Multi-Sport Scoreboard

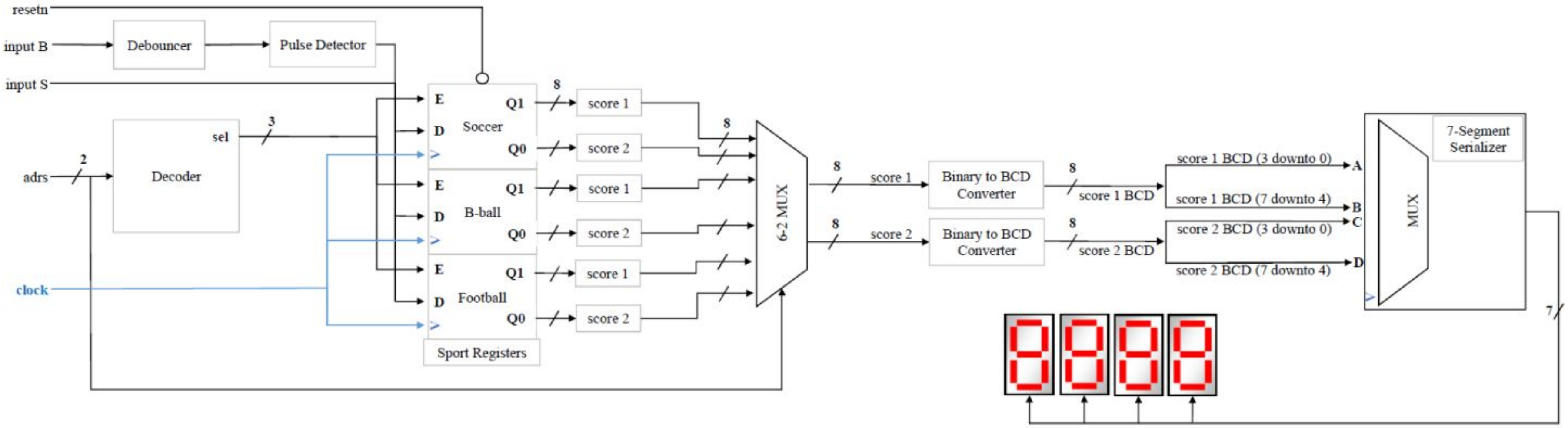
BY: Danijel Spasic, Alexandru Stoica, Mark Srbinovski, John Pak

Introduction

- Competitive scoreboards are essentially applied to all games.
- The most basic electronics use counters to keep track of rates or time.
- Exploration of counting scores for different sports are explored.
- Sports covered are Soccer, Football, and Basketball

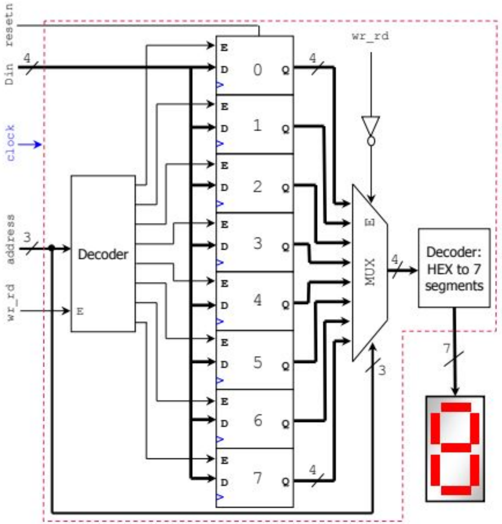
Methodology

- The idea is to implement the RAM emulator to record and count values.
- The recorded values are represented as numbers in the 7 segment displays.



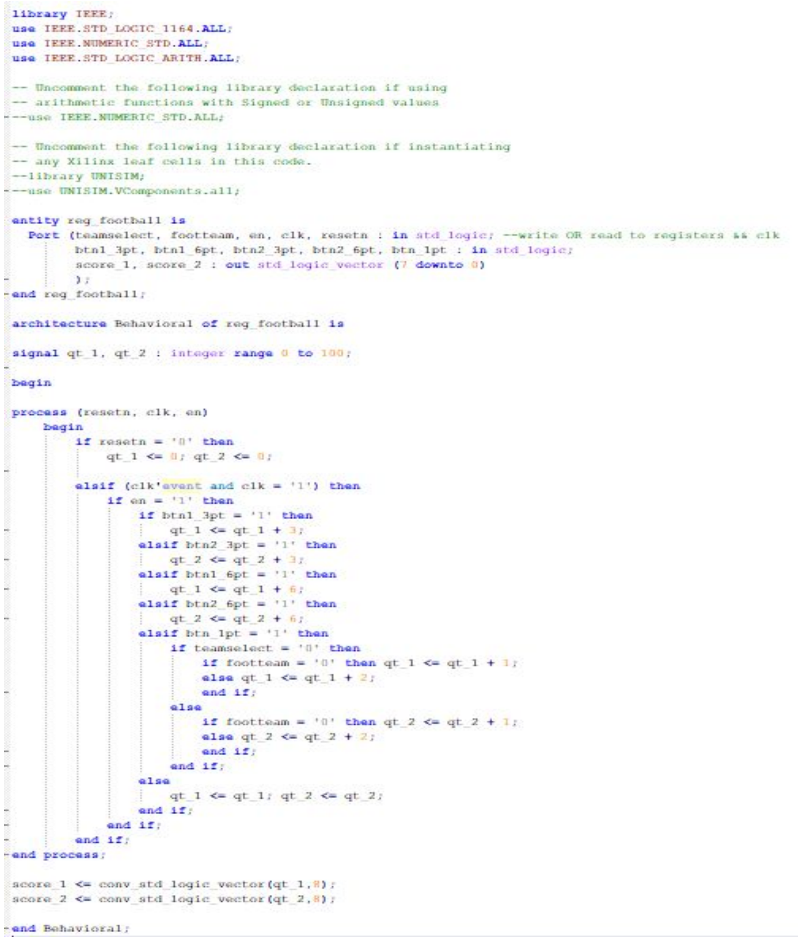
Experimental Setup

- Through the Nexus A7 board, the circuitry has been implemented.
- Codes from previous Lab 5 has been implemented and built upon.



Football Register

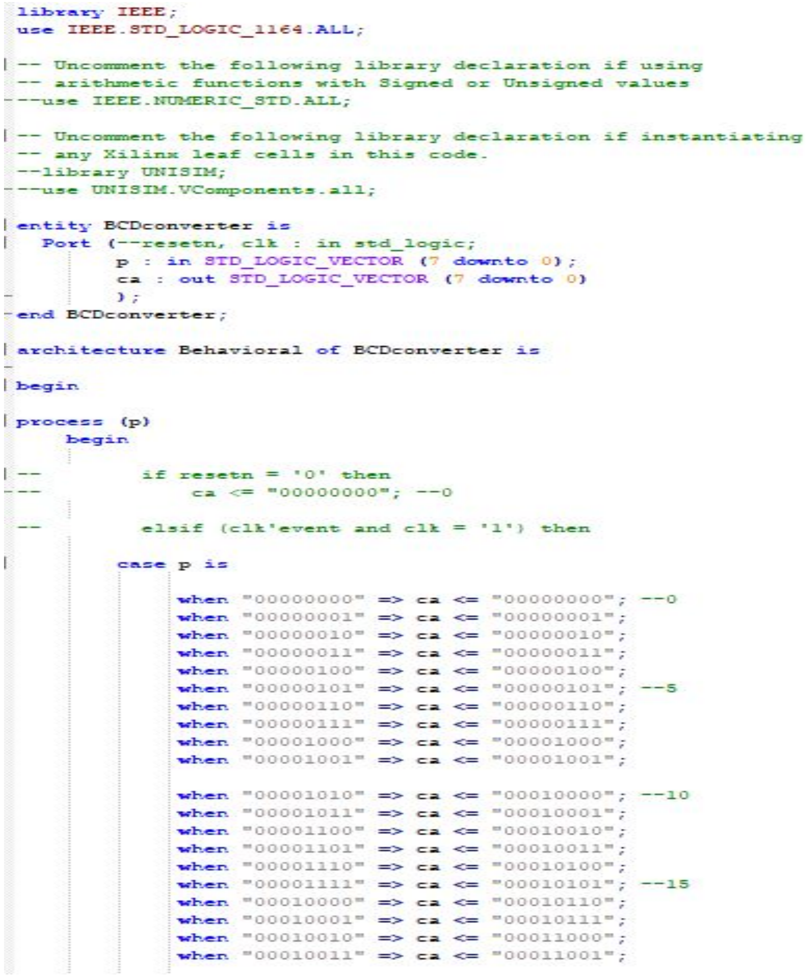
- Depends on the button pushed
- Current score incremented by a certain value
- If statements are used to account for different situations in the sport
- The other sports registers are similar



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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity reg_football is
  Port (teamselect, footteam, en, clk, resetn : in std_logic; --write OR read to registers && clk
        btn1_3pt, btn1_6pt, btn2_3pt, btn2_6pt, btn_1pt : in std_logic;
        score_1, score_2 : out std_logic_vector (7 downto 0)
        );
end reg_football;

architecture Behavioral of reg_football is

signal qt_1, qt_2 : integer range 0 to 100;

begin

process (resetn, clk, en)
    begin
        if resetn = '0' then
            qt_1 <= 0; qt_2 <= 0;

        elsif (clk'event and clk = '1') then
            if en = '1' then
                if btn1_3pt = '1' then
                    qt_1 <= qt_1 + 3;
                elsif btn2_3pt = '1' then
                    qt_2 <= qt_2 + 3;
                elsif btn1_6pt = '1' then
                    qt_1 <= qt_1 + 6;
                elsif btn2_6pt = '1' then
                    qt_2 <= qt_2 + 6;
                elsif btn_1pt = '1' then
                    if teamselect = '0' then
                        if footteam = '0' then qt_1 <= qt_1 + 1;
                        else qt_1 <= qt_1 + 2;
                        end if;
                    else
                        if footteam = '0' then qt_2 <= qt_2 + 1;
                        else qt_2 <= qt_2 + 2;
                        end if;
                    end if;
                else
                    qt_1 <= qt_1; qt_2 <= qt_2;
                end if;
            end if;
        end if;
end process;

score_1 <= conv_std_logic_vector(qt_1,8);
score_2 <= conv_std_logic_vector(qt_2,8);

end Behavioral;
```

Binary to BCD Converter

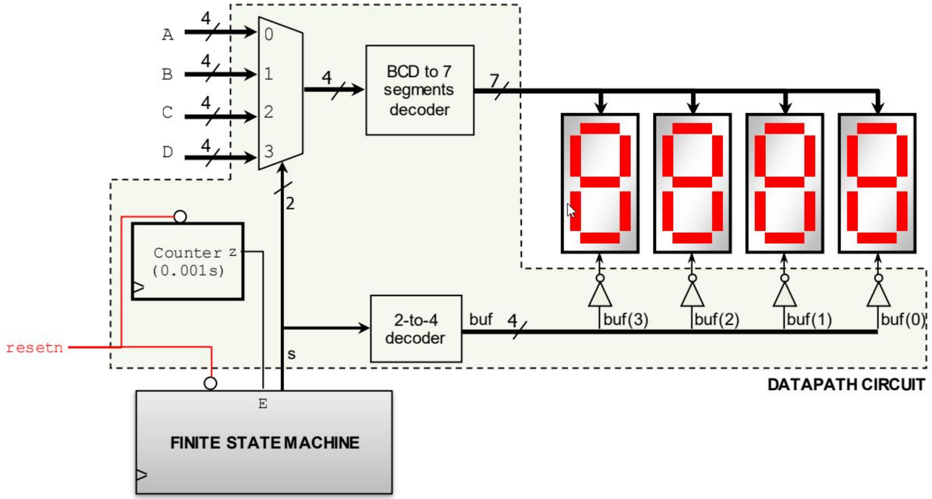
- Modified decoder code
- 100 cases (0-99)
- Converts Binary to BCD
- BCD used to show the correct numbers on the 7-segment display



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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity BCDconverter is
  Port (--resetn, clk : in std_logic;
        p : in STD_LOGIC_VECTOR (7 downto 0);
        ca : out STD_LOGIC_VECTOR (7 downto 0)
        );
end BCDconverter;

architecture Behavioral of BCDconverter is

begin

process (p)
    begin

--          if resetn = '0' then
--              ca <= "00000000"; --0

--          elsif (clk'event and clk = '1') then

        case p is

            when "00000000" => ca <= "00000000"; --0
            when "00000001" => ca <= "00000001";
            when "00000010" => ca <= "00000010";
            when "00000011" => ca <= "00000011";
            when "00000100" => ca <= "00000100";
            when "00000101" => ca <= "00000101"; --5
            when "00000110" => ca <= "00000110";
            when "00000111" => ca <= "00000111";
            when "00001000" => ca <= "00001000";
            when "00001001" => ca <= "00001001";

            when "00001010" => ca <= "00010000"; --10
            when "00001011" => ca <= "00010001";
            when "00001100" => ca <= "00010010";
            when "00001101" => ca <= "00010011";
            when "00001110" => ca <= "00010100";
            when "00001111" => ca <= "00010101"; --15
            when "00010000" => ca <= "00010110";
            when "00010001" => ca <= "00010111";
            when "00010010" => ca <= "00011000";
            when "00010011" => ca <= "00011001";
```

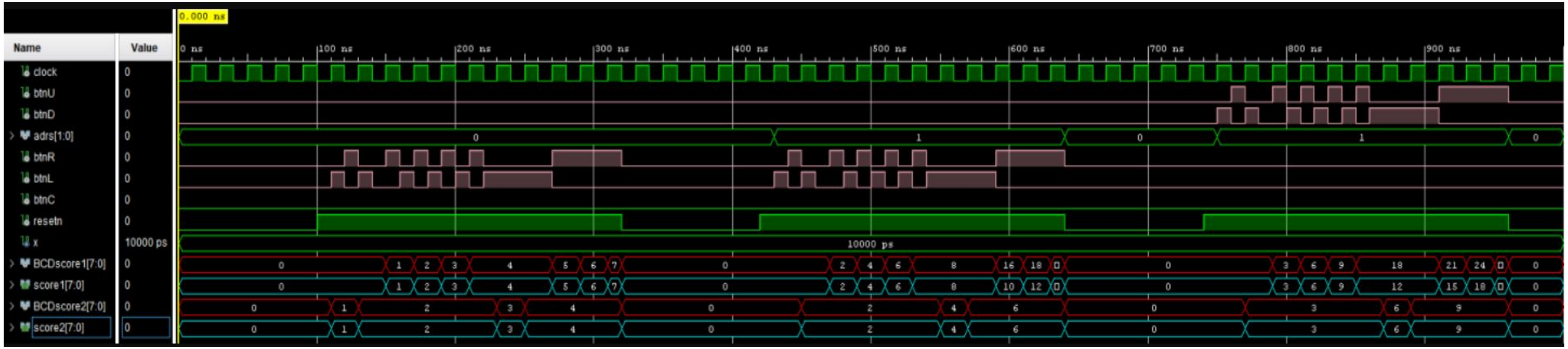
Challenges

- One of the biggest was to work on project without human interaction.
- Implementing the Serializer and BCD (Binary Converter Decimal) codes.



Conclusion

- Uses addresses to select the sport
- Different buttons and switch combinations control the increment and team
- The Binary score is then converter to BCD for the 7-segment display
- The 7-segment display shows the scores for 2 teams


0.000 ns
Name
Value
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
clock 0
btnU 0
btnD 0
adrs[1:0] 0
0 1 0 1 0
btnR 0
btnL 0
btnC 0
resetn 0
x 10000 ps
10000 ps
BCDscore1[7:0] 0
0 1 2 3 4 5 6 7 0 2 4 6 8 16 18 0 3 6 9 18 21 24 0
score1[7:0] 0
0 1 2 3 4 5 6 7 0 2 4 6 8 10 12 0 3 6 9 12 15 18 0
BCDscore2[7:0] 0
0 1 2 3 4 0 2 4 6 0 3 6 9 0
score2[7:0] 0
0 1 2 3 4 0 2 4 6 0 3 6 9 0