# به نام خدا

تمرین ششم

# محمدمهدی آقاجانی

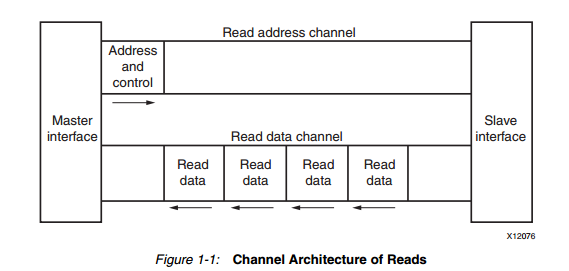
9331056

استاد : دکتر صاحب الزمانی

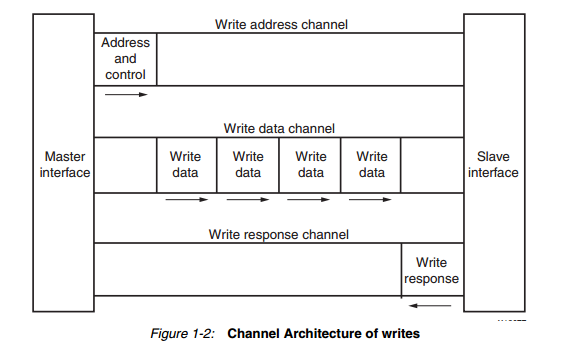
سوال 3

شرکت زایلینکس برای تراشه های جدید خود از گذرگاه AXI ( Advanced eXtensible Interface ) استفاده میکند. این گرگاه مبتین بر گدرگاه امبا از شرکت ARM می باشدو برای اتصال وسایل جانبی سریع مانند حافظه DDR و اترنت به کار میرود.

معماری این گذرگاه برای read به صورت زیر است :



برای write کردن نیز معماری به صورت زیر است :



از آنجایی که این گذرگاه ممکن است منابع زیادی مصرف کند نسخه سبک آن به نام AXI-Lite نیز وجود دارد که برای وسایل جانبی کندتر مانند UART یا اتصال به صفحه کلید مناسب است.برای ارتباط نقطه به نقطه یک جهته نیز ، اتصال AXI – stream معرفی شده است.مد AXI full نیز همان AXI کامل است.

سوال 4

برای این ماژول کنترلر دو تا ماژول Round و Registers را نوشتیم که به صورت زیر است :

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Registers is Port (

input : in std\_logic\_vector( 15 downto 0);

clk , load , clr : in std\_logic;

output : out std\_logic\_vector( 15 downto 0 )

);

end Registers;

architecture Behavioral of Registers is

begin

process( clk )

begin

if( load = '1' )then

output <= input;

end if;

if( clk'event and clk = '1' )then

if( clr = '1' ) then

end if;

end if;

end process;

end Behavioral;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.std\_logic\_unsigned.all;

entity Round is Port (

left,right,key : in std\_logic\_vector( 7 downto 0 );

leftOut , rightOut : out std\_logic\_vector( 7 downto 0 )

);

end Round;

architecture Behavioral of Round is

signal l1,r1 : std\_logic\_vector( 7 downto 0 );

signal r2,l2 : bit\_vector( 7 downto 0 );

signal r3 , l3 : std\_logic\_vector( 7 downto 0 );

begin

l1 <= left xor right;

l2 <= to\_bitvector(l1) sla 1;

l3 <= to\_stdlogicvector(l2) + key;

leftOut <= l3( 7 downto 0 );

r1 <= right xor l3;

r2 <= to\_bitvector(r1) sla 1;

r3 <= key + to\_stdlogicvector(r2) ;

rightOut <= r3( 7 downto 0 );

end Behavioral;

برای ماژول کنترلر نیز به صورت زیر عمل میکنیم :

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.std\_logic\_unsigned.all;

entity controller is Port (

input : in std\_logic\_vector( 15 downto 0 );

key : in std\_logic\_vector( 7 downto 0 );

clk , rst : in std\_logic;

output : out std\_logic\_vector( 15 downto 0 ) );

end controller;

architecture Behavioral of controller is

component Round is Port (

left,right,key : in std\_logic\_vector( 7 downto 0 );

leftOut , rightOut : out std\_logic\_vector( 7 downto 0 )

);

end component;

component Registers is Port (

input : in std\_logic\_vector( 15 downto 0);

clk , load , clr : in std\_logic;

output : out std\_logic\_vector( 15 downto 0 )

);

end component;

type state\_type is( S0 , S1 , S2 , S3 , S4 , S5 , S6 , S7 );

signal state : state\_type := S0;

signal tempOR , tempOL : std\_logic\_vector( 7 downto 0 );

signal right , left : std\_logic\_vector( 7 downto 0 );

signal load , clr : std\_logic := '0' ;

signal registeredOutput : std\_logic\_vector( 15 downto 0 );

signal muxOut : std\_logic\_vector( 15 downto 0 );

signal initialTempL : std\_logic\_vector( 7 downto 0 );

signal initialTempR : std\_logic\_vector( 7 downto 0 );

signal selectLine : std\_logic;

begin

left <= input ( 15 downto 8 );

right <= input ( 7 downto 0 );

module1: Round port map( muxOut( 15 downto 8 ) , muxOut( 7 downto 0 ) , key , tempOL , tempOR );

reg: Registers port map( tempOL & tempOR , clk , load , clr , registeredOutput );

initialTempL <= key + left;

initialTempR <= key + right;

with selectLine select muxOut <=

initialTempL&initialTempR when '0',

registeredOutput when '1',

initialTempL&initialTempR when others;

process(clk)

begin

if( clk'event and clk = '1' )then

load <= '0';

if( rst = '1' )then

state <= S0;

end if;

case state is

when S0 =>

selectLine <= '0';

state <= S1;

when S1 =>

selectLine <= '1';

state <= S2;

when S2 =>

selectLine <= '1';

state <= S3;

when S3 =>

selectLine <= '1';

state <= S4;

when S4 =>

selectLine <= '1';

state <= S5;

when S5 =>

selectLine <= '1';

state <= S6;

when S6 =>

selectLine <= '1';

state <= S7;

when S7 =>

load <= '0';

output <= tempOL & tempOR;

end case;

end if;

if( clk'event and clk = '0' )then

if( state /= S7 )then

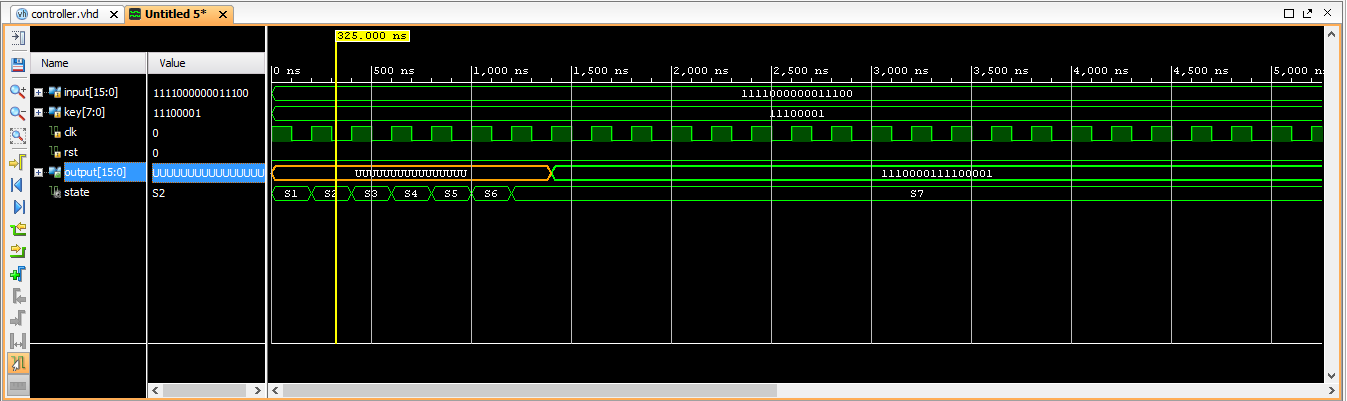
load <= '1';

end if;

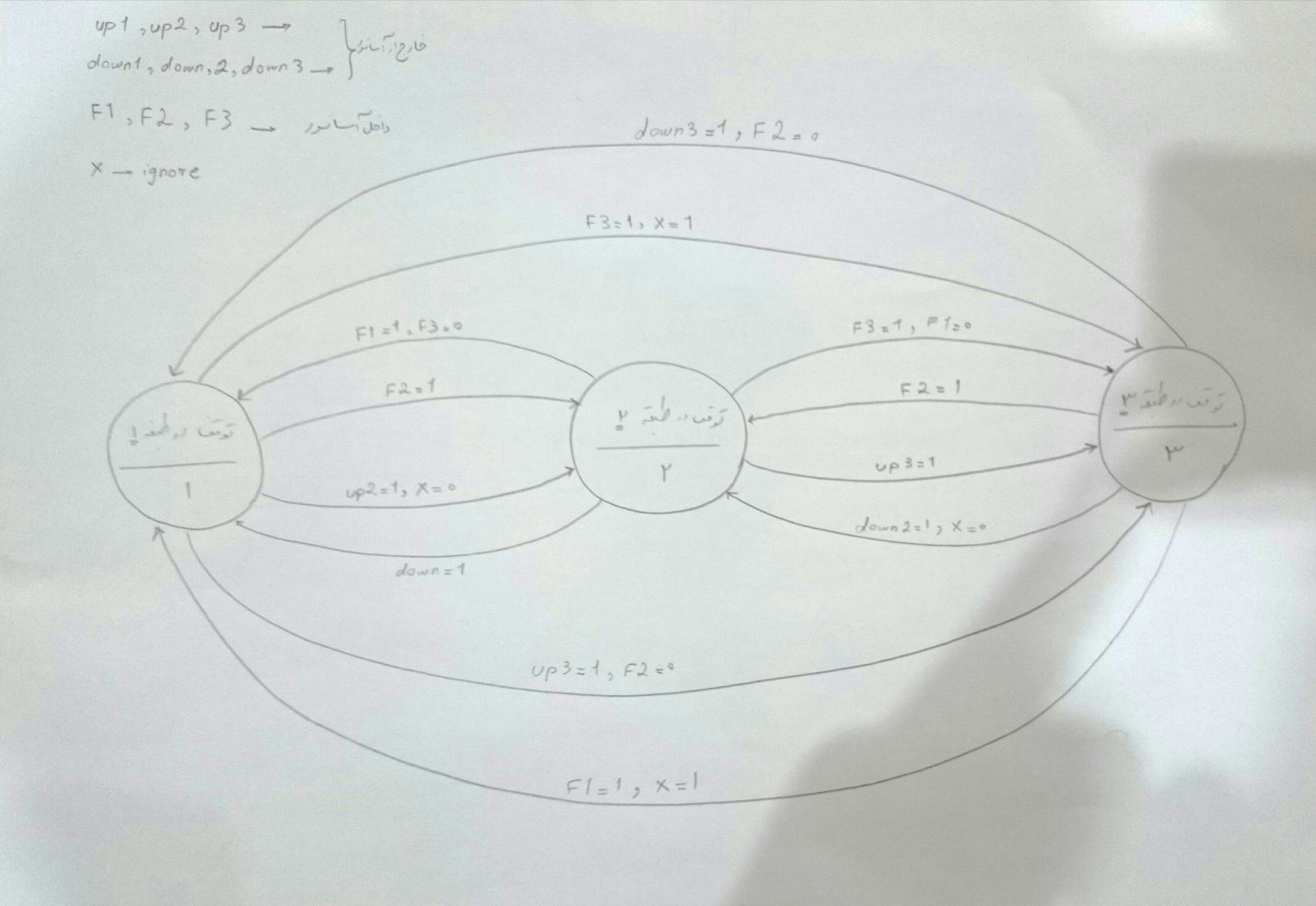
end if;

end process;

end Behavioral;

شکل موج آن نیز به صورت زیر است :

سوال 5 :



نکته : در transition از "توقف در طبقه 3" به "توقف در طبقه 1" به جای down3=1 باید down1=1 باشد.

کد کنترلر به صورت زیر است :

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity controller is Port (

clk , rst : in std\_logic;

up1,up2,up3,down1,down2,down3,f1,f2,f3,x : in std\_logic;

floorNum : out integer range 0 to 3

);

end controller;

architecture Behavioral of controller is

type state\_type is ( stay\_in\_first , stay\_in\_second , stay\_in\_third );

signal state : state\_type := stay\_in\_first ;

begin

process(clk)

begin

if( clk'event and clk = '1' )then

if( rst = '1' )then

state <= stay\_in\_first;

end if;

case state is

when stay\_in\_first =>

floorNum <= 1;

if( f2 = '1' )then

state <= stay\_in\_second;

elsif( up2 = '1' and x = '0' )then

state <= stay\_in\_second;

elsif( f3 = '1' and x='1' )then

state <= stay\_in\_third;

elsif( up3 = '1' and f2 = '0' )then

state <= stay\_in\_third;

end if;

when stay\_in\_second =>

floorNum <= 2;

if( f3 = '1' and f1 = '0' )then

state <= stay\_in\_third;

elsif( up3 = '1' )then

state <= stay\_in\_third;

elsif( f1 = '1' and f3 = '0' )then

state <= stay\_in\_first;

elsif( down1 = '1' )then

state <= stay\_in\_first;

end if;

when stay\_in\_third =>

floorNum <= 3;

if( f2 = '1' )then

state <= stay\_in\_second;

elsif( down2 = '1' and x = '0' )then

state <= stay\_in\_second;

elsif( down1 = '1' and f2 = '0' ) then

state <= stay\_in\_first;

elsif( f1 = '1' and x = '1' ) then

state <= stay\_in\_first;

end if;

end case;

end if;

end process;

end Behavioral;

در آخر هم نتیجه سنتز را مشاهده میکنید:

