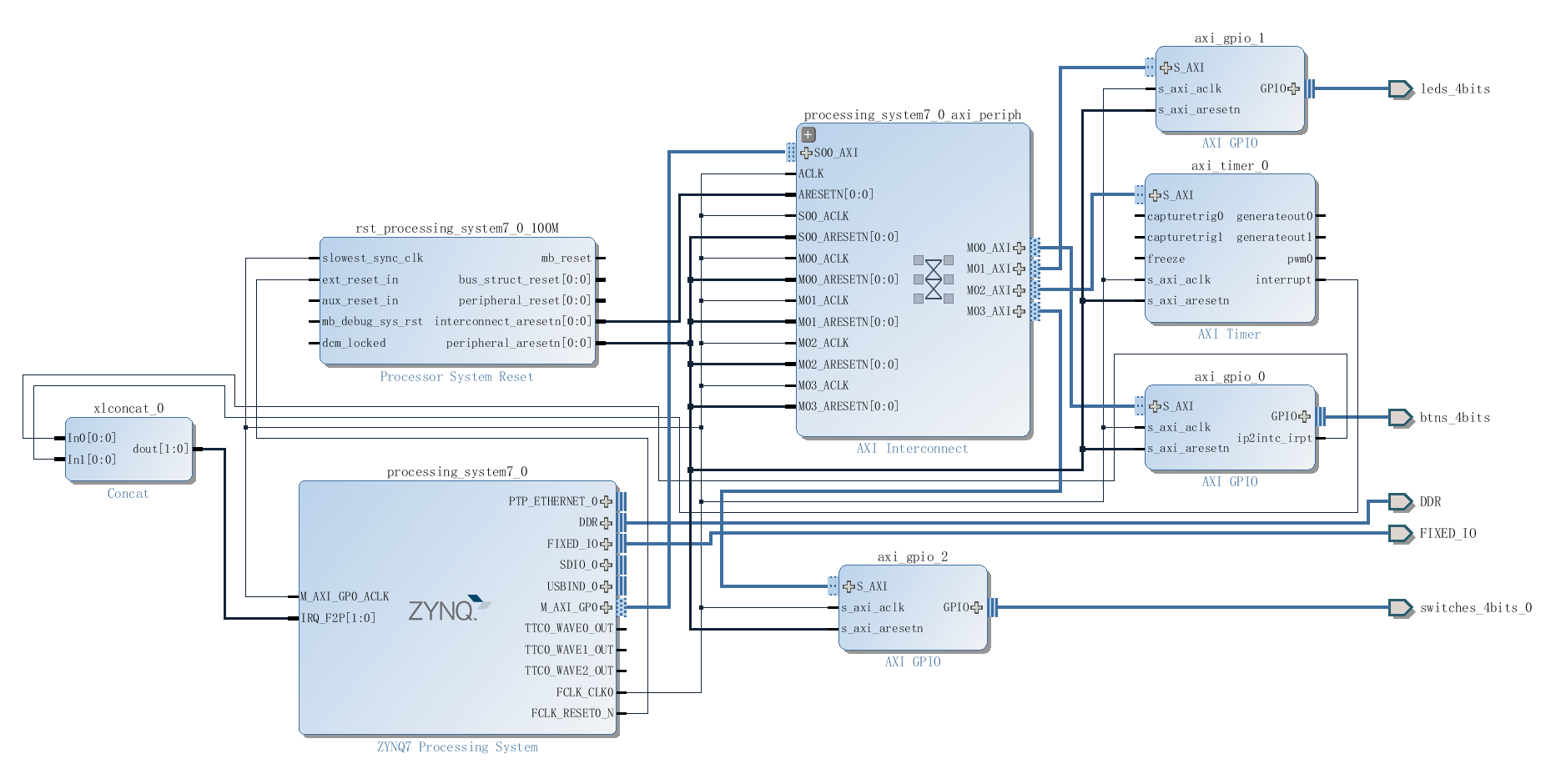
Lab8 Interrupt

Name : Yi Zihong ID : U201613634

**Part One**

In this lab, I uses the FPGA to add numbers with timer and button interrupt.



**Part Two**

In this lab, I finished the Ping-Pong game with timer and button interrupt. Also it can change the speed with reading the values of switches. These are my code.

1. /\*
2. \* interrupt\_counter\_tut\_2B.c
3. \*
4. \*  Created on:     Unknown
5. \*      Author:     Ross Elliot
6. \*     Version:     1.1
7. \*/
9. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
11. \* VERSION HISTORY
12. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
13. \*   v1.1 - 01/05/2015
14. \*       Updated for Zybo ~ DN
15. \*
16. \*   v1.0 - Unknown
17. \*       First version created.
18. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
20. #include "xparameters.h"
21. #include "xgpio.h"
22. #include "xtmrctr.h"
23. #include "xscugic.h"
24. #include "xil\_exception.h"
25. #include "xil\_printf.h"
26. #include "XScuTimer.h"
28. // Parameter definitions
29. #define INTC\_DEVICE\_ID      XPAR\_PS7\_SCUGIC\_0\_DEVICE\_ID
30. #define TMR\_DEVICE\_ID       XPAR\_TMRCTR\_0\_DEVICE\_ID
31. #define BTNS\_DEVICE\_ID      XPAR\_AXI\_GPIO\_0\_DEVICE\_ID
32. #define LEDS\_DEVICE\_ID      XPAR\_AXI\_GPIO\_1\_DEVICE\_ID
33. #define SWITCES\_DEVICE\_ID   XPAR\_AXI\_GPIO\_2\_DEVICE\_ID
34. #define INTC\_GPIO\_INTERRUPT\_ID XPAR\_FABRIC\_AXI\_GPIO\_0\_IP2INTC\_IRPT\_INTR
35. #define INTC\_TMR\_INTERRUPT\_ID XPAR\_FABRIC\_AXI\_TIMER\_0\_INTERRUPT\_INTR
37. #define BTN\_INT             XGPIO\_IR\_CH1\_MASK
38. #define TMR\_LOAD            0xF8000000
40. #define RESETBUTTON 0b0100
41. #define STARTBUTTON 0b0010
42. #define LEFTPADDLE 0b1000
43. #define RIGHTPADDLE 0b0001
45. #define LED\_PATTERNS\_ORDER\_LEFT\_OUT 0
46. #define LED\_PATTERNS\_ORDER\_RIGHT\_OUT 5
48. #define Paddle\_Left 1
49. #define Paddle\_Right 4
51. #define START 1
52. #define STOP 0
53. #define LEFT 0
54. #define RIGHT 1
56. #define ONE\_TENTH 32500000 // half of the CPU clock speed/10
58. **int** LED\_PATTERNS\_ORDER[6] = { 0b0000, 0b1000, 0b0100, 0b0010, 0b0001, 0b0000 };
60. XGpio LEDInst, BTNInst;
61. XScuGic INTCInst;
62. XTmrCtr TMRInst;
63. **static** **int** btn\_value;
64. **static** **int** tmr\_count;
66. **static** **int** scoreright;
67. **static** **int** scoreleft;
68. //----------------------------------------------------
69. // PROTOTYPE FUNCTIONS
70. //----------------------------------------------------
71. **static** **void** BTN\_Intr\_Handler(**void** \*baseaddr\_p);
72. **static** **void** TMR\_Intr\_Handler(**void** \*baseaddr\_p);
73. **static** **int** InterruptSystemSetup(XScuGic \*XScuGicInstancePtr);
74. **static** **int** IntcInitFunction(u16 DeviceId, XTmrCtr \*TmrInstancePtr, XGpio \*GpioInstancePtr);
76. //----------------------------------------------------
77. // INTERRUPT HANDLER FUNCTIONS
78. // - called by the timer, button interrupt, performs
79. // - LED flashing
80. //----------------------------------------------------
81. **char** GameOver, StartDirection;
82. **int** led\_order;
83. XGpio dip;
84. XScuTimer Timer; /\* Cortex A9 SCU Private Timer Instance \*/
85. XScuTimer \*TimerInstancePtr = &Timer;
86. **int** dip\_check, dip\_check\_prev;
88. **int** number = 3;
90. **void** BTN\_Intr\_Handler(**void** \*InstancePtr)
91. {
92. // Disable GPIO interrupts
93. XGpio\_InterruptDisable(&BTNInst, BTN\_INT);
94. // Ignore additional button presses
95. **if** ((XGpio\_InterruptGetStatus(&BTNInst) & BTN\_INT) !=
96. BTN\_INT) {
97. **return**;
98. }
99. btn\_value = XGpio\_DiscreteRead(&BTNInst, 1);
101. **switch**(btn\_value){
102. **case** RESETBUTTON:
103. xil\_printf("\n\rNew Game - Scores Reset\r\n");
104. scoreright = 0;
105. scoreleft = 0;
106. GameOver = STOP;
107. xil\_printf("Score Left = %d Score Right = %d\r\n", scoreright,scoreleft);
108. **break**;
110. **case** STARTBUTTON:
112. GameOver = START; //start game
114. **break**;
116. **case** LEFTPADDLE:
117. **if** (led\_order != Paddle\_Left) {
119. xil\_printf("Press too early !\r\n");
120. StartDirection = RIGHT;
121. GameOver = STOP;
122. led\_order = LED\_PATTERNS\_ORDER\_LEFT\_OUT;
123. XGpio\_DiscreteWrite(&LEDInst, 1,  LED\_PATTERNS\_ORDER[led\_order]);
124. scoreright += 1;
125. xil\_printf("Score  = Left = %d Score Right %d\r\n", scoreleft, scoreright);
127. } **else** {
128. StartDirection = RIGHT;
129. scoreright += 1;
130. XGpio\_DiscreteWrite(&LEDInst, 1,  LED\_PATTERNS\_ORDER[led\_order]);
131. xil\_printf("Score  = Left = %d Score Right %d\r\n", scoreleft, scoreright);
132. }
134. **break**;
136. **case** RIGHTPADDLE:
137. **if** (led\_order != Paddle\_Right) {
138. StartDirection = LEFT;
139. GameOver = STOP;
141. xil\_printf("Press too early !\r\n");
142. led\_order = LED\_PATTERNS\_ORDER\_RIGHT\_OUT;
143. XGpio\_DiscreteWrite(&LEDInst, 1,  LED\_PATTERNS\_ORDER[led\_order]);
144. scoreleft += 1;
145. xil\_printf("Score  = Left = %d Score Right %d\r\n", scoreleft, scoreright);
146. } **else** {
147. StartDirection = LEFT;
148. scoreright += 1;
149. XGpio\_DiscreteWrite(&LEDInst, 1,  LED\_PATTERNS\_ORDER[led\_order]);
150. xil\_printf("Score  = Left = %d Score Right %d\r\n", scoreleft, scoreright);
151. }
152. **break**;
154. **default**:**break**;
155. }
157. (**void**)XGpio\_InterruptClear(&BTNInst, BTN\_INT);
158. // Enable GPIO interrupts
159. XGpio\_InterruptEnable(&BTNInst, BTN\_INT);
160. }
162. **void** TMR\_Intr\_Handler(**void** \*data)
163. {
164. **if** (XTmrCtr\_IsExpired(&TMRInst,0)){
165. // Once timer has expired 3 times, stop, increment counter
166. // reset timer and start running again
167. **if**(tmr\_count == number\*10){
169. //judge different directions
170. **if** (GameOver != STOP){
171. **switch**(StartDirection){
173. **case** LEFT:
174. XTmrCtr\_Stop(&TMRInst,0);
175. tmr\_count = 0;
177. led\_order--;
178. XGpio\_DiscreteWrite(&LEDInst, 1,  LED\_PATTERNS\_ORDER[led\_order]);
179. **if**(led\_order == LED\_PATTERNS\_ORDER\_LEFT\_OUT){
180. scoreright += 1;
181. xil\_printf("Score  = Left = %d Score Right %d\r\n", scoreleft, scoreright);
182. GameOver = STOP;
183. StartDirection = RIGHT;
184. }
185. XTmrCtr\_Reset(&TMRInst,0);
186. XTmrCtr\_Start(&TMRInst,0);
187. **break**;
189. **case** RIGHT:
190. XTmrCtr\_Stop(&TMRInst,0);
191. tmr\_count = 0;
193. led\_order++;
194. XGpio\_DiscreteWrite(&LEDInst, 1,  LED\_PATTERNS\_ORDER[led\_order]);
196. **if**(led\_order == LED\_PATTERNS\_ORDER\_RIGHT\_OUT){
197. scoreleft += 1;
198. xil\_printf("Score  = Left = %d Score Right %d\r\n", scoreleft, scoreright);
200. GameOver = STOP;
201. StartDirection = LEFT;
202. }
204. XTmrCtr\_Reset(&TMRInst,0);
205. XTmrCtr\_Start(&TMRInst,0);
206. **break**;
208. **default**:**break**;
209. }
210. }
211. } **else** {
212. tmr\_count++;
213. xil\_printf("tmr\_count %d\r\n", tmr\_count);
214. }
215. }
216. }


220. //----------------------------------------------------
221. // MAIN FUNCTION
222. //----------------------------------------------------
223. **int** main (**void**)
224. {
225. **int** status;
226. //initialize variables, timers, ports
227. XGpio\_Initialize(&dip, SWITCES\_DEVICE\_ID);
228. XGpio\_SetDataDirection(&dip, 1, 0xffffffff);
230. //----------------------------------------------------
231. // INITIALIZE THE PERIPHERALS & SET DIRECTIONS OF GPIO
232. //----------------------------------------------------
233. // Initialise LEDs
234. status = XGpio\_Initialize(&LEDInst, LEDS\_DEVICE\_ID);
235. **if**(status != XST\_SUCCESS) **return** XST\_FAILURE;
236. // Initialise Push Buttons
237. status = XGpio\_Initialize(&BTNInst, BTNS\_DEVICE\_ID);
238. **if**(status != XST\_SUCCESS) **return** XST\_FAILURE;
239. // Set LEDs direction to outputs
240. StartDirection = RIGHT;
241. led\_order = 1;
242. XGpio\_DiscreteWrite(&LEDInst, 1,  LED\_PATTERNS\_ORDER[led\_order]);
244. //----------------------------------------------------
245. // SETUP THE TIMER
246. //----------------------------------------------------
247. status = XTmrCtr\_Initialize(&TMRInst, TMR\_DEVICE\_ID);
248. **if**(status != XST\_SUCCESS) **return** XST\_FAILURE;
249. XTmrCtr\_SetHandler(&TMRInst, TMR\_Intr\_Handler, &TMRInst);
250. XTmrCtr\_SetResetValue(&TMRInst, 0, TMR\_LOAD);
251. XTmrCtr\_SetOptions(&TMRInst, 0, XTC\_INT\_MODE\_OPTION | XTC\_AUTO\_RELOAD\_OPTION);
253. // Initialize interrupt controller
254. status = IntcInitFunction(INTC\_DEVICE\_ID, &TMRInst, &BTNInst);
255. **if**(status != XST\_SUCCESS) **return** XST\_FAILURE;
257. XTmrCtr\_Start(&TMRInst, 0);
258. xil\_printf("\n\rInit all finished\r\n");

261. **while**(1){
262. dip\_check = XGpio\_DiscreteRead(&dip, 1);
263. **if** (dip\_check != dip\_check\_prev) {
264. xil\_printf("DIP Switch Status %x, %x\r\n", dip\_check\_prev,
265. dip\_check);
266. dip\_check\_prev = dip\_check;
267. number = dip\_check;
268. }
269. }
271. **return** 0;
272. }
274. //----------------------------------------------------
275. // INITIAL SETUP FUNCTIONS
276. //----------------------------------------------------
278. **int** InterruptSystemSetup(XScuGic \*XScuGicInstancePtr)
279. {
280. // Enable interrupt
281. XGpio\_InterruptEnable(&BTNInst, BTN\_INT);
282. XGpio\_InterruptGlobalEnable(&BTNInst);
284. Xil\_ExceptionRegisterHandler(XIL\_EXCEPTION\_ID\_INT,
285. (Xil\_ExceptionHandler)XScuGic\_InterruptHandler,
286. XScuGicInstancePtr);
287. Xil\_ExceptionEnable();

290. **return** XST\_SUCCESS;
292. }
294. **int** IntcInitFunction(u16 DeviceId, XTmrCtr \*TmrInstancePtr, XGpio \*GpioInstancePtr)
295. {
296. XScuGic\_Config \*IntcConfig;
297. **int** status;
299. // Interrupt controller initialisation
300. IntcConfig = XScuGic\_LookupConfig(DeviceId);
301. status = XScuGic\_CfgInitialize(&INTCInst, IntcConfig, IntcConfig->CpuBaseAddress);
302. **if**(status != XST\_SUCCESS) **return** XST\_FAILURE;
304. // Call to interrupt setup
305. status = InterruptSystemSetup(&INTCInst);
306. **if**(status != XST\_SUCCESS) **return** XST\_FAILURE;
308. // Connect GPIO interrupt to handler
309. status = XScuGic\_Connect(&INTCInst,
310. INTC\_GPIO\_INTERRUPT\_ID,
311. (Xil\_ExceptionHandler)BTN\_Intr\_Handler,
312. (**void** \*)GpioInstancePtr);
313. **if**(status != XST\_SUCCESS) **return** XST\_FAILURE;

316. // Connect timer interrupt to handler
317. status = XScuGic\_Connect(&INTCInst,
318. INTC\_TMR\_INTERRUPT\_ID,
319. (Xil\_ExceptionHandler)TMR\_Intr\_Handler,
320. (**void** \*)TmrInstancePtr);
321. **if**(status != XST\_SUCCESS) **return** XST\_FAILURE;
323. // Enable GPIO interrupts interrupt
324. XGpio\_InterruptEnable(GpioInstancePtr, 1);
325. XGpio\_InterruptGlobalEnable(GpioInstancePtr);
327. // Enable GPIO and timer interrupts in the controller
328. XScuGic\_Enable(&INTCInst, INTC\_GPIO\_INTERRUPT\_ID);
330. XScuGic\_Enable(&INTCInst, INTC\_TMR\_INTERRUPT\_ID);
332. **return** XST\_SUCCESS;
333. }