Lab 4 Deliverables

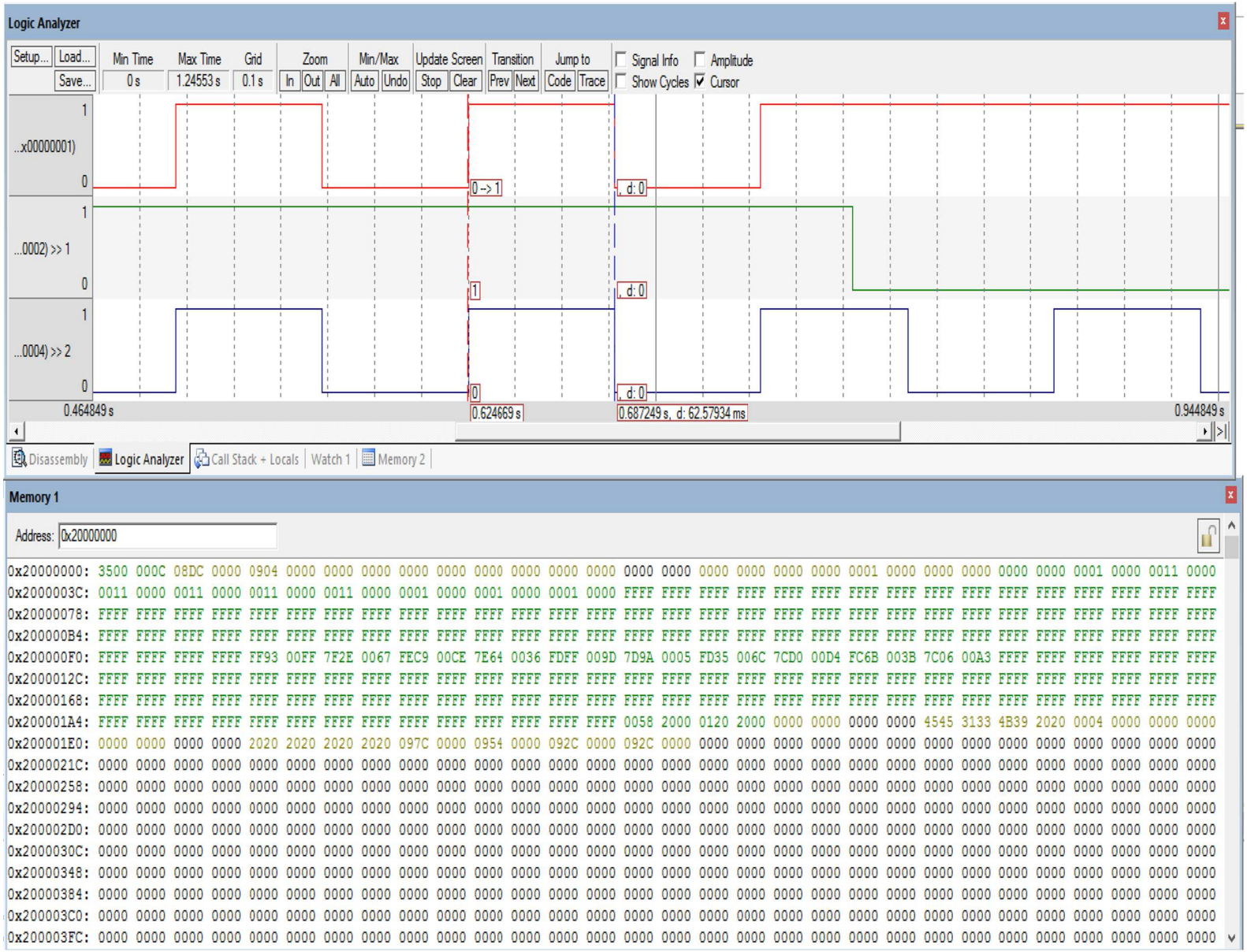
Akaash Chikarmane and Milan Feliciello

Section: 16085

Spring 2016

3/1/16

Simulation


Logic Analyzer
Setup...
Load...
Save...
Min Time
0 s
Max Time
1.24553 s
Grid
0.1 s
Zoom
In
Out
All
Min/Max
Auto
Undo
Update Screen
Stop
Clear
Transition
Prev
Next
Jump to
Code
Trace
Signal Info
Amplitude
Show Cycles
Cursor
...x00000001)
...0002) >> 1
...0004) >> 2
0 -> 1
1
0
d: 0
0.464849 s
0.624669 s
0.687249 s, d: 62.57934 ms
0.944849 s
Disassembly
Logic Analyzer
Call Stack + Locals
Watch 1
Memory 2
Memory 1
Address: 0x20000000
0x20000000: 3500 000C 08DC 0000 0904 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0000 0000 0000 0000 0000 0001 0000 0011 0000
0x2000003C: 0011 0000 0011 0000 0011 0000 0011 0000 0001 0000 0001 0000 0001 0000 FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
0x20000078: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
0x200000B4: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
0x200000F0: FFFF FFFF FFFF FFFF FF93 00FF 7F2E 0067 FEC9 00CE 7E64 0036 FDFF 009D 7D9A 0005 FD35 006C 7CD0 00D4 FC6B 003B 7C06 00A3 FFFF FFFF FFFF FFFF FFFF FFFF
0x2000012C: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
0x20000168: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
0x200001A4: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF 0058 2000 0120 2000 0000 0000 0000 0000 4545 3133 4B39 2020 0004 0000 0000 0000
0x200001E0: 0000 0000 0000 0000 2020 2020 2020 2020 097C 0000 0954 0000 092C 0000 092C 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x2000021C: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x20000258: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x20000294: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x200002D0: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x2000030C: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x20000348: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x20000384: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x200003C0: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x200003FC: 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Final Program

```
;****************** main.s ***************
; Program written by: Akaash Chikarmane/Milan Feliciello
; Date Created: 2/22/2016
; Last Modified: 2/29/2016
; Section Tuesday 2-3
; Instructor: Ramesh Yerraballi
; Lab number: 4
; Brief description of the program
;   If the switch is presses, the LED toggles at 8 Hz
; Hardware connections
;  PE1 is switch input  (1 means pressed, 0 means not pressed)
;  PE0 is LED output (1 activates external LED on protoboard)
;Overall functionality of this system is the similar to Lab 3, with three changes:
;1-  initialize SysTick with RELOAD 0x00FFFFFF
;2-  add a heartbeat to PF2 that toggles every time through loop
;3-  add debugging dump of input, output, and time
; Operation
;     1) Make PE0 an output and make PE1 an input.
;     2) The system starts with the LED on (make PE0 =1).
;   3) Wait about 62 ms
;   4) If the switch is pressed (PE1 is 1), then toggle the LED once, else turn the LED on.
;   5) Steps 3 and 4 are repeated over and over


SWITCH                  EQU 0x40024004  ;PE0
LED                     EQU 0x40024008  ;PE1
SYSCTL_RCGCGPIO_R       EQU 0x400FE608
SYSCTL_RCGC2_GPIOE      EQU 0x00000010   ; port E Clock Gating Control
SYSCTL_RCGC2_GPIOF      EQU 0x00000020   ; port F Clock Gating Control
GPIO_PORTE_DATA_R       EQU 0x400243FC
```

```
GPIO_PORTE_DIR_R       EQU 0x40024400
GPIO_PORTE_AFSEL_R     EQU 0x40024420
GPIO_PORTE_PUR_R       EQU 0x40024510
GPIO_PORTE_DEN_R       EQU 0x4002451C
GPIO_PORTF_DATA_R      EQU 0x400253FC
GPIO_PORTF_DIR_R       EQU 0x40025400
GPIO_PORTF_AFSEL_R     EQU 0x40025420
GPIO_PORTF_DEN_R       EQU 0x4002551C
NVIC_ST_CTRL_R         EQU 0xE000E010
NVIC_ST_RELOAD_R       EQU 0xE000E014
NVIC_ST_CURRENT_R      EQU 0xE000E018
count EQU 0x131000
countone EQU 50
counttwo EQU 50
FILLone EQU 0xFFFFFFFF
FILLtwo EQU 0xFFFFFFFF
        THUMB
        AREA    DATA, ALIGN=4
SIZE      EQU    50
;You MUST use these two buffers and two variables
;You MUST not change their names
;These names MUST be exported
        EXPORT DataBuffer
        EXPORT TimeBuffer
        EXPORT DataPt [DATA,SIZE=4]
        EXPORT TimePt [DATA,SIZE=4]

DataBuffer SPACE SIZE*4
TimeBuffer SPACE SIZE*4
DataPt SPACE 4
TimePt SPACE 4
```

```
    ALIGN
      PRESERVE8
    AREA    |.text|, CODE, READONLY, ALIGN=2
    THUMB
    EXPORT  Start
    IMPORT  TExaS_Init


Start BL   TExaS_Init
       BL Debug_Init; running at 80 MHz, scope voltmeter on PD3
; initialize Port E
; initialize Port F
; initialize debugging dump, including SysTick

    LDR R1, =SYSCTL_RCGCGPIO_R                  ;initialize port E
    LDR R0, [R1]
    ORR R0, R0, #0x10
    STR R0, [R1]
    NOP
    NOP
    NOP
    NOP
    LDR R1, =GPIO_PORTE_DEN_R
    ORR R0, #0x03
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_AFSEL_R
    BIC R0, #0x03
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_DIR_R               ;pin 0 output, pin 1 input
    ORR R0, #0x01
```

```
    BIC R0, #0x02
    STR R0, [R1]
    LDR R1, =SYSCTL_RCGCGPIO_R
    LDR R0, [R1]
    ORR R0, #0x20
    STR R0, [R1]
    NOP
    NOP
    LDR R1, =GPIO_PORTF_DIR_R
    LDR R0, [R0]
    ORR R0, #0x04
    STR R0, [R1]
    LDR R1, =GPIO_PORTF_AFSEL_R
    LDR R0, [R1]
    AND R0, #0xFB
    STR R0, [R1]
    LDR R1, =GPIO_PORTF_DEN_R
    LDR R0, [R1]
    ORR R0, #0x04
    STR R0, [R1]


    CPSIE  I    ; TExaS voltmeter, scope runs on interrupts
;loop  BL   Debug_Capture

loop
    BL Debug_Capture
    BL delay
    LDR R1, =GPIO_PORTF_DATA_R
    LDR R0,[R1]                                    ;heartbeat off
    BIC R0, #0x04
```

```
    STR R0, [R1]



    LDR R1, =GPIO_PORTE_DATA_R
    LDR R0, [R1]                        ;checks if switch is pressed
    AND R2,R0, #0x02                    ;selects on input bits we need
    LSR R2,R2, #1
    EOR R2, #0x01
    BIC R0, R0, #0x01           ;gets the output into the appropriate place for PF4
    ORR R0,R0,R2                        ;makes code friednly
    LDR R1, =GPIO_PORTE_DATA_R
    STR R0, [R1]
    BL delay
    ORR R0, R0, #0x01                   ;consitently turns LED on to toggle the LED
    STR R0, [R1]




    LDR R1, =GPIO_PORTF_DATA_R
    LDR R0, [R1]
    ORR R0, #0x04                       ;heartbeat on
    STR R0,[R1]


;heartbeat
; Delay
;input PE1 test output PE0
      B    loop

delay
```

```
    LDR R2, =count

again
    SUBS R2,R2, #0x01
    BNE again
    BX LR



;------------Debug_Init------------
; Initializes the debugging instrument
; Input: none
; Output: none
; Modifies: none
; Note: push/pop an even number of registers so C compiler is happy
Debug_Init
    PUSH{R14}
    PUSH{R0,R12}
    LDR R1, =DataBuffer                    ;initialize pointers by filling them with starting locations
    LDR R0, =DataPt
    STR R1, [R0]
    LDR R1, =TimeBuffer
    LDR R0, =TimePt
    STR R1, [R0]

    LDR R0, =countone          ;R0 is the count
    LDR R2, =FILLone
    LDR R1, =DataPt
    LDR R1, [R1]

againinit
```

```
                ;pointer
    STR R2, [R1]                ;fill DataBuffer array with 0xFFFFFFFF
    ADD R1, #0x04
    SUB R0, #0x01
    CMP R0, #0
    BNE againinit
    LDR R0, =counttwo
    LDR R2, =FILLtwo
    LDR R1, =TimePt
    LDR R1, [R1]

againinittwo


    STR R2, [R1]                ;fill TimeBuffer array with 0xFFFFFFFF
    ADD R1, #0x04
    SUB R0, #0x01
    CMP R0, #0
    BNE againinittwo

    LDR R1, =NVIC_ST_CTRL_R
    MOV R0, #0
    STR R0, [R1]                        ;iniitalize systick
    LDR R1, =NVIC_ST_RELOAD_R
    LDR R0, =0x00FFFFFF
    STR R0, [R1]
    LDR R1, =NVIC_ST_CURRENT_R
    MOV R0, #0
    STR R0, [R1]
    LDR R1, =NVIC_ST_CTRL_R
    MOV R0, #0x05
```

```
    STR R0, [R1]
    POP{R0,R12}
    POP{R14}
    BX LR



; init SysTick



;------------Debug_Capture------------
; Dump Port E and time into buffers
; Input: none
; Output: none
; Modifies: none
; Note: push/pop an even number of registers so C compiler is happy
Debug_Capture
    PUSH{R14}
    PUSH{R0,R12}
    LDR R0, =countone
    LDR R1, [R0] ;If we have already filled the array we return to main
    SUB R1, #0x01
    CMP R1, #0
    BLS stop
    STR R1, [R0]
    LDR R1, =GPIO_PORTE_DATA_R
    LDR R0, [R1]
    AND R2, R0, #0x02
    AND R0, #0x01
```

```
    LSL R2, #3
    ORR R0, R0, R2
    LDR R1, =DataPt
    LDR R2, [R1]
    STR R0, [R2]
    ADD R2, #0x04
    STR R2, [R1]
    LDR R1, =NVIC_ST_CURRENT_R
    LDR R0, [R1]
    LDR R2, =TimePt
    LDR R3, [R2]
    STR R0, [R3]

    ADD R3, #0x04
    STR R3, [R2]
stop
    POP{R0,R12}
    POP{R14}
    BX LR



  ALIGN                 ; make sure the end of this section is aligned
  END                   ; end of file
```

Estimation of Execution Time

Execution: $\frac{12.5\ ns}{1\ cycle} * 58\ cycles = 725\ ns$; 29 instructions in Debug_capture subroutine @ 2 cycles per instruction

Time between cycles: $5{,}000{,}042\ cycles * \frac{12.5\ ns}{1\ cycle} = 62{,}500{,}252\ \text{ns}$;there are 21 instructions (42 cycles), not including the delay, between captures. Including the delay, there are 5,000,042 cycles @ 2 cycles per instruction

Intrusiveness: $\frac{725\ ns}{62500252\ ns} * 100\% = .0011599\%$

Results of Debugging Instrument

Actual: 62.489 ms

Calculated:

0x00D9FAD7 = 14,285,527

0x00417A27 = 4,291,111

14285527 – 4291111 = 9994416

((9994416 cycles)/2) = 4997208 cycles ;Divide by two because it stores data/time every 2 periods

4997208 $cycles$ * 12.5e-9 $\frac{seconds}{cycle}$ = 62.4651 ms