HW4 Report
林哲毅
R11922148

Note: 沒有改memory

Results

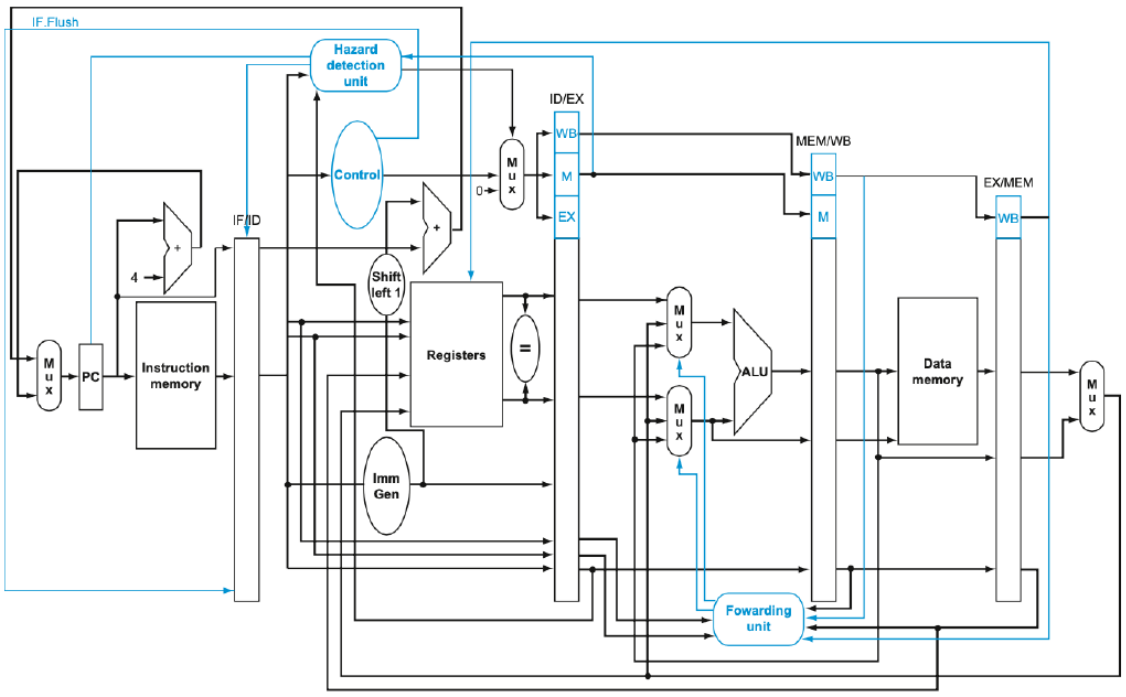
Workload	Cycles
Workload1	15674
Workload2	32078
Workload3	40690

Area = 1463.00
Delay = 1411.47ps

Note: 原本Area跟Delay都異常地低，大概是15跟200左右，照著notification的文件修改完之後，雖然Area離7000還有一段距離，但看起來比較合理一點，warning message沒有出現、cpu_syn.v中也沒有DLATCH了

Pipeline Design

- Pipeline架構是參考spec裡面的圖，如下圖所示
- 拆成IF, ID, EX, MEM, WB五個stages
- 每個stage設定4個cycles去跑，因為data memory跟instruction memory的latency，所以再少會有錯
- 跟圖中比較不一樣的是，ImmGen出來的線要連到ALU Source下面的Mux，也就是這個Mux應該要有4個input，而我是用2個Mux來實作這個Mux
- 變數很多，所以變數的命名可能很混亂，命名的幾個原則:
    - 在哪個stage產生的data就在前面冠上那個stage的名字，eg. ID_imm
    - pipeline register裡面的data就在前面冠上那個pipeline register的名字，eg. IDEX_read_data1_r
    - 各個module，凡是input就在前面冠上i_，eg. i_data_a；onput就在前面冠上o_，eg. o_data



Critical Path

```
ABC: Start-point = pi4011 ($abcloop$59966).  End-point = po125 (\IF_PC_r [63]).
ABC: + write_blif <abc-temp-dir>/output.blif
```

Data Hazard

- 一般的data hazard
  - 可以用forwarding來解決，也就是不等指令完成，就提早把資料送回去
  - 在Forwading Unit中，去偵測EX hazard與MEM hazard，也就是拿ID/EX的來源register，去比對EX/MEM與MEM/WB的目的register
- load-use data hazard
  - 需要forwarding跟stall才能解決
  - 實作一個Hazard Detection Unit在ID stage中運作，檢查ID/EX的MemRead是否被設成1，再檢查IF/ID的來源register是否與ID/EX的目的register相同

Control Hazard

- 在branch進到ID stage時就去判斷是否要跳，也就是去看兩個data是否相等
- 如果要跳，則PC Source Mux讓branch target通過；如果不跳，則是讓PC+4通過

Workloads

- Workload1: 100次iterations，每次指令數最多的程式
- Workload2: 500次iterations，有很多branch，會跳來跳去

- Workload3: 1000次iteratoins，每次指令數最少的程式
- Branch predictor對Workload2影響最大

Multiple Stages

不一定，因為stage數增加也會讓設計的complexity增加，所以frequency也有可能下降，舉例來說有更多的hazard，就要多做forwarding unit去處理，若forwarding處理不了的，還要stall pipeline，這些都會讓frequency下降