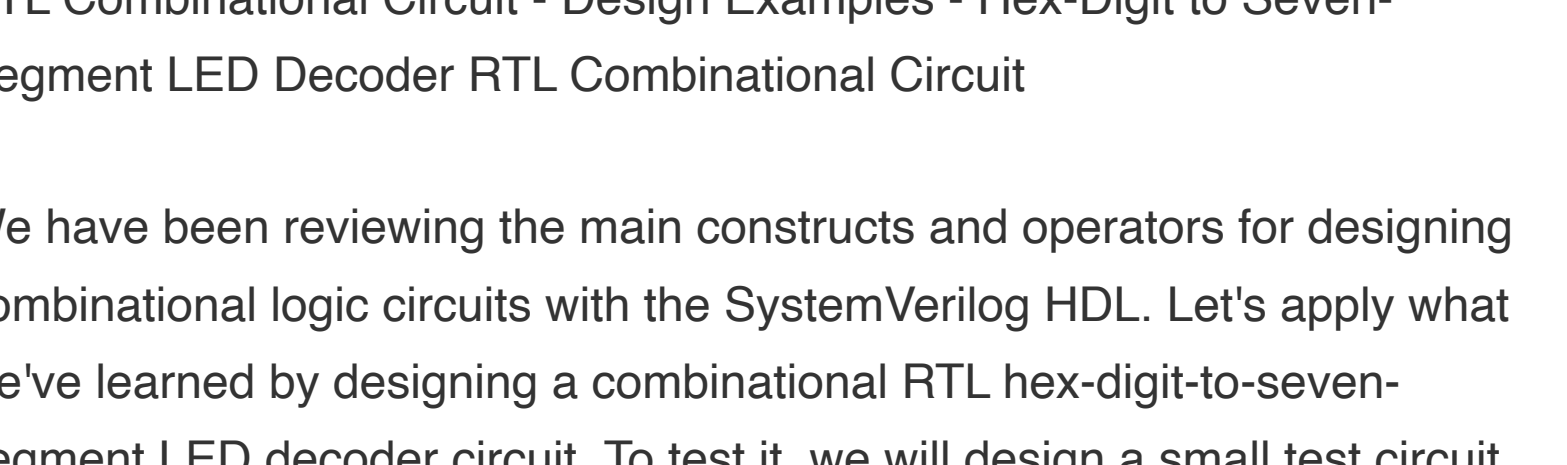
Related
SystemVerilog Study Notes. RTL Combinational Circuit Operators
I continue my series of notes on Sys
SystemVerilog Study Notes. Barrel Shifter RTL Combinational Circuit
RTL Combinational Circuit - Design
SystemVerilog Study Notes. BCD Number Format. RTL Combinational Circuit
Finishing up with combinational circ
SystemVerilog Study Notes. RTL Combinational Circuit - Concurrent and Control Constructs
In the previous chapter we reviewed
SystemVerilog Study Notes. Simplified Floating Point Arithmetic. RTL Combinational Circuit
We continue with combinational circ

SystemVerilog Study Notes. Hex-Digit to Seven-Segment LED Decoder RTL Combinational Circuit

javagoza 22 Aug 2022



RTL Combinational Circuit - Design Examples - Hex-Digit to Seven-Segment LED Decoder RTL Combinational Circuit

We have been reviewing the main constructs and operators for designing combinational logic circuits with the SystemVerilog HDL. Let's apply what we've learned by designing a combinational RTL hex-digit-to-seven-segment LED decoder circuit. To test it, we will design a small test circuit that will introduce us to the technique of Multi-Digit Seven-Segment LED Displays with Time Multiplexing.

Table of Contents

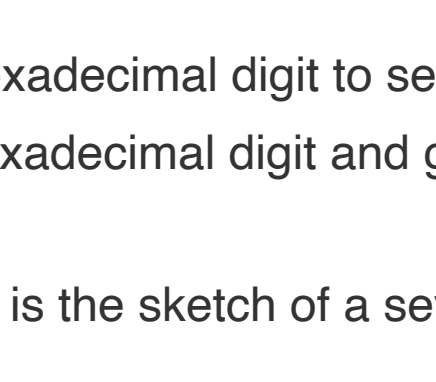
SystemVerilog Study Notes Chapters

Multi-Digit Seven-Segment LED Display Time-Multiplexing

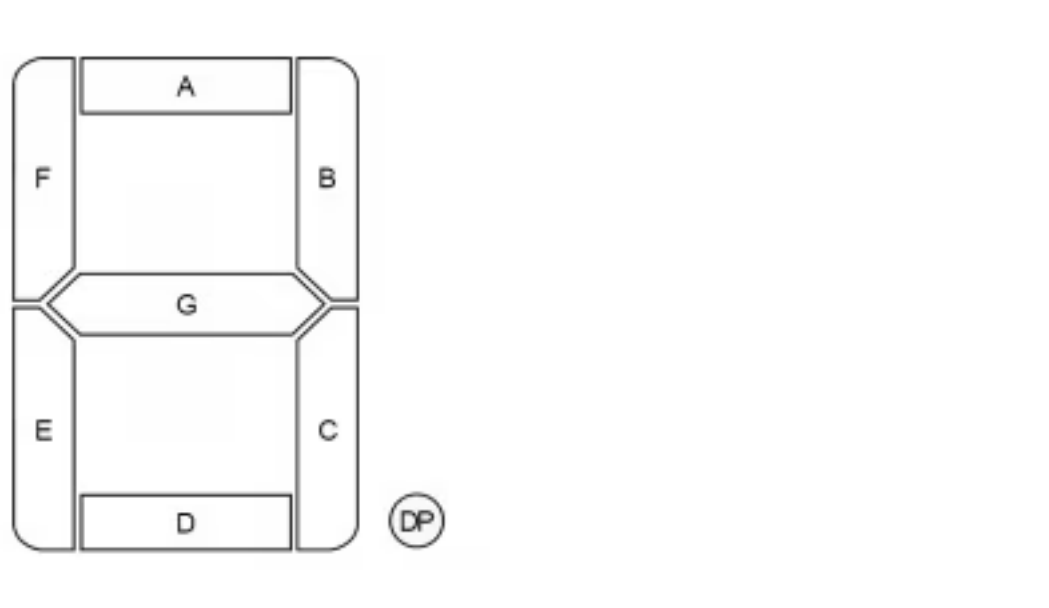
Hexadecimal Digit to Seven-Segment LED Decoder

A hexadecimal digit to seven-segment LED decoder treats a 4-bit input as a hexadecimal digit and generates appropriate LEDs patterns

This is the sketch of a seven-segment LED display:



Seven-segment LED display and hexadecimal patterns



We can group together the LED control signals as g, f, e, d, c, b and a as a single 7-bit string. We suppose digits with a common cathode, so we need active high signals, then we can express the required patterns with the following bit strings:

- `0: 0111111`
- `1: 0000110`
- `2: 1011011`
- `3: 1001111`
- `4: 1100110`
- `5: 1101101`
- `6: 1111101`
- `7: 0000111`
- `8: 1111111`
- `9: 1101111`
- `A: 1110111`
- `B: 1111100`
- `C: 1011000`
- `D: 1011110`
- `E: 1111001`
- `F: 1110001`

SystemVerilog Hex-digit to Seven-Segment LED decoder implementation.

A possible implementation in SystemVerilog code for the hexadecimal digit to seven-segment LED decoder.

The code uses one selected signal assignment to list all the desired patterns for the seven LSBs of the sseg signal.

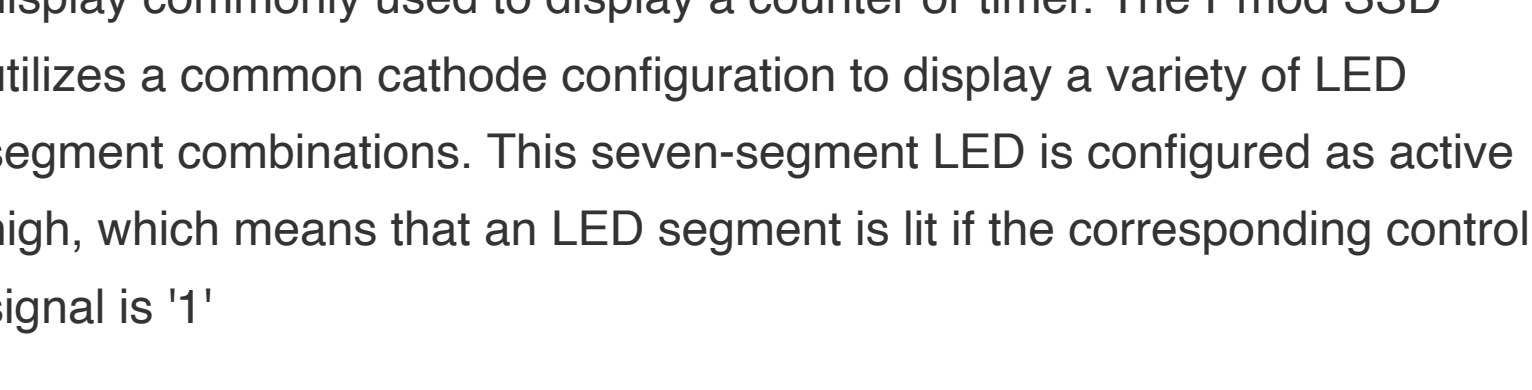
```
            4'h0: sseg[6:0] = 7'b0111111;
            4'h1: sseg[6:0] = 7'b0000110;
            4'h2: sseg[6:0] = 7'b1011011;
            4'h3: sseg[6:0] = 7'b1001111;
            4'h4: sseg[6:0] = 7'b1100110;
            4'h5: sseg[6:0] = 7'b1101101;
            4'h6: sseg[6:0] = 7'b1111101;
            4'h7: sseg[6:0] = 7'b0000111;
            4'h8: sseg[6:0] = 7'b1111111;
            4'h9: sseg[6:0] = 7'b1101111;
            4'hA: sseg[6:0] = 7'b1110111;
            4'hB: sseg[6:0] = 7'b1111100;
            4'hC: sseg[6:0] = 7'b1011000;
            4'hD: sseg[6:0] = 7'b1011110;
            4'hE: sseg[6:0] = 7'b1111001;
            default : sseg[6:0] = 7'b1110001; // 4'hF
        endcase

    end
endmodule
```

Hex-digit to Seven-Segment LED decoder test-bench

We can implement a simple test-bench to simulate all the possible hex digits values with a for-loop.

```
module hex_to_sseg_testbench;

    logic [3:0] hex;
    logic [6:0] sseg;

    hex_to_sseg uut(.*);

    initial
    begin
        for (int i = 0; i< 16; ++i)
        begin
            hex = i; #10;
        end
        $stop;
    end
endmodule
```

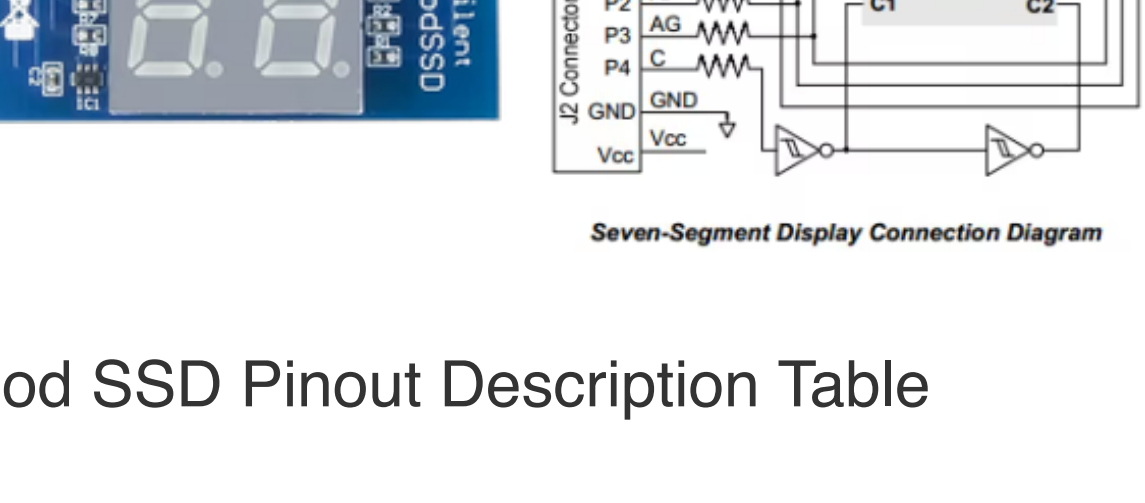
This is the Simulation test-bench scope view in Vivado:



Testing the Hex-digit to Seven-Segment LED decoder with the Digilent Pmod SSD

We will use the Digilent Pmod SSD . It is a 2 digit seven-segment display commonly used to display a counter or timer. The Pmod SSD utilizes a common cathode configuration to display a variety of LED segment combinations. This seven-segment LED is configured as active high, which means that an LED segment is lit if the corresponding control signal is '1'

The Pmod SSD communicates with the host board via the GPIO protocol. A logic level high signal on a particular anode will light up that respective segment on whichever digit is currently enabled. A particular digit can be selected by driving the Digit Selection pin (C) to a logic high or low voltage. Pmod SSD Reference Manual - Digilent Reference



Pmod SSD Pinout Description Table

Header J1			Header J2		
Pin	Signal	Description	Pin	Signal	Description
1	AA	Segment A	1	AE	Segment E
2	AB	Segment B	2	AF	Segment F
3	AC	Segment C	3	AG	Segment G
4	AD	Segment D	4	C	Digit Selection pin
5	GND	Power Supply Ground	5	GND	Power Supply Ground
6	VCC	Positive Power Supply	6	VCC	Positive Power Supply

The PmodSSD does not have the dp LEDs connected to the Pmod connector.

Driving Multi-Digit Seven-Segment LED Displays with Time-Multiplexing Scheme

To save the number of I/O pins, a time-multiplexing scheme is sometimes used in multi-digit seven-segment LED display. In the case of the Pmod SSD only one digit can be lit at a particular time, to display both digits a particular value will need to alternately light up the two digits at least every 20 milliseconds (50 Hz). This will correlate to each digit being lit up for 10 milliseconds each before the other segment needs to be "turned on". Higher refresh rates can be achieved by alternating which digit is currently powered at shorter time intervals.

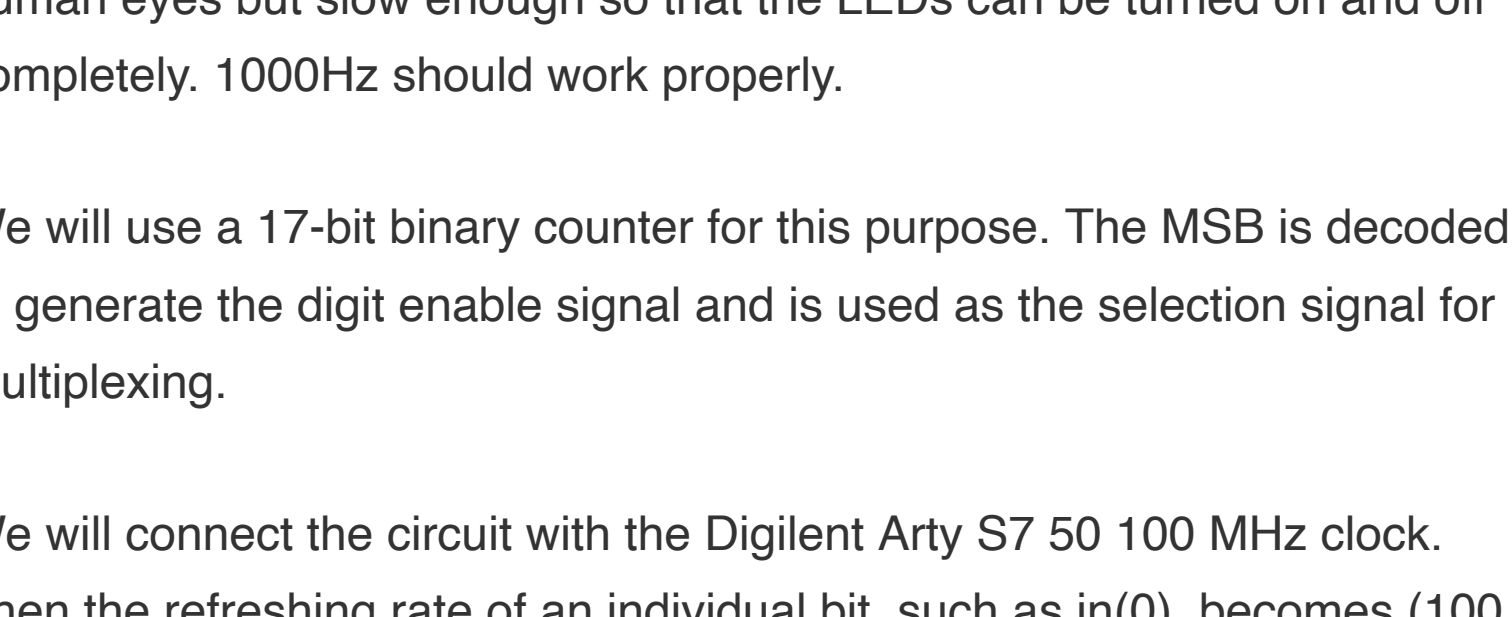
We will design a time-multiplexing circuit with a 1-to-2 decoder. It takes two seven patterns, in1, in0 and passes them to the output, sseg, in accordance with the digit enable signal digitn.

The refresh rate of the enable signal has to be fast enough to fool the human eyes but slow enough so that the LEDs can be turned on and off completely. 1000Hz should work properly.

We will use a 17-bit binary counter for this purpose. The MSB is decoded to generate the digit enable signal and is used as the selection signal for multiplexing.

We will connect the circuit with the Digilent Arty S7 50 100 MHz clock. Then the refreshing rate of an individual bit, such as in(0), becomes (100 000 000 / 2^16) Hz, 1526 Hz approx..

Block diagram of our time-multiplexing circuit with a 1-to-2 decoder.



2-to1 Time Multiplexing Circuit SystemVerilog Implementation.

A possible implementation for our 2-to1 Time Multiplexing circuit.

```
    // next-state logic
    assign q_next = q_reg + 1;

    // 1 MSBs of counter to control 2 to 1 multiplexing
    // and to generate active enable signal
    always_comb
    begin
        case( q_reg[N-1])
            1'b0: begin
                sseg = in1;
                digitn = 1'b0;
            end
            default:  begin
                sseg = in0;
                digitn = 1'b1;
            end
        endcase
    end

endmodule
```

Multi-Digit Seven-Segment LED Display Time-Multiplexing Testing Circuit

We will use the following circuit to verify the operation of the LED display time-multiplexing circuit.

It is a simple 4-bit increment circuit to verify the operation of the decoder.

The sw input is connected to 4 slide switches of the prototyping board. It is fed to an incrementor to obtain sw+1.

The red leds on the board are also connected to the sw input so we can also "debug" the binary input.

The original sw and the incremented one sw+1 are passed to two hex-digit to seven-segment decoders to display the two hexadecimal digits on the Pmod SSD 2-digits seven-segment LED display.

LED display time-multiplexing circuit SystemVerilog Implementation

This is a possible implementation in SystemVerilog of the incrementor circuit with LED display time-multiplexing circuit :

```
    logic [6:0] led0;
    logic [6:0] led1;

    logic [3:0] inc;

    // increment input
    assign inc = sw + 1;
    assign led = sw;

    // instantiate 2 instances of hex decoders
    // instantiate for sw input
    hex_to_sseg sseg_unit_0(.hex(sw[3:0]), .sseg(led0));
    // instantiate for incremented value
    hex_to_sseg sseg_unit_1(.hex(inc[3:0]), .sseg(led1));

    // instantiate 7-seg LED display time multiplexing module
     disp_mux_2_1 disp_unit(.clk(clk), .reset(1'b0), .in0(led0), .in1(led1


endmodule
```

Constraint file

We need to mapping the logical port names of the test circuit to the physical signals on the Arty-S7 50 prototyping board.

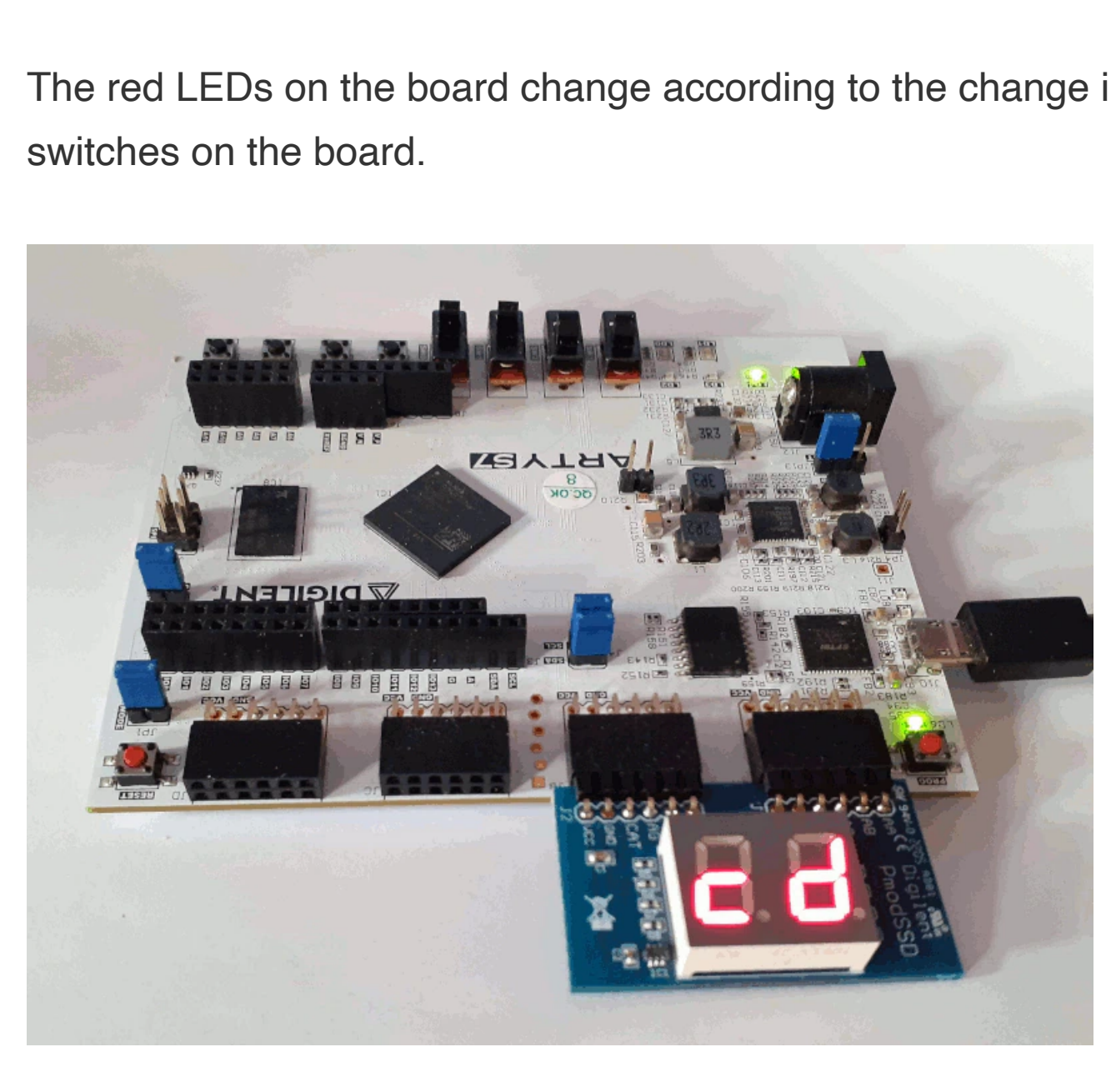
```
set_property -dict { PACKAGE_PIN P17   IOSTANDARD LVCMOS33 } [get_ports {
set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports {
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports {
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports {


## Configuration options, can be used for all designs
set_property BITSTREAM.CONFIG.CONFIGRATE 50 [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
set_property BITSTREAM.CONFIG.SPI_BUSWIDTH 4 [current_design]
set_property CONFIG_MODE SPIx4 [current_design]

## SW3 is assigned to a pin M5 in the 1.35v bank. This pin can also be use
## the VREF for BANK 34. To ensure that SW3 does not define the reference
## and to be able to use this pin as an ordinary I/O the following propert
## be set to enable an internal VREF for BANK 34. Since a 1.35v supply is
## used the internal reference is set to half that value (i.e. 0.675v). Nc
## this property must be set even if SW3 is not used in the design.
set_property INTERNAL_VREF 0.675 [get_iobanks 34]
```

The LED display time-multiplexing test circuit in action

The seven-segment LED digit on the left shows a hexadecimal representation of the 4-bit unsigned binary number represented by the positions of the 4 sliders.

The seven-segment LED digit to the right shows the value incremented by 1.

The red LEDs on the board change according to the change in the slide switches on the board.



SystemVerilog Study Notes Chapters