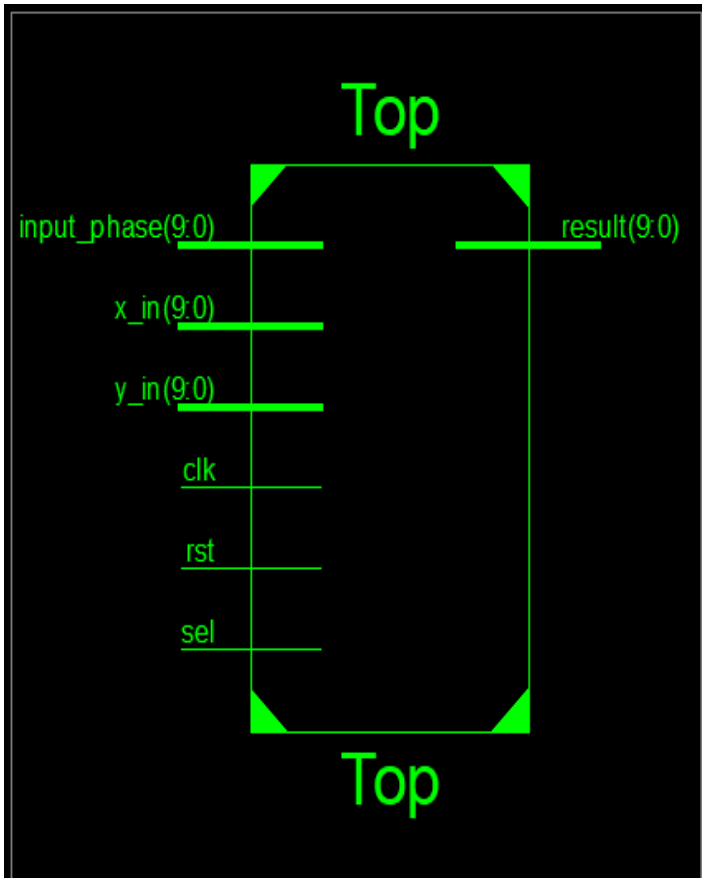
زمانی که محاسبات cosh و arctan تمام شود پایه RDY یک می‌شود و پس است که میتوان خروجی آن را به خروجی داد.

به همین علت باید از سازوکار process استفاده کنیم تا مدام بررسی کنیم که چه زمان RDY برابر با ۱ می‌شود آن زمان آن را به خروجی دهیم.

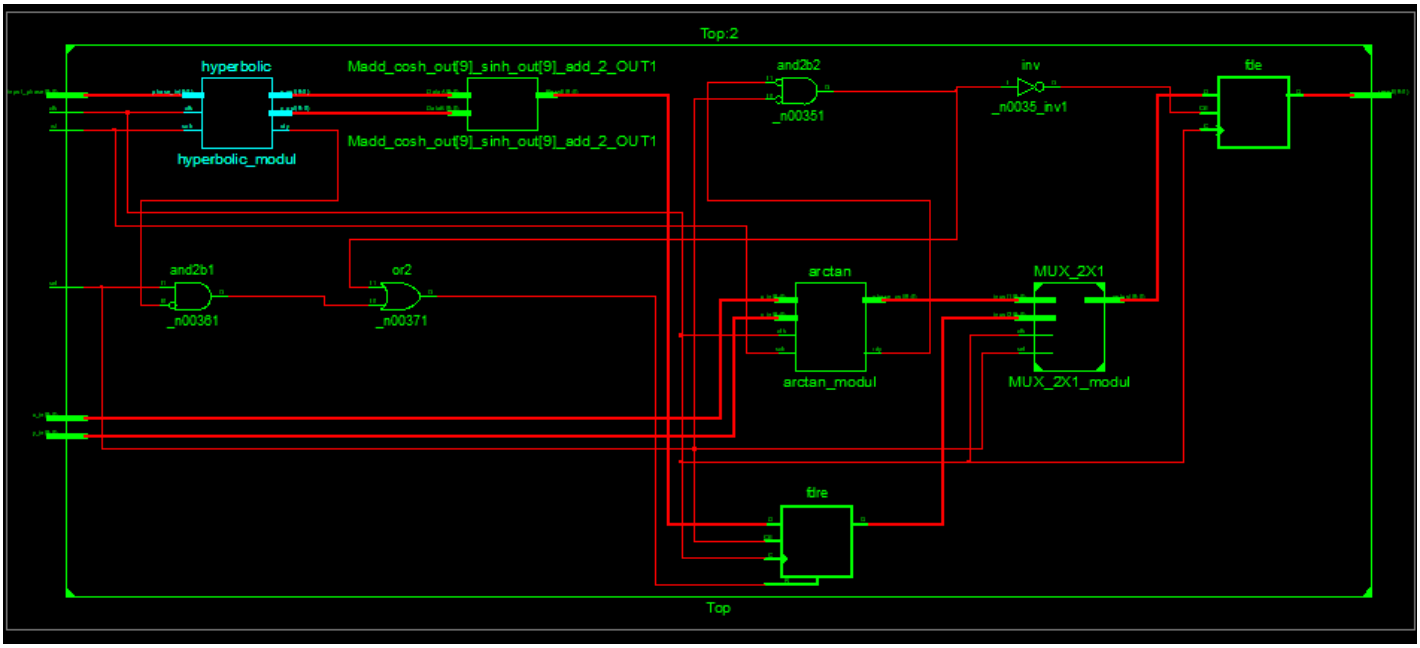
به این منظور پروسس به کلاک و تغییر RDY ماژول ها وابسته است و زمانی که هر کدام از آنها تغییر کند برنامه زیر پروسس اجرا می‌شود.

۱. در if اول شرط لبه بالا رونده گذاشته شده است که کد فقط در زمان لبه های بالا رونده اجرا شود.
۲. در if دو بررسی شده است که که کدام پایه مالتیپلکسر انتخاب شده است و کاربر خروجی کدام ماژول را می‌خواهد تا بعد ready بودن آن ماژول را بررسی کند.
۳. در if سوم هم بررسی شده است که پایه RDY فعال است یا خیر برای هرکدام از ماژول ها. اگر فعال بود خروجی انتقال داده می‌شود در غیر این صورت صفر به خروجی داده می‌شود. در برای ماژول sinh چون ما در واقع اکسپوننشال را می‌خواهیم پس sinh و cosh را جمع می‌کنیم و در خروجی می‌ریزیم.

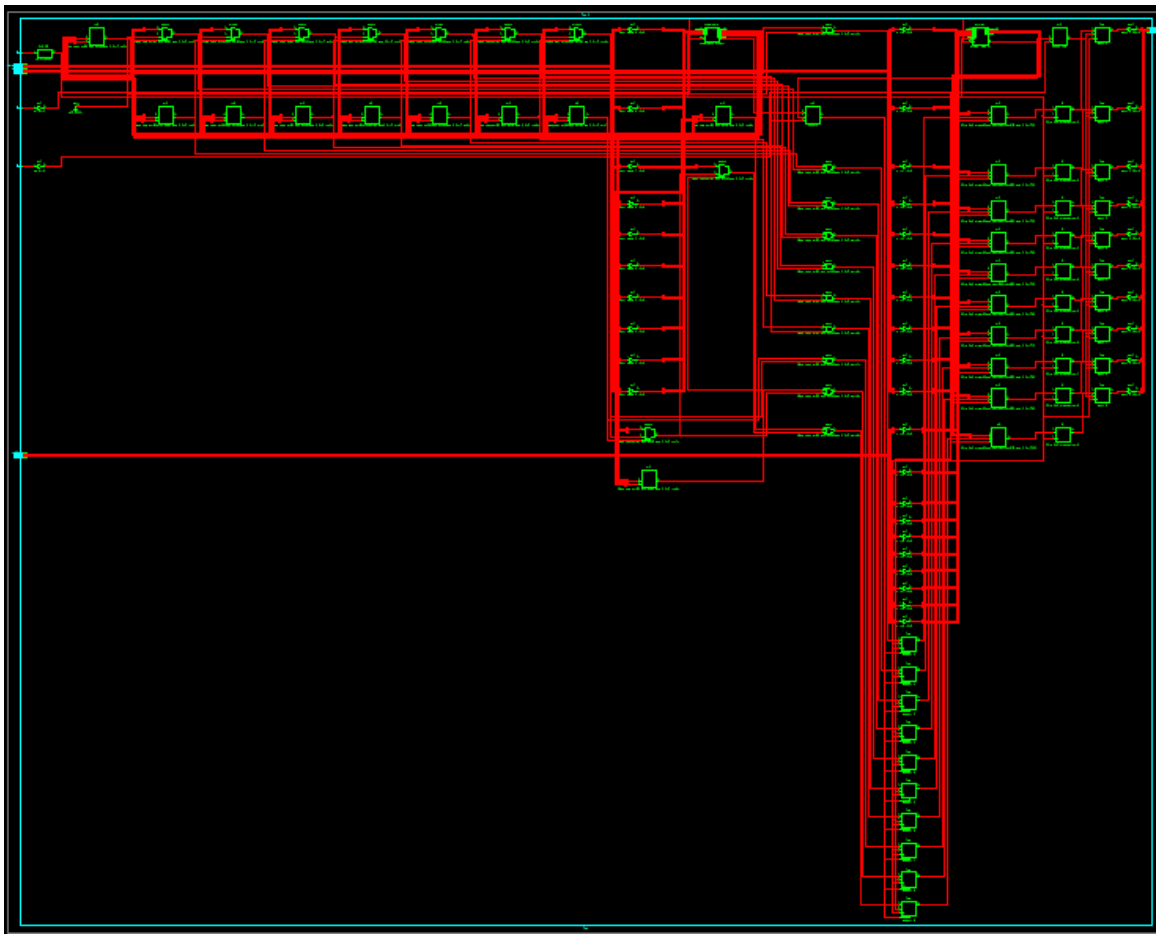
تصویر RTL مدار و سطوح مصرفی



تصویر ۵ – RTL کلی



تصویر ۶ – جزئیات RTL مدار



تصویر ۷ – تصویر تکنولوژی مدار

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	931	126800	0%
Number of Slice LUTs	1052	63400	1%
Number of fully used LUT-FF pairs	29	1954	1%
Number of bonded IOBs	43	210	20%
Number of BUFG/BUFGCTRLs	1	32	3%

تصویر ۸ – سطوح مصرفی FPGA

طراحی تست بنچ

برای اطمینان حاصل کردن از درستی عملکرد کد برای تمام بخش های کد تست بنچ طراحی شده است. و خروجی تمام آنها درست می‌باشد. برای اطمنیان از درستی عملکرد کلی کد یک تست بنچ هم برای ماژول top نوشته شده است که هم arctan و هم exp را تست می‌کند.

نکته در تست ها متوجه شده ام که پردازش هر کدام از توابه حدود ۱۵۰ ns با کلاک های ۱۰ ns ای طول می‌کشد پس زمان های wait را اندکی طولانی تر گذاشتم.

بخش های اولیه تست بنچ مانند پروژهای قبلی است و در ادامه بدنه تست بنچ توضیح داده شده است.

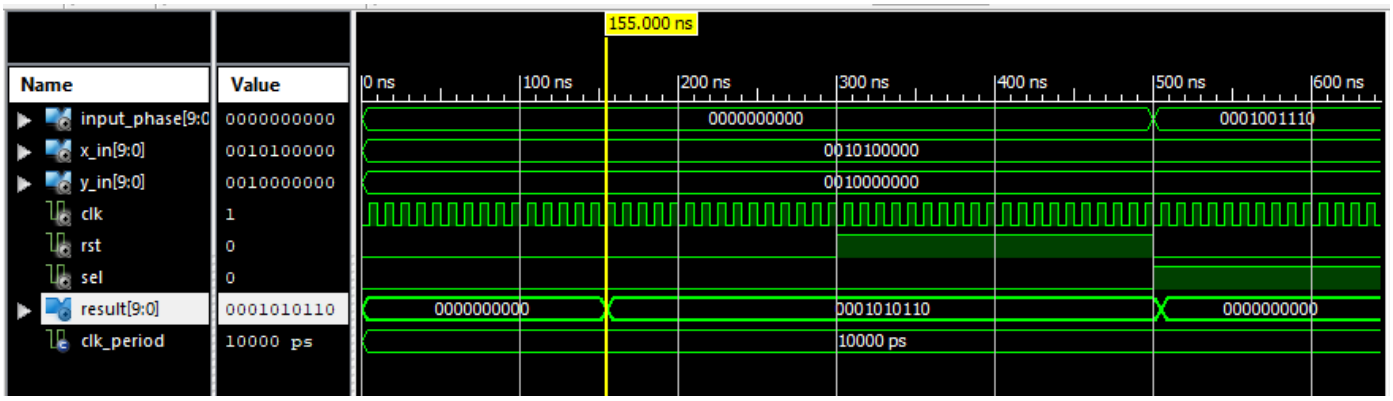
ابتدا برای تست arctan ریست غیرفعال شده است و sel بر روی 0 قرار گرفته است تا خروجی ماژول arctan در صورت آماده بودن به خروجی رود.

با توجه به فرمت دیتا ها داریم:

$$\begin{cases} 0010100000 \rightarrow 00.10100000 = 0.625 \\ 0010000000 \rightarrow 00.10000000 = 0.5 \end{cases}$$

تصویر تست بنچ و خروجی به صورت زیر است

```
-- Stimulus process
stim_proc: process
begin
-- First test arctan
   x_in <= "0010100000";
   y_in <= "0010000000";
   rst <= '0';
   sel <= '0';
wait for 300 ns;
   rst <= '1';
wait for 200 ns;
```



همین طور مشخص است با توجه به اینکه ورودی ها از ابتدا داده شده اند اما جواب بعد از ۱۵۵ ns آماده شده است و خروجی تغییر کرده است. در ۱۱۵ ns خروجی RDY برابر ۱ می‌شود و خروجی ماژول arctan به خروجی منتقل می‌شود.

خروجی به صورت زیر می‌باشد.

$$0001010110 \rightarrow 000.1010110 = 0.672 \text{ رادیان}$$

$$0.672 \times \frac{180}{\pi} = 38.5 \text{ درجه}$$

بررسی

$$\tan(38.5) = 0.79543 \; then \; we \; have \; \frac{sin}{cos} = \frac{0.5}{0.625} = 0.8 \rightarrow 0.01 \text{ دقت}$$

در ادامه قطعه برای مدتی ریست می‌شود و مقدار جدید را برای تست بعدی می‌دهیم.

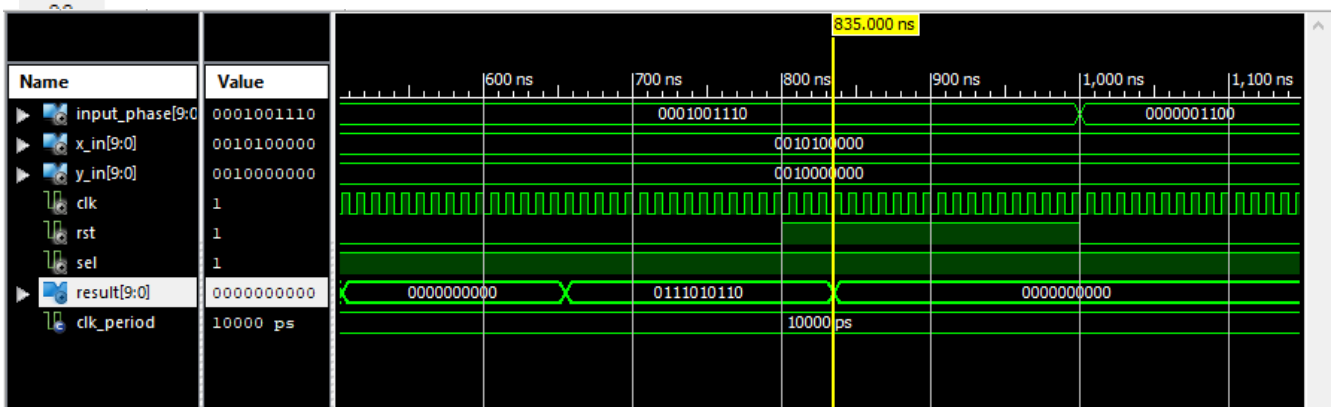
تست بعدی برای Exp می‌باشد و ورودی ها به صورت زیر است.

وروی اول:

```
74
75    -- Second test exp
76       input_phase <= "0001001110";
77       rst <= '0';
78       sel <= '1';
79    wait for 300 ns;
80       rst <= '1';
81    wait for 200 ns;
82       input_phase <= "0000001100";
83       rst <= '0';
84       sel <= '1';
85
86    wait;
87    end process;
```

$$0001001110 \rightarrow 000.1001110 = 0.781\ rad$$
$$= 45\ dgree$$

ریست را غیرفعال و sel را ۱ میکنیم تا خروجی exp به خروجی رود.



جواب به صورت زیر است

$$result = e^x = \cosh x + \sinh x \rightarrow 011010110$$

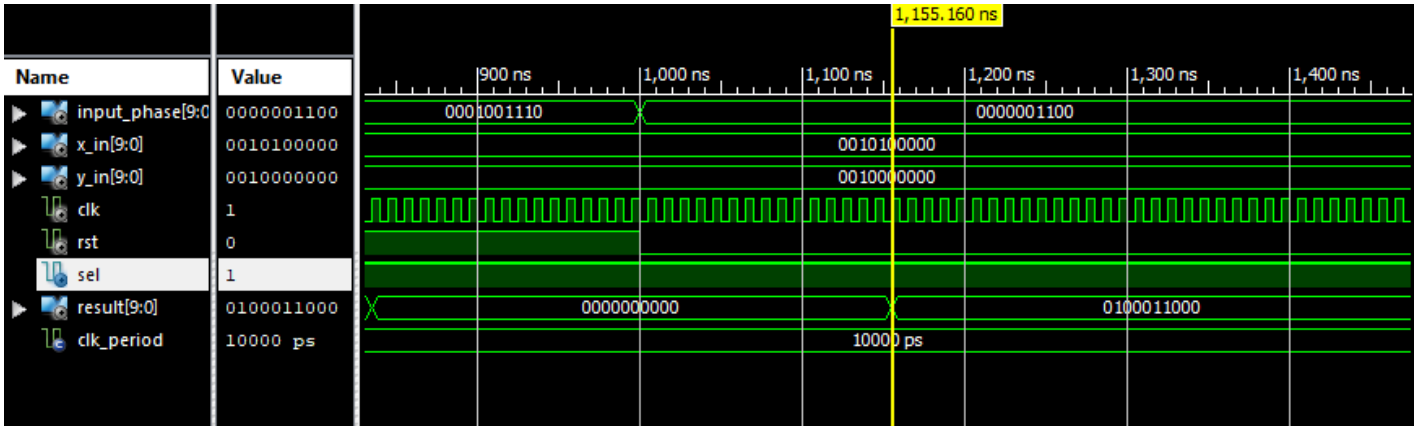
$$011010110 \rightarrow 01.1010110 = 1.839$$

بررسی

$$e^{0.781} \cong 2 \ \ so\ we\ have\ 0.1\ accurecy$$

ورودی دوم

$$0000001100 \rightarrow 000.0001100 \rightarrow 0.9375 \cong 0.1\ rad = 5.72958$$



$$result = e^x = coshx + sinhx = 0100011000 \rightarrow 01.00011000$$

$$= 1.09375$$

$$e^{0.1} = 1.1051\ so\ we\ 0.01\ accuracy$$

پس جواب های حاصل با وجود مقدار کم تعداد بیت های ورودی مقدار بسیار نزدیک به مقادیر حقیقی دارند و ماژول ها به درستی کار می‌کنند.