Assignment #2

EEL4713

Peter Borenstein

January 28th, 2012

Introduction

This assignment focused on propagation delays of two basic devices, multiplexers and D-flipflops. The registers used in this course are 32 bits wide. Various sized muxes were made leading up to a 32 bit 2:1 mux. A component was built that zero or sign extends a 16 bit number to a 32 bit number. It is important to pay attention to propagation to set the right clock speed. If a clock were to trigger in the middle of propagation, the device would be processing garbage data.

Design and testing

The multiplexers were built first. The 2:1 multiplexer has to inputs, in0 and in1, and the select input, sel, chooses which input is seen at the output. The inputs used were 1, 5 and 32 bits wide. There were only two inputs , so the select signal was only one bit.

	1 bit mux	5 bit mux	32 bit mux
Worst delay	11.7ns	13.6ns	18.5ns

Q 3.2: Rs, Rt, and Rd are represented by 5 bit numbers because there are 32 registers. 5 bits can make 32 combinations, or one for ever register.
Q 3.5: The propagation increased as the size of the mux increased. This indicates that the chip being used cant simple arrange muxes in parallel.

The 32 bit register was created next. The register has an input, D, and an output Q. Q acquires and holds the value of D if the write bit is true during a rising clock edge. The clock edge could be falling if the designer chose it to be so. There is an asynchronous clear that sets the values of Q to zero. The delay for Q to acquire the value D is 13 ns.
Q 3.5: The clock period must be greater than the largest propagation delay. For the register to work properly, the clock cannot be faster than 77MHz.

The extender was built last. Immediate instructions only have 16 bits to work with due to limited instruction length. A 32 bit number is needed to perform arithmetic functions with 32 bit numbers. This device compensates for that by filling in the upper 16 bits with 0's or 1's for negative numbers. A zero extender was built that takes in a 16 bit number and outputs a 32 bit number with its upper 16 bits set to 0. Then a sign extender was built to fill in the upper 16 bits of the output with 1's. These devices are combined and a one bit select signal is created to choose between the two.

Textbook questions

2.11.4 a) R-type

b) I-type

2.11.5 a) add

b) sw

2.11.6 a)00000000001000100001100000010000

b)10101110000010100000000000000010

Appendix

Mux code:

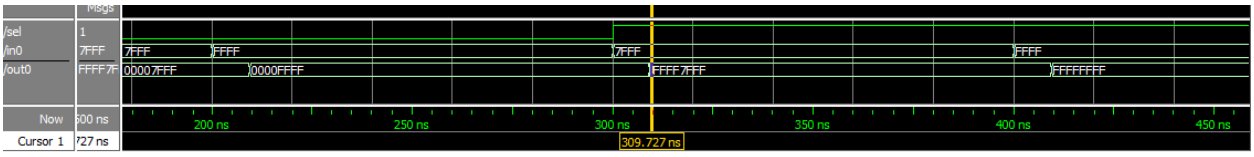
```
library ieee;
use ieee.std_logic_1164.all;
entity mux1 is
port (   sel : in std_logic;
        in1, in0 : in std_logic_vector(31 downto 0); --31 can be changed for the 5 and 1 bit muxes
         output : out std_logic_vector(31 downto 0)
         );
end mux1;
architecture behavior of mux1 is
begin
         with sel select                          --with select select lol
         output<=       in1 when '1',             -- output= sel and in1 or not sel and in0
                        in0 when others;
end behavior;
```

Register code:

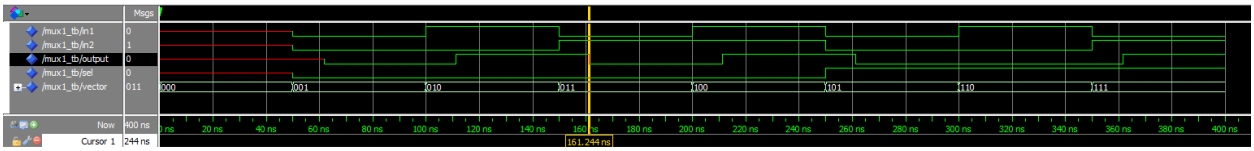
```
library ieee;
use ieee.std_logic_1164.all;
entity reg32 is
port (   D              : in std_logic_vector(31 downto 0);
         clk, clr, wr   : in std_logic;
         Q              : out std_logic_vector(31 downto 0)
         );
end reg32;
architecture behavior of reg32 is
begin
         process(clk, clr)
         begin
                  if clr='1' then                 --clr sets Q to 0 asynchronously
                           Q<=(others=>'0');
                  elsif(clk'event and clk='1' and wr='1')then  -- conditions for Q to obtain a value
                           Q<=D;
                  end if;
         end process;
```

Zeroext, Signext, andExtender code:

```
library ieee;
use ieee.std_logic_1164.all;
entity extender is
        port (  in0     : in std_logic_vector(15 downto 0);          --input is 16 bits
                sel     :       in std_logic;
                out0    : out std_logic_vector(31 downto 0)          --output is 32 bits
                );
end extender;

architecture behavior of extender is
begin
        with sel select
        out0<= x"ffff"&in0 when '1',
                x"0000"&in0 when others;
end behavior;

architecture behavior of zeroext is
begin
        out0<= x"0000"&in0;          --concatenate with 0's
end behavior;

architecture behavior of signext is
begin
        out0<= x"ffff"&in0           --concatenate with 1's
end behavior;
```

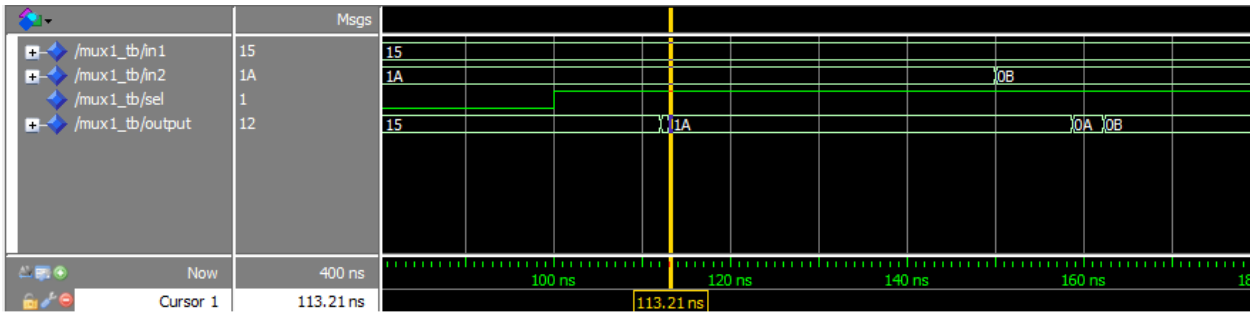


The extender fills the upper 16 bits with 1s or 0s determined by sel. On the left half of the waveform, sel is 0 so the uppper bits are filled with 0. On the right, sel=1 and the uppper bits of out0 are all 1. the delay is 9.7ns

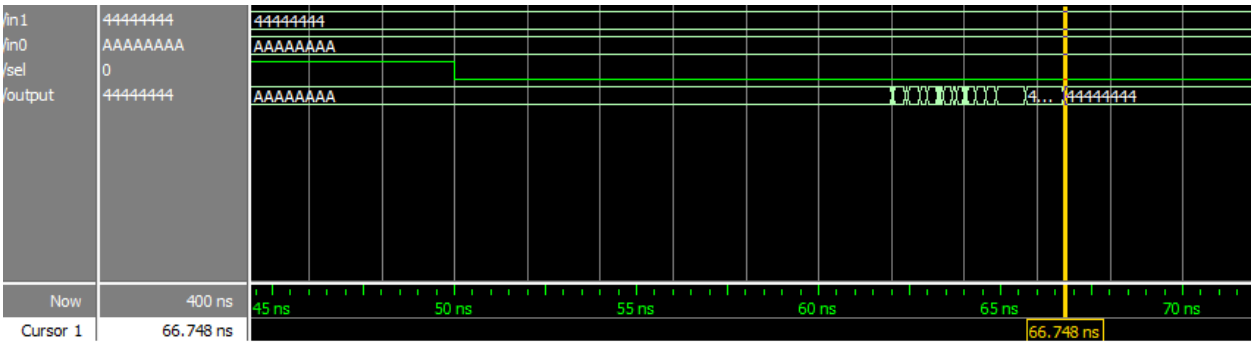


vector is not important. I used it to simulate all the input values.

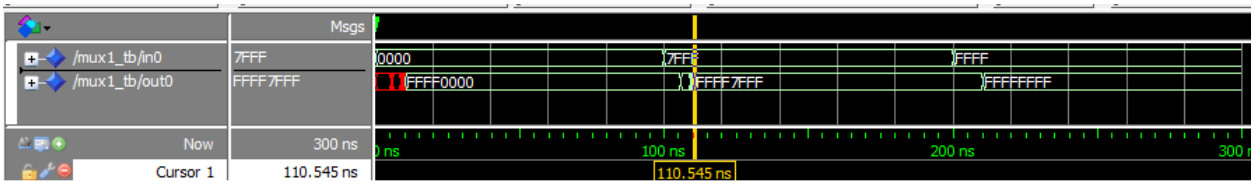
At the cursor we see that the ouput changed at 161.2 ns. this is because in1 chagned at 100 ns. Our delay is 11.2 ns.



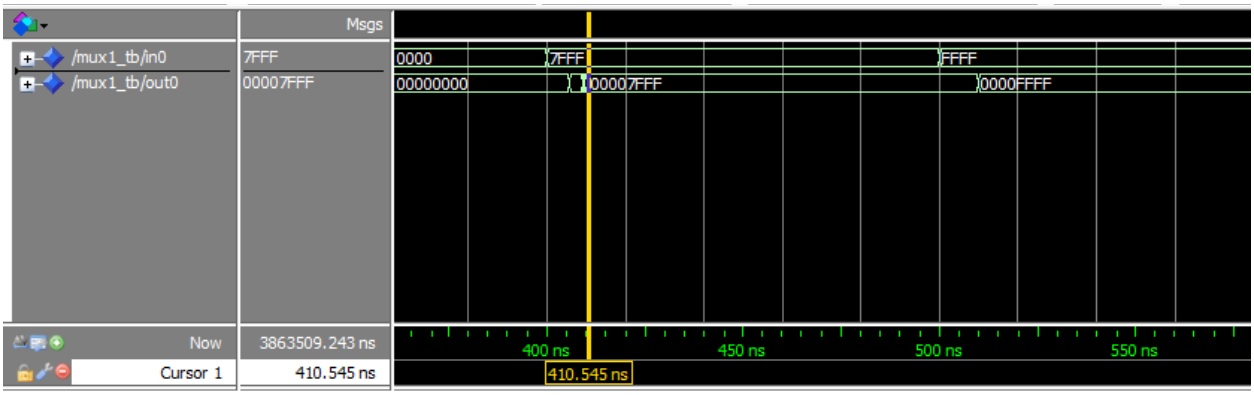
Here we have a 5 bit 2:1 mux. The select is changed. The output changes from in1 to in2. The output changes 13.2ns after the select bit switched due to propigation delay.



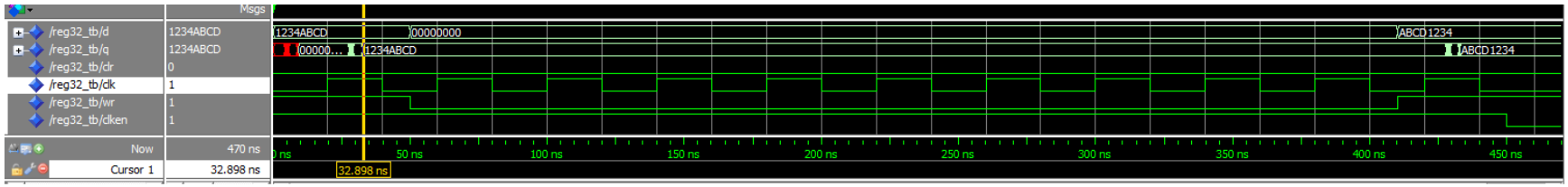
The is the waveform of a 32 bit 2:1 mux. A select change is shown. The output changes to in0 16.7ns after the select bit changes.



The sign extender sets the upper 16 bits of the output to 1. the cursor points out a 10.5 ns delay. the redix is hexadecimal



Here we have a zero extention. The inputs are 7FFF and FFFF. We see the upper 16 bits of the output are set to zero after a 10.5ns delay.



This is a 32 bit register. It is set to 0x00000000 initialy, then to 0x1234abcd, and then to 0xABCD1234. The clk period is 40 ns. The propagation delay is about 13ns as seen at the cursor (the clock triggered the change at 20 ns).