110-2 數電實驗 Final Project Report

team09 吳育丞 陳宥辰 王政邦

層級架構

1. Motion Detection

- DE2_115_CAMERA
  - DE2_115_CAMERA
  - v
    - CCD_Capture(鏡頭畫面擷取)
    - I2C_CCD_Config(D5M模組的I2C)
    - I2C_Controller
    - Line_Buffer
    - Line_Buffer1
    - RAW2RGB(把D5M的資料轉成RGB格式)
    - Reset_Delay(輸出reset信號)
    - sdram_pll
    - SEG7_LUT
    - SEG7_LUT_8(七段顯示器)
    - VGA_Controller(處理VGA顯示)
    - Motion_Detection(運動偵測)
    - CalcDir(計算index)
  - Sdram_Control(控制SDRAM資料讀寫)
    - command
    - control_interface
    - sdr_data_path
    - Sdram_Control
    - Sdram_Params
    - Sdram_RD_FIFO
    - Sdram_WR_FIFO
  - Purify
    - Purify(消除雜訊)
    - Purify_buffer_3(儲存先前row資料的Line Buffer)
    - Purify_Line_Buffer
    - Purify_Line_Buffer_bb

2. Statues Game

- DE2_115
  - src
    - Image_Processing
      - display (處理VGA輸出訊號)
      - item, sprite (處理遊戲圖片顯示)
      - rom_async (控制FPGA LUT-ROM)
      - rom_sync (控制BRAM)
    - Audio_Processing

                - AudTop(根據遊戲的Top傳入的訊號, 選擇播放哪一段音效)
                - AudDSP(控制播放速度)
                - AudRecorder(用來把音效寫入SRAM)
                - AudPlayer(播放音效)
                - I2cInitializer(控制I2C protocal, 初始化wm8731)
            - Top (統整所有子模組, 控制遊戲FSM)
            - Debounce(處理按鍵debounce問題)
            - lfsr(用來產生隨機亂數，當作AudDSP播放速度的輸入)
            - SevenHexDecoder(控制七段顯示器所顯示內容)
            - timer(在螢幕上顯示遊戲的倒數計時)
        - pictures (儲存遊戲圖片資料)
            - game1, gmae2, start, win, die, timer(0~9).

3. Arduino Module

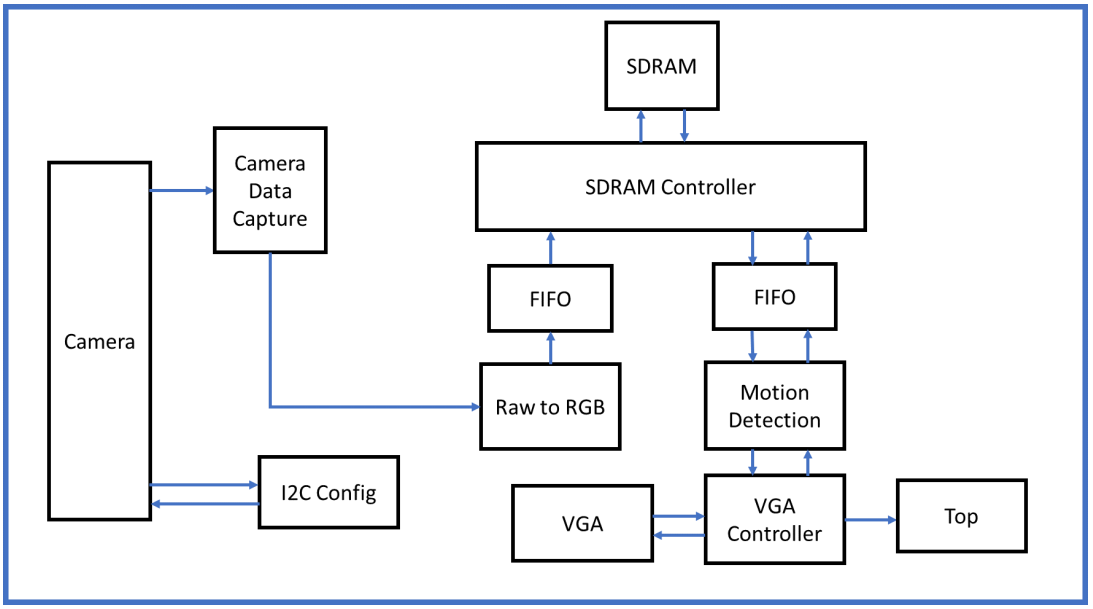
- DE2_115
    - src
        - Arduino_Module
            - DCLab_Servo_Laser.ino(接收 Top 經由GPIO傳輸的訊號, 控制Arduino完成雷射功能)

4. Python

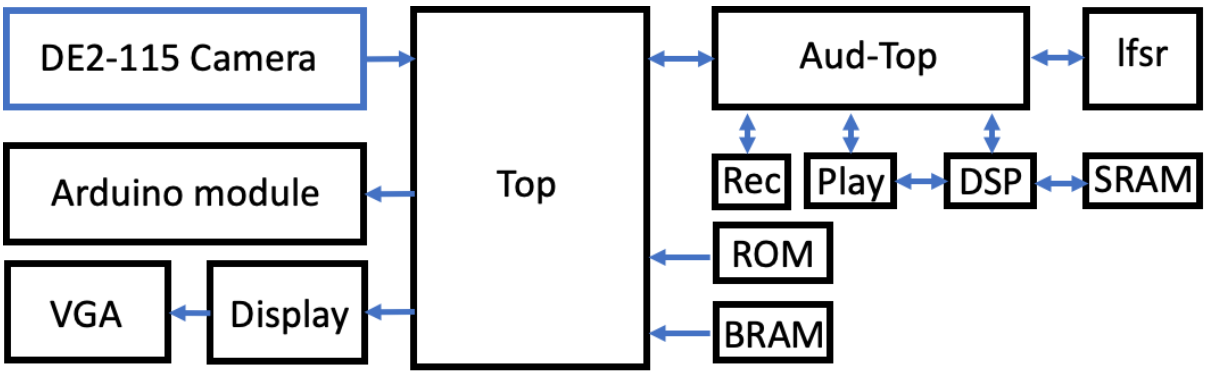
- DE2_115
    - src
        - Python
            - motion_simulation.py(模擬motion detection的演算法之效果)
            - img2mem.py(用來把圖片轉成16進位的.mem檔)[1]

Block Diagram

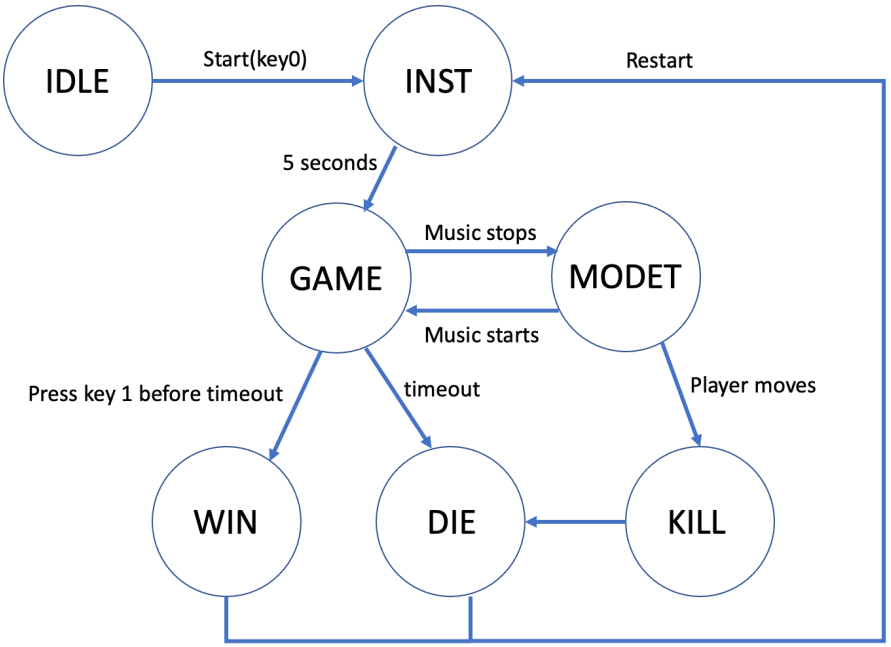
- motion detection



- game



FSM



Fitter Summary

1. Motion Detection

Fitter Summary	
Fitter Status	Successful - Thu Jun 09 22:55:57 2022
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Full Version
Revision Name	DE2_115_CAMERA
Top-level Entity Name	DE2_115_CAMERA
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	2,209 / 114,480 (2 %)
Total combinational functions	1,858 / 114,480 (2 %)
Dedicated logic registers	1,271 / 114,480 (1 %)
Total registers	1271
Total pins	428 / 529 (81 %)
Total virtual pins	0
Total memory bits	71,192 / 3,981,312 (2 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	1 / 4 (25 %)

2. Game

Fitter Summary

Fitter Status	Successful - Thu Jun 09 23:01:45 2022
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Full Version
Revision Name	DE2_115
Top-level Entity Name	DE2_115
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	6,755 / 114,480 (6 %)
Total combinational functions	6,724 / 114,480 (6 %)
Dedicated logic registers	601 / 114,480 (< 1 %)
Total registers	601
Total pins	518 / 529 (98 %)
Total virtual pins	0
Total memory bits	3,351,584 / 3,981,312 (84 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	1 / 4 (25 %)

Timing Analyzer

1. Motion Detection

Slow 1200mV 85C Model Recovery Summary

	Clock	Slack	End Point TNS
1	u6|altpll_component|auto_generated|pll1|clk[4]	-4.297	-1392.330
2	u6|altpll_component|auto_generated|pll1|clk[0]	2.996	0.000
3	CLOCK2_50	9.985	0.000

Slow 1200mV 0C Model Recovery Summary

	Clock	Slack	End Point TNS
1	u6|altpll_component|auto_generated|pll1|clk[4]	-3.561	-1096.243
2	u6|altpll_component|auto_generated|pll1|clk[0]	3.770	0.000
3	CLOCK2_50	10.750	0.000

Fast 1200mV 0C Model Recovery Summary

	Clock	Slack	End Point TNS
1	u6|altpll_component|auto_generated|pll1|clk[4]	-1.172	-302.906
2	u6|altpll_component|auto_generated|pll1|clk[0]	6.237	0.000
3	CLOCK2_50	14.565	0.000

Multicorner Timing Analysis Summary

	Clock	Setup	Hold	Recovery	Removal	Minimum Pulse Width
1	Worst-case Slack	1.361	0.116	-4.297	0.942	4.679
1	CLOCK2_50	14.235	0.181	9.985	0.977	9.272
2	CLOCK3_50	N/A	N/A	N/A	N/A	16.000
3	CLOCK_50	N/A	N/A	N/A	N/A	16.000
4	u6|altpll_component|auto_generated|pll1|clk[0]	1.361	0.116	2.996	2.752	4.679
5	u6|altpll_component|auto_generated|pll1|clk[4]	2.696	0.132	-4.297	0.942	12.182
2	Design-wide TNS	0.0	0.0	-1392.33	0.0	0.0
1	CLOCK2_50	0.000	0.000	0.000	0.000	0.000
2	CLOCK3_50	N/A	N/A	N/A	N/A	0.000
3	CLOCK_50	N/A	N/A	N/A	N/A	0.000
4	u6|altpll_component|auto_generated|pll1|clk[0]	0.000	0.000	0.000	0.000	0.000
5	u6|altpll_component|auto_generated|pll1|clk[4]	0.000	0.000	-1392.330	0.000	0.000

Unconstrained Paths

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	3	3
3	Unconstrained Input Ports	70	70
4	Unconstrained Input Port Paths	361	361
5	Unconstrained Output Ports	131	131
6	Unconstrained Output Port Paths	247	247

2. Game

Slow 1200mV 85C Model Setup Summary

	Clock	Slack	End Point TNS
1	qsys0|altpll_0|sd1|pll7|clk[0]	-7.167	-87.885
2	qsys0|altpll_0|sd1|pll7|clk[1]	-7.109	-167.534
3	AUD_BCLK	-4.600	-22.832
4	KEY[3]	-2.445	-15.501
5	qsys0|altpll_0|sd1|pll7|clk[2]	9992.729	0.000

Slow 1200mV 85C Model Recovery Summary

	Clock	Slack	End Point TNS
1	qsys0|altpll_0|sd1|pll7|clk[0]	-3.553	-1424.052
2	qsys0|altpll_0|sd1|pll7|clk[1]	-3.275	-194.481
3	qsys0|altpll_0|sd1|pll7|clk[2]	-2.314	-46.382
4	AUD_BCLK	0.046	0.000

Slow 1200mV 85C Model Removal Summary

	Clock	Slack	End Point TNS
1	AUD_BCLK	-0.098	-0.404
2	qsys0|altpll_0|sd1|pll7|clk[0]	1.310	0.000
3	qsys0|altpll_0|sd1|pll7|clk[1]	1.335	0.000
4	qsys0|altpll_0|sd1|pll7|clk[2]	1.352	0.000

Slow 1200mV 85C Model Minimum Pulse Width Summary

	Clock	Slack	End Point TNS
1	AUD_BCLK	-3.210	-11.400
2	KEY[3]	-3.000	-3.000
3	CLOCK_50	9.819	0.000
4	qsys0|altpll_0|sd1|pll7|clk[0]	19.705	0.000
5	qsys0|altpll_0|sd1|pll7|clk[1]	41.362	0.000
6	qsys0|altpll_0|sd1|pll7|clk[2]	4999.709	0.000

Slow 1200mV 0C Model Setup Summary

	Clock	Slack	End Point TNS
1	qsys0|altpll_0|sd1|pll7|clk[1]	-6.423	-147.232
2	qsys0|altpll_0|sd1|pll7|clk[0]	-6.421	-78.574
3	AUD_BCLK	-4.504	-22.531
4	KEY[3]	-2.285	-14.459
5	qsys0|altpll_0|sd1|pll7|clk[2]	9993.402	0.000

Slow 1200mV 0C Model Recovery Summary

	Clock	Slack	End Point TNS
1	qsys0|altpll_0|sd1|pll7|clk[0]	-3.071	-1216.838
2	qsys0|altpll_0|sd1|pll7|clk[1]	-2.876	-171.744
3	qsys0|altpll_0|sd1|pll7|clk[2]	-1.945	-38.776
4	AUD_BCLK	0.010	0.000

Slow 1200mV 0C Model Removal Summary

	Clock	Slack	End Point TNS
1	AUD_BCLK	-0.168	-0.856
2	qsys0|altpll_0|sd1|pll7|clk[0]	1.135	0.000
3	qsys0|altpll_0|sd1|pll7|clk[1]	1.145	0.000
4	qsys0|altpll_0|sd1|pll7|clk[2]	1.174	0.000

Slow 1200mV 0C Model Minimum Pulse Width Summary

	Clock	Slack	End Point TNS
1	AUD_BCLK	-3.210	-10.920
2	KEY[3]	-3.000	-3.000
3	CLOCK_50	9.799	0.000
4	qsys0|altpll_0|sd1|pll7|clk[0]	19.692	0.000
5	qsys0|altpll_0|sd1|pll7|clk[1]	41.353	0.000
6	qsys0|altpll_0|sd1|pll7|clk[2]	4999.699	0.000

Fast 1200mV 0C Model Setup Summary

	Clock	Slack	End Point TNS
1	qsys0|altpll_0|sd1|pll7|clk[1]	-3.613	-73.363
2	qsys0|altpll_0|sd1|pll7|clk[0]	-3.584	-44.492
3	AUD_BCLK	-2.090	-9.226
4	KEY[3]	-0.790	-4.658
5	qsys0|altpll_0|sd1|pll7|clk[2]	9996.363	0.000

Fast 1200mV 0C Model Hold Summary

	Clock	Slack	End Point TNS
1	AUD_BCLK	-0.130	-0.130
2	qsys0|altpll_0|sd1|pll7|clk[0]	0.138	0.000
3	qsys0|altpll_0|sd1|pll7|clk[1]	0.173	0.000
4	qsys0|altpll_0|sd1|pll7|clk[2]	0.173	0.000
5	KEY[3]	0.317	0.000

Fast 1200mV 0C Model Recovery Summary

	Clock	Slack	End Point TNS
1	qsys0|altpll_0|sd1|pll7|clk[0]	-2.162	-866.904
2	qsys0|altpll_0|sd1|pll7|clk[1]	-2.162	-133.161
3	qsys0|altpll_0|sd1|pll7|clk[2]	-1.399	-28.322
4	AUD_BCLK	0.794	0.000

Fast 1200mV 0C Model Removal Summary

	Clock	Slack	End Point TNS
1	AUD_BCLK	-0.339	-1.930
2	qsys0|altpll_0|sd1|pll7|clk[0]	0.660	0.000
3	qsys0|altpll_0|sd1|pll7|clk[1]	0.662	0.000
4	qsys0|altpll_0|sd1|pll7|clk[2]	0.665	0.000

Fast 1200mV 0C Model Minimum Pulse Width Summary

	Clock	Slack	End Point TNS
1	AUD_BCLK	-3.000	-14.526
2	KEY[3]	-3.000	-3.000
3	CLOCK_50	9.400	0.000
4	qsys0|altpll_0|sd1|pll7|clk[0]	19.749	0.000
5	qsys0|altpll_0|sd1|pll7|clk[1]	41.445	0.000
6	qsys0|altpll_0|sd1|pll7|clk[2]	4999.780	0.000

Multicorner Timing Analysis Summary

	Clock	Setup	Hold	Recovery	Removal	Minimum Pulse Width
1	Worst-case Slack	-7.167	-0.130	-3.553	-0.339	-3.210
1	AUD_BCLK	-4.600	-0.130	0.010	-0.339	-3.210
2	CLOCK_50	N/A	N/A	N/A	N/A	9.400
3	KEY[3]	-2.445	0.317	N/A	N/A	-3.000
4	qsys0|altpll_0|sd1|pll7|clk[0]	-7.167	0.138	-3.553	0.660	19.692
5	qsys0|altpll_0|sd1|pll7|clk[1]	-7.109	0.173	-3.275	0.662	41.353
6	qsys0|altpll_0|sd1|pll7|clk[2]	9992.729	0.173	-2.314	0.665	4999.699
2	Design-wide TNS	-293.752	-0.13	-1664.915	-1.93	-17.526
1	AUD_BCLK	-22.832	-0.130	0.000	-1.930	-14.526
2	CLOCK_50	N/A	N/A	N/A	N/A	0.000
3	KEY[3]	-15.501	0.000	N/A	N/A	-3.000
4	qsys0|altpll_0|sd1|pll7|clk[0]	-87.885	0.000	-1424.052	0.000	0.000
5	qsys0|altpll_0|sd1|pll7|clk[1]	-167.534	0.000	-194.481	0.000	0.000
6	qsys0|altpll_0|sd1|pll7|clk[2]	0.000	0.000	-46.382	0.000	0.000

Unconstrained Paths

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	0	0
3	Unconstrained Input Ports	24	24
4	Unconstrained Input Port Paths	144	144
5	Unconstrained Output Ports	90	90
6	Unconstrained Output Port Paths	453	453

使用器材

DE2-115*2、TRDB-D5M鏡頭、Arduino Nano、SG90 伺服馬達、雷射激光模組、杜邦線、麥克風、喇叭、螢幕、VGA線、電源線、魷魚遊戲工作人員老大。

遊戲方法與規則

按下key0之後倒數五秒就會進入遊戲，開始播放遊戲音樂，也就是玩家可以移動的狀態。此時遊戲畫面左上角會顯示遊戲所剩的時間，並且木頭人會背對玩家。當音樂播放結束，計時器會暫停，木頭人會轉過來，進入偵測狀態。如果玩家此時有任何動作，就會被木頭人偵測然後用雷射光射擊，也代表遊戲失敗。反之如果木頭人沒有偵測到任何移動的物體，會在五秒後轉回去，此時會重新播放背景音樂，代表玩家可以移動。若玩家在倒數計時器歸零前按下key1，即代表遊戲獲勝。反之如果在時間歸零時仍然沒有按下key1，即使沒有被雷射光射中也宣告遊戲失敗。

實作方法與設計細節

1. Motion Detection:

Algorithm 1: Basic ΣΔ

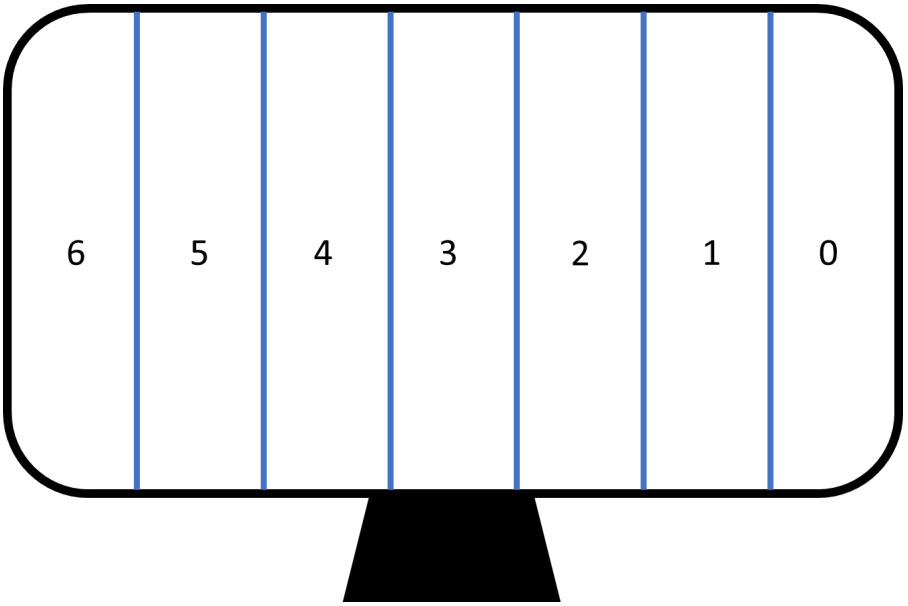
foreach *pixel* x **do** [step #1: M_t estimation]
  if $M_{t-1}(x) < I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) + 1$
  if $M_{t-1}(x) > I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1}(x) - 1$
  otherwise $M_t(x) \leftarrow M_{t-1}(x)$
foreach *pixel* x **do** [step #2: O_t computation]
  $O_t(x) = |M_t(x) - I_t(x)|$
foreach *pixel* x **do** [step #3: V_t update]
  if $V_{t-1}(x) < N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) + 1$
  if $V_{t-1}(x) > N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) - 1$
  otherwise $V_t(x) \leftarrow V_{t-1}(x)$
  $V_t(x) \leftarrow max(min(V_t(x), V_{max}), V_{min})$
foreach *pixel* x **do** [step #4: $\hat{E}_t$ estimation]
  if $O_t(x) < V_t(x)$ **then** $\hat{E}_t(x) \leftarrow 0$ **else** $\hat{E}_t(x) \leftarrow 1$

我們這次的Motion Detection是選擇上圖中的演算法，會這樣的選擇的原因是考量到運算速度還有在FPGA上實作的複雜程度。這個演算法會從相機讀入影像It，然後跟估計出來的背景Mt計算difference，如果difference的絕對值小於variance Vt的話，那這個pixel就會被判定為沒有在運動，反之則為有在運動。Mt跟Vt會根據difference的大小來做更新。

在做完Motion Detection的運算後，我們還有做去除雜訊的步驟，對於每一個pixel，如果周遭的pixels都被判定為沒有在運動的話，則將此pixel也設為沒有在運動的點，此處理降低了圖片上的噪點，使Motion Detection的效果可以更精準。

以下為運動偵測的結果:

2. Index Calculation:
   為了辨識物體運動的左右位置，我們將畫面等比例切成七等分，並分別計算每個區段被偵測為運動的pixel數量，選出擁有最多pixel的區段，如果該區段的pixel數量大於設定的閾值，就將對應的移動訊號傳送給遊戲控制的FPGA板。畫面由右至左分別對應到0到6:



3. Image_Display:
   在圖片顯示的部分，我們參考(https://projectf.io/posts/hardware-sprites/)的作法，利用img2mem.py 將每張圖片分成pixel和palette兩個檔案儲存。palette代表這張圖片使用到的顏色，pixel則代表的是每一個pixel對應到palette中的第幾個顏色。我們將遊戲圖片用16種顏色的palette儲存，每一個顏色由RGB各4 bits儲存，如下圖所示：
   Palette:

```
EE9 ACC CC9 BB9 9CC 9BD 9BC 8BC 9BB AA7 B84 986 776 653 923 233
```

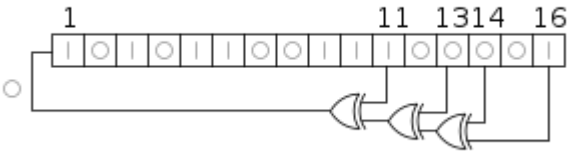
Pixel:

以上是範例的pixel內容，1(紅色框框處)代表此picel的顏色為Palette檔案中的第1種顏色，也就是ACC(紅色框框)，而pixel為5(綠色框框處)則代表Palette檔案中第5種顏色，也就是9BD(綠色框框)，依此類推。
我們把檔案較小的Palette存在容量較小的ROM裡，而把檔案較大的pixel檔案存在容量較大的Block RAM裡，減少latency和記憶體的損耗。
將圖片存入FPGA的設計也使得我們整個遊戲裝置不需要電腦主機的輔助，能夠在各個地方執行，也更貼近實際遊戲的環境設定。

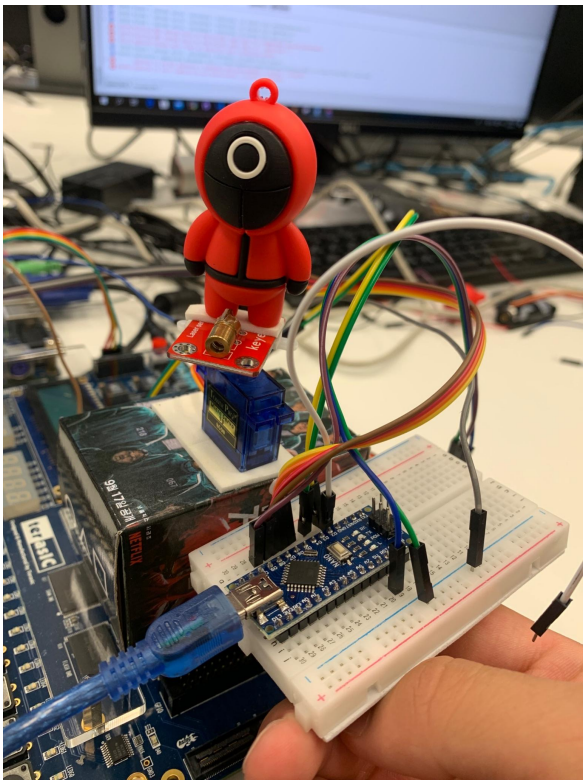
4. Audio_Processing:
遊戲音樂的部分，為了提升遊戲體驗和獲得更好的效果，我們想要在不同的遊戲狀態使用不同的音效，於是採用Lab3的數位錄音機的方法並加以修改。最終的方法會在遊戲設置時先分段錄音，也就是每錄一段我們要的音檔，就按暫停，並記錄當下顯示在七段顯示器上，SRAM儲存的address，用來當作此段音檔的finish address。因此記錄完每段音檔的start/finish address後，Top可以根據不同的state assign AudDSP到不同的address區間，就可以達到在遊戲中播放多段不同音效的效果了，如下圖：

```
//// MUSIC
// GAME
parameter START_ADDR_0    = 20'b0;
parameter FINISH_ADDR_0   = {20'h24FD7};
// KILL
parameter START_ADDR_1    = {20'h24FD8};
parameter FINISH_ADDR_1   = {20'h599CF};
// WIN
parameter START_ADDR_2    = {20'h599D0};
parameter FINISH_ADDR_2   = {20'h7E056};
// DIE
parameter START_ADDR_3    = {20'h7E057};
parameter FINISH_ADDR_3   = {20'h97D8F};
```

5. Random Speed:
為了提升遊戲的趣味性，我們也採用了Lab1所使用的LFSR(Linear Feedback Shift Register)(如下圖)作為隨機亂數的產生器，在每次木頭人轉過去的時候會產生隨機的速度傳給AudDSP，提供不同的音樂速度，達到隨機調整玩家可移動時間的效果，提高遊戲難度和趣味性。



6. Laser Module:
   為了更貼近劇中場景, 我們加入了雷射槍的部分。我們使用了Arduino Nano的開發板、SG-90伺服馬達和雷射模組, 用來控制雷射槍發射的方向和訊號。遊戲控制的 Top module首先會接收到D5M camera提供的訊號, 包含是否有移動的物體以及物體位在哪一個區間, 如果遊戲是在木頭人偵測的狀態, Top 就會透過GPIO將訊號傳遞給Arduino。Arduino收到發射的訊號後, 會把物體區間的訊號轉換成角度訊號, 控制伺服馬達轉到特定的角度後發射雷射光。比起單獨只有螢幕和鏡頭的互動方式, 加入雷射會讓整個遊戲變得更有可玩性, 雷射模組上的人偶娃娃也是一大亮點！



遇到的問題與解決辦法

1. 在處理Motion Detection的時候, 因為影像的資訊轉換為RGB之後會有30個bits, 但是SDRAM的2個FIFO一次最多只能夠讀寫32個bits的資料，如果想要用SDRAM來儲存演算法裡面的Mt跟Vt的話一定會不夠。我們的解決方式為只選擇RGB中G的channel來做Motion Detection, 因此影像buffer的部分只會用到SDRAM的其中一個FIFO, 另外一個FIFO就用來儲存Mt跟Vt, Mt

因為會需要比較精確的數值，所以使用跟影像一樣的bits數，也就是10個bits，FIFO剩下的6個bits就用來儲存Vt。在儲存Mt跟Vt的時候有遇到讀出來的資料顯示在螢幕上像是雜訊的問題，嘗試了很多的方式之後才發現在寫入SDRAM時，clock要設定成跟Motion Detection的module一樣，不然會沒辦法正確的寫入資料，這些細節都是在純軟體設計時不會遇到的。

2. 另外在組合各個module，包括兩塊FPGA板時，由於訊號的clock不同步，也時常會有接收不到訊號的問題。像是當遊戲中的木頭人處在偵測狀態時，會在接收到鏡頭模組偵測到移動物體的訊號瞬間進入到發射雷射光的狀態，同時也要播放相對應的音效。起初音效時不時會失敗，當時還一度以為將SRAM address 區分成多段區域的方法不可行，但是在逐一排錯之後發現是因為Aud的clock比鏡頭模組慢了一倍，所以後來我們讓傳送的高電位pulse訊號多停留幾個cycle之後就成功解決了這個問題。

心得

在修這門課以前就有耳聞數電實驗是一門相當紮實的課程，甚至號稱是電機系大學部最紮實的課，在打這篇report的此時此刻，我回想過去的3個lab, bonus lab和這次的final project，雖然沒有哪一次的lab是很輕鬆的完成，但隨著解決一個又一個bug，我們對於FPGA和Quartus的掌握度也越來越高。從一開始連Quartus建立project都會卡一整個晚上，然後完全不懂qsys的意義，到最後我們成功完成了一個完全由硬體設計的123木頭人遊戲。在最後final project的部分，我們花了整整一週瞭解D5M相機該如何接收到影像並存在SDRAM中，再透過VGA顯示到螢幕上，也花了整整一週瞭解該如何把圖片存在BRAM和ROM中，再透過VGA顯示到螢幕上。看似簡單的一件事，當要脫離電腦，並以不熟悉的硬體裝置實現時，對於FPGA新手的我們而言都變得格外困難，儘管如此我們還是一一克服，成功完成proposal提出的所有功能，非常有成就感。