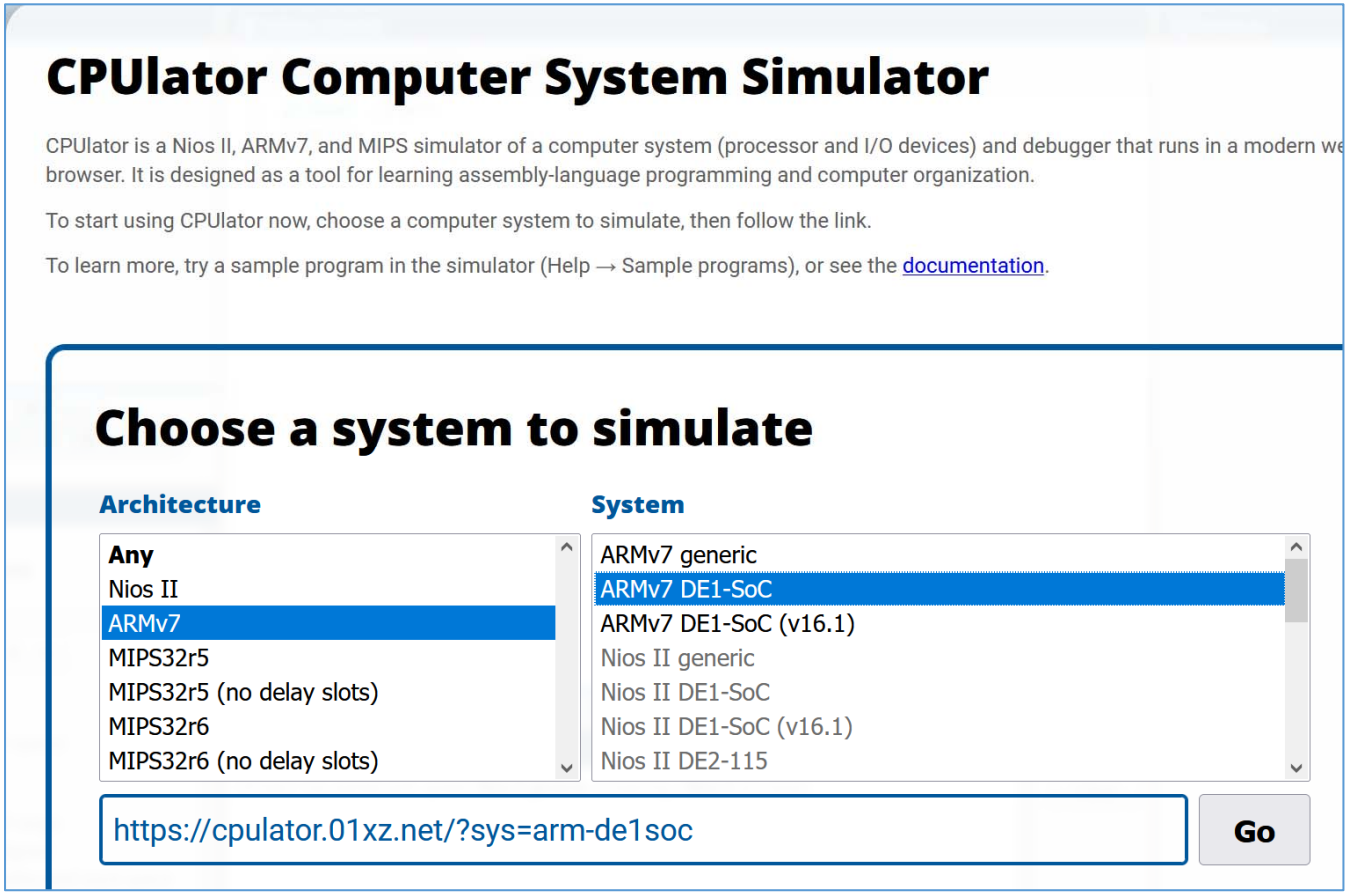
Chapter 8: ARM Assembly Input/Output with DE1-SoC

Selection change in CPUlator -- We will study I/O using the virtual DE1-SoC

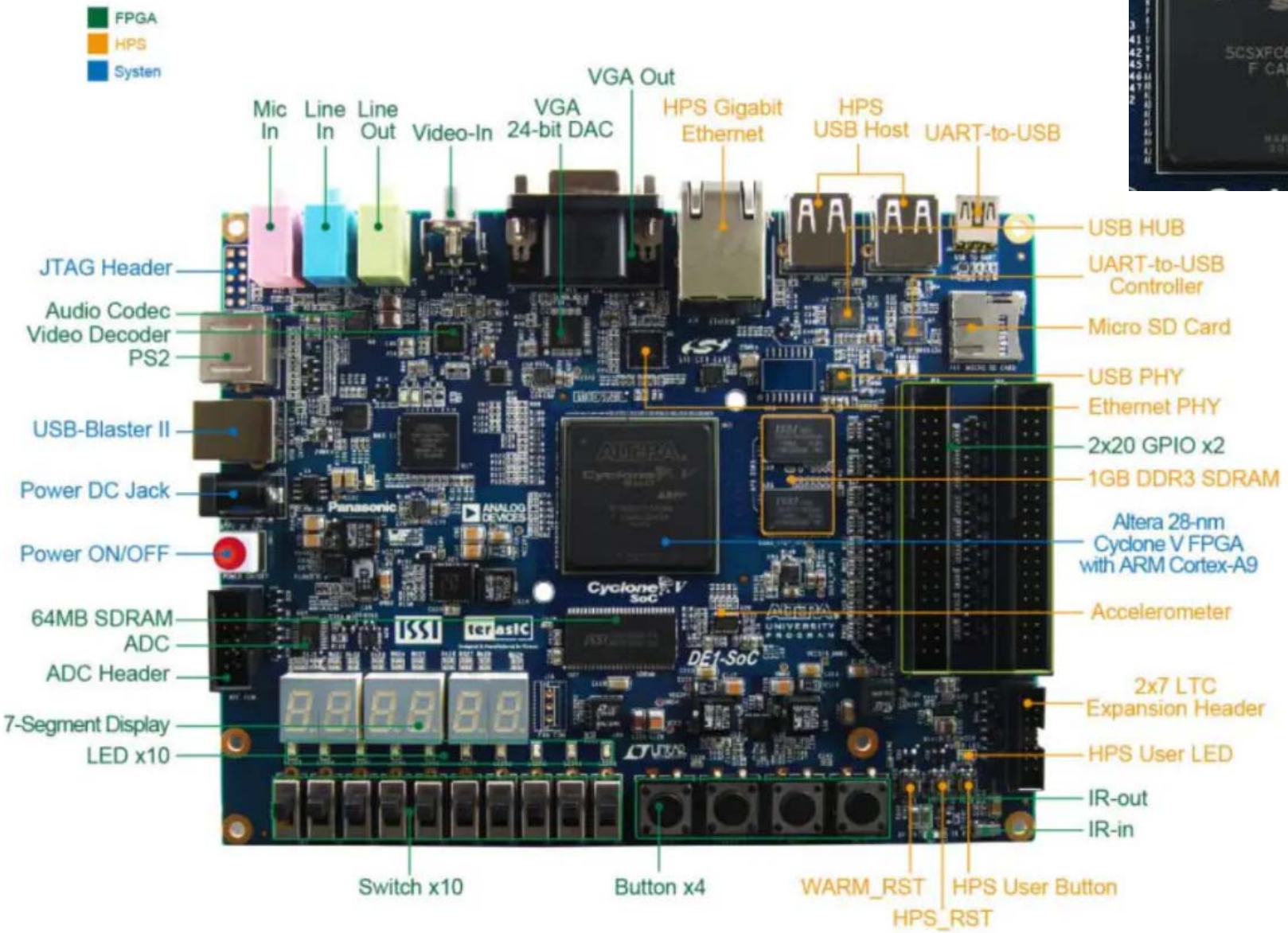
Architecture: ARMv7 (the same)

System: ARMv7 DE1-SoC (New)

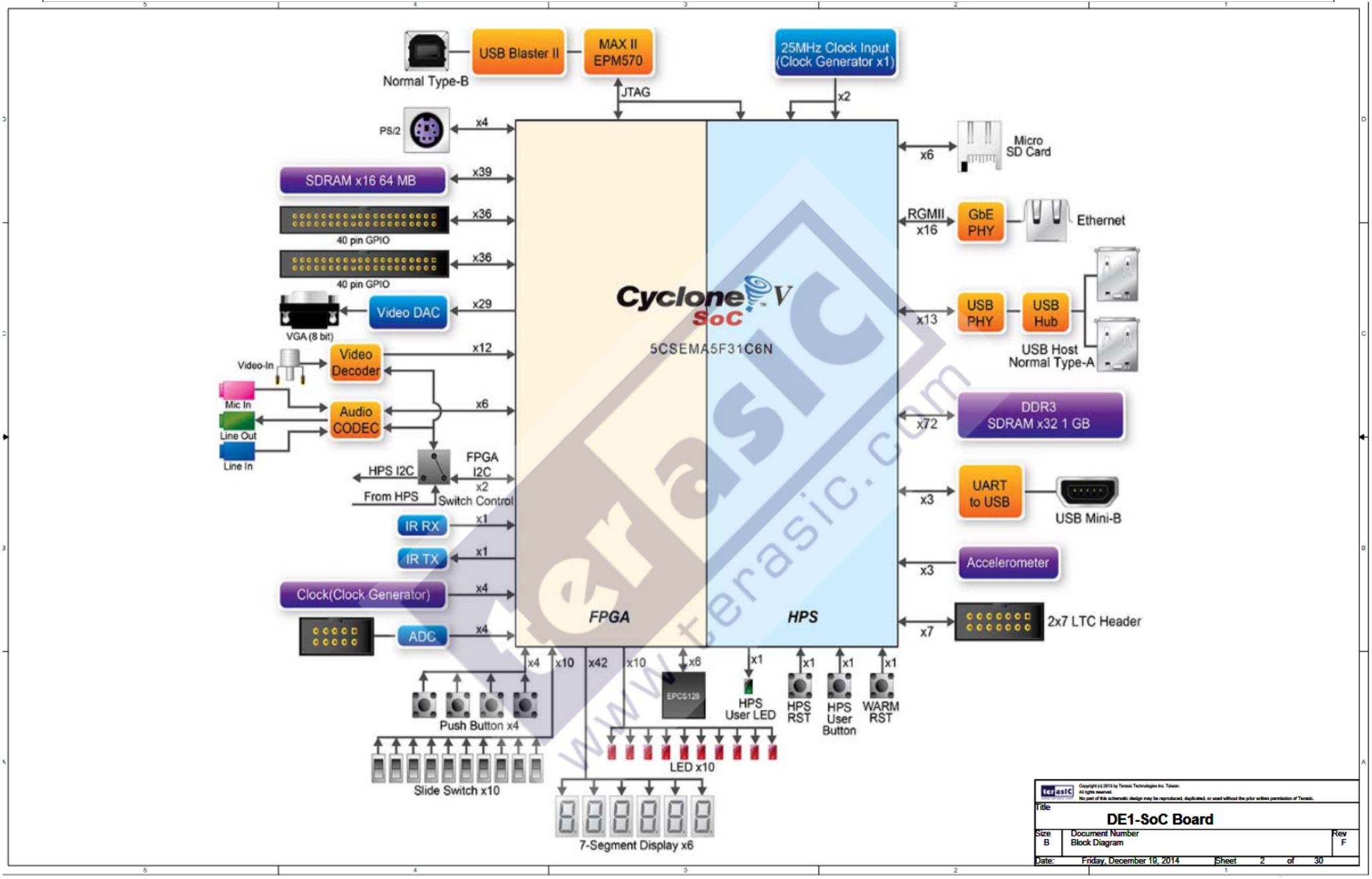


DE1-SoC

Altera's SoC integrates an ARM-based hard-processor system (HPS)



DE1-SoC Schematic Diagram

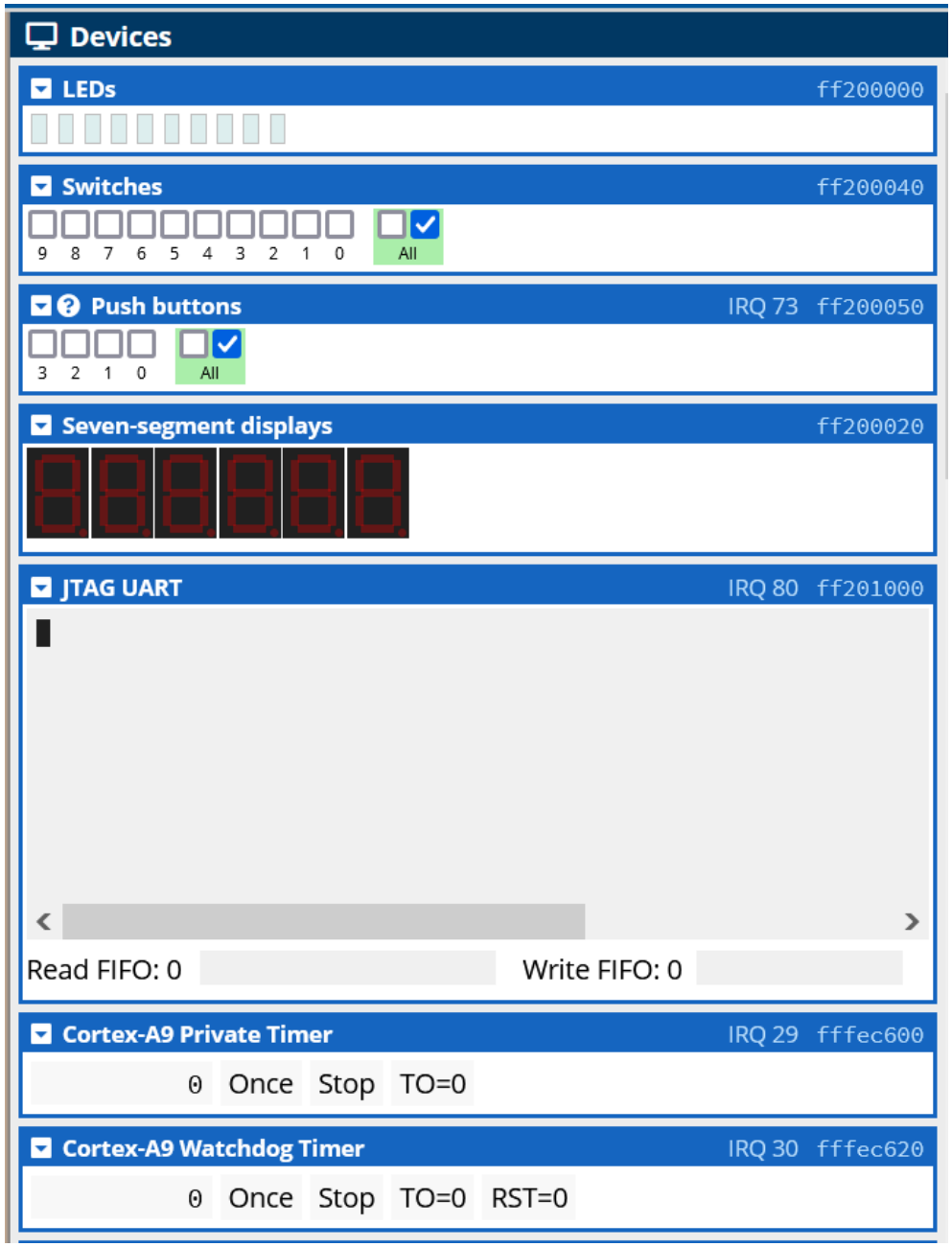


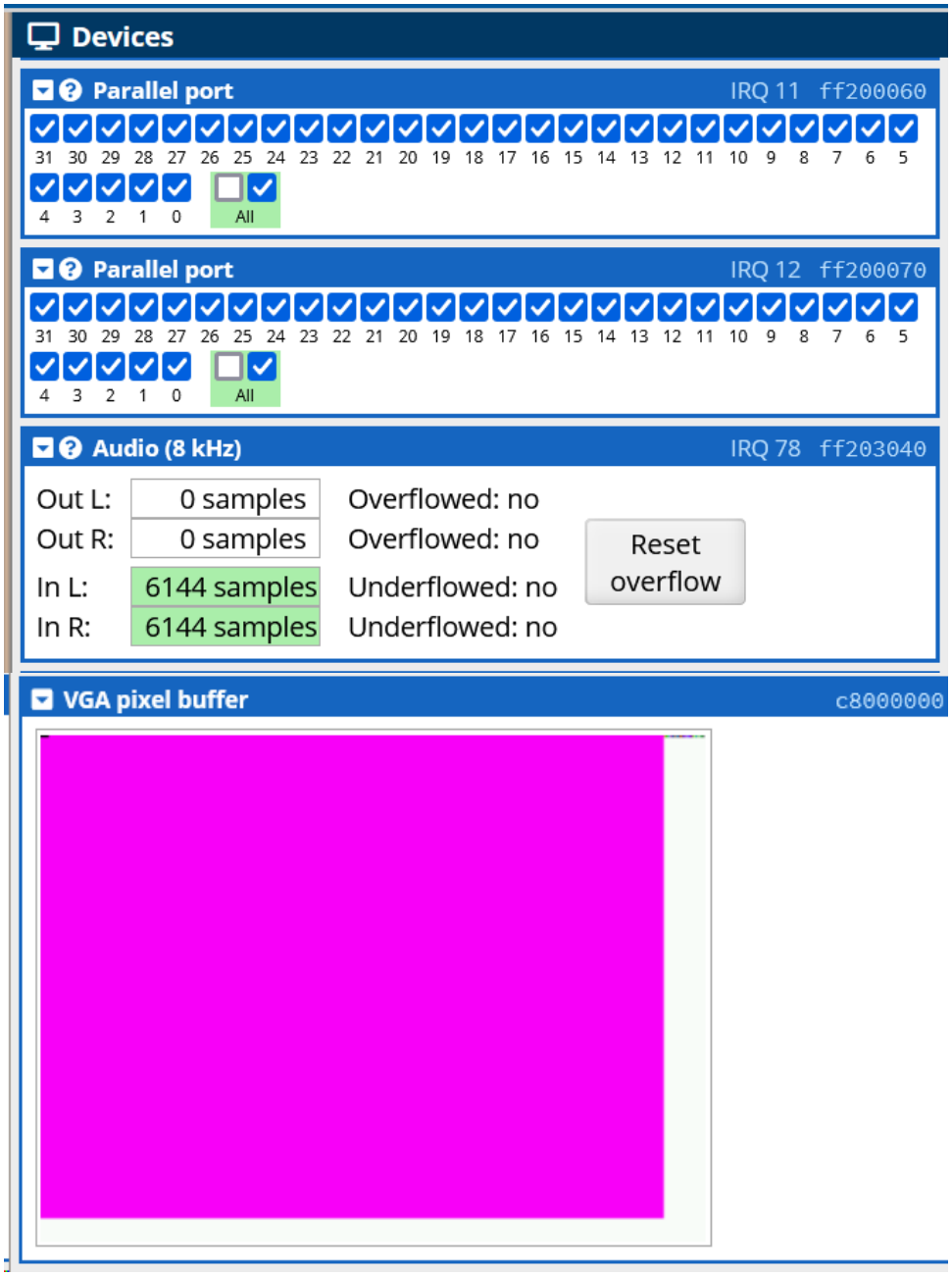
Title: **DE1-SoC Board**

Size	Document Number	Rev
B	Block Diagram	F

Date: Friday, December 19, 2014 Sheet 2 of 30

DE1-SoC Devices in CUPlator





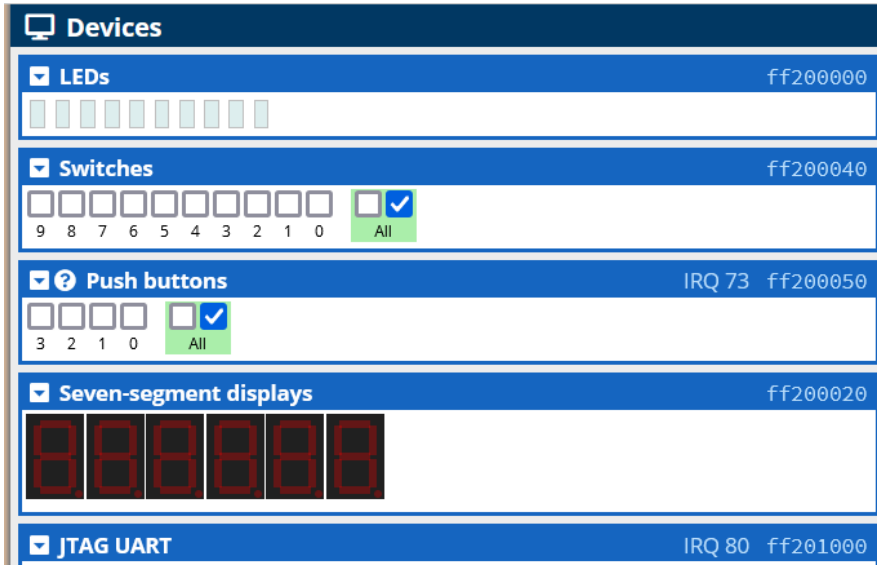
DE1-SoC Memory Map

Base Address	End Address	I/O Peripheral
0x00000000	0x3FFFFFFF	DDR3 Memory
0xFFFF0000	0xFFFFFFFF	A9 On-chip Memory
0xC0000000	0xC3FFFFFF	SDRAM
0xC8000000	0xC803FFFF	FPGA On-chip Memory
0xC9000000	0xC9001FFF	FPGA On-chip Memory Character Buffer
0xFF200000	0xFF20000F	Red LEDs
0xFF200020	0xFF20002F	7-segment HEX3–HEX0 Displays
0xFF200030	0xFF20003F	7-segment HEX5–HEX4 Displays
0xFF200040	0xFF20004F	Slider Switches
0xFF200050	0xFF20005F	Pushbutton KEYs
0xFF200060	0xFF20006F	JP1 Expansion
0xFF200070	0xFF20007F	JP2 Expansion
0xFF200100	0xFF200107	PS/2
0xFF200108	0xFF20010F	PS/2 Dual
0xFF201000	0xFF201007	JTAG UART
0xFF201008	0xFF20100F	Second JTAG UART
0xFF201020	0xFF201027	Infrared (IrDA)
0xFF202000	0xFF20201F	Interval Timer
0xFF202020	0xFF20202F	Second Interval Timer
0xFF203000	0xFF20301F	Audio/video Configuration
0xFF203020	0xFF20302F	Pixel Buffer Control
0xFF203030	0xFF203037	Character Buffer Control
0xFF203040	0xFF20304F	Audio
0xFF203060	0xFF203070	Video-in
0xFF203020	0xFF20302F	Pixel Buffer Control
0xFF203030	0xFF203037	Character Buffer Control
0xFF203040	0xFF20304F	Audio
0xFF203060	0xFF203070	Video-in
0xFF204000	0xFF20401F	ADC
0xFF709000	0xFF709063	HPS GPIO1
0xFFC04000	0xFFC040FC	HPS I2C0
0xFFC08000	0xFFC08013	HPS Timer0
0xFFC09000	0xFFC09013	HPS Timer1
0xFFD00000	0xFFD00013	HPS Timer2
0xFFD01000	0xFFD01013	HPS Timer3
0xFFD0501C	0xFFD0501F	FPGA Bridge
0xFFFEC100	0xFFFEC1FC	GIC CPU Interface
0xFFFED000	0xFFFEDFFC	GIC Distributor Interface
0xFFFEC600	0xFFFEC60F	ARM A9 Private Timer

DE1-SoC LED blinking

- LEDs --- 10 of them
- Address: 0xFF200000

Device	Red LEDs
Configuration	32-bit registers
Input/Output	Output only
Address Base	red LEDs: 0xFF200000
Address Map	base - each map directly to the register



- Example

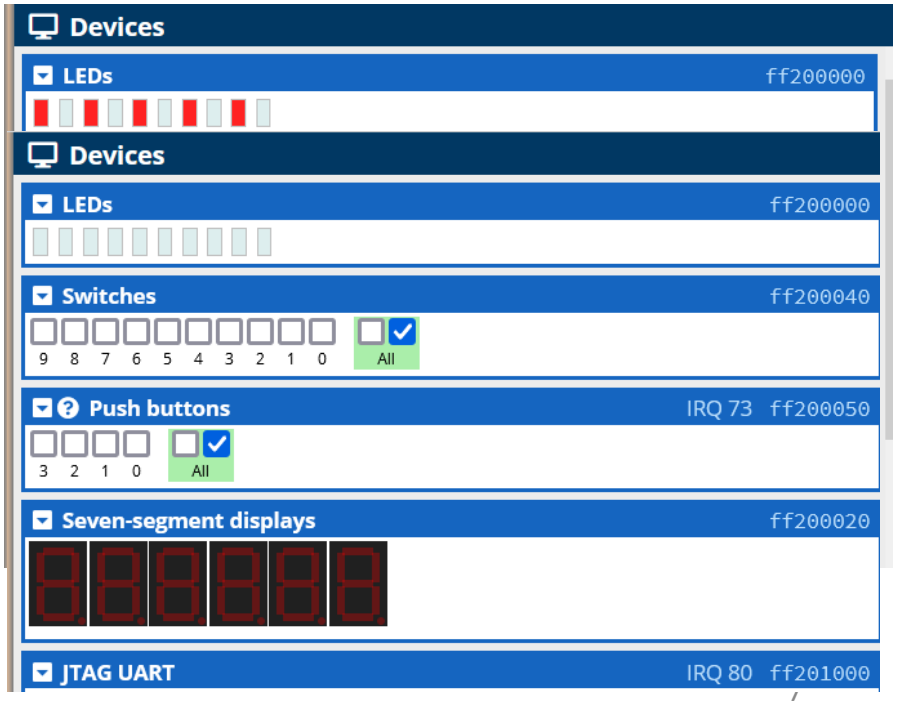
```
.equ LED_REG,0xFF200000

    LDR r0, LED_REG
    LDR r1, =0X0000001  //Turn on the right-most LED
    STR r1, [r0]
```

DE1-SoC LED blinking

- LEDs --- 10 of them
- Address: 0xFF200000

r0	ff200000	
r1	000002aa	
r2	000002aa	
r3	00000000	
r4	00000000	
r5	00000000	
r6	00000000	
r7	00000000	
r8	000cd94f	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	0000001c	
pc	00000024	
cpsr	200001d3	NZCVI SVC
spsr	00000000	NZCVI ?



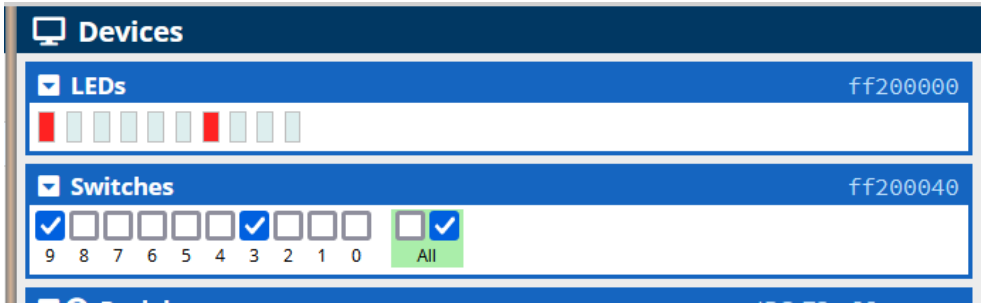
DE1-SoC LED Blinking

- LEDs --- 10 of them
- Address: 0xFF200000
- Q1) Blink only the left-most (LED 9) and the right-most (LED 0) LEDs
- Q2) Blink LEDs 0,2,4,6, and 8.

```
1  //Example8-2.s
2  //10 Red LEDs in 0xFF200000
3  // 2 second delayed blinking
4  .equ LED,0xFF200000
5  .global _start
6  _start:
7
8      LDR r1, =0x00000201  //lower 10 bits
9      //0000 0155 for LEDs 0, 2, 4, 6, 8
10     //0000 01AA for LEDs 1, 3,5, 7, 9
11     //0000 0201 for LEDs 0 and 9
12     LDR r0,=LED
13     STR r1, [r0]
14 again:
15     LDR r2, [r0]  //read
16     EOR r3, r1, r2  //modify
17     STR r3, [r0]  //write
18     BL delay
19     B again
20
21 delay:
22 //manual delay without
23 //using time delay
24     LDR r8, =0x200000 //2 seconds
25 keep:
26     SUBS r8, r8, #1
27     BNE keep
28     BX lr
```

DE1-SoC LED by Switches

Device	Slider Switches
Configuration	One 32-bit register
Input/Output	Input only
Address Base	0xFF200040
Address Map	base - maps directly to the register



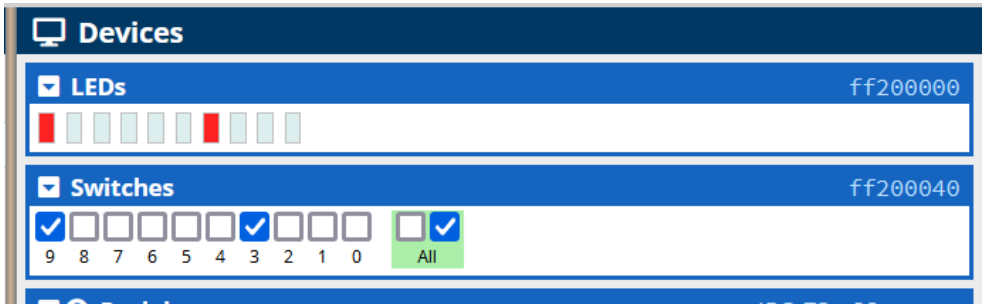
Slider Switch: 0xFF200040
10 switches used
Bits[31:10] are not used

EXAMPLE (reading the switch position: 1/0)

```
.equ  SW_REG, 0xFF200040

      LDR r0, =SW_REG
      LDR r1, [r0] //Read the SW
```

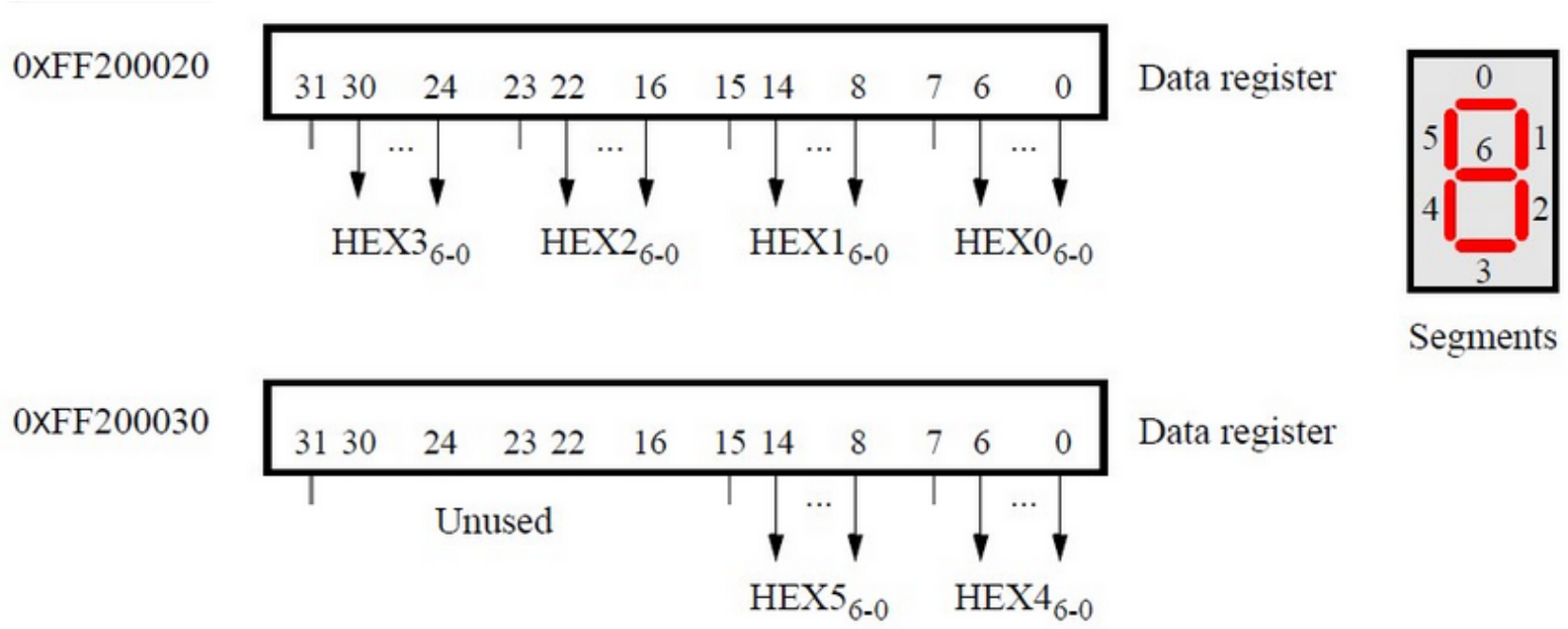
DE1-SoC LED by Switches



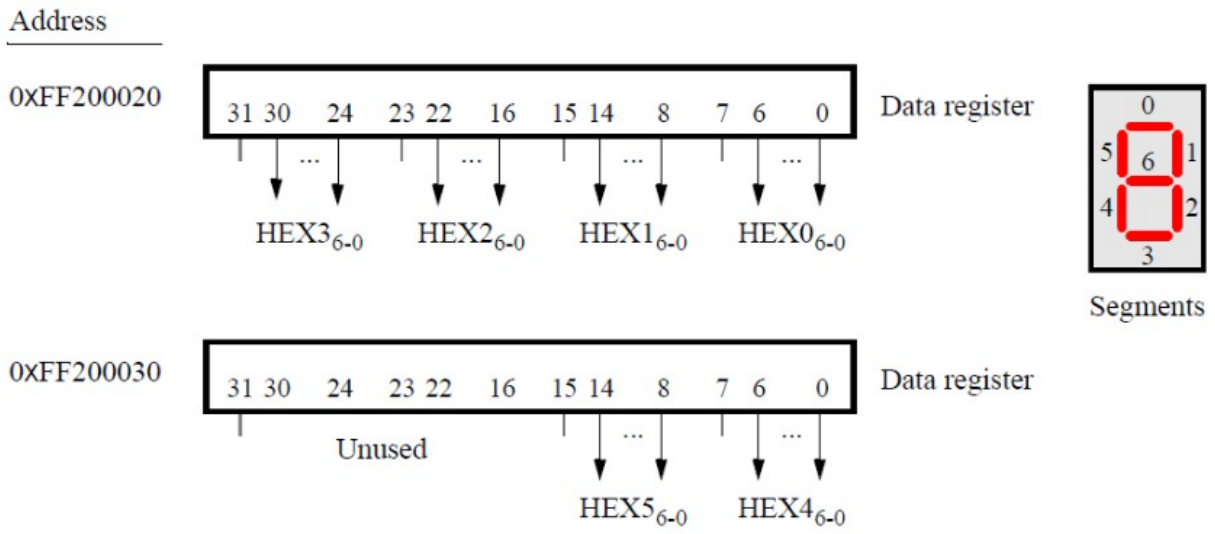
r0	ff200040	
r1	00000208	
r2	ff200000	
r3	00000201	
r4	00000000	
r5	00000000	
r6	00000000	
r7	00000000	
r8	0011ccb5	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000014	
pc	00000020	
cpsr	200001d3	NZCVI SVC
spsr	00000000	NZCVI ?

DE1-SoC 7-Segment Display

- 2 registers with base addresses
- HEX3 – HEX0: 0xFF200020
- HEX5 – HEX4: 0xFF200030

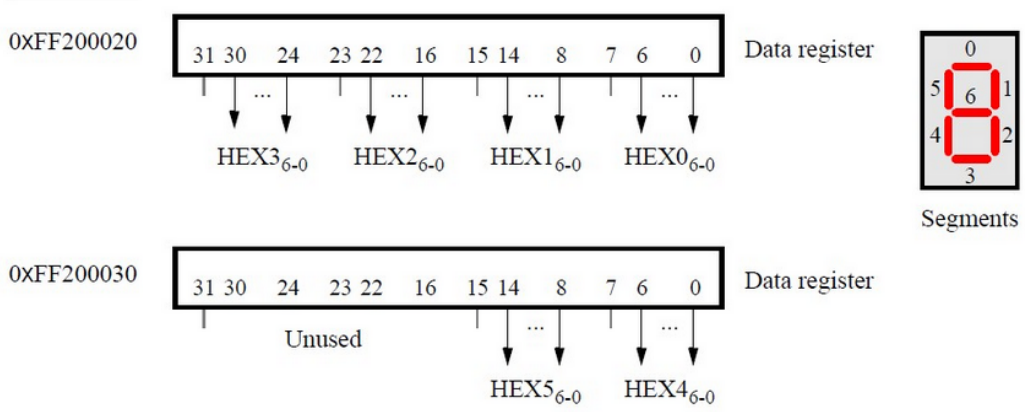


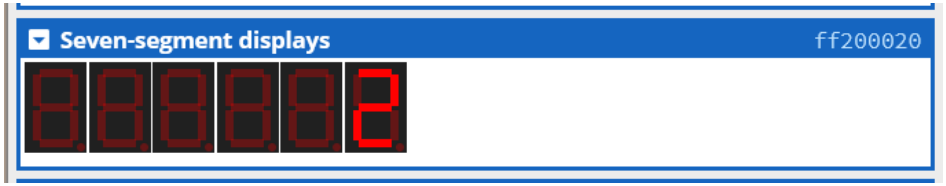
Device	7-segment Display
Configuration	Two 32-bit registers
Input/Output	Output only
Address Base	HEX3 to HEX0: 0xFF200020, HEX5 to HEX4: 0xFF200030
Address Map	0xFF00020: • Bits 6 to 0: $HEX0_{6-0}$ • Bits 14 to 8: $HEX1_{6-0}$ • Bits 22 to 16: $HEX2_{6-0}$ • Bits 30 to 24: $HEX3_{6-0}$ 0x10000030: • Bits 6 to 0: $HEX4_{6-0}$ • Bits 14 to 8: $HEX5_{6-0}$ • Bits 22 to 16: Unused • Bits 30 to 24: Unused



DE1-SoC 7-Segment Display

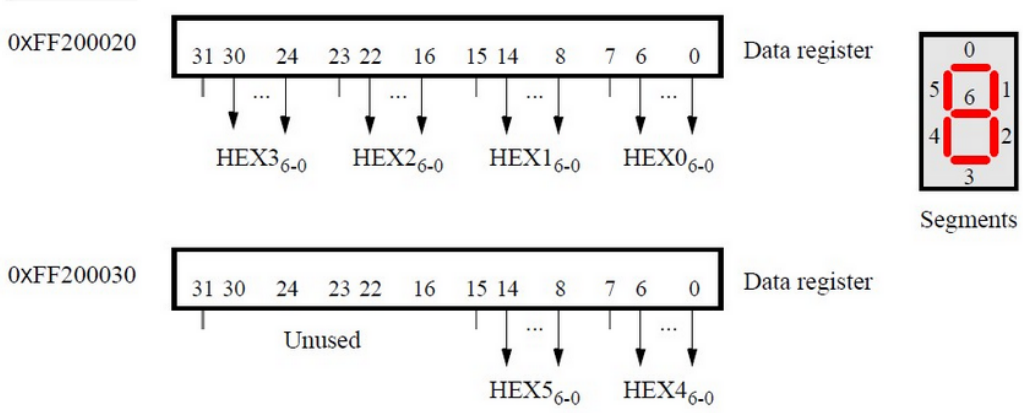
- 2 registers with base addresses
- HEX3 – HEX0: 0xFF200020
- HEX5 – HEX4: 0xFF200030
- **EX) Display “1” and “2” in HEX0 display**





DE1-SoC 7-Segment Display

- **Q) Write a code which Displays “1867” and “2020” alternatingly.**



DE1-SoC (Coding Practice)

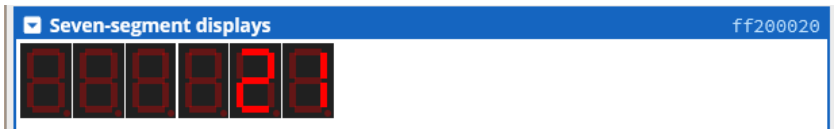
- Q) Write a code which displays (a) the last 6 digits of your ID and (b) your last (or first) name (up to 6 characters), alternatingly, in the 7-segments with 4 second interval.

DE1-SoC (7-segment)

- Q) Write a code which displays 0 – 9 on the right-most 7-segment with 1 second time delay.

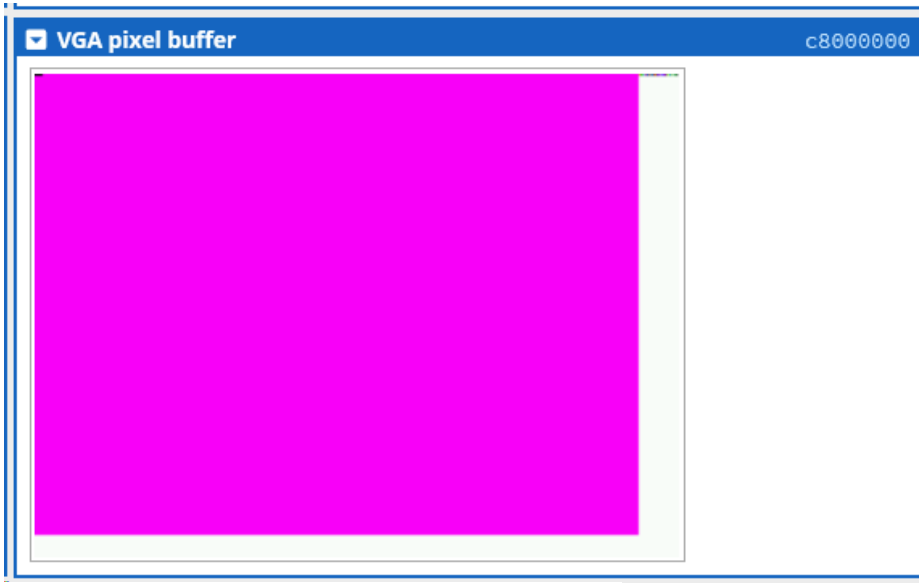
DE1-SoC Coding Practice

- Q) Write a code which displays 00 to 99 on the two right-most 7-segments with 1 second time delay.



DE1-SoC VGA Adapter

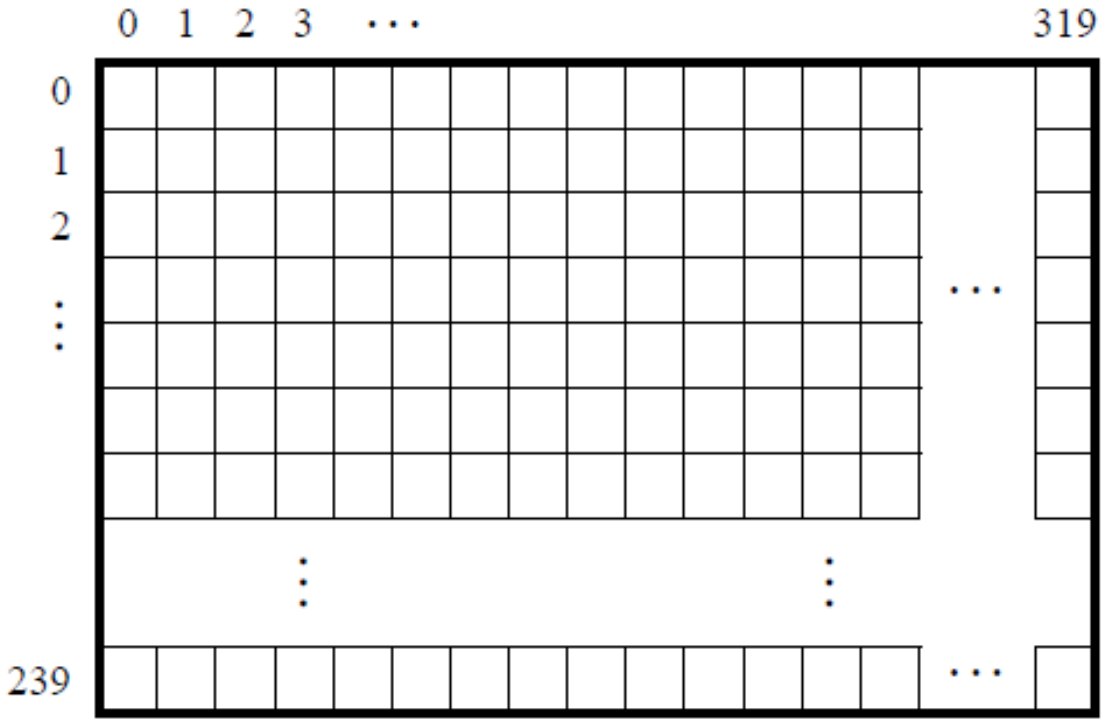
- VGA Adapter
  - Draw images on the monitor
  - An image: rectangular array of picture elements (“pixel”)
  - Each pixel: appears as a dot

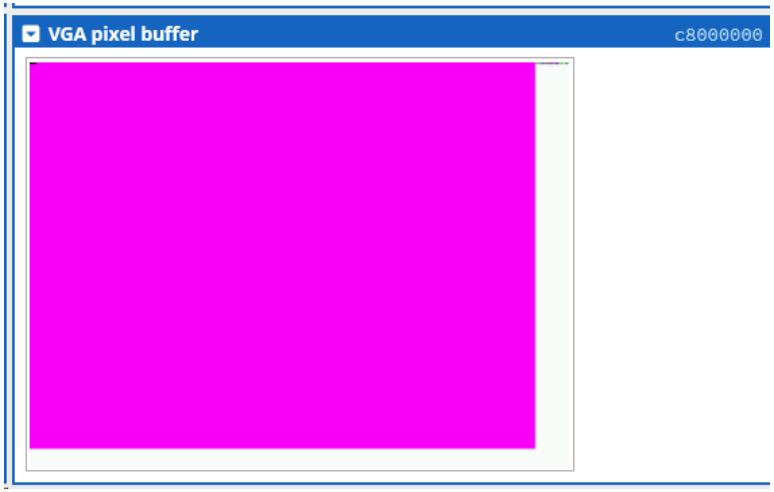


Device	VGA Adapter
Configuration	320x240 pixel resolution, 80x60 character resolution, 16-bit colour
Input/Output	Input and Output
Address Base	Pixel Buffer: 0xC8000000, Character Buffer: 0xC9000000
Address Map	pixel base+{y[7:0],x[8:0],1'b0} corresponds to pixel (x,y) character base+{y[5:0],x[6:0]} corresponds to character (x,y)
Initialization	None needed
Interrupts	None
Hardware Setup	Connect the VGA plug of your monitor to the VGA port of the DE1-SoC
Reference	ADV7123 Video DAC datasheet

DE1-SoC VGA Adapter

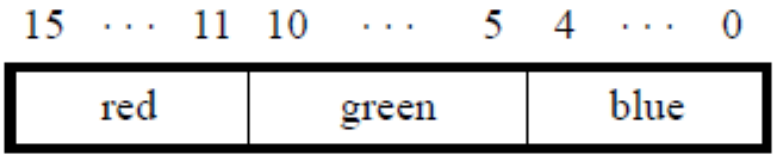
- VGA Adapter
  - Screen size: 240 row (y) x 320 columns(x)
  - Coordinate: (x,y).
    - (0,0) top-left corner
    - (319,239) bottom-right corner

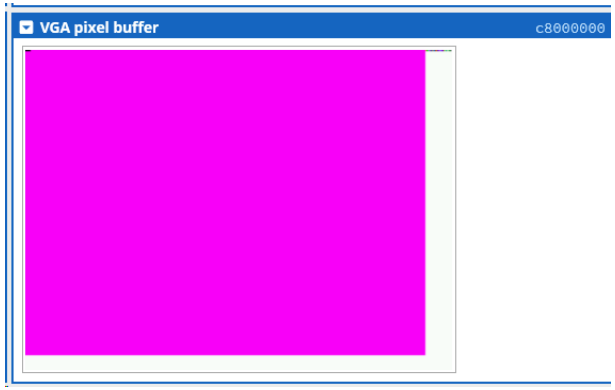




DE1-SoC VGA Adapter

- Pixel Color
  - 16-bit is used to represent the color of a pixel

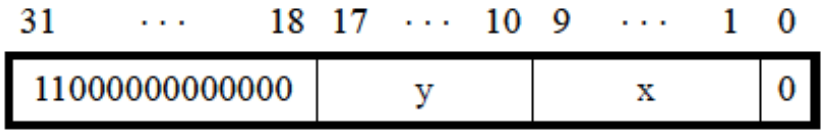




  - (EX): 0b1111 1000 0000 0000 = 0xF800(Red)
  - 0b0000 0111 1110 0000 = 0x07E0 (Green)
  - 0b0000 0000 0001 1111 = 0x001F (Blue)
  - 0xF81F (purple) 0xFFFF (White)
  - 0x0 (Black) 0x8410 (Gray)

DE1-SoC VGA Adapter

- Pixel Address

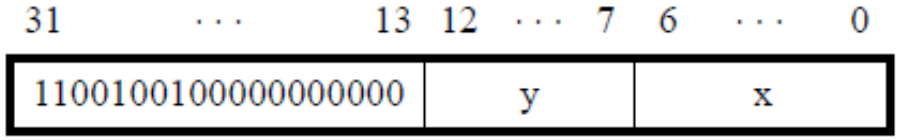


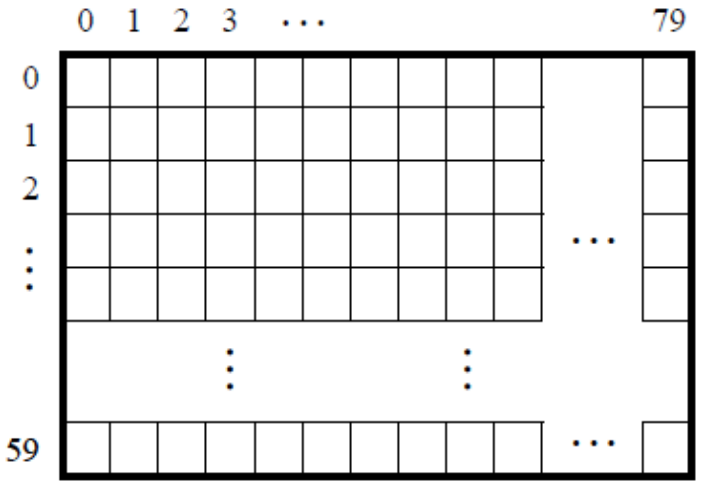

```
3 //PIXEL address
4 // 240 rows and 320 cols
5 //3 2             1
6 //1098 7654 3210 9876 5432 1098 7654 3210
7 //1100 1000 0000 00yy yyyy yyxx xxxx xxx0
8 //Location = PIXEL + y<<10 + x<<1
```

- Pixel Address = 0xC8000000 + y[7:0] <<10 + x[8:0] <<1
- (EX) (x,y) = (0, 0) → Pixel Addr = 0xC8000000
- (x,y) = (1,1) → Pixel Addr = 0xC80000000 + 0x400 + 0x20
- = 0xC80000420
- **(x.y)=(319,239) → pixel Addr ??**

DE1-SoC VGA Adapter

- Character Buffer
  - Screen: 80 x 60 characters
  - (x,y)=(0,0) top-left corner
  - (x,y)=(79,59) bottom-right corner
- Character buffer address

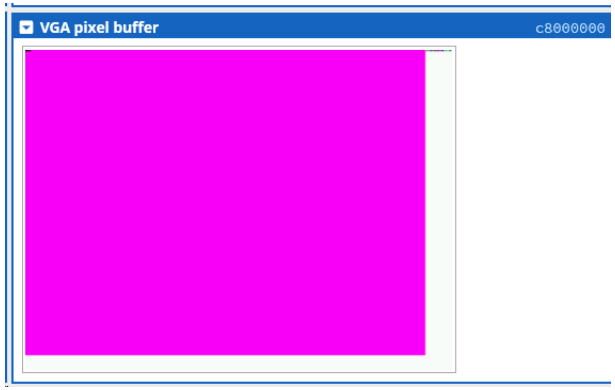




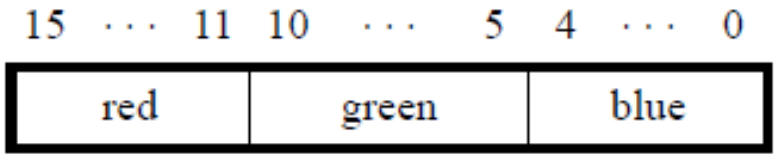
- Character Address= 0xC9000000 + y[5:0]<<7 + x[6:0]
- (EX) (x,y)=(0,0) Character addr = 0xC9000000
- (x,y)=(1,0) Character addr = 0xC9000000 + 0 + 1 = 0xC9000001
- (x,y)=(0,1) Character addr = 0xC9000000 + 0x80 + 0 = 0xC9000080
- **(x,y)=(79,59) Character addr = ???**

DE1-SoC VGA Adapter – Practice 1

- **PRACTICE:**
  - (a) Place a red dot on (10,10)
  - (b) Change the entire screen to Green



- Pixel Color

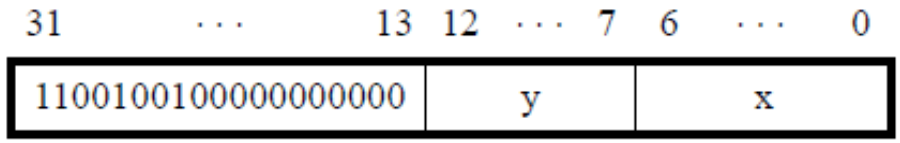


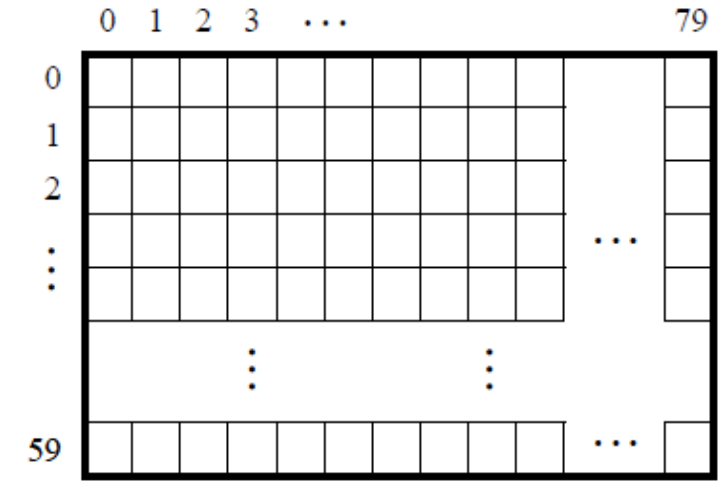
  - (EX): 0b1111 1000 0000 0000 = 0xF800(Red)
  - 0b0000 0111 1110 0000 = 0x07E0 (Green)
- Pixel Address
  - Pixel Address = 0xC8000000 + y[7:0] <<10 + x[8:0] <<1

```
3 //PIXEL address
4 // 240 rows and 320 cols
5 //3 2             1
6 //1098 7654 3210 9876 5432 1098 7654 3210
7 //1100 1000 0000 00yy yyyy yyxx xxxx xxx0
8 //Location = PIXEL + y<<10 + x<<1
```

DE1-SoC VGA Adapter – Practice 2

- **Practice**
  - (a) Print ‘A’ on (10,10)
  - (b) Print “Hello World!” in the row 10
- Character buffer address





- Character Address= 0xC9000000 + y[5:0]<<7 + x[6:0]

```
19 //CHARACTER address
20 //Register addr + character buffer
21 //3 2             1
22 //1098 7654 3210 9876 5432 1098 7654 3210
23 //1100 1001 0000 0000 000y yyyy yxxx xxxx
24 // 60 rows (y) x 70 cols (x)
25 //y - vertical from top to bottom
26 //x - horizontal from left to right
27 //Location = VGA + y<<7 + x
```

DE1-SoC VGA Adapter – functions and Code

- **Write a function**
  - **onePixel: place a dot with a given color at a given (x,y) location**
  - **oneChar: write a character at a given (x,y) location**
- Write a code (using above 2 functions) to (a) make the entire screen background Purple and (b) write “Hello World!” in the row 10

DE1-SoC (Project Idea)

- Display from 0000 to 9999
- Conversion of Roman number and display its decimal equivalent on 7-Seg
- 1 minute timer: 60 → 0
  - Stop by a SW
  - Resume by another SW
- Digital clock: HH MM SS format on 7-Segment Display.
- Display on the 7-segment display a decimal number which is to the binary number made by the 10 switches (ON – 1, OFF – 0)
- Sum of decimal digits and display on the VGA monitor
- Conversion of Roman number and display its decimal equivalent on the Monitor

DE1-SoC UART(Universal Async. Receiver/Transmitter)

- JTAG UART Box simulates (a) keyboard and (b) display
- Our objective
  - Write (and thus display) a string (in the memory) on the JTAG (Joint Test Action Group) UART box.
  - Read a string from the JTAG UART box in to the memory
- UART base address: 0xFF201000

Device	JTAG UART		
Input/Output	Both		
Address Base	0xFF201000		
Address Map	**Address**	**R/W**	**Description**
	base	R/W	Data Register 31:16 - Number of characters available to read (before this read) 15 - read data is valid 7:0 - the data itself
	base+4	R/W	Control Register 31:16 - Spaces available for writing 10 - AC (has value of 1 if JTAG UART has been accessed by host, cleared by writing 1 to it) 9 - Write interrupt is pending 8 - Read interrupt is pending 1 - Enable write interrupts 0 - Enable read interrupts
Initialization	None		

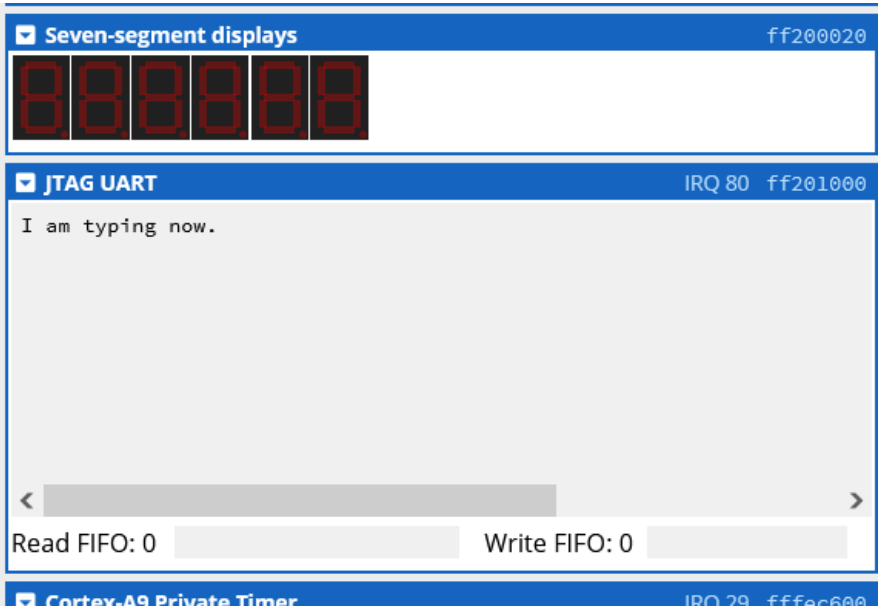
DE1-SoC UART(Universal Async. Receiver/Transmitter)

- Writing a string (Which is stored in the data section by **.asciz** (which is ended with null 0. Null 0 is automatically added.):
  - Read a character (i.e., a Byte) at a time
  - Store the byte to the UART register
  - Repeat until the read character is null 0.

- Reading a string from JTAG UART Box
  - In the code, make a space for the string in the memory by .space
  - Need to do:
    - Size (buffer_ of the string in Byte → 80
    - UART "data valid" bit which indicates that new data is in or not: bit **15**

DE1-SoC UART(Universal Async. Receiver/Transmitter)

- Writing a string
  - Read a character (i.e., a Byte) at a time
  - Store the byte to the UART address
  - Repeat until the read character is null 0.



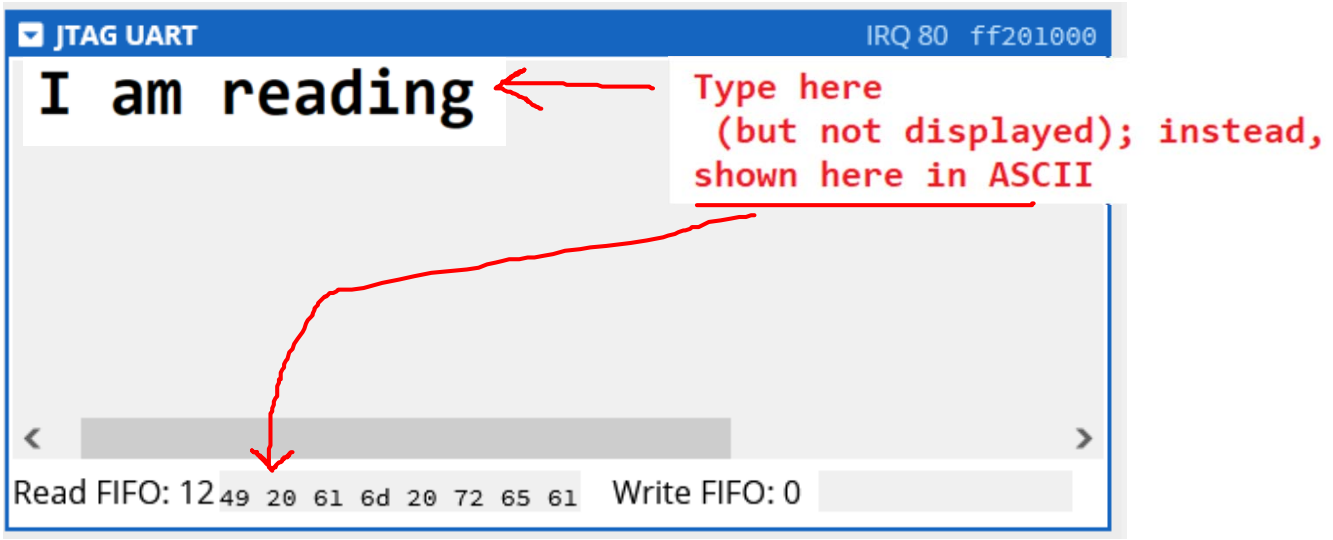
DE1-SoC UART(Universal Async. Receiver/Transmitter)

- Reading a string from JTAG UART Box
  - Make a space (or buffer) for the string in the memory by .space
  - Size of the string in Byte → 80
  - Check the "data valid" bit (bit 15)→ Read UART register and check the bit 15 (remember bits 7-0 hold the data!!).

DE1-SoC UART(Universal Async. Receiver/Transmitter)

- Reading a string from JTAG UART Box
  - Before running the code, type a string in the JTAG UART BOX (main display)
  - It does not display as you type; instead, it's ASCII code (in hex number) is displayed in the "Read FIFO" tray.

I		a	m		r
0x49	0x20	0x61	0x6D	0x20	0x72



Now run the code

```
                    String:
00000038  6d612049      stclvs  p0, cr2, [r1, #-292]!
0000003c  61657220      ?
00000040  676e6964      strbvs  r6, [lr, -r4, ROR #18]
```

Result:

DE1-SoC (Project Idea)

- Display from 0000 to 9999
- Conversion of Roman number and display its decimal equivalent on 7-Seg
- 1 minute timer: 60 → 0
  - Stop by a SW
  - Resume by another SW
- Digital clock: HH MM SS format on 7-Segment Display.
- Display on the 7-segment display a decimal number which is to the binary number made by the 10 switches (ON – 1, OFF – 0)
- Sum of decimal digits and display on the VGA monitor
- Conversion of Roman number and display its decimal equivalent on the Monitor