機能設計仕様書

1029-28-1547 住江 祐哉

2018 年 6 月 8 日

目 次

中間報告からの追加

中間報告以降はデバッグとアルゴリズム考案が主なタスクであり、機能拡張はあまりしていない.
パイプラインにする予定であったがデバッグに時間を要し、実現できなかった.

性能評価

主に担当したのは、p3、p4 なので、そこに関して性能評価を記す.

p4

LU 数

250

遅延時間 (メモリを持つためクロックは自動生成されるものである)

Setup -> 45.872 ns Hold -> 0.415 ns

クリティカルパス

```
Setup -> sld_hub:auto_hub|
alt_sld_fab_with_jtag_input:\instrumentation_fabric_with_node_gen:fabric_gen_new_way:w
ith_jtag_input_gen:instrumentation_fabric|alt_sld_fab:instrumentation_fabric|
alt_sld_fab_alt_sld_fab:alt_sld_fab|alt_sld_fab_alt_sld_fab_sldfabric:sldfabric|
sld_jtag_hub:\jtag_hub_gen:real_sld_jtag_hub|irsr_reg[2] to :auto_hub|
alt_sld_fab_with_jtag_input:\instrumentation_fabric_with_node_gen:fabric_gen_new_way:w
ith_jtag_input_gen:instrumentation_fabric|alt_sld_fab:instrumentation_fabric|
alt_sld_fab_alt_sld_fab:alt_sld_fab|alt_sld_fab_alt_sld_fab_sldfabric:sldfabric|
sld_jtag_hub:\jtag_hub_gen:real_sld_jtag_hub|tdo

Hold ->
sld_hub:auto_hubalt_sld_fab_with_jtag_input:\instrumentation_fabric_with_node_gen:fabr
ic_gen_new_way:with_jtag_input_gen:instrumentation_fabric|
alt_sld_fab:instrumentation_fabric|alt_sld_fab_alt_sld_fab:alt_sld_fab
alt_sld_fab_alt_sld_fab_sldfabric:sldfabric|
sld_jtag_hub:\jtag_hub_gen:real_sld_jtag_hub|sld_shadow_jsm:shadow_jsm|state[4] to
sld_hub:auto_hub|
alt_sld_fab_with_jtag_input:\instrumentation_fabric_with_node_gen:fabric_gen_new_way:w
ith_jtag_input_gen:instrumentation_fabric|alt_sld_fab:instrumentation_fabric|
alt_sld_fab_alt_sld_fab:alt_sld_fab|alt_sld_fab_alt_sld_fab_sldfabric:sldfabric|
sld_jtag_hub:\jtag_hub_gen:real_sld_jtag_hub|sld_shadow_jsm:shadow_jsm|state[4]
```

p3

LU 数

616

遅延時間 (以下、1 クロック 5.00ns と設定した場合)

Setup -> 0.579 ns
Hold -> 3.020 ns

クリティカルパス

Setup -> zreg to pcsrc~reg0
Hold -> sreg to pcsrc~reg0

考察および感想

考察

p3 は演算が主な役割なので、最も重要な機構は ALU である. 始め、演算は bit ごとに管理しようとしたが Verilog の仕様を確認することで簡便に記すことができた. 加算と減算においては 16bit 同士計算させると 17bit で出力されるので他の演算も 17bit で出力し、あとで 16bit に修正するという風に実装した. シフト演算は bit ごとに計算させるようにした. この部分で場合分けが多くなってしまったので、もっとシンプルに実装することができたのではないかと思った. また、フラグにおいては論理式で判定するようにした. このほうが直感的であったからである.
LED での出力もここで行われるがダイナミック点灯が実装できなかった. 無条件分岐・条件分岐ともにここで行う. p2 から分岐するかどうかのフラグが送られてきてそれで判断するだけなので、実装は簡単である. このモジュールで次のモジュールのメモリに対する動作を決めておくのは、次のフェイズですぐにメモリがデータを読みだしたり、書き込んだりしてしまうので、それに間に合わせるためである.
p4 はメモリの読み書きが大半の役割なので、構造的には単純である. ただし、メモリが動作するタイミングがクロックの立ち上がりに限定され、入力などは前サイクルで用意して置かなければならない.

感想

個人的にはこの実験は非常に有意義なものであった. というのは、プロセッサの構造が今までより具体的に理解することができたからである. 「計算機アーキテクチャ」の授業で一通り習っていたが、座学だけでなく実際に作ることでフェイズごとの働きまで理解できるようになった. しかし、現代市販のプロセッサでは当たり前であるであろうパイプラインまで実装できなかったのは残念である.