DSD - HW4 - FIFO

101062124 戴宏穎

Part1

完全照著講義上的定義來實作，基本上以一個大的 FSM 來掌控全局，再利用 assign 訊號，把相關的線路接好。

我設計了一筆測試資料，會對 FIFO 進行過度的 push & pop 來驗證整個設計，當然裡頭也有使用 invalid 的指令來測試相關的錯誤訊息。 自己定義的 opcode 如下：

```
// op_data
// op 00 -> pushing data (在這指令外的 data 不影響行為)
// op 01 -> popping
// op 10 -> idle
// op 11 -> invalid
```

Part2

這部分我在 5/18 作業 Deadline 以前一直沒辦法順利完成，做出來的結果都有問題，就是在放資料的時候會整個亂掉，這幾天一直找 bug 到 Demo 前夕才終於完成了，原來是我的 bit 開太小，所以超過長度的資料會直接不見導致結果一直不正確。

在這個部分，我把 addr 多加一個位數當成是決定要放到哪一組 16-bit memory 的依據。

那把 data 的部分以 32-bit 分開拆成兩個部分，前面 16-bit 給 front 的 ram, 後面 16-bit 給 end 的 ram。

檔案說明

`fifo.v`, `fifo_ctr.v`, `fifo_t.v` 為 `Part1` 部分的程式碼

`my_t.v`, `pattern1.dat` 為自己測資的程式碼

`fifo_64x16.v`, `fifo_64x16_t.v`, `fifo_64x16_ctr.v` 的程式碼為 `Part2` 部分的程式碼

檔案結構

```
hw4_fifo
|____fifo.v
|____fifo_ctr.v
|____fifo_t.v
|____header.v
|____Makefile
|____RAM32x8.v
|____syn.v
|____my_t.v
|____pattern1.dat
|____fifo_64x16_ctr.v
|____fifo_64x16_t.v
|____fifo_64x16.v
|____my_64x16_t.v
```

Discussion

要把 fifo (queue) 換成是 filo (stack) 的話，只要維護 head 的部分就好。本來在 fifo 裏頭，為了不浪費記憶體，利用環狀 queue 的 head, tail 來處理。到了 filo 以後，因為不會有環狀的狀態產生，所以只要維護 head 的移動就可以了。

額外部分 - git version control

我是在做 part2 的時候，因為結果一直不對，所以不停 checkout 回去 initial 的 commit 直到好不容易試出結果，才加進來 git commit 的。

感覺有版本控制以後，亂改程式碼都不用怕了。

```
➜  hw4_FIFO  git log --graph
* commit 8377b17c44f11a2493a7ddb3cd489ee0f2fe755c
| Author: hydai <z54981220@gmail.com>
| Date:   Mon May 26 17:31:13 2014 +0800
|
|     Finish 64x16 mem
|
* commit 276b079b9d545a3f92ec85c3446888345bd645e1
| Author: hydai <z54981220@gmail.com>
| Date:   Mon May 26 12:57:37 2014 +0800
|
|     Add 64x16 module
|
* commit d73145e1b8c7231bf158ed6ca9fe058f86a7169d
| Author: hydai <z54981220@gmail.com>
| Date:   Wed May 7 10:39:16 2014 +0800
|
|     First commit with sample codes
(END)
```