Adam Blake Bell

LFF734

ENEE 4710 Embedded Systems

Dr. Loveless

9/20/18

HOMEWORK #3

HW Assignment #3 (Due Thursday Sept. 20)

Overview: This assignment will apply your growing knowledge of VHDL through synthesis of your ALU design from HW #2 on the DE1-SoC FPGA Development Kit.

Resources Required: VHDL project from Assignment #2 (or instructor provided code).
Rapid Prototyping of Digital Systems – SOPC Edition
Altera DE1-SoC User Manual
Tutorial on Programming the DE1-SoC FPGA Development Board

Additional Reading Material:
VHDL Reference Manual
Lo-Carb VDHL Tutorial

Description: This lab-based HW assignment leverages the 4-bit ALU you designed for HW Assignment #2. Briefly, the ALU is described with the following operational codes (Op Code) and associated instructions. The ALU inputs include the 4-bit Op Code (S_2, S_1, S_0, C_i) and two 4-bit words: A(3..0) and B(3..0). The outputs include an 4-bit word, G(3..0), and a single bit carry-out (C_{out}).

Op Code (HEX)	Instruction
0	G = A
1	G = A + 1
2	G = A + B
3	G = A + B + 1
4	G = A + $\overline{B}$
5	G = A – B
6	G = A – 1
7	G = A
8	G = A AND B
9	G = A NAND B
A	G = A OR B
B	G = A NOR B
C	G = A XOR B
D	G = A XNOR B
E	G = $\overline{A}$
F	G = $\overline{B}$

Task: Using the VHDL-based ALU design from Assignment #2 (or the instructor provided design), program the DE1-SoC board with the Cyclone V SoC 5CSEMA5F31C6 device. You should use the toggle switches to represent the two input words A and B and single bit Carry Input (C_i). A push button (KEY) should be used to cycle through the most significant three bits of the Op Code (S_2, S_1, S_0). Use five of the red LEDs on the development board to display the binary result (including the Carry Out C_{out} and four-bit output word G). Additionally, use the 7-segment displays to present the HEX equivalent of the Op Code and either the integer OR HEX equivalent of the result G.

Deliverables: Provide the **commented** VHDL code as well as the annotated output-timing diagram (functional simulation) with test bench. The completed program should be demonstrated for the instructor. Submit this sheet to be signed by the instructor. Submit all files electronically on UTCLearn.

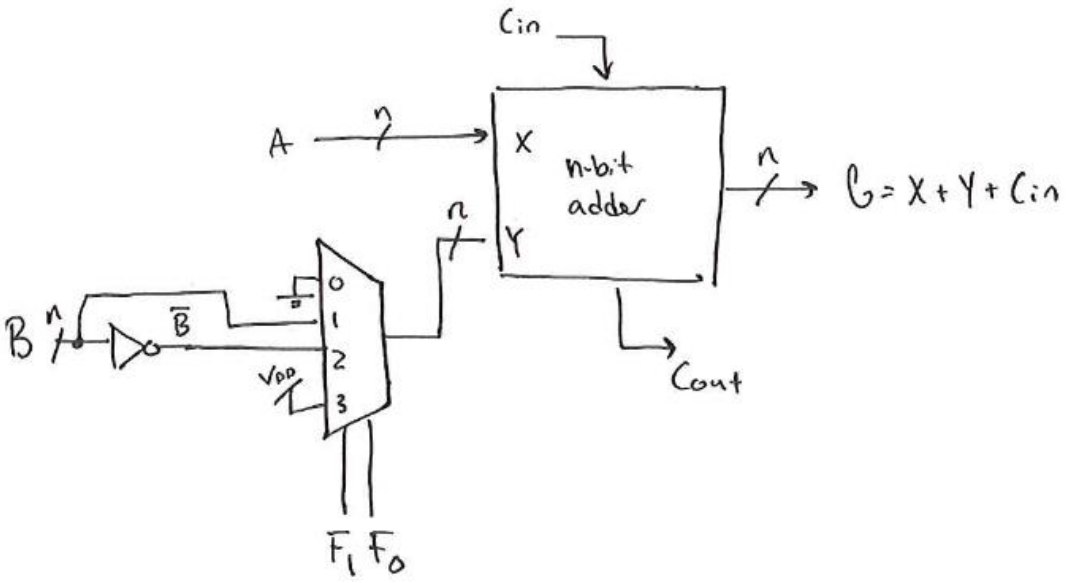


Figure 1 Arithmetic Unit of ALU

· Now, a complete ALU stage :

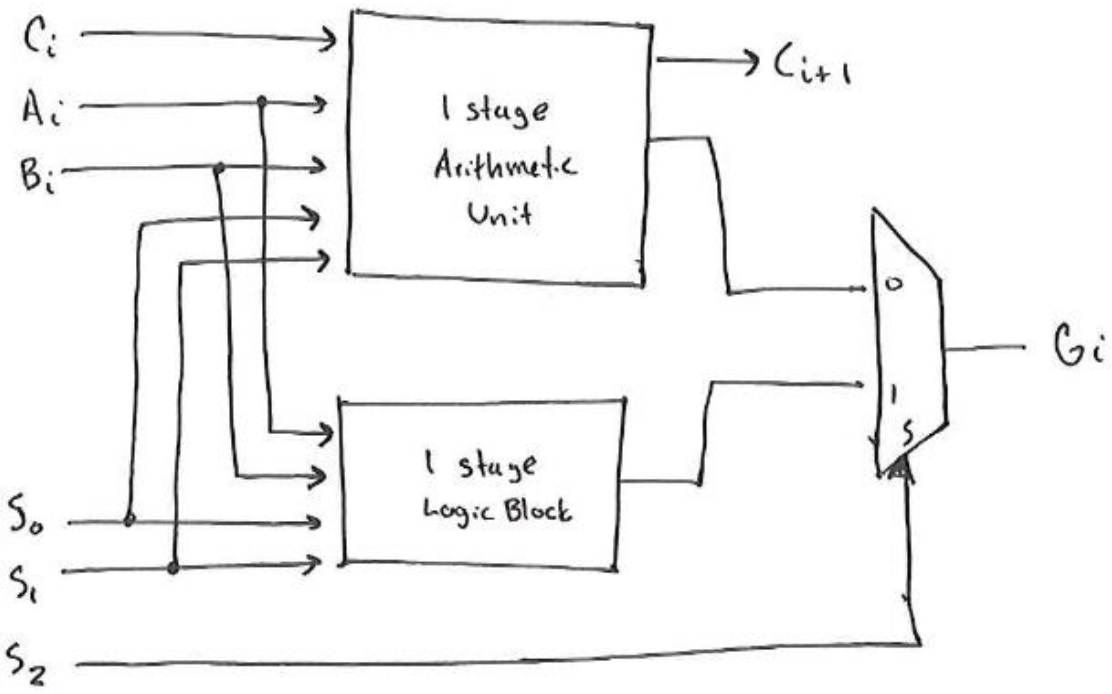


Figure 2 Full ALU architecture with Arithmetic and Logic Unit

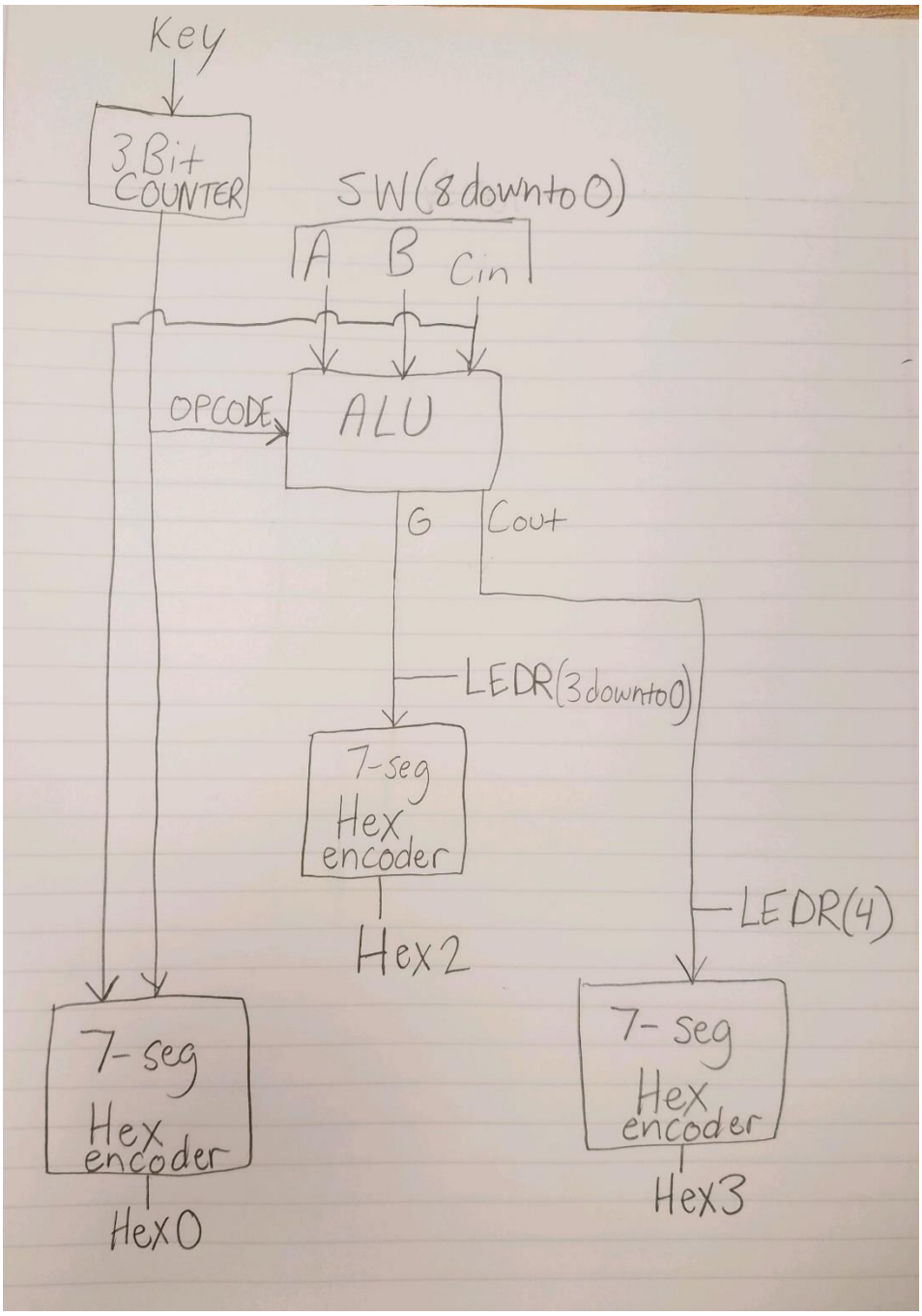
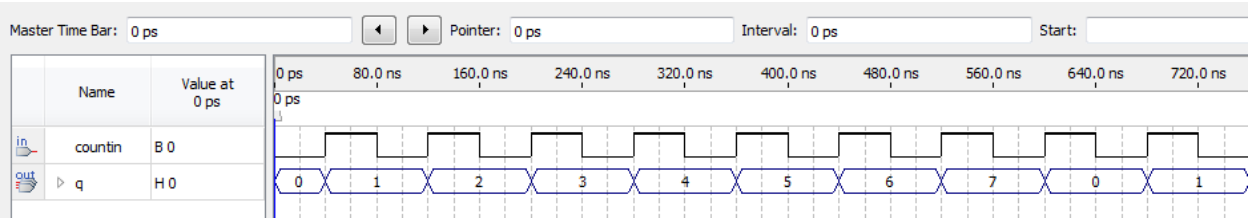


Figure 3 ALU with 3 bit counter used for S2 S1 S0 of the Opcode. The 3 bit counter allows use of a button to cycle through the operations. The inputs A, B, and Cin are controlled by 9 switches. Three 7-segment hex encoders are used to convert the binary signals of G, Cout, and OPCODE to Hex representations on three separate 7-segment displays. Cout and G can also be displayed in binary on LEDs using the output LEDR.

count_3bit.vhd — Compilation Report - Test

```
4   library ieee;
5   use ieee.std_logic_1164.all;
6   use ieee.std_logic_arith.all;
7   use ieee.std_logic_unsigned.all;
8
9   -- The 3-bit counter should accept a 1-bit input (countin) and generate an output count as a 3-bit b
10  entity count_3bit is
11     port
12     (
13        countin     : in std_logic;                          -- 1 bit input
14        q           : out std_logic_vector(2 downto 0)       -- 3 bit output vector denoting binary coun
15     );
16  end entity;
17
18  architecture count3 of count_3bit is
19  -- place any signal definitions or variables here
20     signal cin : std_logic_vector(2 downto 0):= "000"; --signal that holds the starting value of 000
21
22  begin
23     process(countin, cin)
24     begin
25        -- Detect a rising edge (or falling edge) of countin, and increment counter.
26        if(rising_edge(countin)) then
27              if(cin = "111") then
28                 cin <= "000";
29              else
30                 cin <= cin + 1;
31              end if;
32        end if;
33
34        q <= cin;
35
36     end process;
37     -- place any signal/port assignments here
38  end count3;
```

Code for my 3 bit counter. Starts at “000” and adds 1 when a rising clock edge of the input is sensed. Resets at “111”



Functional Simulation shows that the counter can correctly count from 0 to 7 although it is actually using binary.

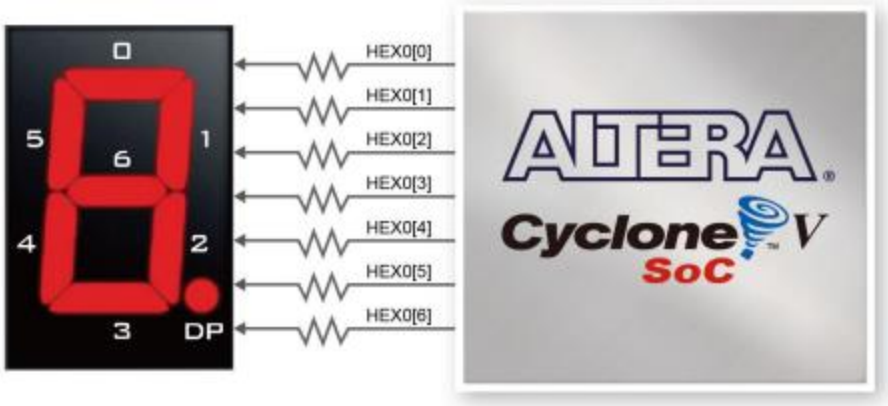


Figure 3-18 Connections between the 7-segment display HEX0 and the Cyclone V SoC FPGA

Figure 4 Diagram showing 7-segment display mappings. Taken from Altera DE1-SOC User Manual

alu_interface.vhd | count_3bit.vhd | hex_encoder.vh

```
3   -- ENEE 4710, Fall 2016
4
5   library ieee;
6   use ieee.std_logic_1164.all;
7
8   entity hex_encoder is
9   port
10     (
11        vin      : in std_logic_vector(3 downto 0);
12        hout     : out std_logic_vector(6 downto 0)
13     );
14  end entity;
15
16  architecture hex_encoder_1 of hex_encoder is
17  begin
18  -- refer to page 25 of the DE1_SOC_Users_Manual for HEX pin ass
19  with vin select
20     hout <="1000000" when "0000", -- 0
21          "1111001" when "0001", -- 1
22          "0100100" when "0010", -- 2
23          "0110000" when "0011", -- 3
24          "0011001" when "0100", -- 4
25          "0010010" when "0101", -- 5
26          "0000010" when "0110", -- 6
27          "1111000" when "0111", -- 7
28          "0000000" when "1000", -- 8
29          "0010000" when "1001", -- 9
30          "0001000" when "1010", -- A
31          "0000011" when "1011", -- b
32          "1000110" when "1100", -- C
33          "0100001" when "1101", -- d
34          "0000110" when "1110", -- E
35          "0001110" when "1111", -- F
36          "1111111" when others; -- turn off all LEDs
37  end hex_encoder_1;
```

Figure 5 7-segment Hex encoder. Takes 4 bit binary and converts it to a HEX encoding to display equivalent HEX on a 7-segment display. The individual LEDs are off when the hout bits are HIGH.

alu_interface.vhd | count_3bit.vhd | hex_encoder.vhd | Compilation Report - HW

```
15  library ieee;
16  use ieee.std_logic_1164.all;
17  use ieee.numeric_std.all;
18
19  entity alu_interface is
20  port
21      (
22  -- Place ports here
23          SW        : in std_logic_vector(8 downto 0); --a,b,cin
24          KEY       : in std_logic;
25
26          HEX0      : out std_logic_vector(6 downto 0); --OPCODE
27          HEX2      : out std_logic_vector(6 downto 0); --LSB of G
28          HEX3      : out std_logic_vector(6 downto 0); --MSB of G *ONLY USED WHEN COUT is high*
29          LEDR      : out std_logic_vector(4 downto 0) --G 4 bits and Cout 1 bit
30
31      );
32  end entity;
33
34  architecture alu_interface_1 of alu_interface is
35
36  -- Place any signals you need here
37      signal counterTOalu : std_logic_vector(2 downto 0); --MSBs of the opcode
38      signal aluTOhexB : std_logic_vector(3 downto 0);--Hex2 G
39      signal aluTOhexC : std_logic;--Hex3 Carry out
40
41
42  -- Place any variables and constants you need here
43
44  -- Place an aliases you need here
45
46  -- Define the 4-bit ALU component. This skeleton refers to Dr. Loveless' ALU from HW #2
47  component alu_4bit                                  -- alu_4bit.vhd (4-bit ALU from HW #2)
48      port
49      (
```

```
49      (
50          a         : in std_logic_vector(3 downto 0);  -- 4 bit input vector a
51          b         : in std_logic_vector(3 downto 0);  -- 4 bit input vector b
52          opcode    : in std_logic_vector(3 downto 0);  -- 4 bit input vector opcode arranged as (s2 s1 s0 cin)
53          g         : out std_logic_vector(3 downto 0); -- 4 bit output vector g
54          cout      : out std_logic                     -- 1 bit output carry out (cout)
55      );
56  end component;
57
58  -- You are going to need to create a counter to cycle through the 3 most-significant bits
59  -- of the ALU's opcode. Here is the component definition. Name the file count_3bit.vhd.
60  component count_3bit                                -- count_3bit.vhd (3-bit counter for cycling through 3 MSBs of opcode
61      port
62      (
63          countin     : in std_logic;                     -- 1 bit input
64          q           : out std_logic_vector(2 downto 0)  -- 3 bit output vector denoting binary count
65      );
66  end component;
67
68  -- You are going to need to create an encoder that accepts a 4-bit input and converts to a 7-bit code for the 7-segments
69  -- displays. Name the file hex_encoder.vhd.
70  -- HINT: Look up the 7-segment display in the DE1-SOC manual and note the order of the bits!
71  component hex_encoder is                            -- hex_encoder.vhd (4:7 HEX encoder for 7-segment displays)
72  port
73      (
74          vin       : in std_logic_vector(3 downto 0);  -- 4-bit input vector (binary)
75          hout      : out std_logic_vector(6 downto 0)  -- 7-bit output vector (7-segment display, hexadecimal)
76      );
77  end component;
78
79
80
81
82  begin
```

```
83
84  -- Describe architecture here. Place component connectivity/port maps and signal assignments.
85  -- Note: You should not require any other behavioral descriptions in this file, other than port mapping of the defined components.
86
87  count_3bit_A : count_3bit port map(countin => KEY, q => counterTOalu);
88  alu_4bit_A : alu_4bit port map(a => SW(8 downto 5), b => SW(4 downto 1), opcode => (counterTOalu & SW(0)), g => alutohexB, cout => aluTOhexC);
89  hex_encoder_A : hex_encoder port map(vin => counterTOalu & SW(0), hout => Hex0); --OPCODE
90  hex_encoder_B : hex_encoder port map(vin => aluTOhexB, hout => Hex2); --G
91  hex_encoder_C : hex_encoder port map(vin => "000" & aluTOhexC, hout => Hex3); --MSB of G *ONLY USED WHEN COUT is high*
92  LEDR(3 downto 0) <= aluTOhexB; --G in binary
93  LEDR(4) <= aluTOhexC; -- Cout in binary
94
95  end alu_interface_1;
```

Figure 6 Code for my ALU interface which uses an ALU, 3 Hex encoders, and a 3 bit counter

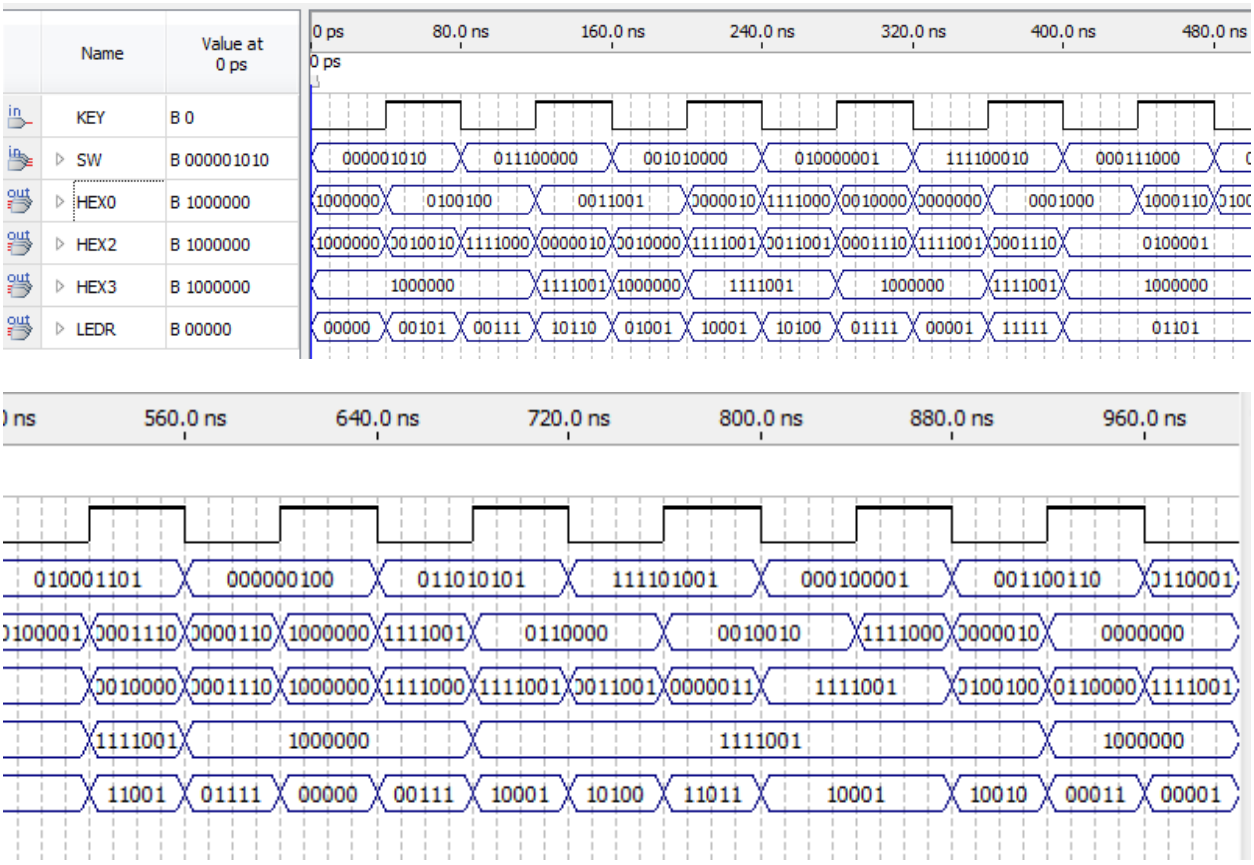


Figure 7 The functional simulation above shows the outputs of the hex encoding when random values are input into SW which is A, B, and C. The Functional Simulation is much less useful in this lab because it is time consuming to decipher the hex encodings. It is much easier to do testing on the board because the HEX outputs can be clearly red on the 7-segment displays.

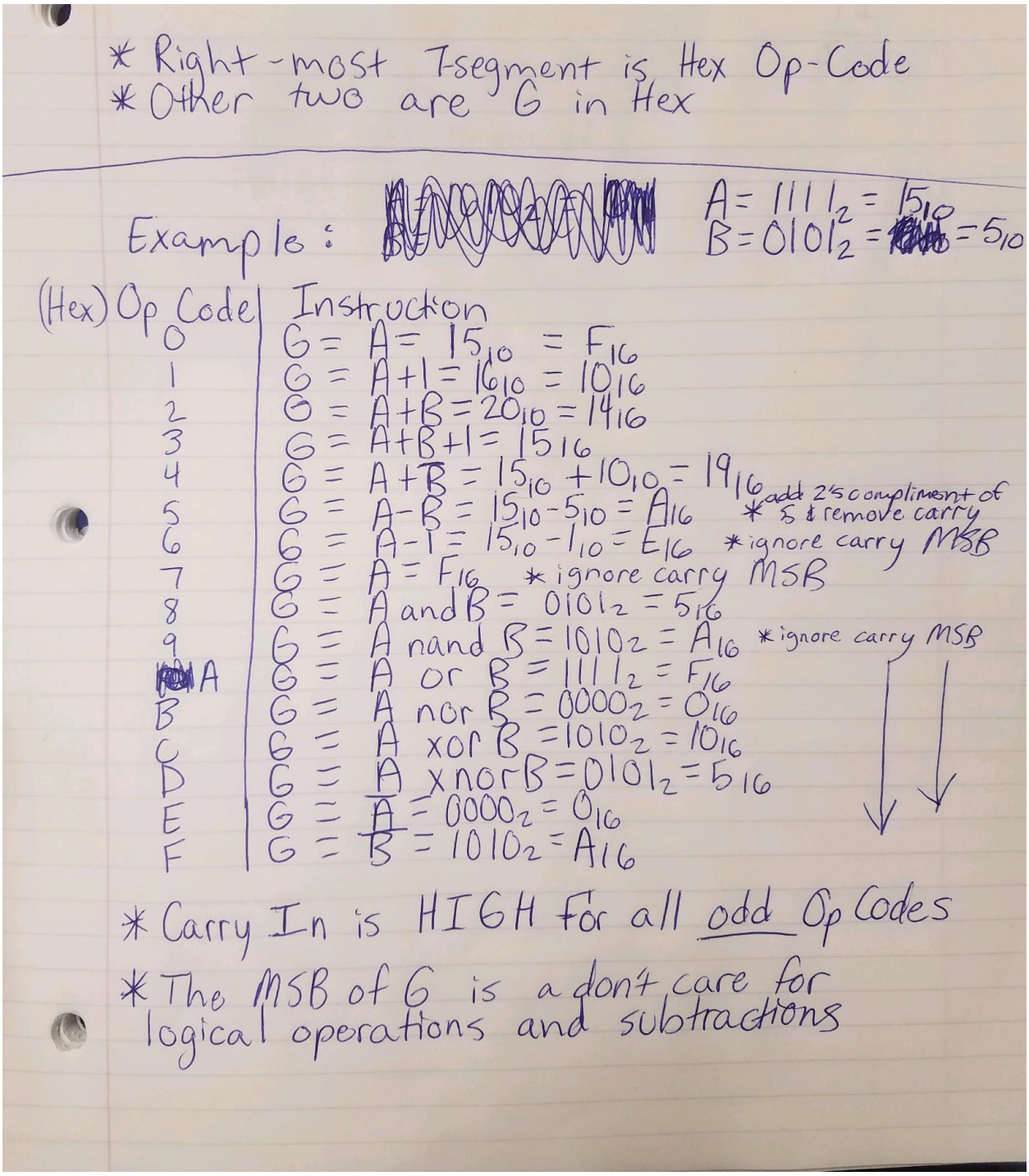
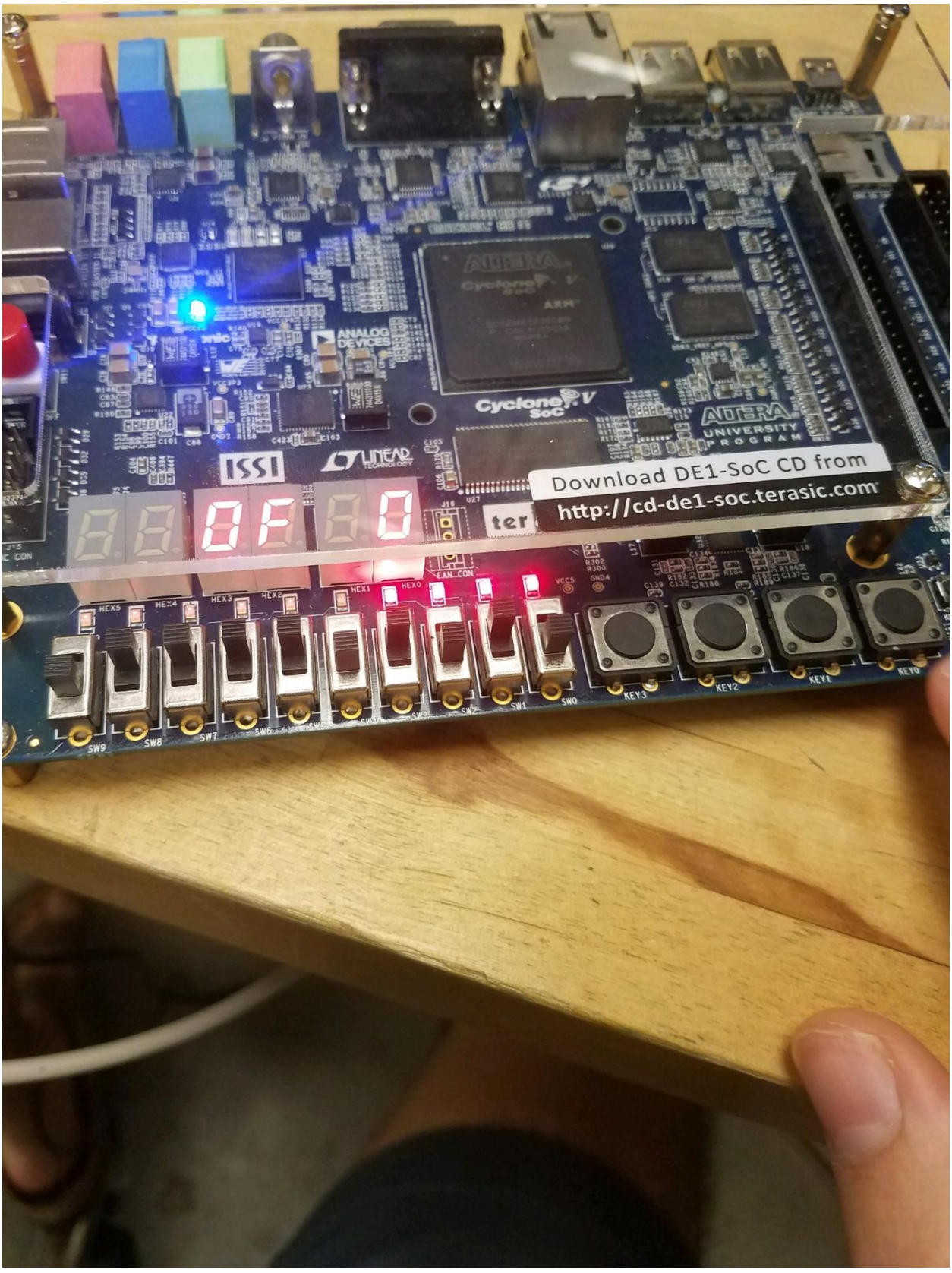


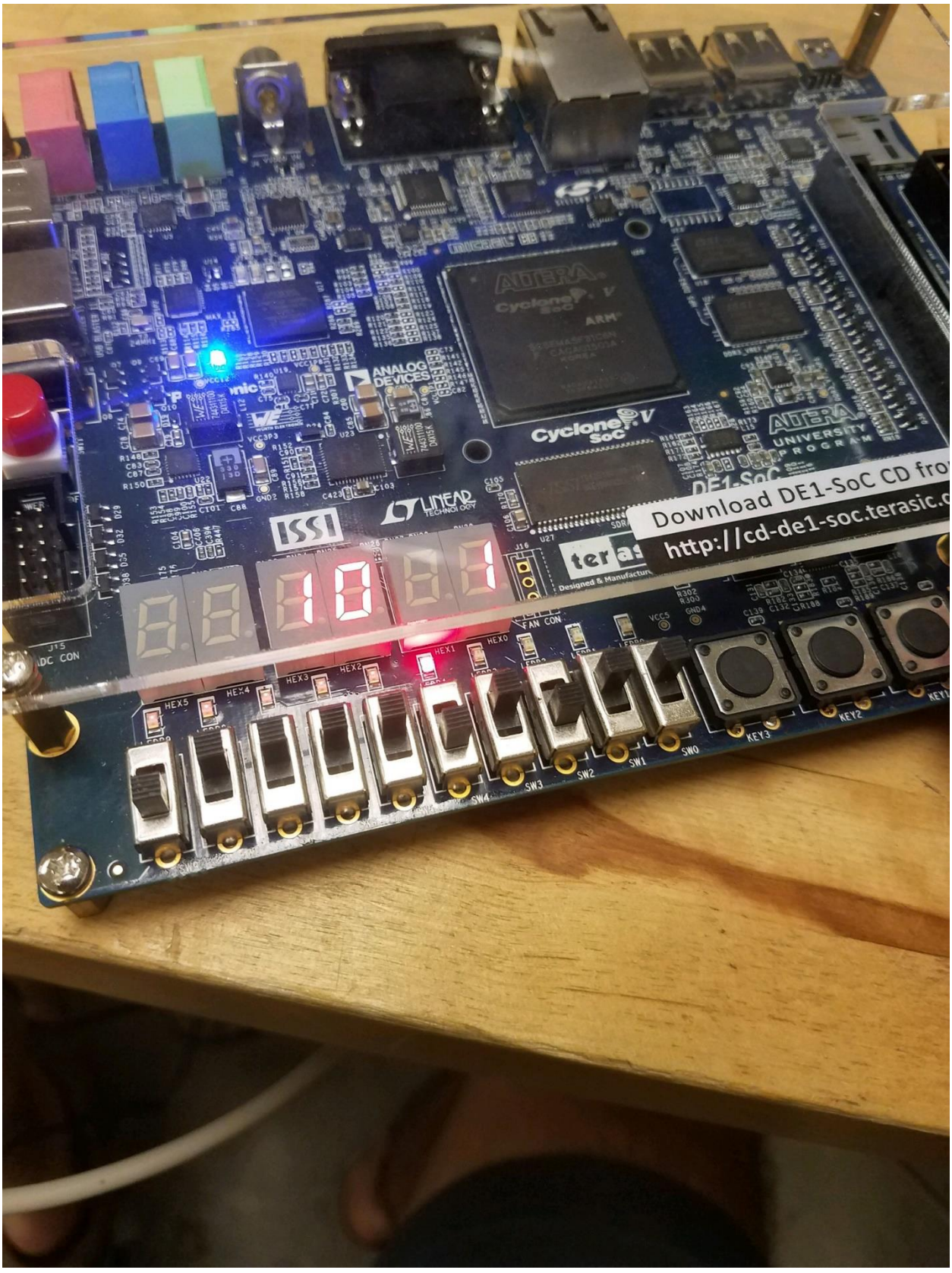
Figure 8 the above image shows the test cases I used for each Opcode to test the vhdl code on the FPGA board. The pictures below match the outputs expected. However in some instances of subtraction and logical operations, the Cout shows 1 on the 7-segment display. This can be ignored due to the nature of subtraction being the addition of 2's compliment and unsigned binary and the outputs having M>N so the carry bit can be discarded. For logical operations the Carry Bit is not even used.

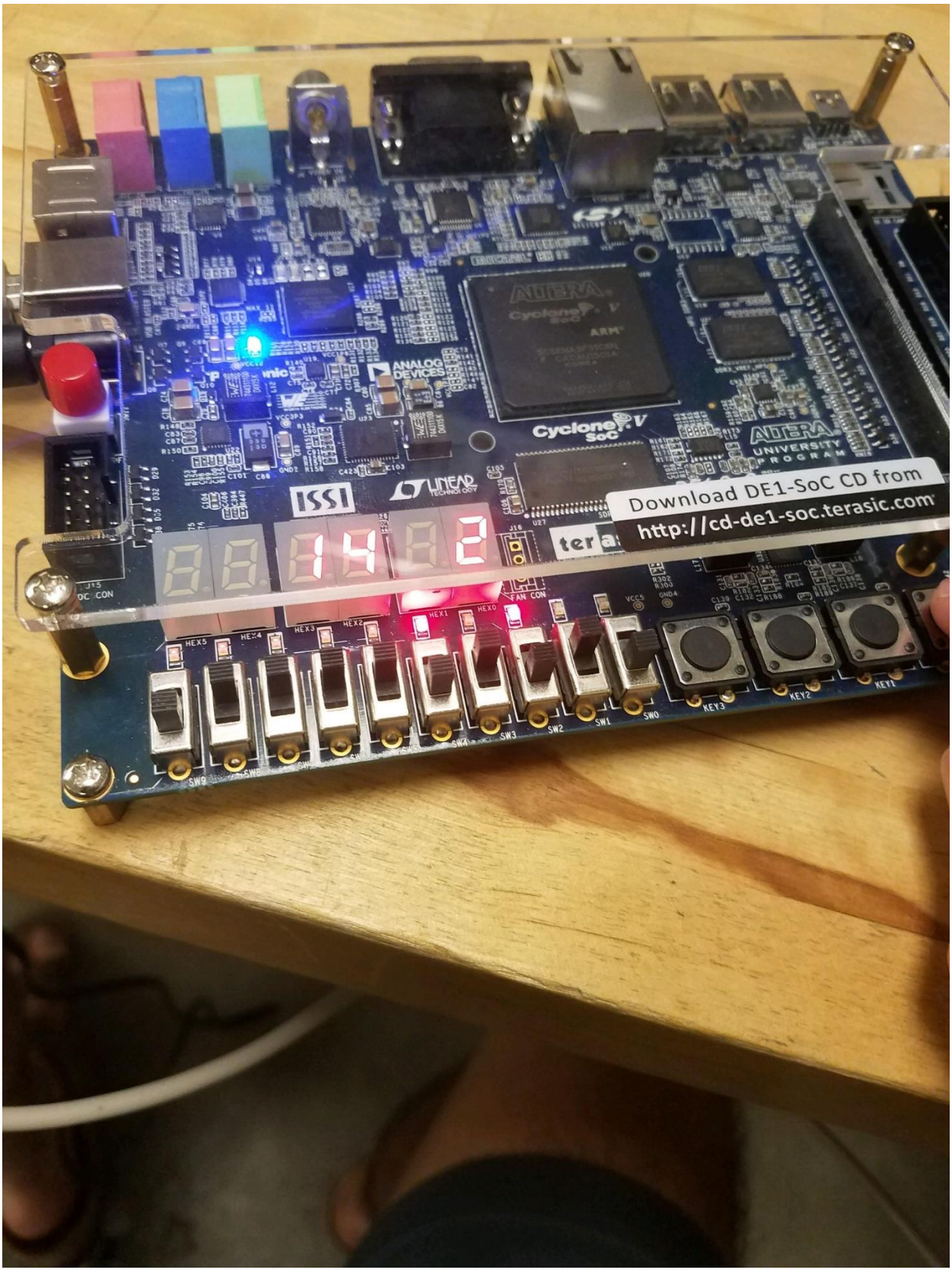
The right-most switch on the FPGA board is mapped to control Carry-In. The four switches to the left of that switch controls the input B and the four switches to the left of B control input A. The MSBs of A and B are on the left and are not inverted. The four right-most LEDs symbolize the output G in binary, while a single LED to the left of these LEDS symbolizes the Carry-out.

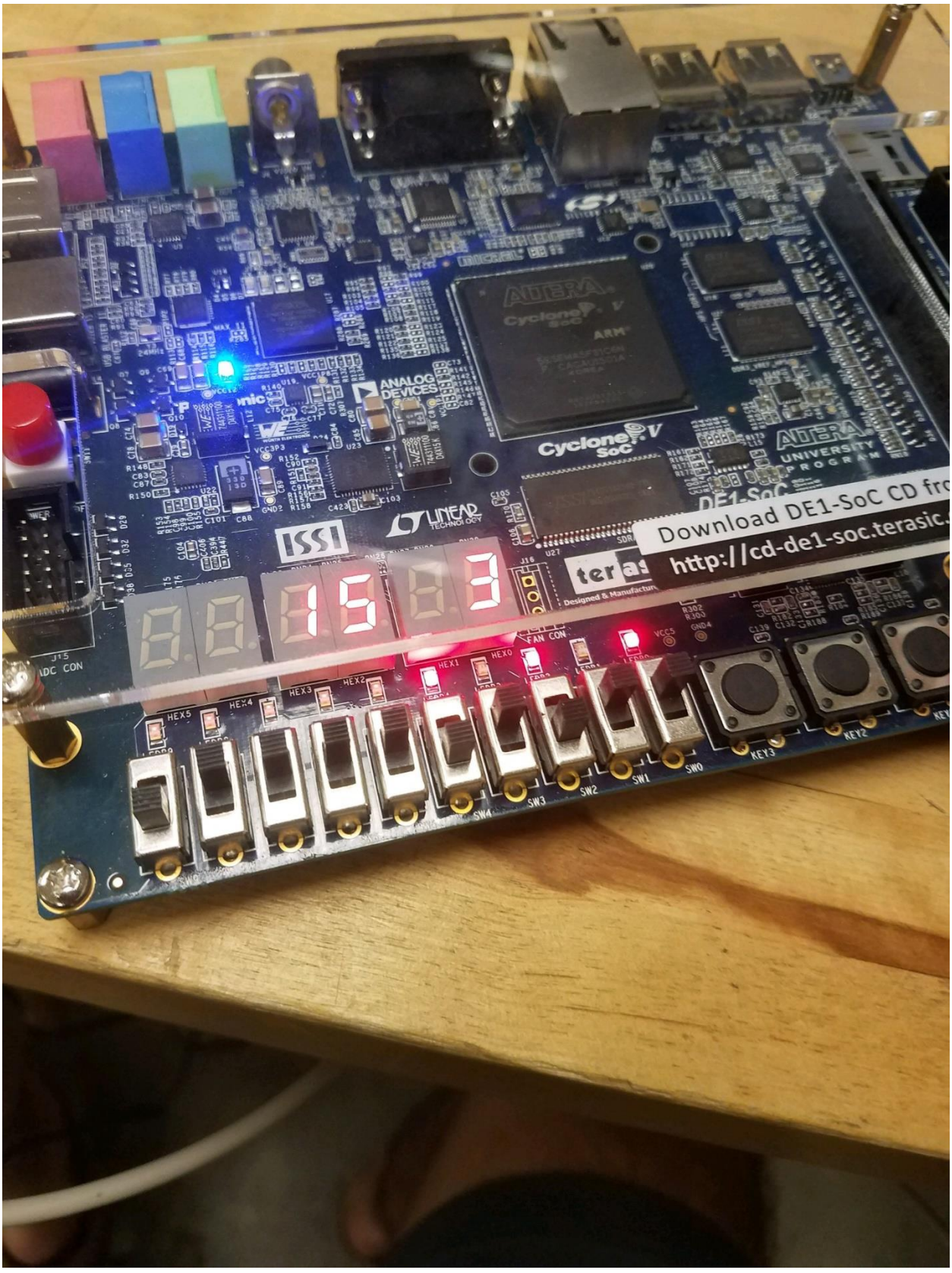
Three Hex 7-segment displays are used here. The right-most Hex bit represents the Opcode, and the left two displays represent Carry-out and G.

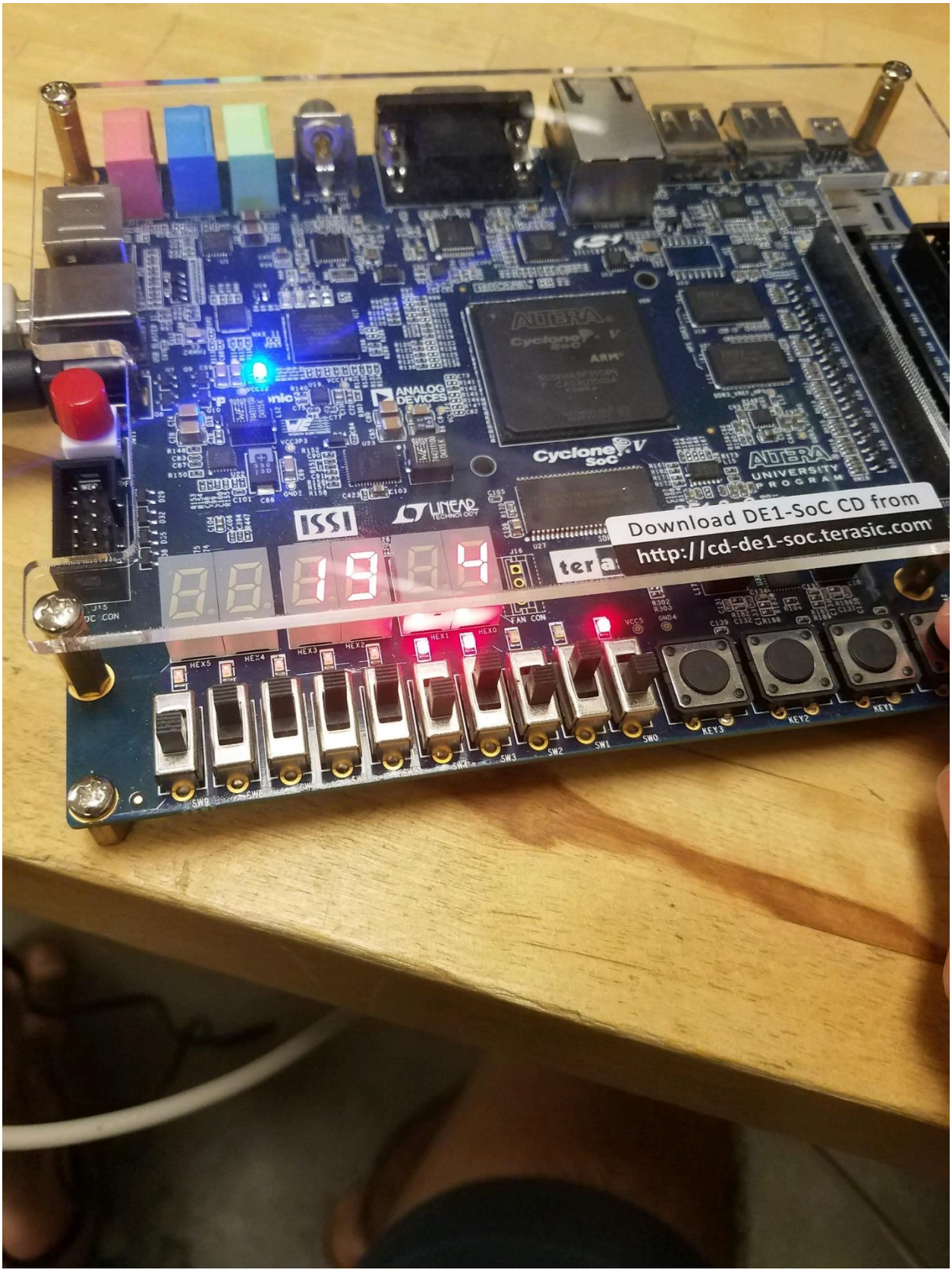
There are four buttons to the right of the switches. The right-most button changes the state of the binary counter adding 1 to the Opcode. The other buttons are left unwired.

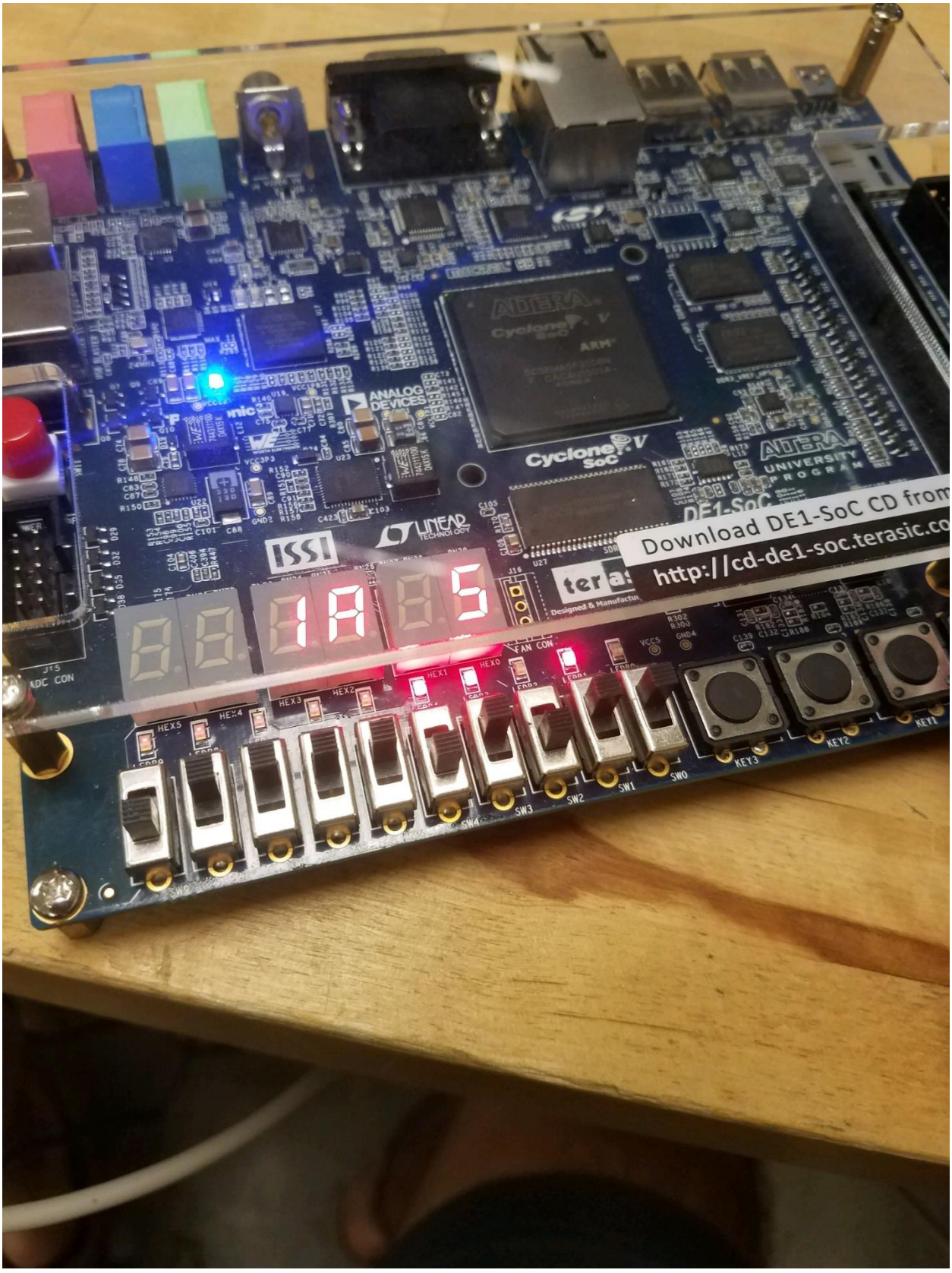

Download DE1-SoC CD from
http://cd-de1-soc.terasic.com
Cyclone V SoC
ALTERA
UNIVERSITY PROGRAM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW9
SW8
SW7
SW6
SW1
SW0
KEY3
KEY2
KEY1
KEY0

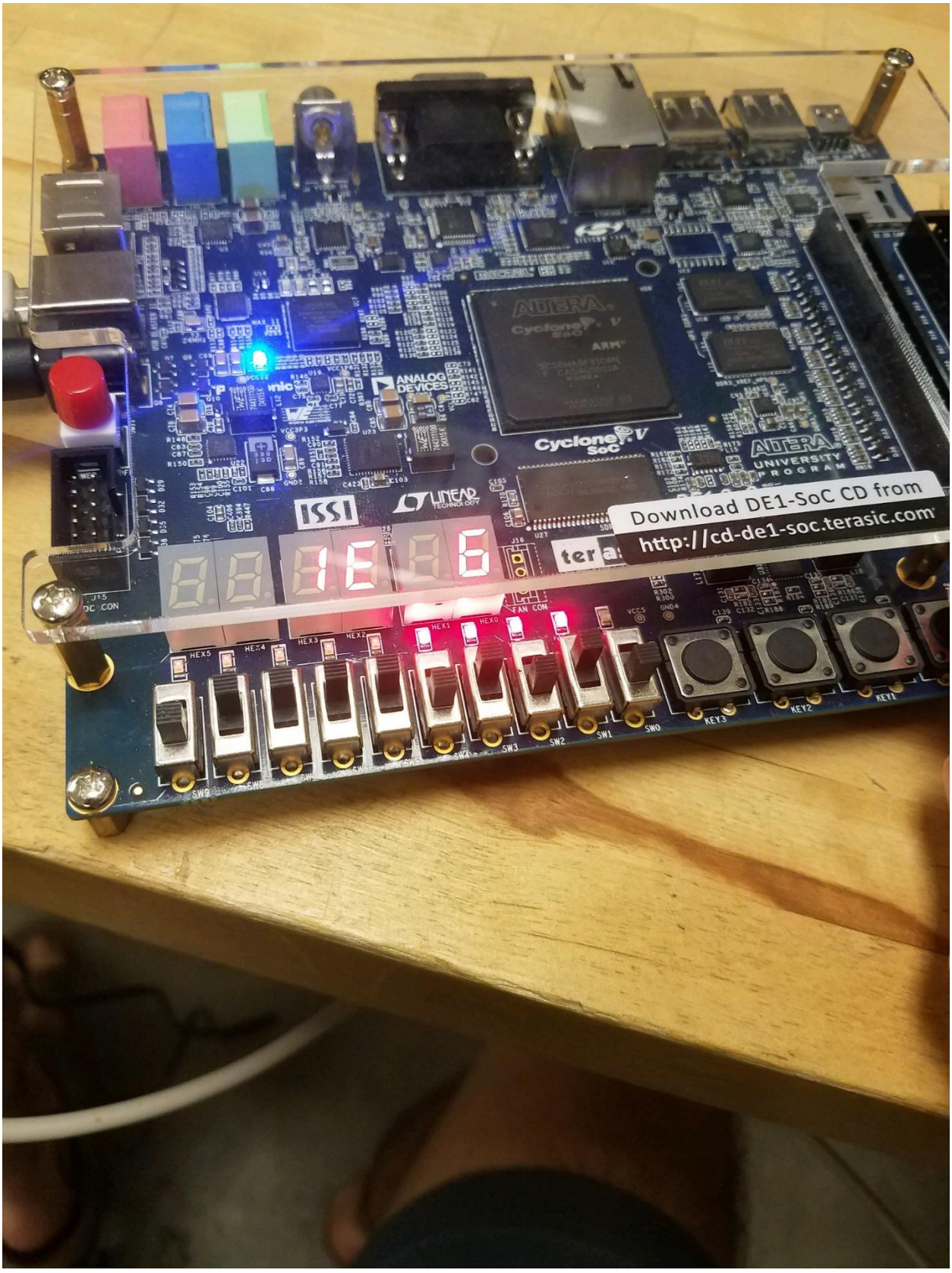

Download DE1-SoC CD fro
http://cd-de1-soc.terasic.c
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
Cyclone V SoC
ALTERA UNIVERSITY PROGRAM
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
FAN CON
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2

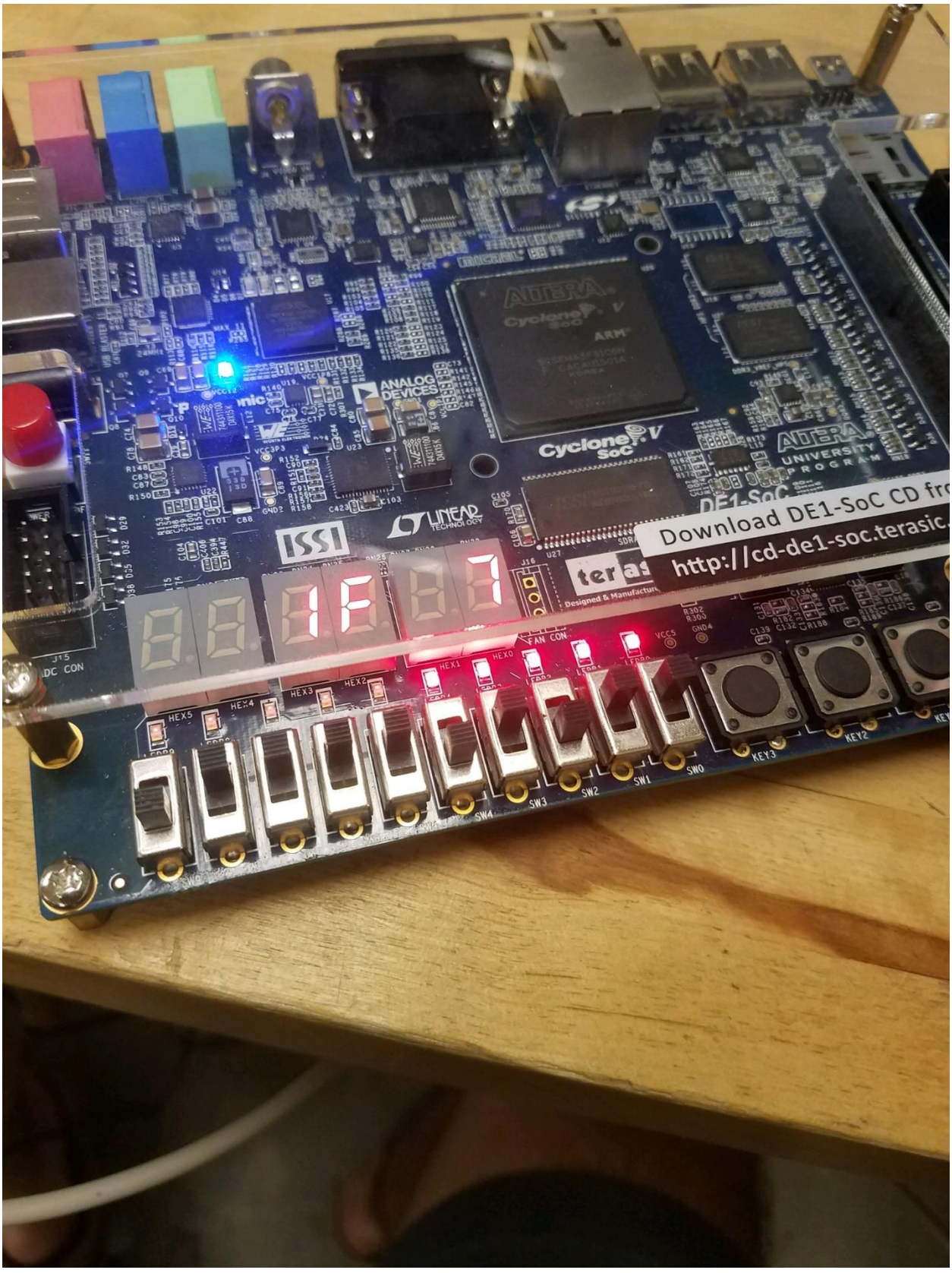

Download DE1-SoC CD from
http://cd-de1-soc.terasic.com
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW9
SW8
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2
KEY1

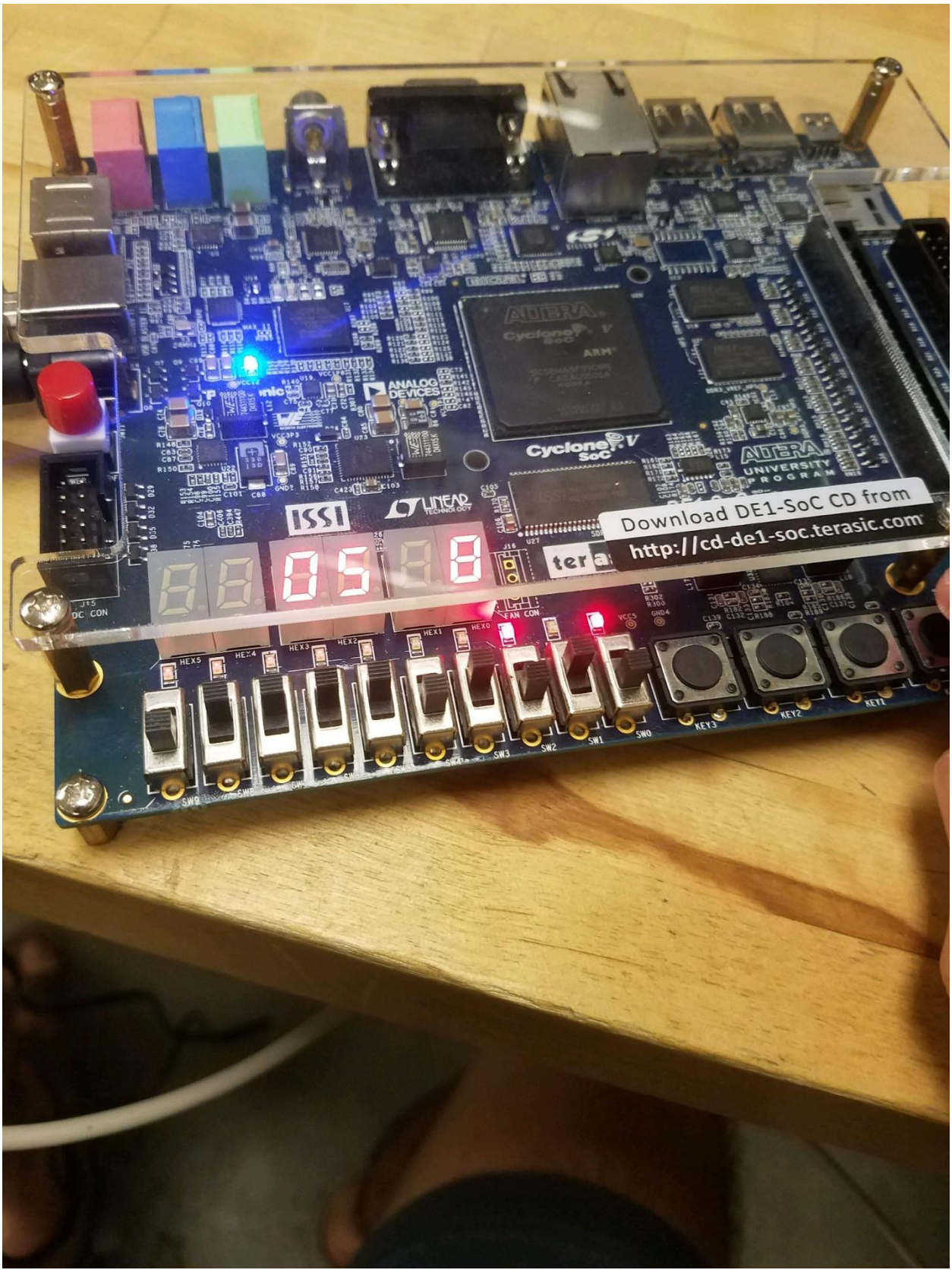

Download DE1-SoC CD fro
http://cd-de1-soc.terasic.
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2

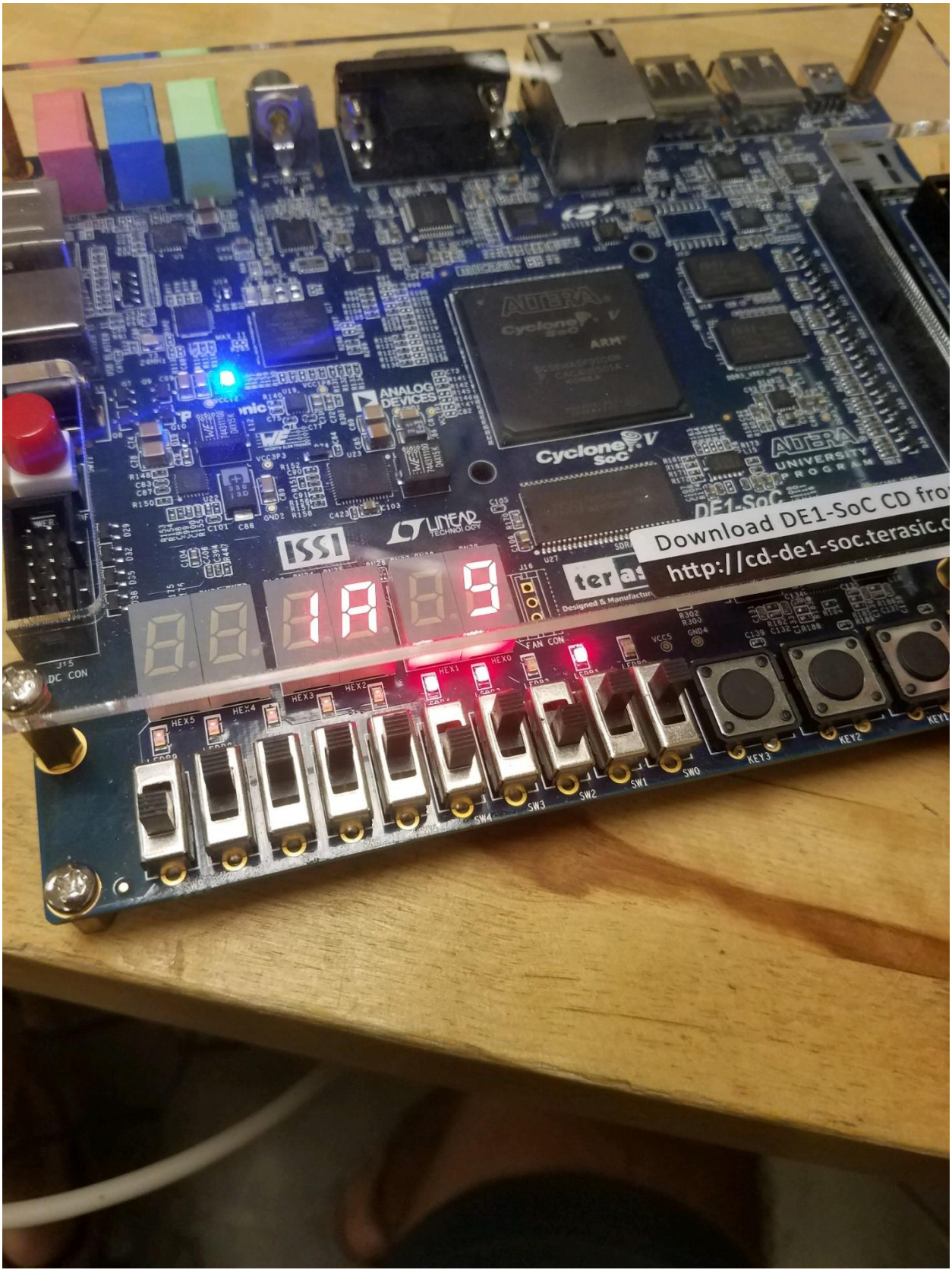

Download DE1-SoC CD from
http://cd-de1-soc.terasic.com
Cyclone V SoC
ALTERA
UNIVERSITY PROGRAM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW9
SW3
SW2
SW1
SW0
KEY3
KEY2
KEY1
FAN CON

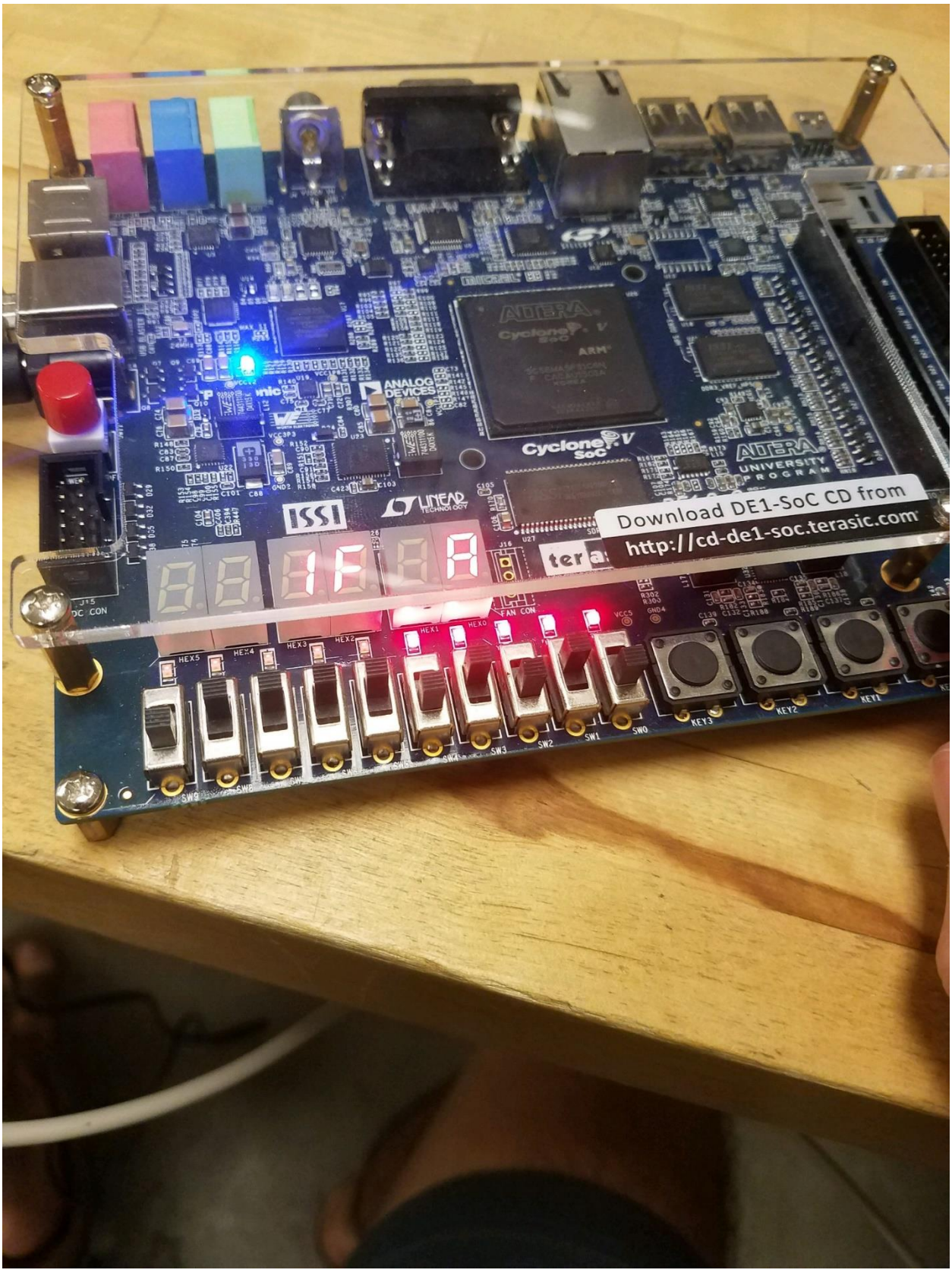

Download DE1-SoC CD from
http://cd-de1-soc.terasic.co
Cyclone V SoC
ARM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
ADC CON
FAN CON
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2
KEY1

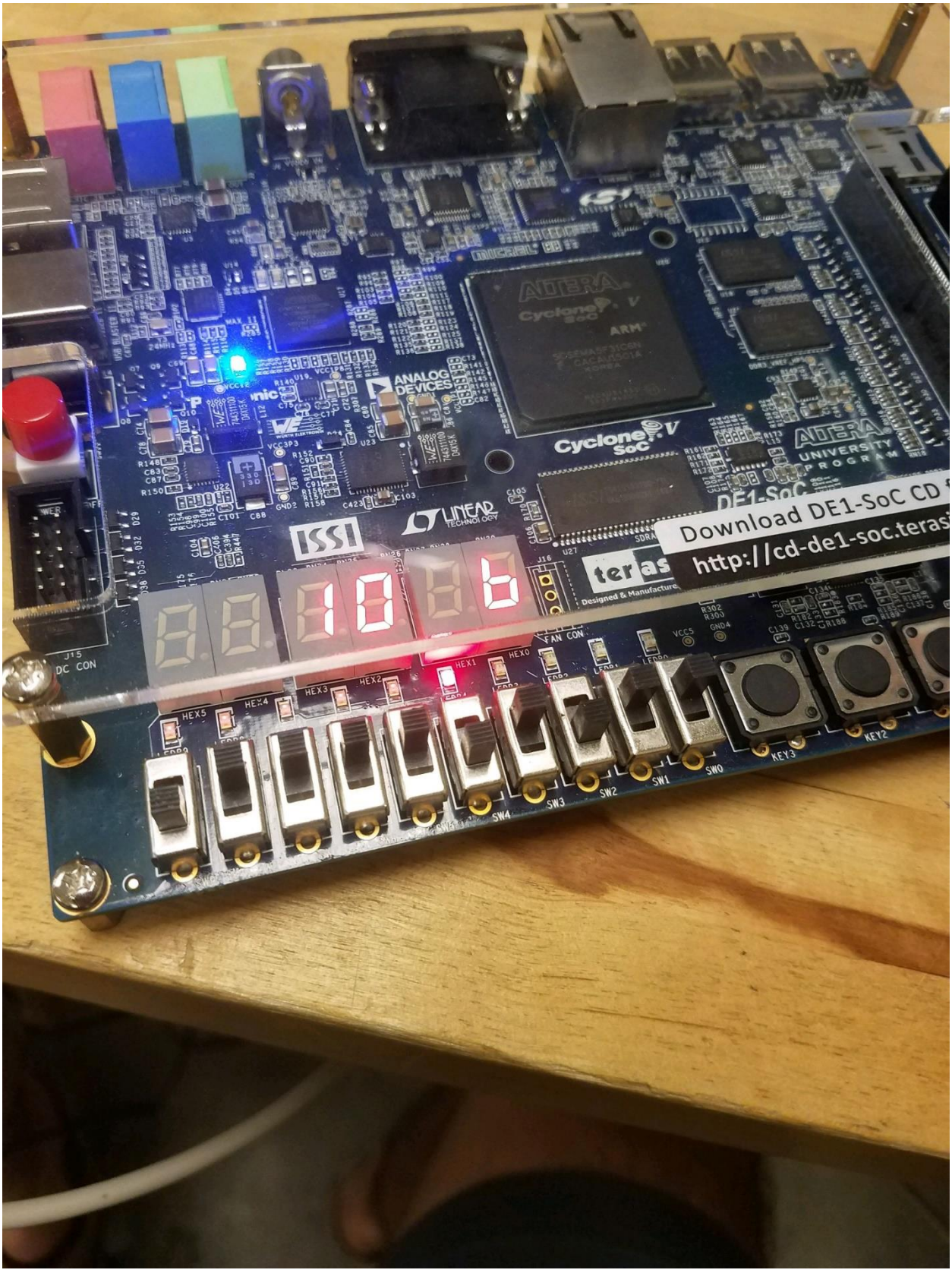

Download DE1-SoC CD from
http://cd-de1-soc.terasic.com
ALTERA
Cyclone V SoC
ARM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
ALTERA UNIVERSITY PROGRAM
FAN_CON
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW9
SW3
SW2
SW1
SW0
KEY3
KEY2
KEY1

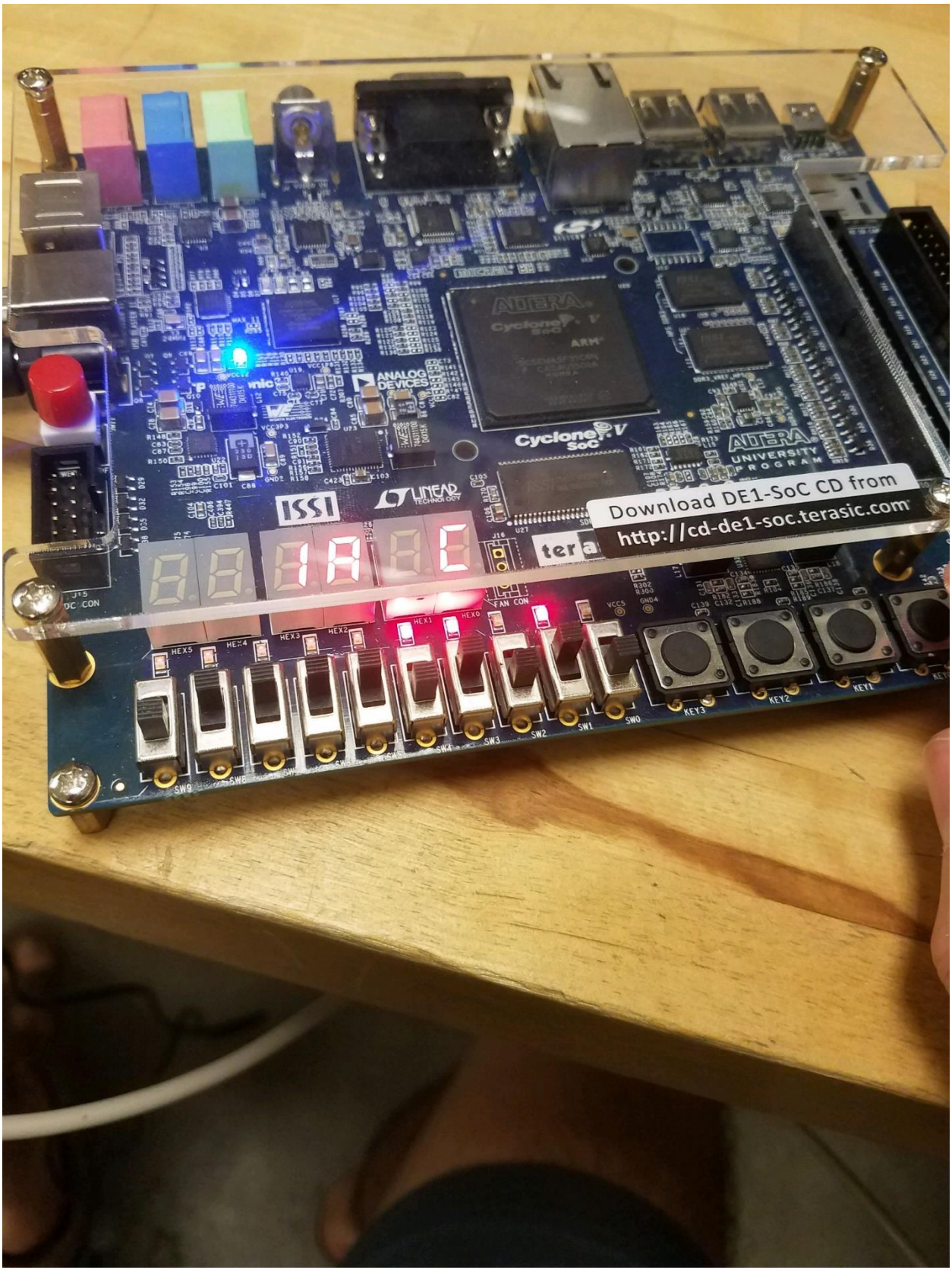

ALTERA
Cyclone V SoC
ARM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
Cyclone V SoC
ALTERA UNIVERSITY PROGRAM
Download DE1-SoC CD fr
http://cd-de1-soc.terasic.
Designed & Manufactu
FAN CON.
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2

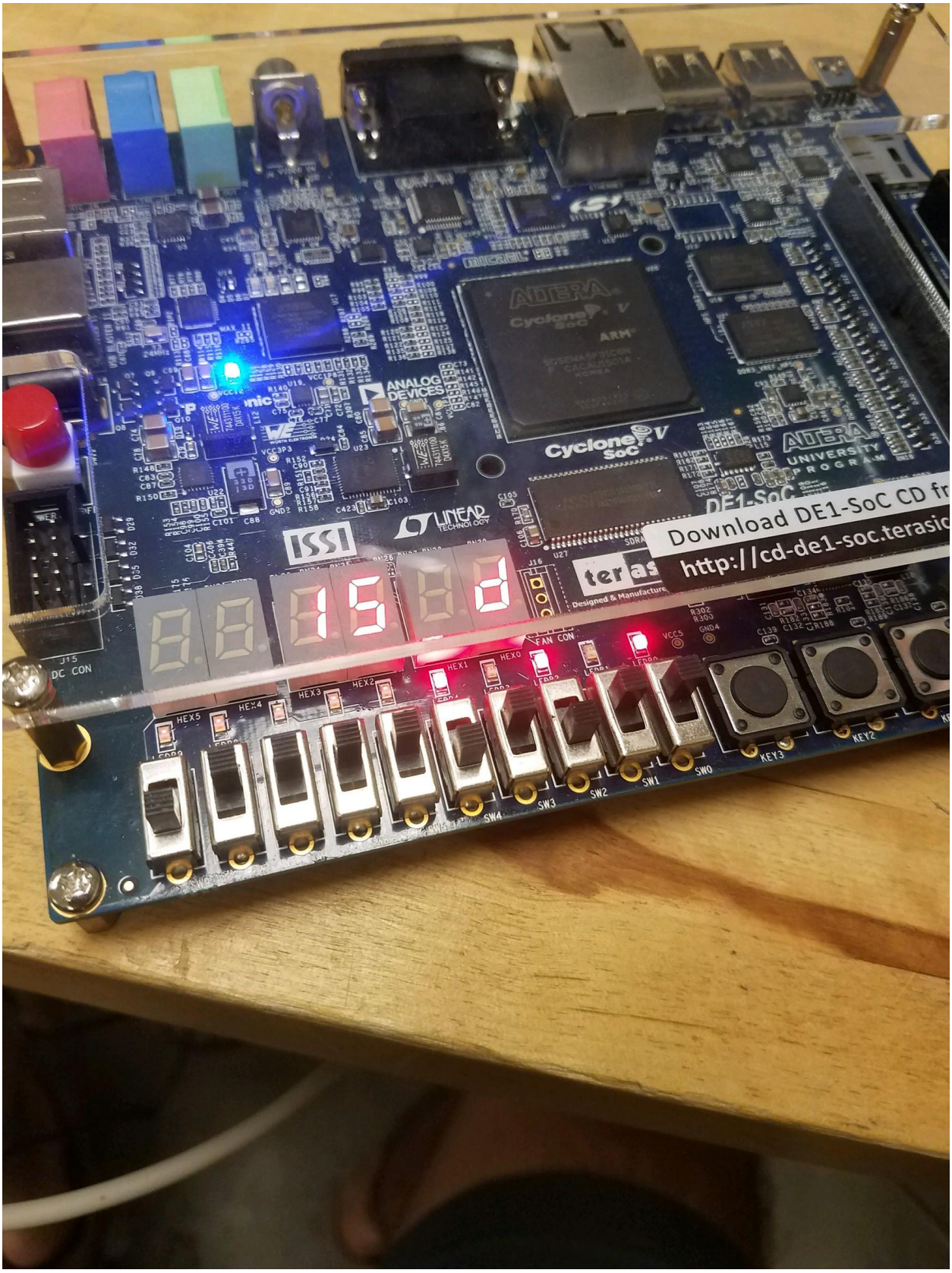

Download DE1-SoC CD from
http://cd-de1-soc.terasic.com
ALTERA
Cyclone V
SoC
ARM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
UNIVERSITY PROGRAM
DC CON
FAN CON
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW9
SW3
SW2
SW1
SW0
KEY3
KEY2
KEY1

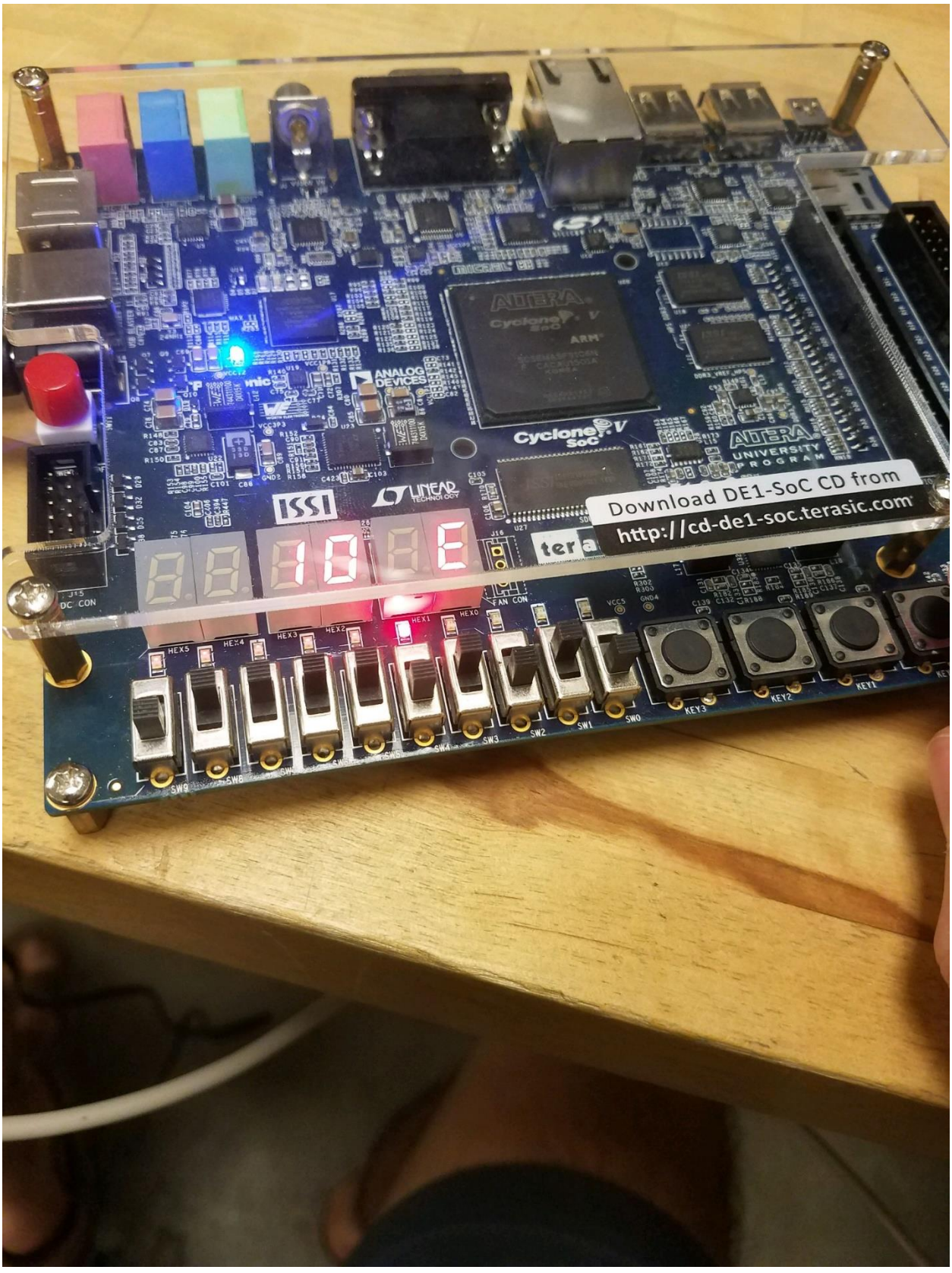

Cyclone V SoC
ARM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
DE1-SoC
Download DE1-SoC CD fro
http://cd-de1-soc.terasic.c
1A9
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
FAN CON
DC CON
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2

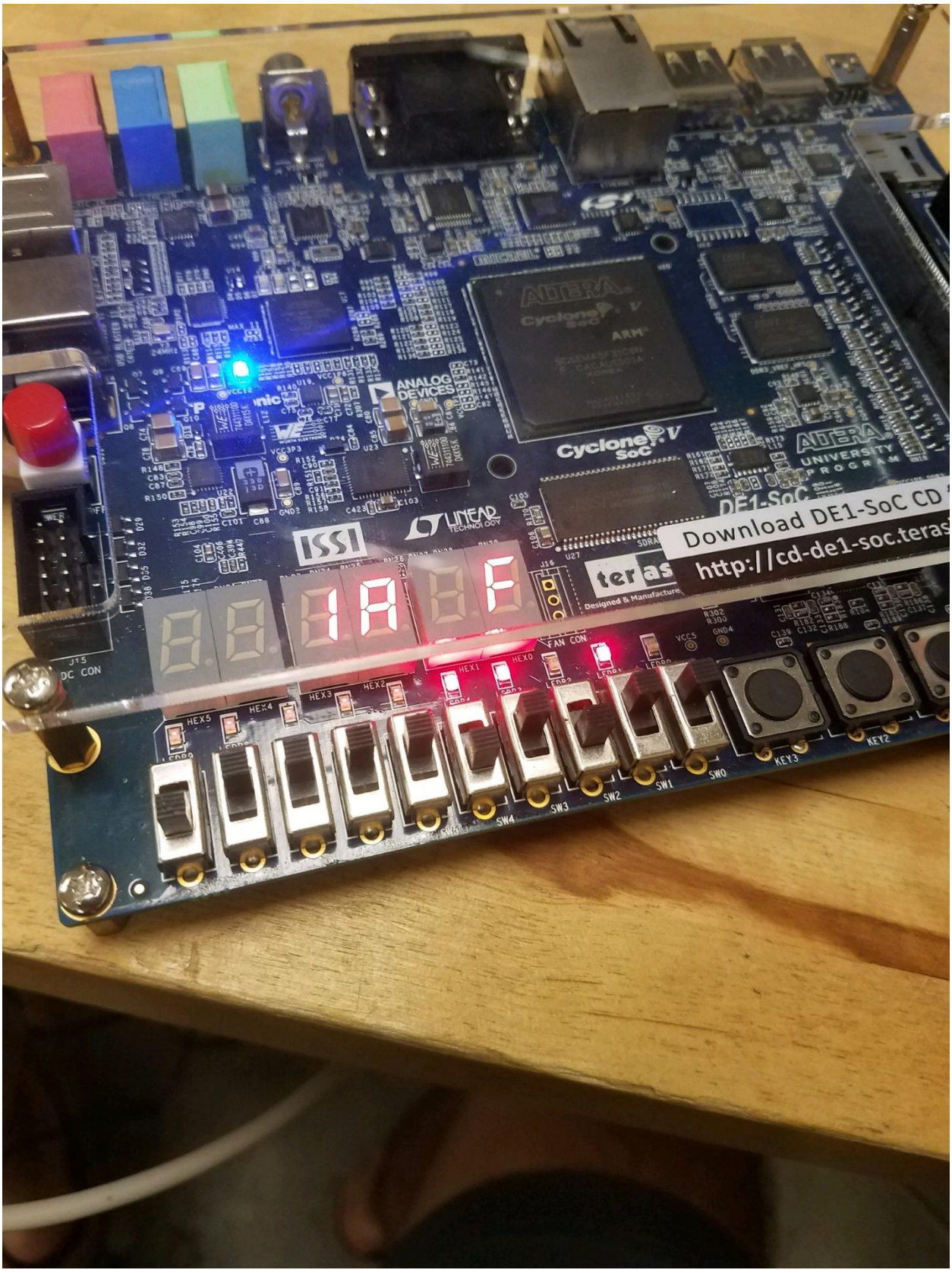

Download DE1-SoC CD from
http://cd-de1-soc.terasic.com
ALTERA
Cyclone V
SoC
ARM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
ALTERA
UNIVERSITY
PROGRAM
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
FAN_CON
SW3
SW2
SW1
SW0
KEY3
KEY2


Cyclone V SoC
ARM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
ALTERA UNIVERSITY PROGRAM
Download DE1-SoC CD
http://cd-de1-soc.teras
Designed & Manufacture
DC CON
FAN CON
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2


Download DE1-SoC CD from
http://cd-de1-soc.terasic.com
ALTERA
Cyclone V
SoC
ARM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
UNIVERSITY PROGRAM
FAN_CON
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW9
SW3
SW2
SW1
SW0
KEY3
KEY2


Download DE1-SoC CD fr
http://cd-de1-soc.terasic
Cyclone V SoC
ARM
ALTERA
UNIVERSITY PROGRAM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
J15
DC CON
FAN CON
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2


Download DE1-SoC CD from
http://cd-de1-soc.terasic.com
Cyclone V SoC
ALTERA
UNIVERSITY PROGRAM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
SW9
SW8
SW3
SW2
SW1
SW0
KEY3
KEY2
KEY1
DC CON
FAN CON


Cyclone V SoC
ALTERA
ARM
ANALOG DEVICES
LINEAR TECHNOLOGY
ISSI
Download DE1-SoC CD
http://cd-de1-soc.teras
Designed & Manufacture
1A
F
HEX5
HEX4
HEX3
HEX2
HEX1
HEX0
FAN CON
DC CON
SW5
SW4
SW3
SW2
SW1
SW0
KEY3
KEY2