University of Dublin,
Trinity College

Computer Architecture I (CS2022)

Project 1a - Datapath Design

Author:
Edmond O'Flynn 12304742

Lecturer:
Dr. Michael Manzke

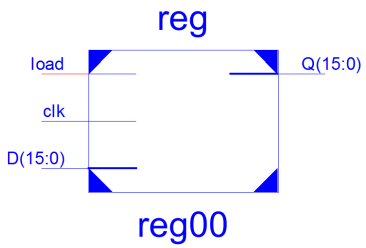
February 16, 2016

Contents

1 VHDL Component Source Code

1.1 Register Component

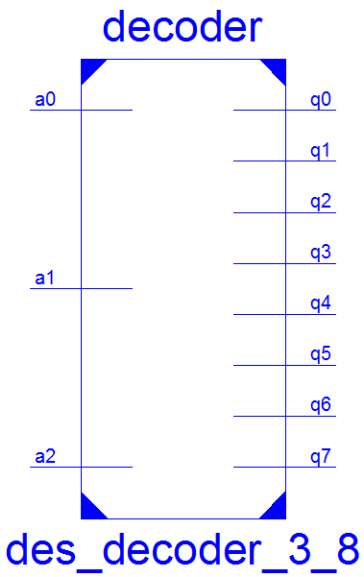
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg is
   Port(
        D : in STD_LOGIC_VECTOR(15 downto 0);
        load, clk : in STD_LOGIC;
        Q : out STD_LOGIC_VECTOR(15 downto 0)
      );
end reg;

architecture Behavioral of reg is
begin process(clk)
   begin
      if(rising_edge(clk)) then
         if(load = '1') then
            Q <= D after 5ns;
         end if;
      end if;
   end process;
end Behavioral;
```



1.2 Decoder

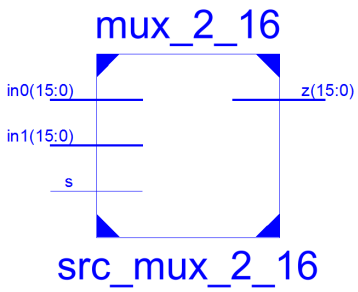
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity decoder is
   Port(
        a0, a1, a2 : in STD_LOGIC;
        q0, q1, q2, q3, q4, q5, q6, q7 : out STD_LOGIC
      );
end decoder;

architecture Behavioral of decoder is
begin
   q0 <= ((not a0) and (not a1) and (not a2)) after 5ns; --000
   q1 <= ((not a0) and (not a1) and (a2)) after 5ns;  --001
   q2 <= ((not a0) and (a1) and (not a2)) after 5ns;  --010
   q3 <= ((not a0) and (a1) and (a2)) after 5ns;    --011
   q4 <= ((a0) and (not a1) and (not a2)) after 5ns;  --100
   q5 <= ((a0) and (not a1) and (a2)) after 5ns;    --101
```

```
   q6 <= ((a0) and (a1) and (not a2)) after 5ns;    --110
   q7 <= ((a0) and (a1) and (a2)) after 5ns;        --111
end Behavioral;
```



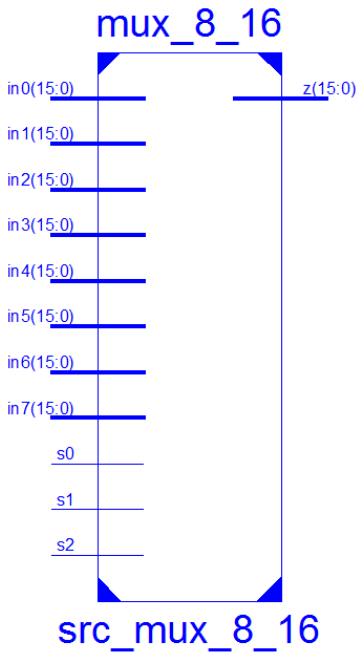
1.3 2 to 16 Bit Multiplexer

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux_2_16 is
   Port(
         in0, in1 : in STD_LOGIC_VECTOR(15 downto 0);
         s : in STD_LOGIC;
         z : out STD_LOGIC_VECTOR(15 downto 0)
      );
end mux_2_16;

architecture Behavioral of mux_2_16 is
begin
   z <=  in0 after 5ns when s = '0' else
         in1 after 5ns when s = '1' else
         x"0000" after 5ns;
end Behavioral;
```



1.4 8 to 16 Bit Multiplexer

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux_8_16 is
   Port(
        s0, s1, s2 : in STD_LOGIC;
        in0, in1, in2, in3, in4, in5, in6, in7 : in STD_LOGIC_VECTOR(15 downto 0);
        z : out STD_LOGIC_VECTOR(15 downto 0)
      );
end mux_8_16;

architecture Behavioral of mux_8_16 is
begin
   z <=  in0 after 5ns when s0 = '0' and s1 = '0' and s2 = '0' else --000
        in1 after 5ns when s0 = '0' and s1 = '0' and s2 = '1' else --001
        in2 after 5ns when s0 = '0' and s1 = '1' and s2 = '0' else --010
        in3 after 5ns when s0 = '0' and s1 = '1' and s2 = '1' else --011
        in4 after 5ns when s0 = '1' and s1 = '0' and s2 = '0' else --100
        in5 after 5ns when s0 = '1' and s1 = '0' and s2 = '1' else --101
        in6 after 5ns when s0 = '1' and s1 = '1' and s2 = '0' else --110
        in7 after 5ns when s0 = '1' and s1 = '1' and s2 = '1' else --111
        x"0000" after 5ns;
end Behavioral;
```



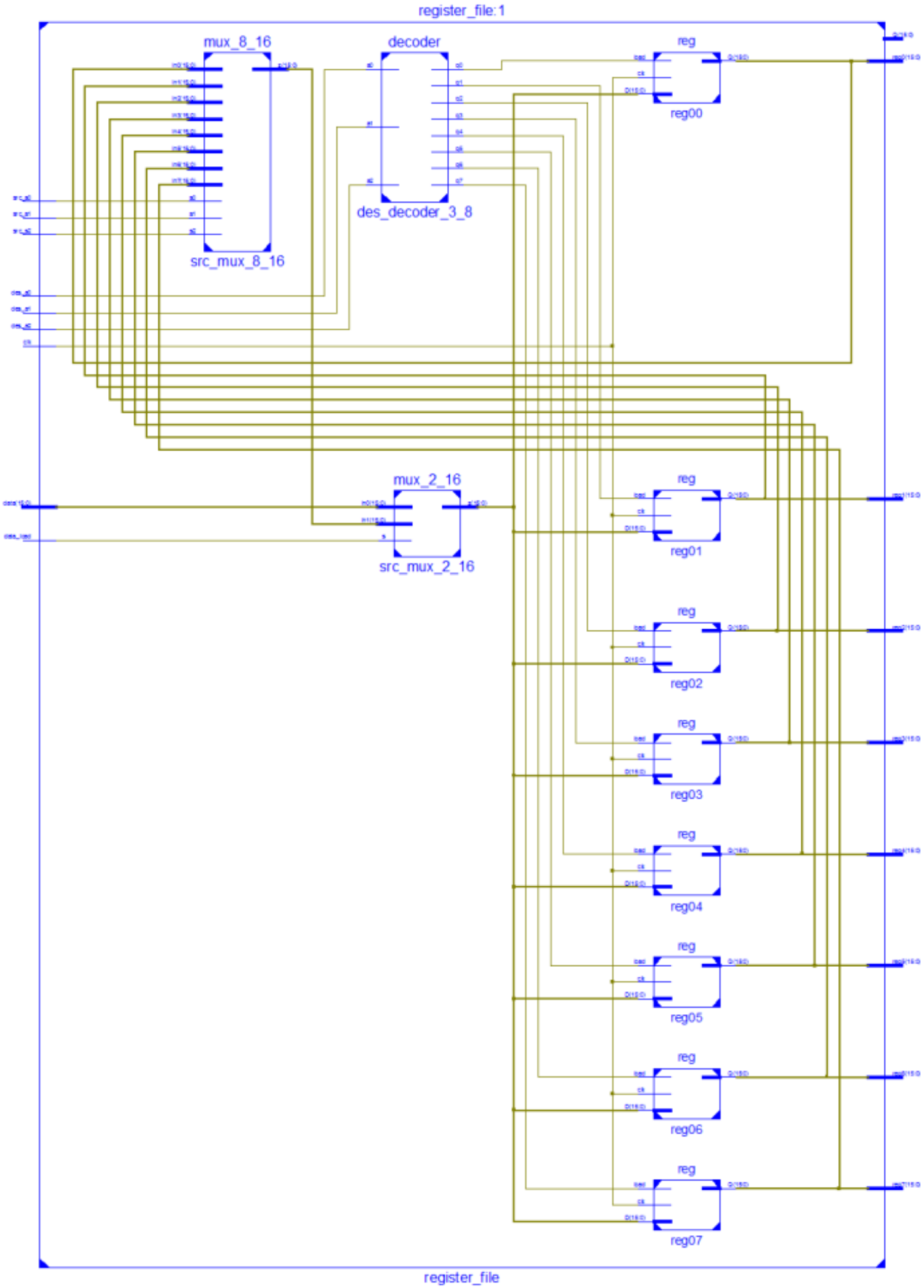
1.5 Top-Level VHDL File

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity register_file is
   Port(
        src_s0, src_s1, src_s2  : in STD_LOGIC;
        des_a0, des_a1, des_a2  : in STD_LOGIC;
        data_load, clk          : in STD_LOGIC;
        data  : in STD_LOGIC_VECTOR(15 downto 0);
```

```
        Q     : out STD_LOGIC_VECTOR(15 downto 0);
        reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7 : out STD_LOGIC_VECTOR(15 downto 0)
        );
end register_file;

architecture Behavioral of register_file is
--register component
   Component reg
      Port(
           load, clk  : in  STD_LOGIC;
           D          : in  STD_LOGIC_VECTOR(15 downto 0);
           Q          : out STD_LOGIC_VECTOR(15 downto 0)
           );
   End Component;
--decoder 3 to 8 bit
   Component decoder
      Port(
           a0, a1, a2 : in STD_LOGIC;
           Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7 : out STD_LOGIC
           );
   End Component;
--mux 2 to 16 bit
   Component mux_2_16
      Port(
           in0, in1 : in STD_LOGIC_VECTOR(15 downto 0);
           s : in STD_LOGIC;
           z : out STD_LOGIC_VECTOR(15 downto 0)
           );
   End Component;
--mux 8 to 16 bit
   Component mux_8_16
      Port(
           in0, in1, in2, in3, in4, in5, in6, in7 : in STD_LOGIC_VECTOR(15 downto 0);
           s0, s1, s2 : STD_LOGIC;
           z : out STD_LOGIC_VECTOR(15 downto 0)
           );
   End Component;

--inter-component signals
   --load in for each register's load
   signal load_reg0, load_reg1, load_reg2, load_reg3, load_reg4, load_reg5, load_reg6,
      load_reg7 : STD_LOGIC;
   --output load for each register
   signal q_reg0, q_reg1, q_reg2, q_reg3, q_reg4, q_reg5, q_reg6, q_reg7 :
      STD_LOGIC_VECTOR(15 downto 0);
   signal data_src_mux_out, src_reg : STD_LOGIC_VECTOR(15 downto 0);

--port mapping for each individual component within the schema
begin
   reg00 : reg PORT MAP(
        load  =>   load_reg0,
        clk   =>   clk,
        D     =>   data_src_mux_out,
        Q     =>   q_reg0
   );
```

```
reg01 : reg PORT MAP(
     load  =>   load_reg1,
     clk   =>   clk,
     D     =>   data_src_mux_out,
     Q     =>   q_reg1
);

reg02 : reg PORT MAP(
     load  =>   load_reg2,
     clk   =>   clk,
     D     =>   data_src_mux_out,
     Q     =>   q_reg2
);

reg03 : reg PORT MAP(
     load  =>   load_reg3,
     clk   =>   clk,
     D     =>   data_src_mux_out,
     Q     =>   q_reg3
);


reg04 : reg PORT MAP(
     load  =>   load_reg4,
     clk   =>   clk,
     D     =>   data_src_mux_out,
     Q     =>   q_reg4
);

reg05 : reg PORT MAP(
     load  =>   load_reg5,
     clk   =>   clk,
     D     =>   data_src_mux_out,
     Q     =>   q_reg5
);

reg06 : reg PORT MAP(
     load  =>   load_reg6,
     clk   =>   clk,
     D     =>   data_src_mux_out,
     Q     =>   q_reg6
);

reg07 : reg PORT MAP(
     load  =>   load_reg7,
     clk   =>   clk,
     D     =>   data_src_mux_out,
     Q     =>   q_reg7
);

des_decoder_3_8 : decoder PORT MAP(
     a0    => des_a0,
     a1    => des_a1,
     a2    => des_a2,
```

```
        q0    => load_reg0,
        q1    => load_reg1,
        q2    => load_reg2,
        q3    => load_reg3,
        q4    => load_reg4,
        q5    => load_reg5,
        q6    => load_reg6,
        q7    => load_reg7
    );

    src_mux_2_16 : mux_2_16 PORT MAP(
        in0   => data,
        in1   => src_reg,
        s     => data_load,
        z     => data_src_mux_out
    );

    src_mux_8_16 : mux_8_16 PORT MAP(
        in0   => q_reg0,
        in1   => q_reg1,
        in2   => q_reg2,
        in3   => q_reg3,
        in4   => q_reg4,
        in5   => q_reg5,
        in6   => q_reg6,
        in7   => q_reg7,
        s0    => src_s0,
        s1    => src_s1,
        s2    => src_s2,
        z     => src_reg
    );

    --register instantiation
    reg0 <= q_reg0;
    reg1 <= q_reg1;
    reg2 <= q_reg2;
    reg3 <= q_reg3;
    reg4 <= q_reg4;
    reg5 <= q_reg5;
    reg6 <= q_reg6;
    reg7 <= q_reg7;

end Behavioral;
```


register_file:1
mux_8_16
decoder
reg
reg00
des_decoder_3_8
src_mux_8_16
mux_2_16
reg
reg01
src_mux_2_16
reg
reg02
reg
reg03
reg
reg04
reg
reg05
reg
reg06
reg
reg07
register_file

2 Component Test Benches

2.1 Register Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY reg_test_bench IS
END reg_test_bench;

ARCHITECTURE behavior OF reg_test_bench IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT reg
    PORT(
         D : IN std_logic_vector(15 downto 0);
         load : IN std_logic;
         clk : IN std_logic;
         Q : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal D : std_logic_vector(15 downto 0) := (others => '0');
   signal load : std_logic := '0';
   signal clk : std_logic := '0';

   --Outputs
   signal Q : std_logic_vector(15 downto 0);

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: reg PORT MAP (
          D => D,
          load => load,
          clk => clk,
          Q => Q
        );

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;

   -- Stimulus process
   stim_proc: process
```

```
   begin
      wait for 10ns;
      D <= x"FFFF";
      load <= '1';

      wait for 10ns;
      load <= '0';

      wait for 10ns;
      D <= x"AAAA";
      load <= '1';

      wait for 10ns;
      load <= '0';
   end process;
END;
```

2.2 Decoder Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY decoder_tb IS
END decoder_tb;

ARCHITECTURE behavior OF decoder_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT decoder
    PORT(
         a0 : IN std_logic;
         a1 : IN std_logic;
         a2 : IN std_logic;
         q0 : OUT std_logic;
         q1 : OUT std_logic;
         q2 : OUT std_logic;
         q3 : OUT std_logic;
         q4 : OUT std_logic;
         q5 : OUT std_logic;
         q6 : OUT std_logic;
         q7 : OUT std_logic
        );
    END COMPONENT;

   --Inputs
   signal a0 : std_logic := '0';
   signal a1 : std_logic := '0';
   signal a2 : std_logic := '0';

   --Outputs
   signal q0 : std_logic;
   signal q1 : std_logic;
   signal q2 : std_logic;
   signal q3 : std_logic;
```

```
   signal q4 : std_logic;
   signal q5 : std_logic;
   signal q6 : std_logic;
   signal q7 : std_logic;

BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: decoder PORT MAP (
          a0 => a0,
          a1 => a1,
          a2 => a2,
          q0 => q0,
          q1 => q1,
          q2 => q2,
          q3 => q3,
          q4 => q4,
          q5 => q5,
          q6 => q6,
          q7 => q7
        );

   -- Stimulus process
   stim_proc: process
   begin
      --000
      wait for 10ns;
      a0 <= '0';
      a1 <= '0';
      a2 <= '0';
      --001
      wait for 10ns;
      a0 <= '0';
      a1 <= '0';
      a2 <= '1';
      --010
      wait for 10ns;
      a0 <= '0';
      a1 <= '1';
      a2 <= '0';
      --011
      wait for 10ns;
      a0 <= '0';
      a1 <= '1';
      a2 <= '1';
      --100
      wait for 10ns;
      a0 <= '1';
      a1 <= '0';
      a2 <= '0';
      --101
      wait for 10ns;
      a0 <= '1';
      a1 <= '0';
      a2 <= '1';
      --110
```

```
      wait for 10ns;
      a0 <= '1';
      a1 <= '1';
      a2 <= '0';
      --111
      wait for 10ns;
      a0 <= '1';
      a1 <= '1';
      a2 <= '1';
   end process;
END;
```

2.3 2 to 16 Bit Multiplexer Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux_2_16_test_bench IS
END mux_2_16_test_bench;

ARCHITECTURE behavior OF mux_2_16_test_bench IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT mux_2_16
    PORT(
         in0 : IN std_logic_vector(15 downto 0);
         in1 : IN std_logic_vector(15 downto 0);
         s : IN std_logic;
         z : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal in0 : std_logic_vector(15 downto 0) := (others => '0');
   signal in1 : std_logic_vector(15 downto 0) := (others => '0');
   signal s : std_logic := '0';

   --Outputs
   signal z : std_logic_vector(15 downto 0);

BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: mux_2_16 PORT MAP (
          in0 => in0,
          in1 => in1,
          s => s,
          z => z
        );

   -- Stimulus process
   stim_proc: process
   begin
      wait for 10ns;
```

```
      in0 <= x"FFFF";
      in1 <= x"AAAA";

      wait for 10ns;
      s <= '1';

      wait for 10ns;
      s <= '0';

      wait for 10ns;
      s <= '1';

   end process;
END;
```

2.4 8 to 16 Bit Multiplexer Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux_8_16_test_bench IS
END mux_8_16_test_bench;

ARCHITECTURE behavior OF mux_8_16_test_bench IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT mux_8_16
    PORT(
         s0 : IN std_logic;
         s1 : IN std_logic;
         s2 : IN std_logic;
         in0 : IN std_logic_vector(15 downto 0);
         in1 : IN std_logic_vector(15 downto 0);
         in2 : IN std_logic_vector(15 downto 0);
         in3 : IN std_logic_vector(15 downto 0);
         in4 : IN std_logic_vector(15 downto 0);
         in5 : IN std_logic_vector(15 downto 0);
         in6 : IN std_logic_vector(15 downto 0);
         in7 : IN std_logic_vector(15 downto 0);
         z : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal s0 : std_logic := '0';
   signal s1 : std_logic := '0';
   signal s2 : std_logic := '0';
   signal in0 : std_logic_vector(15 downto 0) := (others => '0');
   signal in1 : std_logic_vector(15 downto 0) := (others => '0');
   signal in2 : std_logic_vector(15 downto 0) := (others => '0');
   signal in3 : std_logic_vector(15 downto 0) := (others => '0');
   signal in4 : std_logic_vector(15 downto 0) := (others => '0');
   signal in5 : std_logic_vector(15 downto 0) := (others => '0');
   signal in6 : std_logic_vector(15 downto 0) := (others => '0');
```

```
   signal in7 : std_logic_vector(15 downto 0) := (others => '0');

   --Outputs
   signal z : std_logic_vector(15 downto 0);

BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: mux_8_16 PORT MAP (
          s0 => s0,
          s1 => s1,
          s2 => s2,
          in0 => in0,
          in1 => in1,
          in2 => in2,
          in3 => in3,
          in4 => in4,
          in5 => in5,
          in6 => in6,
          in7 => in7,
          z => z
        );

   -- Stimulus process
   stim_proc: process
   begin
      in0 <= x"FFFF";
      in1 <= x"EEEE";
      in2 <= x"DDDD";
      in3 <= x"CCCC";
      in4 <= x"BBBB";
      in5 <= x"AAAA";
      in6 <= x"9999";
      in7 <= x"8888";

      wait for 10ns;
      s0 <= '1';
      s1 <= '0';
      s2 <= '0';

      wait for 10ns;
      s0 <= '0';
      s1 <= '1';
      s2 <= '0';

      wait for 10ns;
      s0 <= '1';
      s1 <= '1';
      s2 <= '0';

      wait for 10ns;
      s0 <= '0';
      s1 <= '0';
      s2 <= '1';

      wait for 10ns;
```

```
      s0 <= '1';
      s1 <= '0';
      s2 <= '1';

      wait for 10ns;
      s0 <= '0';
      s1 <= '1';
      s2 <= '1';

      wait for 10ns;
      s0 <= '1';
      s1 <= '1';
      s2 <= '1';
   end process;
END;
```

2.5 Top Level Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY datapath_test_bench IS
END datapath_test_bench;

ARCHITECTURE behavior OF datapath_test_bench IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT register_file
    PORT(
         src_s0 : IN std_logic;
         src_s1 : IN std_logic;
         src_s2 : IN std_logic;
         des_a0 : IN std_logic;
         des_a1 : IN std_logic;
         des_a2 : IN std_logic;
         data_load : IN std_logic;
         clk : IN std_logic;
         data : IN std_logic_vector(15 downto 0);
         Q : OUT std_logic_vector(15 downto 0);
         reg0 : OUT std_logic_vector(15 downto 0);
         reg1 : OUT std_logic_vector(15 downto 0);
         reg2 : OUT std_logic_vector(15 downto 0);
         reg3 : OUT std_logic_vector(15 downto 0);
         reg4 : OUT std_logic_vector(15 downto 0);
         reg5 : OUT std_logic_vector(15 downto 0);
         reg6 : OUT std_logic_vector(15 downto 0);
         reg7 : OUT std_logic_vector(15 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal src_s0 : std_logic := '0';
   signal src_s1 : std_logic := '0';
   signal src_s2 : std_logic := '0';
```

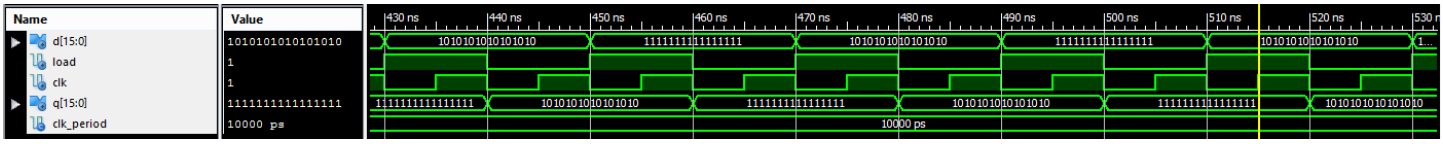
```
   signal des_a0 : std_logic := '0';
   signal des_a1 : std_logic := '0';
   signal des_a2 : std_logic := '0';
   signal data_load : std_logic := '0';
   signal clk : std_logic := '0';
   signal data : std_logic_vector(15 downto 0) := (others => '0');

   --Outputs
   signal Q : std_logic_vector(15 downto 0);
   signal reg0 : std_logic_vector(15 downto 0);
   signal reg1 : std_logic_vector(15 downto 0);
   signal reg2 : std_logic_vector(15 downto 0);
   signal reg3 : std_logic_vector(15 downto 0);
   signal reg4 : std_logic_vector(15 downto 0);
   signal reg5 : std_logic_vector(15 downto 0);
   signal reg6 : std_logic_vector(15 downto 0);
   signal reg7 : std_logic_vector(15 downto 0);

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: register_file PORT MAP (
          src_s0 => src_s0,
          src_s1 => src_s1,
          src_s2 => src_s2,
          des_a0 => des_a0,
          des_a1 => des_a1,
          des_a2 => des_a2,
          data_load => data_load,
          clk => clk,
          data => data,
          Q => Q,
          reg0 => reg0,
          reg1 => reg1,
          reg2 => reg2,
          reg3 => reg3,
          reg4 => reg4,
          reg5 => reg5,
          reg6 => reg6,
          reg7 => reg7
        );

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;

   -- Stimulus process
   stim_proc: process
```

```
  begin
    wait for 10ns;
    des_a0 <= '0';
    des_a1 <= '0';
    des_a2 <= '0';
    data <= x"FFFF";

    wait for 10ns;
    des_a0 <= '0';
    des_a1 <= '0';
    des_a2 <= '1';
    data <= x"EEEE";

    wait for 10ns;
    des_a0 <= '0';
    des_a1 <= '1';
    des_a2 <= '0';
    data <= x"DDDD";

    wait for 10ns;
    des_a0 <= '0';
    des_a1 <= '1';
    des_a2 <= '1';
    data <= x"CCCC";

    wait for 10ns;
    des_a0 <= '1';
    des_a1 <= '0';
    des_a2 <= '0';
    data <= x"BBBB";

    wait for 10ns;
    des_a0 <= '1';
    des_a1 <= '0';
    des_a2 <= '1';
    data <= x"AAAA";

    wait for 10ns;
    des_a0 <= '1';
    des_a1 <= '1';
    des_a2 <= '0';
    data <= x"9999";

    wait for 10ns;
    des_a0 <= '1';
    des_a1 <= '1';
    des_a2 <= '1';
    data <= x"8888";
  end process;
END;
```

3 Results of Test Benches

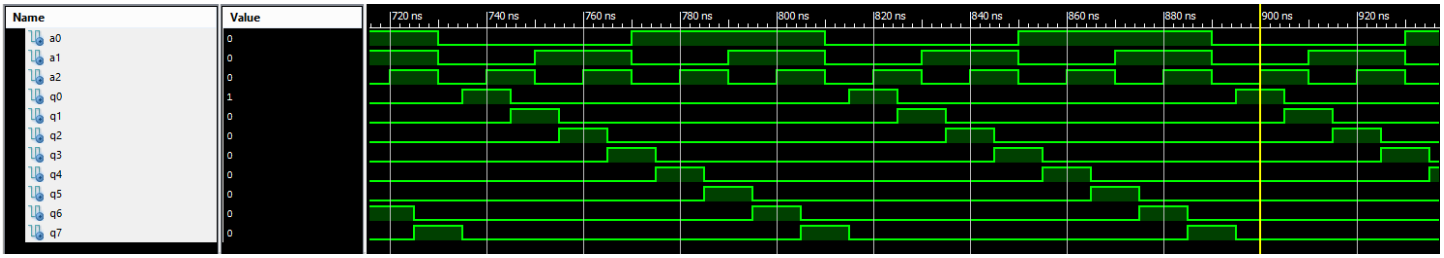
3.1 Register

The register is given a new value ever 5ns and works as expected for each edge on a given rising clock signal where the load is also high.



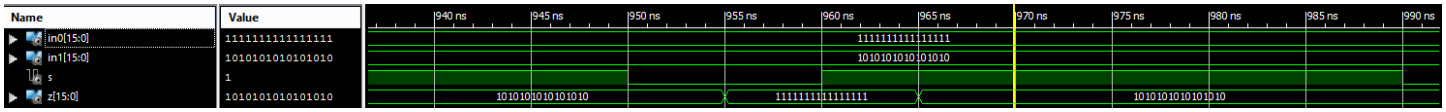
3.2 Decoder

The test bench cycles through each given combination, causing each given output pin to turn high via the register selection.



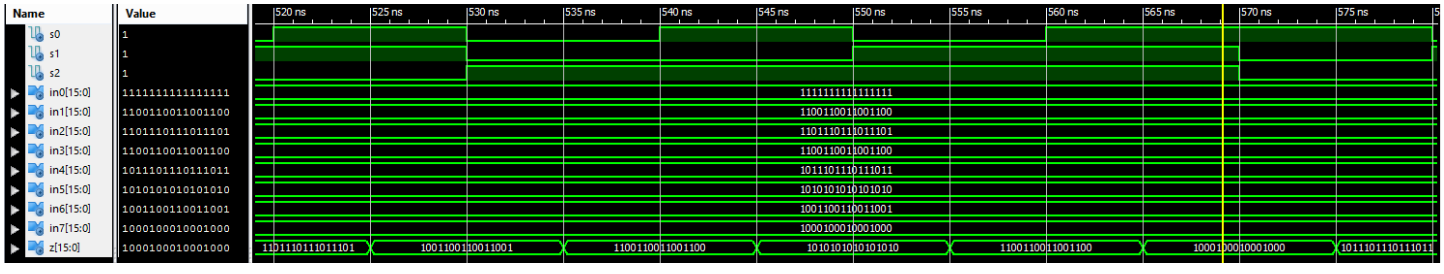
3.3 2 to 16 Multiplexer

The multiplexer cycles through a series of changes via s between 0 and 1. The output of the multiplexer switches between the given outputs for in0 and in1.



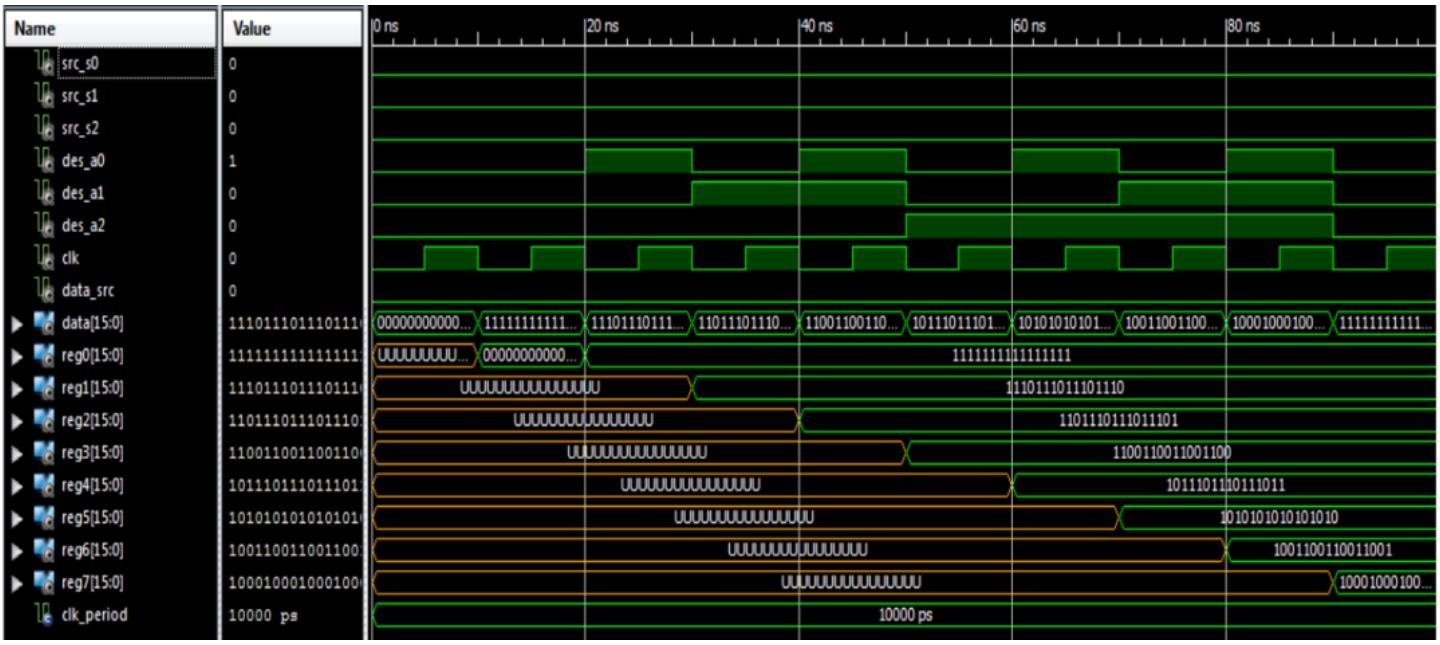
3.4 8 to 16 Multiplexer

This multiplexer cycles through all given combinations of s0, s1 and s2, thus allowing z to cycle through the inputs in0 to in7 as the output pin.



3.5 Top Level

Below is a register simulation given a different hexadecimal value for each subsequent data input, ie $R_i \leftarrow X_i \quad i = 0, 7$



The register r0 is loaded with values from the input data from the testbench. The value is transferred from one given register to the incremented register, r1; ie from r0 to r1, r1 to r2 ... r6 to r7. As each load is cycled through the data inputs, the transfer of data cycling between each register is passed on until the input is reset to 0x0000 after the first input time, which shows that the the values stem from the registers, and not from the data input itself sequentially, ie $R_i \leftarrow R_j \quad i, j = 0, 7$