بسم الله الرحمن الرحیم

ساختار کامپیوتر

امیر حسین رستمی

96101635

گزارش کار فاز سوم پروژه

دکتر موحدیان عطار

دانشگاه صنعتی شریف بهار 98

پاسخ خواسته ی یک : فقط ماژول های متفاوت با فاز قبلی را در اینجا پیاده سازیشان را آورده ام و الباقی مشابه فاز قبل می باشد.

لازم به ذکر است که با توجه به خصلت پایپ لاینی و پردازش کلاک محور این بخش بنده ماژول های کنترلر و ALU را به داخل کلاک می روند! و پیاده سازیشان در فایل آپلود شده ضمیمه شده است.

```
module triStateBuffer #(parameter WIDTH = 8)
(input clk, reset, enable, input [WIDTH-1:0] inp, output reg [WIDTH-1:0] out);
// it is a bus of triState buffers so the input is a bus.
// and also it returns 0 instead of High ampedance (Z) --> different from triStateBuffer
always @(posedge clk,posedge reset)
begin
    if(reset)
        begin
            out <= 0;
        end
    else
        if (enable == 1'b1)
        begin
            out <= inp;
        end
end
endmodule

module Mux2_1 #(parameter WIDTH = 8)
(input  [WIDTH-1:0] inp_ZeroSentical, inp_OneSentical, input senticalSignal, output [WIDTH-1:0] out);
assign out = senticalSignal ? inp_OneSentical : inp_ZeroSentical;
endmodule

module Adder( input [31:0] firstInput , input [31:0] secondInput , output [31:0] out
    );
assign out = firstInput + secondInput ;
endmodule
```

پاسخ خواسته ی دو :

(پیاده سازی ماژول ها که در فایل ارسالی کامل آمده است).

برای حل این فاز از مرجع اصلی خود درس کتاب هریس استفاده کردم و طبق گفته های کتاب ماژول های زیر را پیاده سازی کردم.

1- دیتا هازارد
2- ماکس های دو به یک
3- شبه! tirStateBuffer
4- Adder جهت همان جمع کردن پی سی با 4 می باشد!(pcplus4)

قلب پیاده سازی این بخش، دیتا هازارد پایپ لاین می باشد.

- پیاده سازی دیتا هازارد : همانطور که می دانید جهت پیاده سازی این قسمت ها نیاز به دانستن نکته های فوروارد و Stall می باشد :

- نکات فوروارد :

- Forwarding logic for *ForwardAE*:

```
if      ((rsE != 0) AND (rsE == WriteRegM) AND RegWriteM)
    then   ForwardAE = 10
else if ((rsE != 0) AND (rsE == WriteRegW) AND RegWriteW)
    then   ForwardAE = 01
else       ForwardAE = 00
```

Forwarding logic for *ForwardBE* same, but replace *rsE* with *rtE*

- نکات stall :

```
lwstall =
  ((rsD==rtE) OR (rtD==rtE)) AND MemtoRegE

StallF = StallD = FlushE = lwstall
```

توضیحات شرط ها :

شرط های قسمت فوروارد و Stall در اصل در این جهت است که رخ داد و نیاز به فوروارد و Stall را تشخیص دهد

حال چگونه ؟

مثلا قسمت stall رو نگاه کنید چه زمانی نیاز به ایست است؟ زمانی که رجستر مقصد در دستور Rtype با یکی از رجیستر های مبداعی برابر باشد که این معادل تکه شرط دو بخشی زیر است :

((rsD == rtE) or (rtD == rte))

For example : r1,r2,r3 → two stall need states :

1 : r1 == r2

2 : r1 == r3

اما اگر نیاز به memToReg باشد باید این stall رخ بدهد وگرنه که خب اگه نیاز نباشه چیزی در رجیستر مقصد که مشابه با حداقل یکی از رجیستر های مبدا است نوشته نشود خب اصلا دیگر نیاز به Stall نخواهد بود.

لذا در نهایت باید با memToReg حاصل را & کنیم :

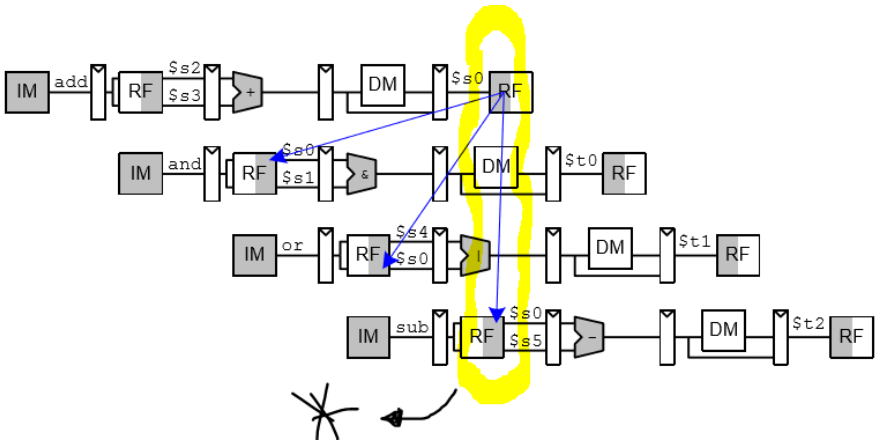
لذا پاسخ نهایی به شکل رو به رو خواهد بود :

➔ ((rsD == rtE) or (rtD == rtE)) and memtoReg

که همان شرط مشاهده در lwstall می باشد.

مشابه توضیح داده شده در بالا هم برای حالت فوروارد وجود دارد که مثلا شرط اول آن را بررسی می کنیم :

هدف از فوروارد چی بود؟ این بود که اگر رجیستری زود اماده شد بفرستیمش مراحل بعد: مثلا به شکل زیر توجه کنید :



حال از کجا میشه این انتقالات رو انجام داد؟

1- Memory stage usage → also known as using the alu out!
2- Writeback stage .
3- Using negedge of clock in order to write into registerfile right at the same time as writing into it. Look at the yellow part above.

شروط ذکر شده در قسمت فوروارد هم همانند توضیحات ذکر شده برای قسمت stall می باشد بدین ترتیب که اول تشخیص می دهیم ایا نیاز است که اصلا فوروارد رخ بده یا نه بعد که این را تشخیص دادیم میایم ماخذ دریافت فوروارد دیتا را تشخیص می دهیم یعنی اینکه از خروجی ALU بگیریم یا خروجی مموری و قس علی ذلک.

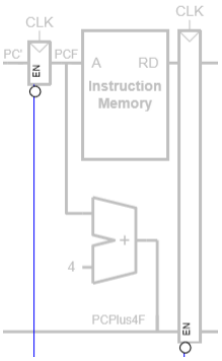
چالش های پیشرو :

اصلی ترین چالش های پیشرو :

- گرفتن x :
  - دلایل : چون در اثر انتقال بین کلاک داریم که مثلا متغیر های کنترلی decode از fetch و execution از decode و به همین ترتیب متغیر های کنترلی مرحله ی بعد از مرحله ی قبل استفاده می کند و چون در هر لبه ی کلاک داریم که متغیر ها مرحله مرحله تعیین می شوند و جلو پیش میروند اما اگر اندکی حواسمون نباشد بلافاصله X تولید و مرحله مرحله progpagate می شود و این x به جلو تولید می شود و مانع از عملکرد صحیح مدار می گردد لذا توجه هر چه بیشتر به اینکه دیتا هازاد و متغیر های ماکس چه باشند و دقیق پیاده سازی شده باشند خیلی به عملکرد هرچه درست مدار و تولید کمتر x منتها می گردد.
- گرفتن خروجی مخالف با خروجی تست بنچ تحویلی :

- بررسی بیشتر ماژول های کنترلر و ALU
- داخل کلاک قرار دادن این ماژول ها

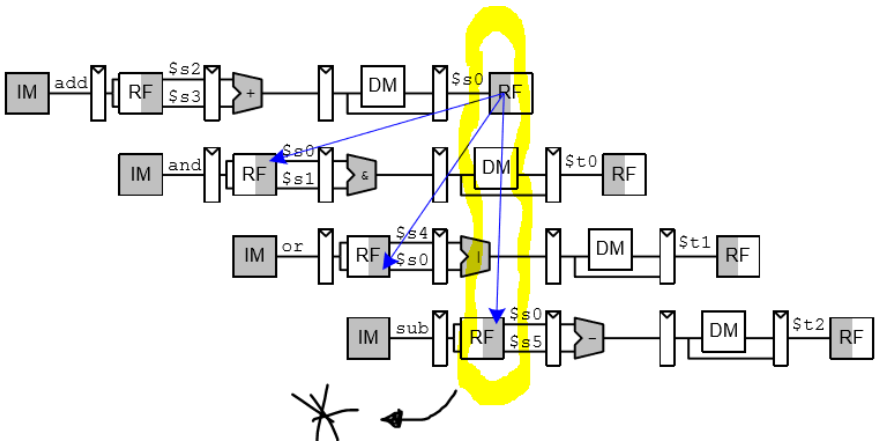
- دقت هر چه بیشتر در active low یا active High بودن متغیر های اعمالی دیتا هازارد به دکل های انتقال متغیر های کنترلی از fetch به decode و decode به execution و ...



همانطورکه مشاهده می کنید داریم که enable ها در برخی مراحل activeLow و activeHigh می باشند و عدم توجه به این نکته باعث می شود که x تولید شود.

هم چنین توجه کنید که یکی از راه های هازارد که از طریق انتقال عمودی در بلوک امکان پذیر است به کمک negedge کلاک این کار را می کند که عدم توجه درست به این باعث می شود عملکرد فوروارد ناقص انجام شود و این موضوع باعث میشود مثلا در خروجی برخی بیت ها یا همه ی انها ایکس شود.این نکته مربوط به قسمت زرد رنگ شکل زیر است :

قسمت زرد رنگ:



پاسخ خواسته ی 3 : در اصل وقتی چنین حالتی رخ بدهد و داشته باشیم که خواندن و نوشتن در مموری با چندین تاخیر رخ بدهد مثلا فرض کنید فقط خواندن از مموری با 3 تاخیر رخ بدهد در اینصورت چه اتفاقی می افتد؟ دیگر نیاز نداریم که نگران رخ داد concurrency باشیم چرا که همین که مموری سه تا تاخیر دارد یعنی اینکه چی؟ یعنی انگار سه پالس کلاک stall داریم و این باعث می شود که خود به خود این ایرادات دستکاری موازی از بین برود.

حالا اگر کمتر از یک پالس کلاک داشتیم چه؟

خب خیلی سادس مثلا فرض کنید خواندن و نوشتن در مموری 1 پالس کلاک تاخیر داشته باشد در اینصورت هم چنان نیاز به بررسی رخ داد هازاد هست اما در این حالت کافی است یک کلاک بررسی را زودتر انجام بدیم یعنی اینکه مثلا این انتقال داده بین بلوک ها یک کلاک زودتر انجام شود و این یعنی اینکه enable های بلوک ها را یک کلاک بلوکا شیفت بدهیم.

در اصل در برآیند اینکه اگر میزان تاخیر ها از حداکثر تعداد کلاک لازم جهت پرهیز کردن از دستکاری موازی بیشتر باشد نیاز نیست که اصلا بلوک هازاردی داشته باشیم! چون انگار مدار ذاتا چند کلاک Stall دارد در سایر حالت ها هم بسته به اینکه چه تعداد تاخیر در خواندن داریم لازم است تا بلوکا مدار تشخیص هازارد را شیفت بدهیم و ...