1. Introduction  
   In this lab the objective was to implement the blinking LED design from previous labs using externally connected components and using the Systick to track time. Systick is the built-in time counter in the FPGA and will be the go-to option when a time delay is needed for practical systems. The use of external components will also be important to understand as the built-in components will be insufficient for more sophisticated design.
2. Procedure/Discussion  
   The design of the LED logic was already established in the previous 2 labs; thus it was adapted for this one as well. The Systick delay constant for 100ms however needed to be reduced to 75000 instead of the theoretical 300000. The addition of external components did add the challenge of a hardware design from scratch. Figuring out how use the external pins came after setting up basic source to ground linear circuits for the switch and LED. Pin 5.0 was connected to the ground/resistor end of the switch to read a potential change from the switch connecting it to the source. Pin 4.0 was connected as if it were a source to the LED anode end to apply the LED logic.  
   The voltage measurements for the no press/ off states are zero because of cutoff, as there is no voltage source driving a current nor potential difference. For the switch, we expect V(P5.0) to be about equal to the source as it has a short circuit to it once the switch is pressed. For the current, the theoretical value is V(source)/10000 because of the resistor value. As such 324 uA is close to the expected value.   
   For the LED, when the LED is off there is no current and as such there should be no potential differences. The values of the potential differences across the LED and Resistor should be about equal to the value of the voltage source, in this case 3.13V is within 10% of the 3.3V source.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Switch |  |  | LED |  |
| V(P5.0) press | 3.24V |  | V(LED off) | 0V |
| I(R1) press | 324 uA |  | V(R2) LED off | 0V |
| V(P5.0) no press | 0V |  | I(LED off) | 0A |
| I(R1) no press | 0A |  | V(LED on) | 1.86V |
|  |  |  | V(R2) LED on | 1.27V |
|  |  |  | I(LED on) | 3mA |
|  |  |  |  |  |

A positive logic interface functions intuitively where 0 is the low state and 1 is the high. Negative logic functions opposite to that where 0 is the high state and 1 is the low. For a switch, a negative logic means that the button being pressed will input a zero when read in, and a one when not pressed. A negative logic LED will only light up when a zero is read in alternatively.   
Changing the resistance value of R1 shouldn’t alter anything in the design as P5.0 will read 3.3V always. A change in current for the switch has no obviously observable effects. Increasing R2 though will decrease the brightness of the bulb, whereas decreasing will increase it.

1. Conclusion  
   The result of this lab was the expected function of the design. The LED flashed at an acceptable rate and the switch functioned as it should. Correct functionality of the design proved that the code *and* the circuit setup were compatible and individually correct. For more sophisticated systems these parts will most likely be more complex both by themselves and together.   
   Completing the lab established how the external pins functioned and how to use the FPGA with external components. I should’ve constructed the LED and switch circuits separately before combining them as the complete scheme of the wiring ended up making it difficult to verify that it was set up correctly.
2. References  
   -Griffon Ye
3. Appendix

.thumb

**.text**

**.align** 2

P4IN .field 0x40004C21,32

P5IN .field 0x40004C40,32

P4OUT .field 0x40004C23,32

P5OUT .field 0x40004C42,32

P4DIR .field 0x40004C25,32

P5DIR .field 0x40004C44,32

P4DS .field 0x40004C29,32

P5DS .field 0x40004C48,32

P4SEL0 .field 0x40004C2B,32

P5SEL0 .field 0x40004C4A,32

P4SEL1 .field 0x40004C2D,32

P5SEL1 .field 0x40004C4C,32

SYSTICK\_STCSR .field 0xE000E010, 32

SYSTICK\_STRVR .field 0XE000E014, 32

SYSTICK\_STCVR .field 0xE000E018, 32

Systick\_cons .field 0x00FFFFFF, 32

DELAY200MS .field 75000, 32

;RED .equ 0x01

;GREEN .equ 0x02

;BLUE .equ 0x04

**.global** main

.thumbfunc main

**main:** .asmfunc

BL Port4\_Init ; initialize P1.1 and P1.4 and make them inputs (P1.1 and P1.4 built-in buttons)

BL Port5\_Init

loop

BL SysTick\_Init

BL SysTick\_Wait200ms

BL Port5\_Input ; read both of the switches on Port 1

**CMP** R0, #0x01 ; R0 == 0x02?

BEQ sw2pressed ; if so, switch 2 pressed

**CMP** R0, #0x00 ; R0 == 0x12?

BEQ nopressed ; if so, neither switch pressed

; if none of the above, unexpected return value

**B** loop

sw2pressed

LDR R1, P4OUT

LDRB R0, [R1]

EOR R0, #0x01

BL Port4\_Output ; turn the red LED on

**B** loop

nopressed

BIC R0, #0x01

BL Port4\_Output ; turn all of the LEDs off

**B** loop

.endasmfunc

**Port4\_Init:** .asmfunc

LDR R1, P4SEL0

**MOV** R0, #0x00

STRB R0, [R1]

LDR R1, P4SEL1

**MOV** R0, #0x00

STRB R0, [R1]

LDR R1, P4DIR

**MOV** R0, #0x01

STRB R0, [R1]

LDR R1, P4OUT

ORR R0, #0x00

STRB R0, [R1]

BX LR

.endasmfunc

**Port5\_Init:** .asmfunc

LDR R1, P5SEL0

**MOV** R0, #0x00

STRB R0, [R1]

LDR R1, P5SEL1

**MOV** R0, #0x00

STRB R0, [R1]

LDR R1, P5DIR

**MOV** R0, #0x00

STRB R0, [R1]

LDR R1, P5OUT

ORR R0, #0x00

STRB R0, [R1]

BX LR

.endasmfunc

**Port5\_Input:** .asmfunc

LDR R1, P5IN

LDRB R0, [R1]

**AND** R0, R0, #0x01

BX LR

.endasmfunc

**Port4\_Output:** .asmfunc

LDR R1, P4OUT

STRB R0, [R1]

BX LR

.endasmfunc

;-----------SysTick\_Init----------------

**SysTick\_Init:** .asmfunc

LDR R1, SYSTICK\_STCSR

**MOV** R0, #0

STR R0, [R1]

LDR R1, SYSTICK\_STRVR

LDR R0, Systick\_cons

STR R0, [R1]

LDR R1, SYSTICK\_STCVR

**MOV** R0, #0

STR R0, [R1]

LDR R1, SYSTICK\_STCSR

**MOV** R0, #0x00000005

STR R0, [R1]

BX LR

.endasmfunc

;-----------SysTick\_Wait------------

**SysTick\_Wait:** .asmfunc

LDR R1, SYSTICK\_STRVR

**SUB** R0, R0, #1

STR R0, [R1]

LDR R1, SYSTICK\_STCVR

**MOV** R2, #0

STR R0, [R1]

LDR R1, SYSTICK\_STCSR

SysTick\_Wait\_loop

LDR R3, [R1]

ANDS R3, R3, #0x00010000

BEQ SysTick\_Wait\_loop

BX LR

.endasmfunc

**SysTick\_Wait200ms:** .asmfunc

**PUSH** {R4, LR}

**MOVS** R4, R0

BEQ SysTick\_Wait200ms\_done

SysTick\_Wait200ms\_loop

LDR R0, DELAY200MS

BL SysTick\_Wait

SUBS R4, R4, #1

BHI SysTick\_Wait200ms\_loop

SysTick\_Wait200ms\_done

**POP** {R4, LR}

BX LR

.endasmfunc

.end