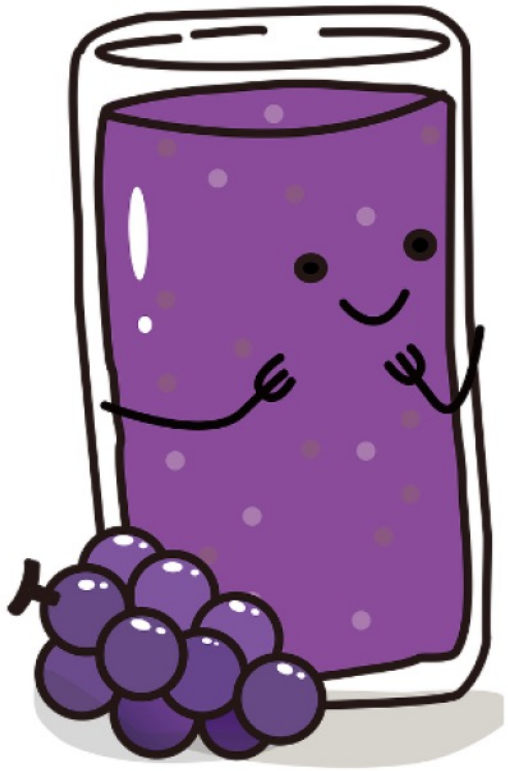
RISCV-32I プロセッサ設計

電気情報工学科４年

ムハマドナウファル

1TE18239M

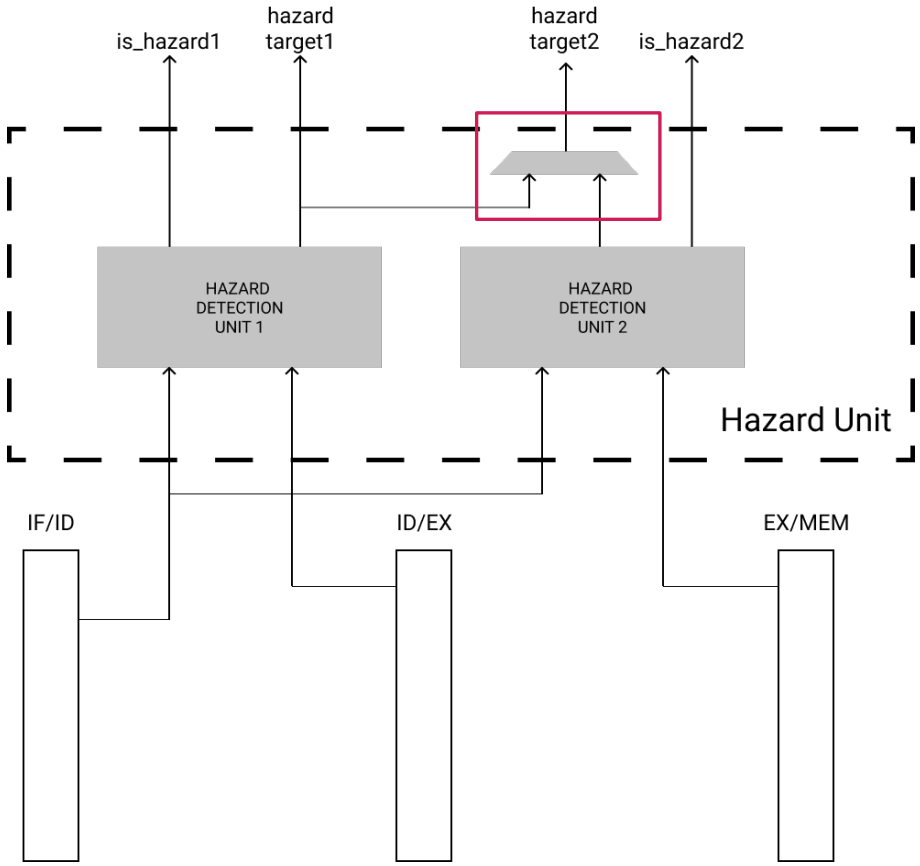
目次



プロセッサ名

GRAPY

データハザード・その解決方法
ハザード検出



赤い四角は注意すべきところである。
同様なユニットからできたハザード検出には
同じターゲットレジスタを出力する可能性がある。

例：

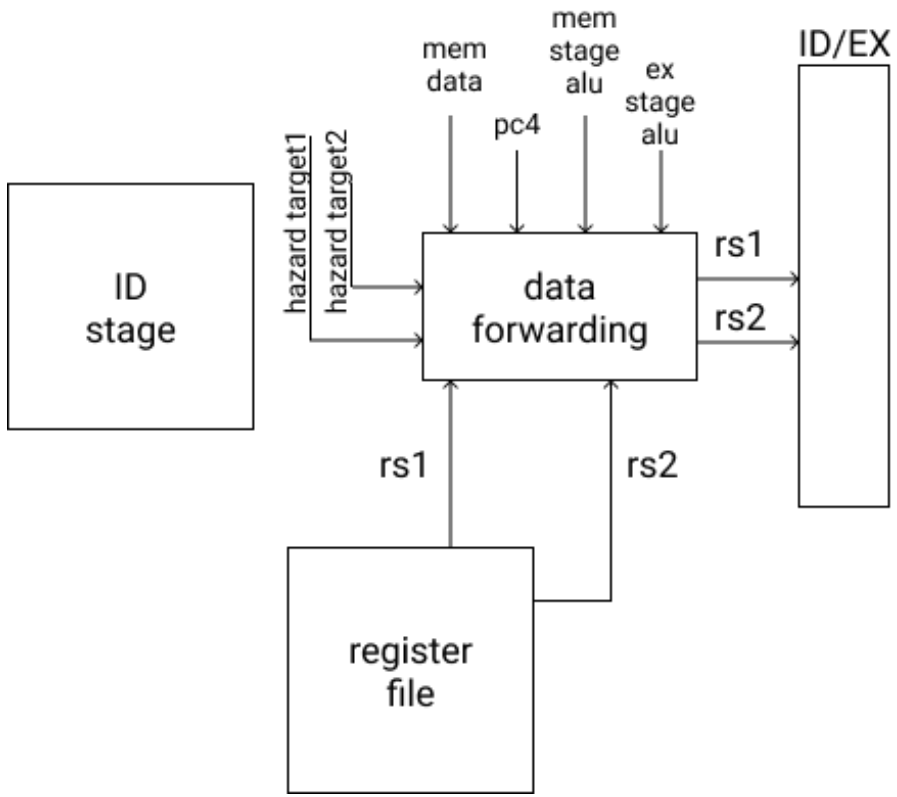
EX/MEM	add	x5, x6, x7
ID/EX	add	x5, x5, x7
IF/ID	add	x9, x5, x7

FROM_EX_RS1	3'd1
FROM_EX_RS2	3'd2
FROM_MEM_RS1	3'd3
FROM_MEM_RS2	3'd4

結果：rs1 = 1(unit1) と 3(unit2) になる

これを選択

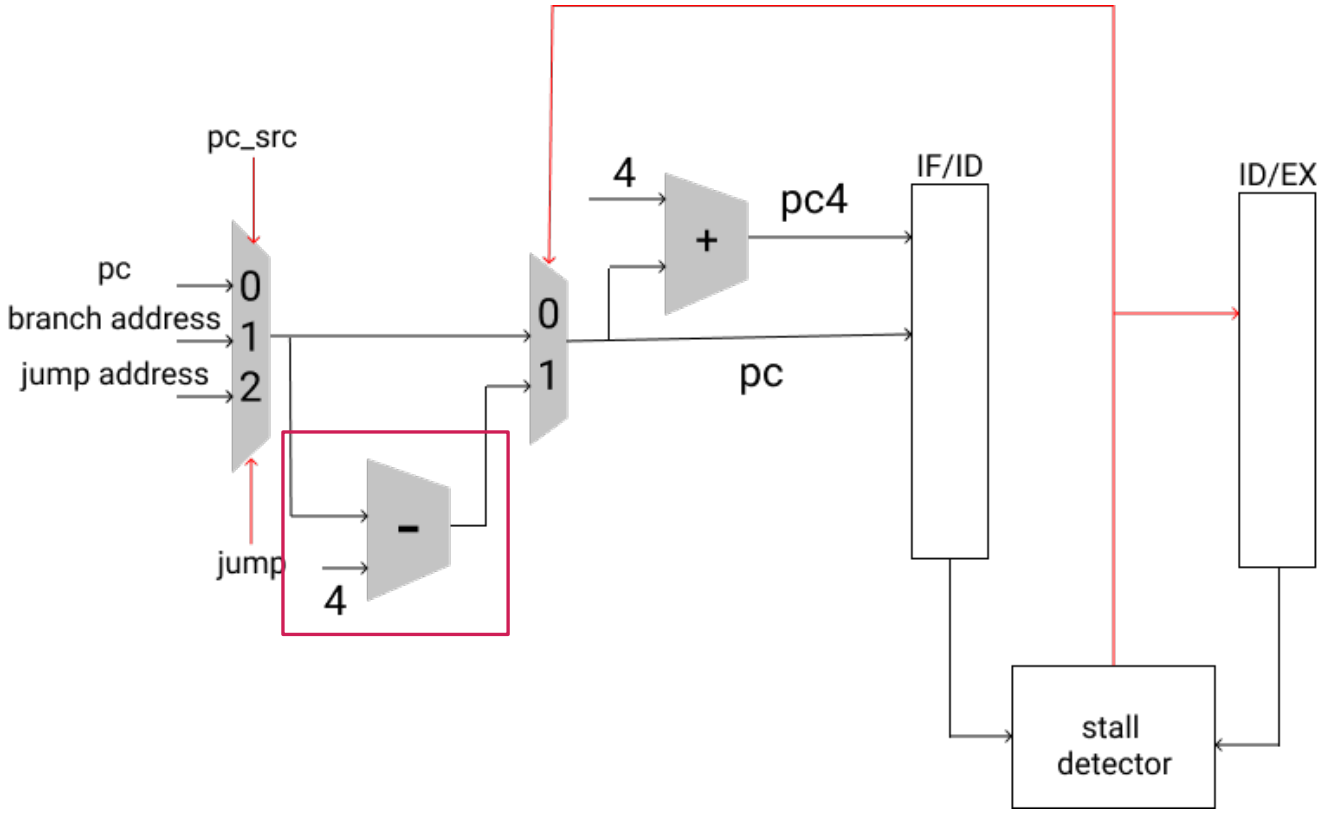
データハザード・その解決方法
データフォワーディング



クリティカルパースになる！

他の信号に比べたら、パイプラインに入る前に
かなりの論理ゲートを通過しないといけなくなるから。

データハザード・その解決方法
パイプラインストール



減算機が要るのは新たに入ったPC値が
既に4と加算されたから、次の命令を指している

効率が悪く、パイプラインストールではなく、
ステージストールである。

特にメモリアクセスには1クロックサイクル以上、
かかるときに、困る。

改善したこと

面積の削減

面積の削減に挑戦した。面積が小さいほど消費電力も削減できる。

もちろん、チップのコストも下がるだろう。

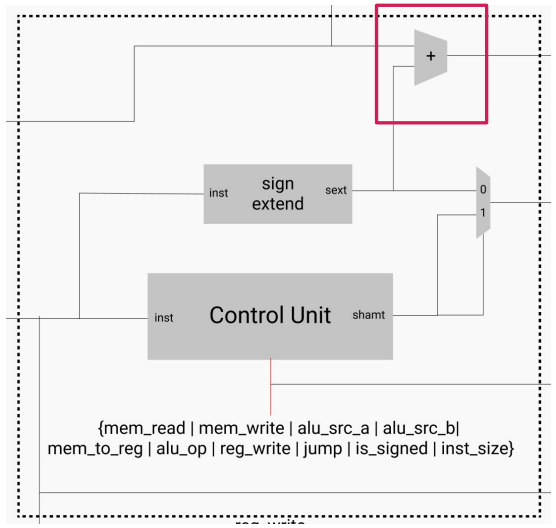
ただVerilogの工夫をしただけである。

- Don’t Care 信号をなくした。（特にmultiplexer）
- 一部の制御信号も変えなといけない。
- If-else文をconditional operator(条件文？True : False)に書きかえた。

結果は約29%の面積が削減できた。

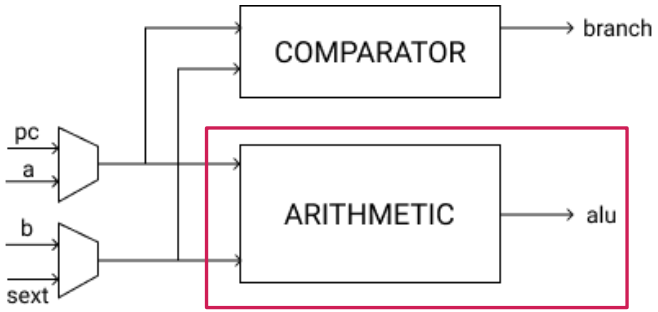
改善したこと
クリティカルパース(フォワーディングなし)

分岐先アドレスの計算



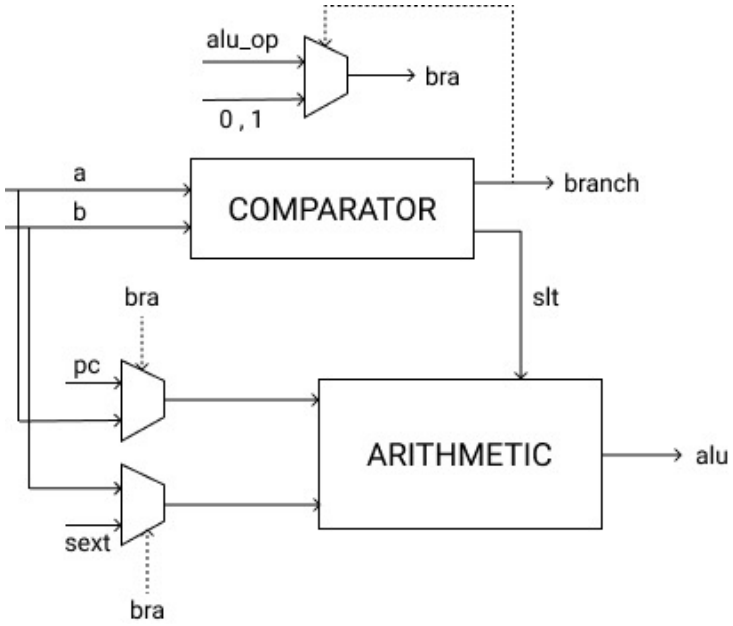
ID Stage

+



EX Stage

もったいない！



検証の結果

1クロックサイクル = 10ns

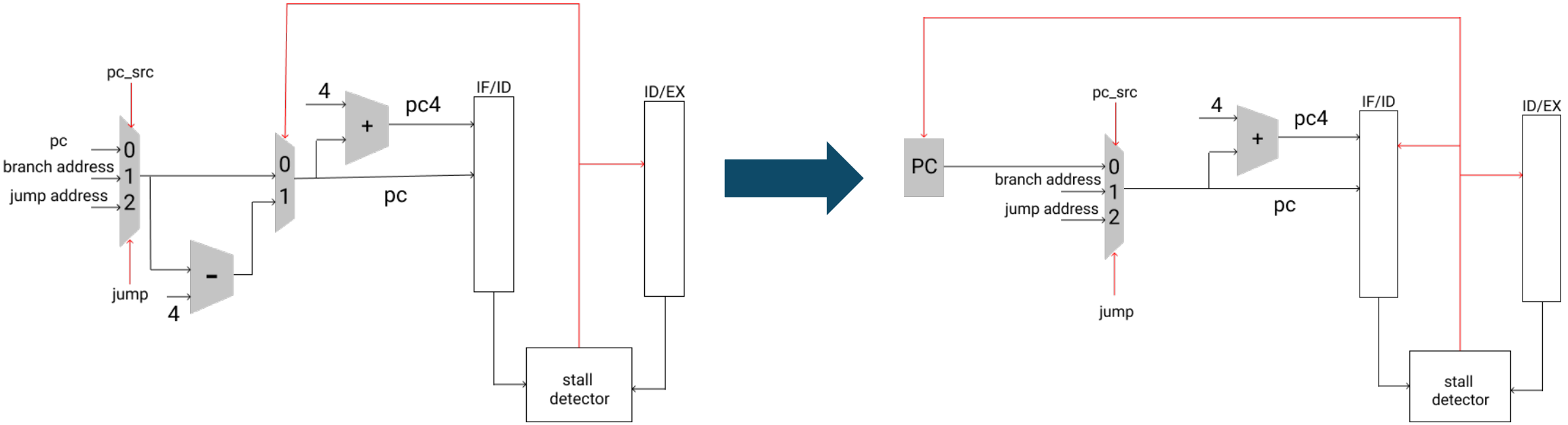
ベンチマークプログラム	総クロック数
bitcnts:test	200285
bitcnts:small	1417166995
dijkstra:test	107521545
dijkstra:small	985645445
stringsearch:test	351885
stringsearch:small	3007895

Largeのプログラムの実行が終わらない。
問題点は見つからない。

論理合成の結果

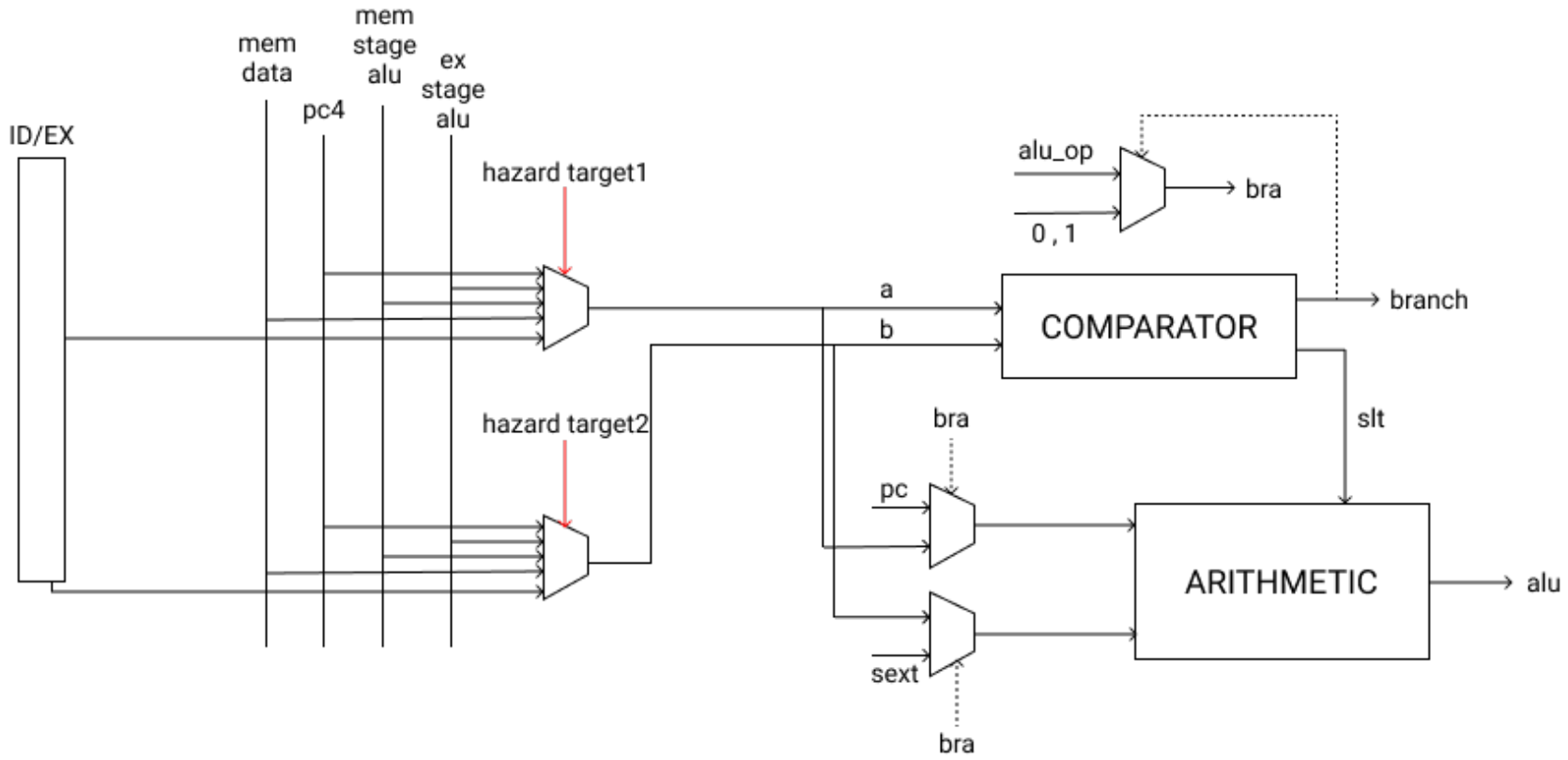
最小クロック周期の制約 [ns]	**5.15**
面積 [μm^2]	265559
消費電力 [mW]	4.7185

改善できること
パイプラインストール



これで他のパイプラインもストールできる。

改善できること
クリティカルパース（フォワーディング）



まとめ

演習を始めたとき、図などの作成にはうまく行ったが、Verilogでは苦労した。

- 書いたとき、一般的なプログラミング言語のマインドセットを持ち、進んだ。
- よって、書きかえにはかなりの時間を使った。

自分のパソコンで同じ環境を立ち上げるときに、様々なソフトとスクリプティングの能力が伸びた。

最後に、例外処理が完成できなく、悔しい。

- これは、最初からきちんとした計画を立てなかっただろう。
- これからの卒論研究に同じなミスが起こらないように頑張る