آزمایشگاه معماری کامپیوتر

آزمایش پنجم : افزودن بخش forwarding به پردازنده

پارسا حداد منفرد : 810198380

ایمان رسولی پرتو : 810199425

بخش اول کد وریلاگ بخش forwarding :

عملکرد ماژول forwarding بدین صورت است که در مواقعی که وابستگی داده ای ما بین دستورات پایپ برقرار است به جای نگهداشتن پایپ توسط سیگنال هازارد داده مورد نیاز دستورات را از میانه پایپ تامین و ارسال می کند.

در حالت کلی اگر مخاطره داده ای خواندن پس از نوشتن (read after write) داشته باشیم مقدار داده ای که باید قبل از خواندن در رجیستر فیال نوشته شود در یکی از سه مرحله زیر است

الف) در قسمت حافظه و به عنوان خروجی واحد محاسباتی (ALU result)

ب) در قسمت باز نویسی (write back)

ج) در قسمت حافظه و به عنوان خروجی خوانده شده از حافظه

به وسیله ماژول forwarding می توان دیتا های موجود در قسمت های الف و ب را از داخل پایپ استخراج و در زمان مناسب به واحد محاسباتی رساند و از توقف پایپ جلوگیری نمود

اما چنانچه داده مورد نظر در حالت ج باشد به ساختار عملکرد حافظه نمی توانیم داده مد نظر را در همان سیکل در اختیار واحد محاسباتی قرار دهیم لذا باید یک سیکل پایپ را توسط هازارد متوقف کنیم و سپس با رسیدن داده مد نظر به قسمت بازنویسی واحد forwarding آن را در اختیار بخش محاسباتی قرار میدهد

بنا بر توضیحات بالا ماژول forwarding به شکل زیر طراحی می شود

```
module Forwarding_unit(src1_id_reg, src2_id_reg, WB_EN_exe_reg, WB_EN_mem_reg, Dest_exe_reg,
          Dest_mem_reg, fu_en, Sel_src1, Sel_src2);

  input [3:0] src1_id_reg, src2_id_reg;
  input WB_EN_exe_reg, WB_EN_mem_reg;
  input [3:0] Dest_exe_reg, Dest_mem_reg;
  input fu_en;

  output reg [1:0] Sel_src1, Sel_src2;

  always@*begin
    Sel_src1 = 2'b00;
    Sel_src2 = 2'b00;
    if(fu_en)begin
      if((src1_id_reg == Dest_exe_reg) & WB_EN_exe_reg)begin
         Sel_src1 = 2'b01;
      end
      else if((src1_id_reg == Dest_mem_reg) & WB_EN_mem_reg)begin
          Sel_src1 = 2'b10;
      end
      if((src2_id_reg == Dest_exe_reg) & WB_EN_exe_reg)begin
          Sel_src2 = 2'b01;
      end
      else if((src2_id_reg == Dest_mem_reg) & WB_EN_mem_reg)begin
          Sel_src2 = 2'b10;
      end
    end

  end


endmodule
```

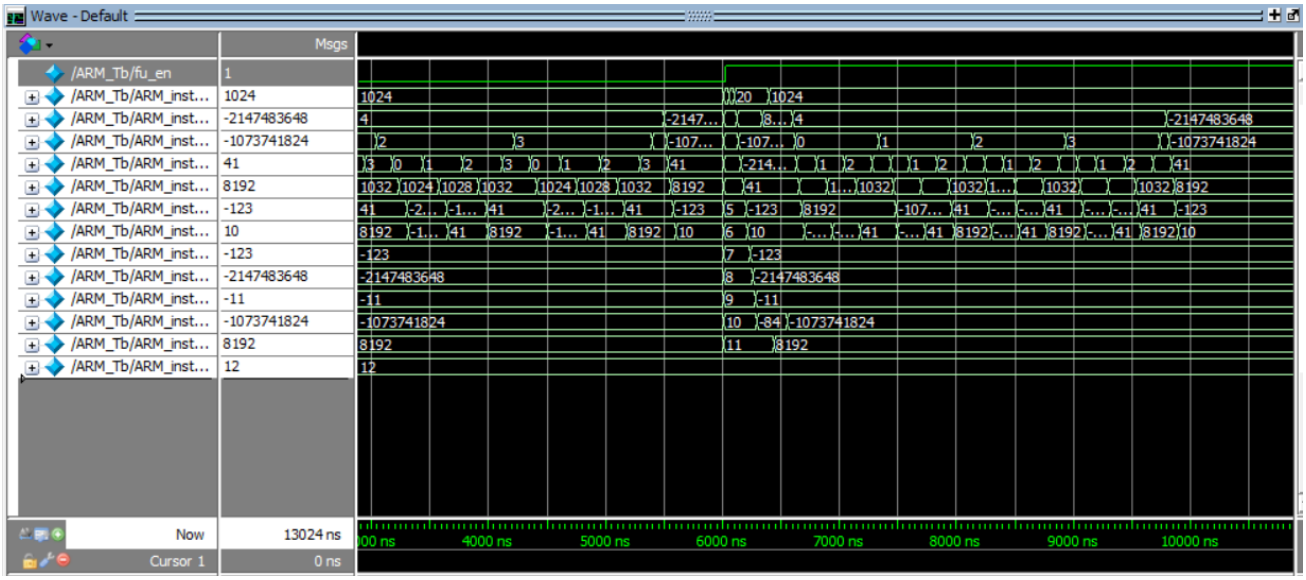
شکل 1 : کد وریلاگ ماژول forwarding

همچنین همانطور که گفته شد حضور ماژول forwarding مخاطره های داده ای را رفع می کند لذا باید ماژول hazard detection را نیز مطابق با آن تغییر دهیم ساختار این ماژول در حضور forwarding unit به شکل زیر است

```
module hazard_Detection_Unit(src1_id, src2_id, Exe_Dest, Exe_WB_EN, move_id, Mem_Dest,
          Mem_WB_EN, Two_src_id, fu_en,EXE_MEM_R, hazard);
  input [3:0] src1_id, src2_id;
  input [3:0] Exe_Dest;
  input Exe_WB_EN, move_id;
  input [3:0] Mem_Dest;
  input Mem_WB_EN;
  input Two_src_id;

  //Add Forwarding unit
  input fu_en;
  input EXE_MEM_R;
  output reg hazard;

  //hazard <= mem_read_mem && ((exe_dest == src1) || (two_src && exe_dest == src2));

  always@* begin
    hazard = 1'b0;
    if(fu_en)begin
     if(EXE_MEM_R & ((src1_id==Exe_Dest)|(src2_id==Exe_Dest))) begin
       hazard = 1'b1;
     end
   end
```

شکل 2 : تغییرات hazard detection در حضور forwarding unit

همچنین نتایج شبیه سازی در modelsim برای هر دو حالت حضور و عدم حضور forwarding unit به شکل زیر است

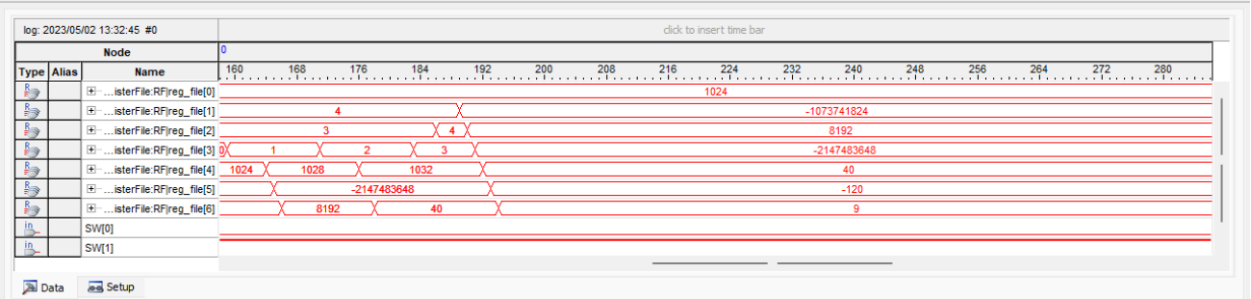


شکل 3 : نتایج شبیه سازی در modelsim

بخش دوم سنتز:

پس از شبیه سازی، پردازنده را در quartus سنتز کرده و نتایج را در signal tab بررسی می کنیم

شکل 4 : خروجی پردازنده بدون استفاده از forwarding unit



شکل 5 : خروجی پردازنده با استفاده از forwarding unit

همانطور که در شکل های بالا مشخص است در حالتی که forwarding unit در پردازنده حضور ندارد زمان اتمام پردازش در حدود 280 دوره کلاک است و پس از افزودن forwarding unit این زمان به حدود 195 دوره کلاک کاهش می یابد

بر این مبنا می توان افزایش بازدهی پردازنده را در حدود 30 درصد برآورد کرد

بخش سوم میزان هزینه سخت افزاری:

در اینجا تعداد المان های مصرفی FPGA در حضور و عدم حضور forwarding unit را بررسی می کنیم

Flow Status	Successful - Fri May 05 17:53:52 2023
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	ARM_Processor
Top-level Entity Name	ARM_Processor
Family	Cyclone II
Device	EP2C35F484C8
Timing Models	Final
Total logic elements	2,350 / 33,216 (7 %)
Total combinational functions	2,226 / 33,216 (7 %)
Dedicated logic registers	815 / 33,216 (2 %)
Total registers	815
Total pins	4 / 322 (1 %)
Total virtual pins	0
Total memory bits	2,048 / 483,840 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

شکل 6 : تعداد المان های منطقی مصرف شده در حضور forwardimg unit

Flow Status	Successful - Fri May 05 18:10:43 2023
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	ARM_Processor
Top-level Entity Name	ARM_Processor
Family	Cyclone II
Device	EP2C35F484C8
Timing Models	Final
Total logic elements	2,248 / 33,216 (7 %)
Total combinational functions	2,038 / 33,216 (6 %)
Dedicated logic registers	811 / 33,216 (2 %)
Total registers	811
Total pins	4 / 322 (1 %)
Total virtual pins	0
Total memory bits	2,048 / 483,840 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

شکل 7: تعداد المان های منطقی بدون حضور forwarding unit

با توجه به نتایج بالا می توان گفت هزینه سخت افزاری در حدود 9.2 درصد افزایش یافته است

بخش چهارم محاسبه میزان کارایی بر هزینه :

با توجه به قسمت های سه و چهار نرخ افزایش کارایی برابر 30 درصد و نرخ افزایش هزینه سخت افزاری برابر 9 درصد است بنابراین میزان کارایی بر حسب هزینه حدودا برابر 3.3 خواهد بود

بخش پنجم روش پیشنهادی افزایش عملکرد:

هانند آنچه در جلسه مطرح شد یکی از راه های افزایش عملکرد پردازنده ها استفاده از روش out of order است در این ساختار به کمک قطعات سخت افزاری دستورات پیش از ارسال به پردازنده از نظر وابستگی داده ای به یکدیگر مورد بررسی قرار میگیرند و دستوراتی که وابستگی داده به یکدیگر ندارند خارج از ترتیب وارد پایپ شده و از این رو نیاز به توقف پایپ در حین اجرای دستورات وجود ندارد

شکل زیر خروجی نرم افزار شبیه ساز gem5 را برای اجرای یک برنامه تست روی دو پردازنده یکی به صورت in order و دیگری به صورت out of order نشان می دهد که حاکی از افزایش تقریبا 3 برابری سرعت در پردازنده های out of order است

```
---------- Begin Simulation Statistics ----------
simSeconds                                   0.325332
simTicks                                  325332457000
finalTick                                 325332457000
simFreq                                  1000000000000
hostSeconds                                    982.48
hostTickRate                                331134433
hostMemory                                    695244
simInsts                                    171966832
simOps                                      262462940
hostInstRate                                   175034
hostOpRate                                     267144
```

شکل 8 : زمان مصرفی در پردازنده های in order برای یک برنامه تست

```
---------- Begin Simulation Statistics ----------
simSeconds                                   0.099743
simTicks                                  99743431500
finalTick                                 99743431500
simFreq                                  1000000000000
hostSeconds                                   7287.91
hostTickRate                                13686144
hostMemory                                    700628
simInsts                                    171966832
simOps                                      262462940
hostInstRate                                   23596
hostOpRate                                     36013
```

شکل 9 : زمان مصرفی در پردازنده های out of order برای یک برنامه تست