Computer Architecture 2019 Fall - project 2
Team: Cashy Cache & Pipeline Pie

1. Member & Team work

Member				Work
資工四	b05902133	范勝禹	45%	1. Overall debug 2. Cache Controller 3. report : Difficulties Encountered & Solutions
資工三	T08902109	賈成銷	24%	1. report 2. Cache Controller
資工四	B03902002	高新造	31%	1. report 2. Overall debug

2. Cache Controller Implementation

將Project1中的PC.v, Registers.v, Instruction_Memory.v, Data_Memory.v，Testbench.v更換為助教提供的 module

(2)　將 Project1中的 pipeline register皆加上stal的input port（當 cache miss時，dcache_top會輸出stall 的singal給PC和pipeline register，如此便可以stall整個CPU）

(3)　將助教提供的dcache.v, dcache_tag _sram.v, dcache_data_sram.v接上。

(4)　修改dcache.v:

1. 若p1tag==sram_tag&&sram_valid成立, 即 address中的tag和要access. cache entry所存的tag相同，且該 cache_entry為valid，則代表hit，否則為miss

2.若 cache hit，則代表 cache_entry裡存有我們要access的data而若 cache miss，則必須將要access的 data_block自data_memory帶到cache中，此時再access cache entry便會hit，因此不論如何, r_hit_data的值 assign為 sram_cache_data，即要access的 cache_entry所存的值即可.

（i） Read_data和 write_data中，必須找到要 access的data在 block中的哪一個位置，即r_hit_data[(p1_offset>2*32]開始的32個bits (因為每次讀

寫的data為32bits,即4bytes，為了alignment，因此先將p1_offset向右shift2位再乘以4，如此一來 byte_offset必為4的倍數，再乘以8後即為bit_offset，由此開始的32bits即為要讀寫的data)

(A)STATE_IDLE：
初始state，有cache miss以及要對data memory讀或寫的時候(p1_MemRead_i或p1_MemWrite_i是1)才進入STATE_MISS

(B)STATE_MISS：
因為cache miss時會依block是否Dirty分兩條路分別進入STATE_WRITEBACK和STATE_READMISS，到最後都一定會需要讀取data memory，因此將 mem_enable設為1。
如果cache block是dirty(sram_dirty=1)則需要先write back，要先將該block data存回data memory中與cache entry相對應的位置，並進入STATE_WRITEBACK，此時會將mem_write和write_back設為 1 。
如果cache block不是dirty(sram_dirty=0)則省略write back，直接進入STATE_READMISS，此時將mem_write和 write_back設為0 。

(C)STATE_READMISS：
去看data memory是否已經準備好將要存取的block data存入cache中(並更新一列cache entry)，若已經準備好(mem_ack_i=1)，就進入STATE_READMISSOK，此時因為不用再存取data memory而是要存取cache，因此mem_enable設為0、 cache_we設為1。

若data memory還沒準備好(mem_ack_i=0)，則繼續保持STATE_READMISS。

(D)STATE_READMISSOK：
進入這個state代表需要的block data已經從data memory存入cache中(並更新好一列cache entry)，因此將 cache_we設為0，回到 STATE_IDLE。

(E)STATE_WRITEBACK：
若mem_ack_i=1，代表dirty block data已經write back回data memory中，現在data memory中與cache entry相對應位置的block data已經是一樣

的了，下一步是把要存取的block data自data memory存入cache，因此將mem_write和write_back都設回0，進入STATE_READMISS。
若mem_ack_i=0，代表dirty block data還沒write back回data memory中，則繼續保持 STATE_WRITEBACK。

3. Difficulties Encountered & Solutions of This Project

##(A) Remember to add `mem_enable <= 1'b1` after `STATE_MISS`

##(B) Change TA's PC.v to my PC.v in Project 1

##(C) When writing data from 32 bits to 256 bits : `w_hit_data`, we shall use `p_data_i` , which is an input wire , not `p_data` , which is a register modified by `r_hit_data`.

So the codes below only have one difference : `p_data_i` and `p_data`
At line 140, if we use `p1_data` , then our Data_Memory is never written.

```
126   // read data :  256-bit to 32-bit
127   always@(p1_offset or r_hit_data) begin
128       // TODO: add you code here! (p1_data=...?)
129       for(i = 0 ; i < 32 ; i = i + 1) begin
130               p1_data[i] = r_hit_data[ (p1_offset >> 2) * 32 + i ];
131       end
132   end
133
134
135   // write data :  32-bit to 256-bit
136   always@(p1_offset or r_hit_data or p1_data_i) begin
137       // TODO: add you code here! (w_hit_data=...?)
138       w_hit_data = r_hit_data;
139       for(i = 0 ; i < 32 ; i = i + 1) begin
140               w_hit_data[ (p1_offset >> 2) * 32 + i ] = p1_data[i];
141       end
142   end
```

Then we change to `p_data_i` at line 135, then we got the correct results.

```
118  // Read data : 256-bit to 32-bit.
119  always @(p1_offset or r_hit_data)
120  begin
121          // Add your code here!!! (p1_data = ..... ?)
122          for (i = 0 ; i < 32 ; i = i + 1)
123          begin
124                  p1_data[i] = r_hit_data[(p1_offset >> 2) * 32 + i];
125          end
126  end
127
128  // Write data : 32-bit to 256-bit.
129  always @(p1_offset or r_hit_data or p1_data_i)
130  begin
131          // Add your code here!!! (w_hit_data = ..... ?)
132          w_hit_data = r_hit_data;
133          for (i = 0 ; i < 32 ; i = i + 1)
134          begin
135                  w_hit_data[(p1_offset >> 2) * 32 + i] = p1_data_i[i];
136          end
137  end
```

////////////////////////////////////// END //////////////////////////////////////
