

به نام خدا

تمرین ششم

محمد مهدی آقاجانی

۹۳۳۱۰۵۶

استاد : دکتر صاحب الزمانی

سوال ۳

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

معماری این گذرگاه برای read به صورت زیر است :

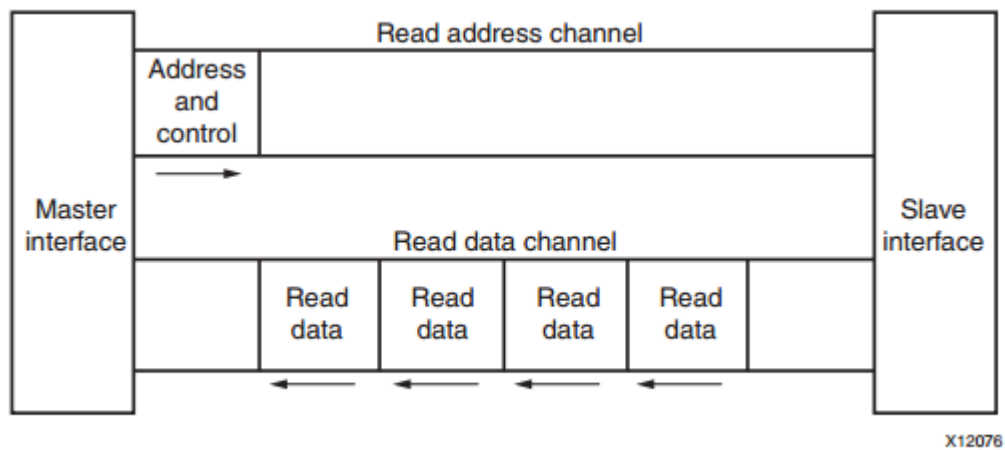


Figure 1-1: Channel Architecture of Reads

برای write کردن نیز معماری به صورت زیر است :

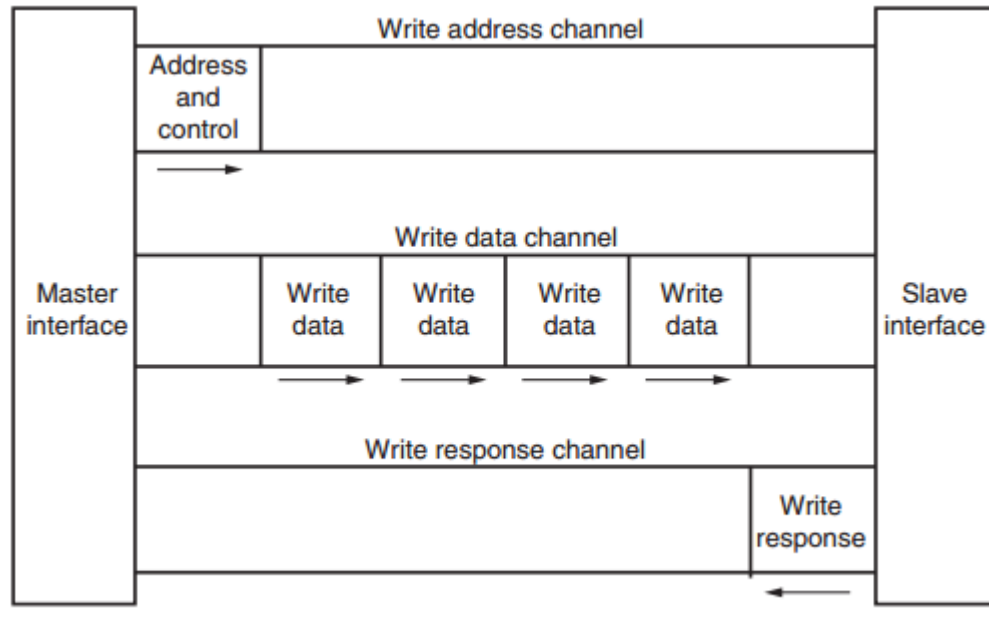


Figure 1-2: Channel Architecture of writes

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 جهت نیز، اتصال AXI – stream معرفی شده است. مد AXI full نیز همان AXI کامل است.

سوال ۴

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;

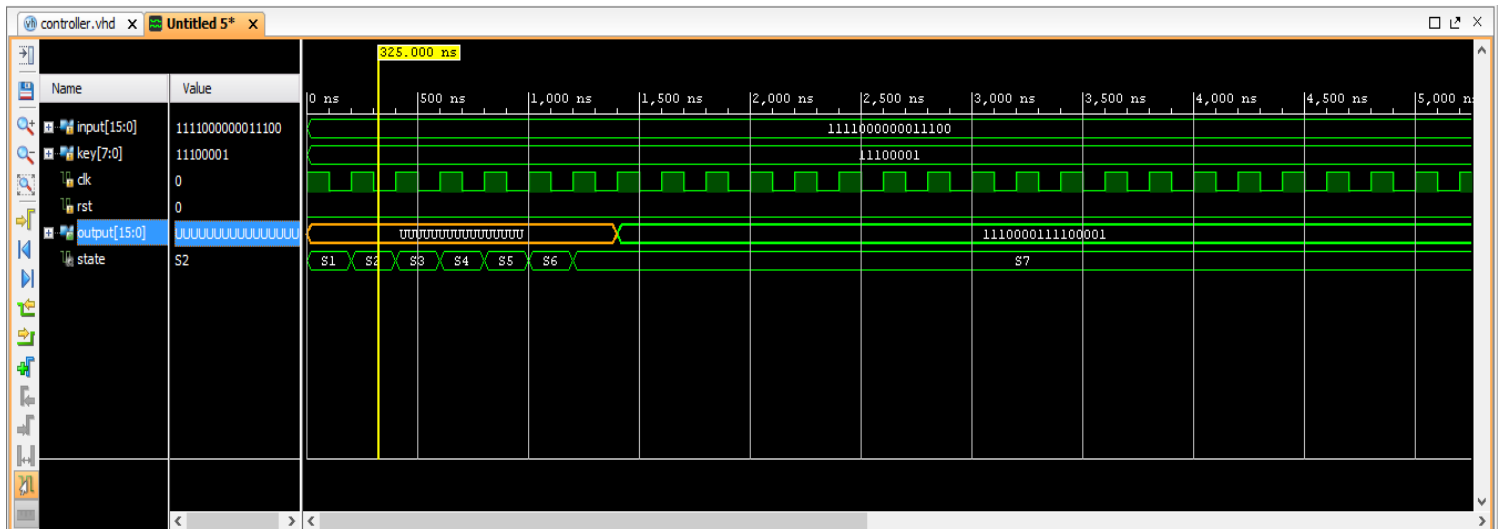
```

```

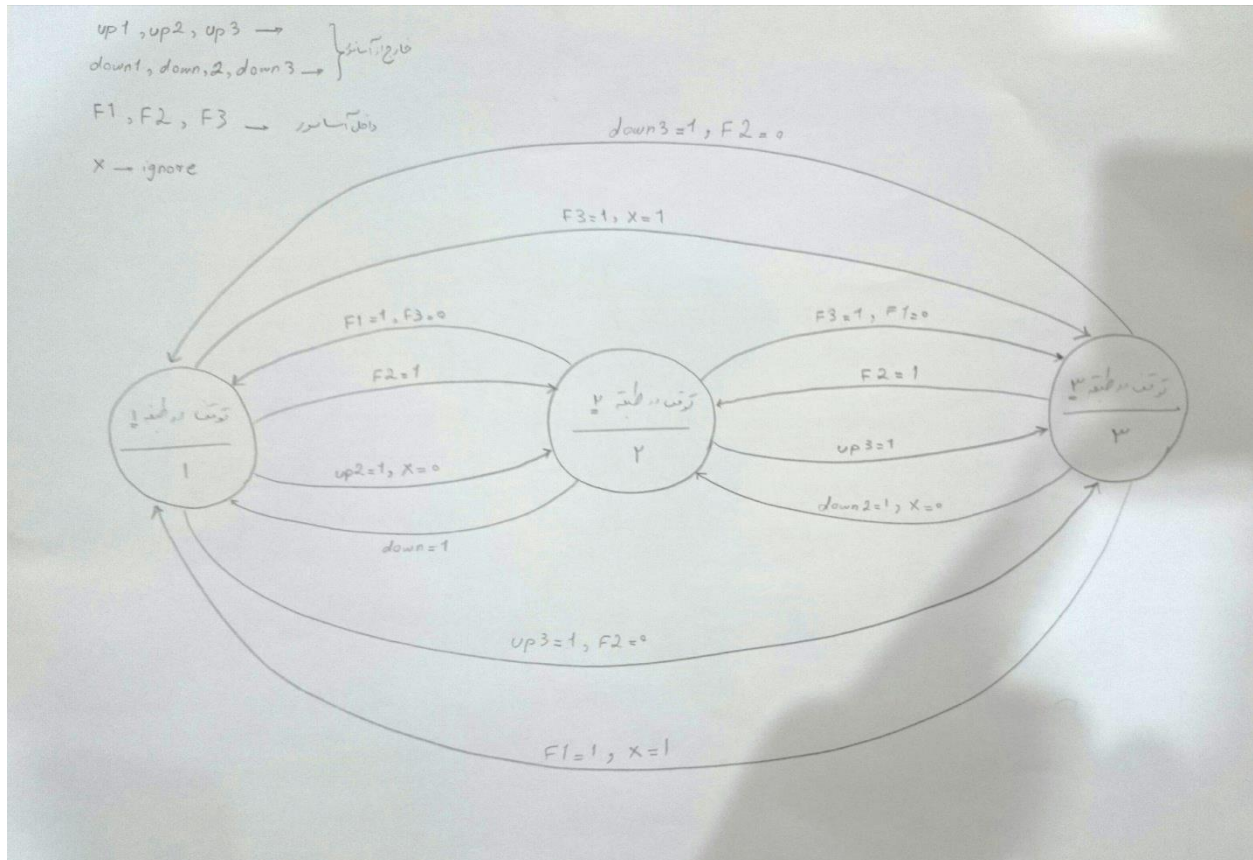
when S4 =>
    selectLine <= '1';
    state <= S5;
when S5 =>
    selectLine <= '1';
    state <= S6;
when S6 =>
    selectLine <= '1';
    state <= S7;
when S7 =>
    load <= '0';
    output <= temp0L & temp0R;
end case;
end if;
if( clk'event and clk = '0' )then
    if( state /= S7 )then
        load <= '1';
    end if;
end if;
end process;
end Behavioral;

```

شکل موج آن نیز به صورت زیر است :



سوال ۵ :



نکته : در transition از "توقف در طبقه ۳" به "توقف در طبقه ۱" 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;
            end if;
        end case;
    end if;
end process;
```

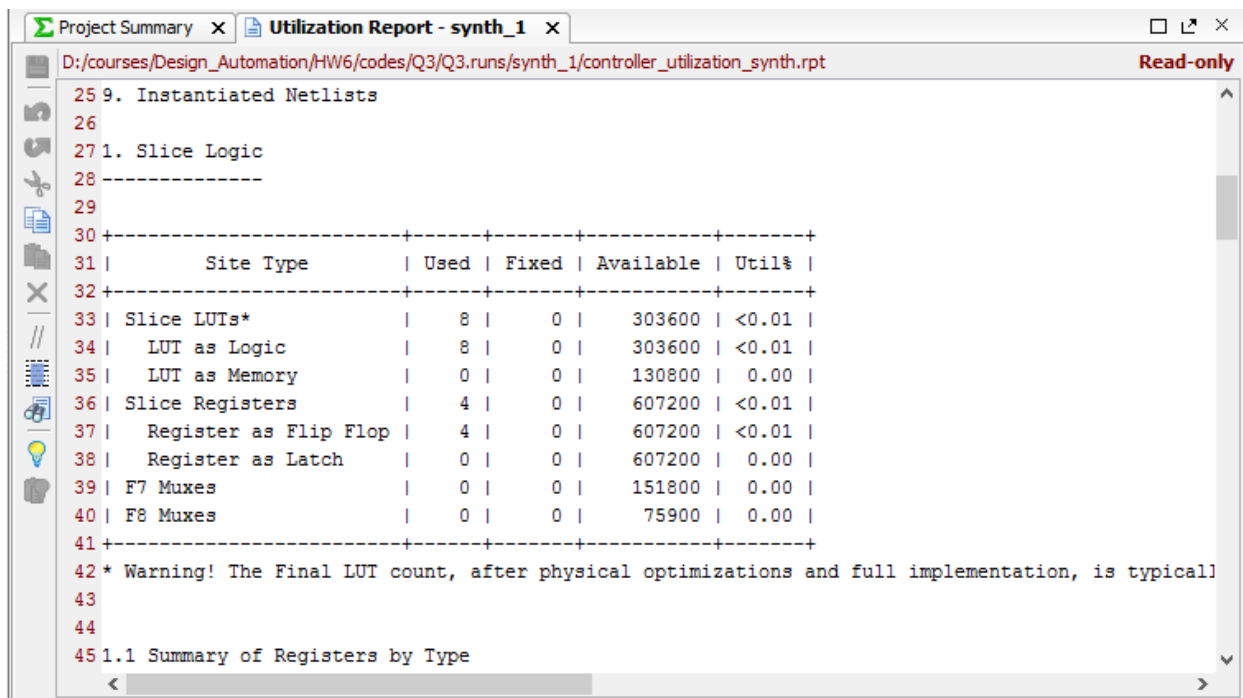
```

        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:



Project Summary x Utilization Report - synth_1 x

D:/courses/Design_Automation/HW6/codes/Q3/Q3.runs/synth_1/controller_utilization_synth.rpt Read-only

25 9. Instantiated Netlists

26

27 1. Slice Logic

28 -----

29

	Site Type	Used	Fixed	Available	Util%
33	Slice LUTs*	8	0	303600	<0.01
34	LUT as Logic	8	0	303600	<0.01
35	LUT as Memory	0	0	130800	0.00
36	Slice Registers	4	0	607200	<0.01
37	Register as Flip Flop	4	0	607200	<0.01
38	Register as Latch	0	0	607200	0.00
39	F7 Muxes	0	0	151800	0.00
40	F8 Muxes	0	0	75900	0.00

41 +-----+

42 * Warning! The Final LUT count, after physical optimizations and full implementation, is typically

43

44

45 1.1 Summary of Registers by Type