計算機科学実験3HW中間レポート

橘大佑
1029-31-6811
2019年度入学

2021/05/05

工学部情報学科計算機科学コース
1029-31-6811
橘 大佑

機能設計仕様書

1　全体のコンポーネントへの分割方法

プロセッサを設計する上でまず、部品をそれぞれモジュールとして作成した後にそれらを用いてフェーズ 1 からフェーズ 5 を構成した。以上 5 つのモジュール化したフェーズとそれらを制御する部品の併合わせて 6 つを組み合わせることで、最終的なプロセッサを完成させた。図 1 に方式設計仕様書のブロック図を示す。

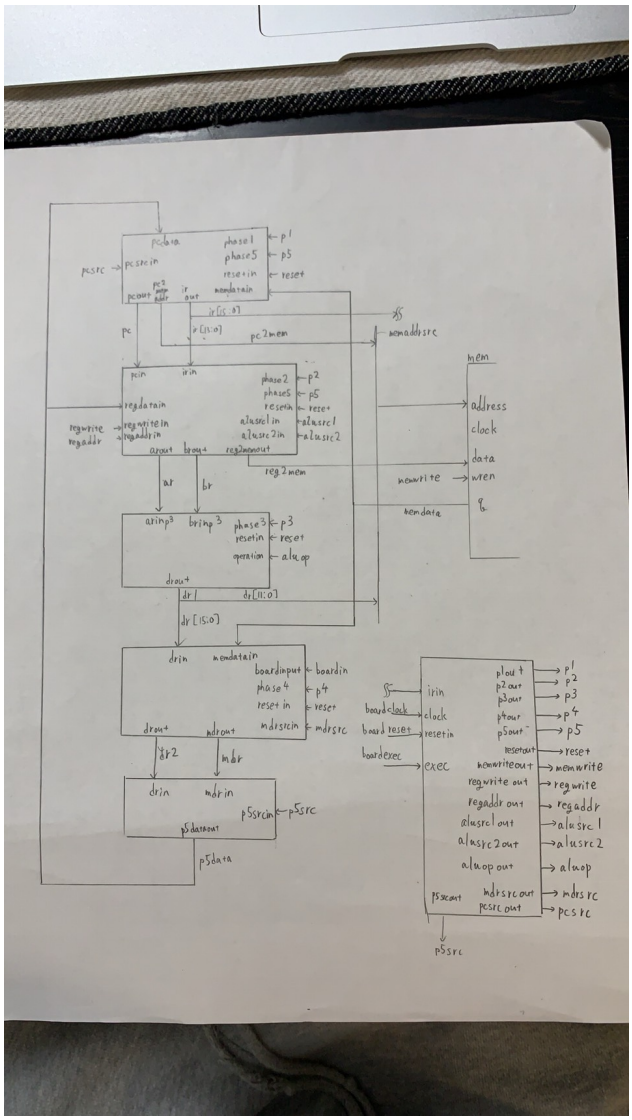

図 1: 方式設計仕様書のブロック図

2 設計を担当するコンポーネントの外部仕様

プログラムカウンター、5 つのフェーズのうち現在どのフェーズなのかを示す部品、alu、レジスター、符号拡張、LED への出力方法を決定する部品、の以上 6 つのコンポーネントを自分が設計を担当した。以下でそれぞれのコンポーネントの外部仕様を説明する。

2.1 pc(プログラムカウンター)

2.1.1 入力

1 ビットの clock,reset と 16 ビットの pcnext

2.1.2 出力

16 ビットの pcvalue

2.1.3 機能

clock の立ち上がり、または reset の立ち下がりという条件下で reset が 1 のときは pcvalue を 0 に初期化し、reset が 0 のときは pcvalue に pcnext を代入する。

2.2 phasecounter(現在どのフェーズにあるかを示すモジュール)

2.2.1 入力

1 ビットの clock,reset

2.2.2 出力

1 ビットの phase1,phase2,phase3,phase4,phase5

2.2.3 機能

clock の立ち上がり、または reset の立ち下がりという条件下で reset が 1 のときはフェーズを 1 に設定し、reset が 0 のときはフェーズを 1 すすめる。phase1 から phase5 の出力にはクロックとの論理積をとる。

2.3 alu(演算装置)

2.3.1 入力

演算対象となる符号付き 16 ビットの arin,brin、上位ビットから順に条件コードを表す 4 ビットの flagin, 場合分けのための 4 ビットの opin

2.3.2 出力

16 ビットの dr、4 ビットの flagalu、1 ビットの write

2.3.3 機能

4 ビットの opin に応じて arin と brin に対して 16 種類の異なる演算を行い、dr にその結果を格納する。write には SZCV の 4 ビットの条件コードを書き換えるかどうかの場合分けを行い、その結果を格納する。1 のときのみ書き込む可能となる。flagalu は演算に基づく条件コードを格納する。flagalu は上位ビットから順に S,Z,C,V という名前を簡単のためにつける。S は演算結果が負、つまり最上位ビットが 1 となるときに 1、それ以外は 0 を格納する。Z は演算結果が 0 のときに 1、そうでないときは 0 を格納する。C は加算、減算、比較演算の場合は最上位ビットでの桁上げ、シフト演算では最後にシフトアップされたビットの値、それ以外の演算では 0 を格納する。V は演算結果が符号付き 16 ビットで表せる範囲を越えた場合は 1、そうでない場合は 0 を格納する。つまり 4 通りの演算 (正+正=負, 負+負=正, 正-負=負, 負-正=正) 結果となった場合である。

2.4 register(レジスター)

2.4.1 入力

1 ビットの clock,reset,regwrite とフェーズ 5 からの入力を表す 16 ビットの regdata、IR の 16 ビットのうちの一部を表す 6 ビットの regin、制御部からの入力を表す 3 ビットの regaddr

2.4.2 出力

16 ビットの out1,out2

2.4.3 機能

register は 16 ビットのレジスタ 8 個の配列 register_file を内部に持っており、クロックの立ち上がり、またはリセットの立ち下がりという条件下でリセットが 1 のときは内部のレジスタを全て 0 に初期化する。リセットが 0 のときで、かつ regwrite が 1 のときのみ regaddr で指定されたレジスタの値を regdata で書き換える。regwrite が 0 の場合はレジスタの値は更新しない。二つの出力には regin で指定された上位 3 ビットのアドレスのレジスタの値を out1 に、下位 3 ビットのアドレスのレジスタの値を out2 に格納する。

2.5 signextention(符号拡張)

2.5.1 入力

8 ビットの irin

2.5.2 出力

16 ビットの tobr

2.5.3 機能

最上位の 1 ビットを入力された 8 ビットの左に 8 つ並べることで符号付きでの符号拡張を行う。

2.6 decodefrom2to16(外部出力の表示方法)

2.6.1 入力

16 ビットの in

2.6.2 出力

8 ビットの out

2.6.3 機能

フェーズ 3 からのびる 2 進数 16 ビットの AR の値を 4 桁 16 進数表記に変換するためのモジュール。16 ビットの入力を 4 ビットごとに分割し、4 つに分けられた 4 ビットの入力から、課題 2 で用いた LED に 0 から F の文字で点点灯させるための 8 ビットの出力にする。課題 2 で用いたコードをそのまま利用した。

3 実装を担当するコンポーネントの内部仕様

それぞれのモジュールのシミュレーション結果を示し、正しく動作していることを示す。なお、シミュレーションでは共通してクロックを 10ns ごとに反転するようにしている。

3.1 pc

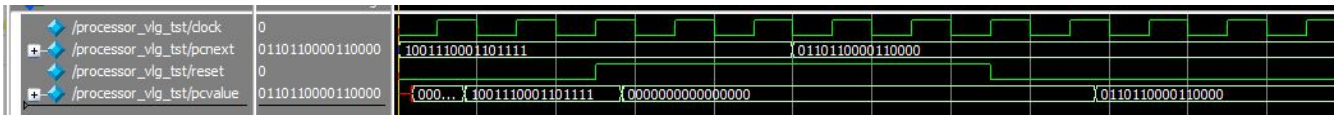

図 2: pc のシミュレーション結果

pc のシミュレーション結果を図 2 に示す。リセットが 0 のときはクロックの立ち上がりで pcvalue に pcnext の値が格納され、リセットが 1 のときはクロックの立ち上がりで pcvalue に 0 が代入されていることが分かる。また、使用した論理要素、レジスタ、ピンの数はそれぞれ 16,16,34 個であった。

3.2 phasecounter

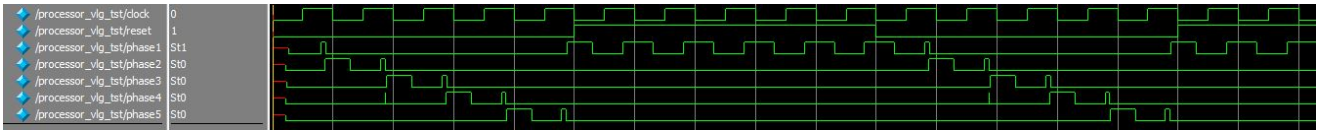

図 3: phasecounter のシミュレーション結果

phasecounter のシミュレーション結果を図 3 に示す。クロックの立ち上がり、またはリセットの立ち下がりの条件下で、リセットが 0 のときは 3 ビットのカウンター count を 0 に再定義し、リセッ

トが 1 のときは count が 1 づつ増加し、count がそれぞれ 3'b000,3'b001,3'b010,3'b011,3'b100 のときにそれぞれ phasecounter 内で定義した関数 phase1,phase2,phase3,phase4,phase5 が 1 になる。こうすることで、フェーズが 1 から 5 までクロックの立ち上がりで順に遷移している状態ができるように設計されている。また、使用した論理要素、レジスタ、ピンの数はそれぞれ 8,3,7 個であった。

3.3 alu

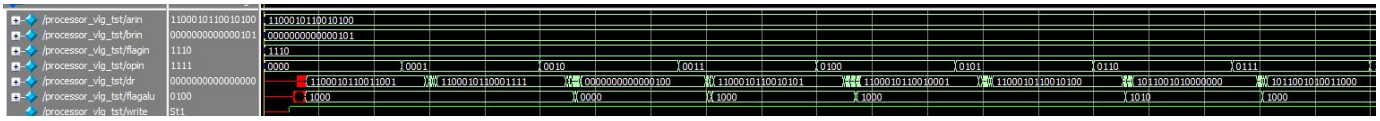

図 4: alu のシミュレーション結果 1

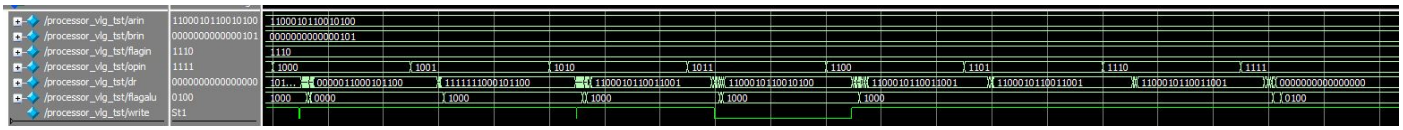

図 5: alu のシミュレーション結果 2

alu 内に 3 つの関数 fordr,for_szcv,for_write を定義し、それぞれ出力 dr,flagalu,write に対して入力 opin で場合分けされるように設計した。
opin が 0000 から 0111 の場合の alu のシミュレーション結果を図 4 に、opin が a000 から a111 の場合の alu のシミュレーション結果を図 5 に示す。arin,brin の値がそれぞれ 1100010110010100、0000000000000101 であるとき、演算結果は以下のようになる。表 1 に演算子の種類による演算結果を示す。算術演算、論理演算は演算子をそのまま用いるだけである。シフト演算に関しては、論理シフトは記号 <<, >> を用いた。循環シフトについては、論理シフトの論理和をとることで実現した。算術演算に関しては記号 <<< を用いたが、符号付きの整数を扱うために入力 arin と brin に *signed* と書き加えることで arin と brin を符号付き整数として扱うことができるようになった。下の表の演算結果とシミュレーション結果で表示された 16 ビットの出力を比較してみると演算が正しく行われていることが分かる。また、SZCV に関しても演算結果から正しく計算されている。1 ビットの出力 write は opin が 11 のときのみ 0 となる。

また、使用した論理要素、レジスタ、ピンの数はそれぞれ 531,0,61 個であった。表 1 に opin によって異なる演算の種類による演算結果を示す。

機能設計仕様書

表 1: 値段表

演算子	演算結果
加算	1100_0101_1001_1000
減算	1100_0101_1000_1110
論理積	0000_0000_0000_0100
論理和	1100_0101_1001_0100
排他的論理和	1100_0101_1001_0000
左論理シフト	1011_0010_1000_0000
左循環シフト	1011_0010_1001_1000
右論理シフト	0000_0110_0010_1100
右算術シフト	1111_1110_0010_1100

opin			S	Z	C	V
0	+	ADD	計算結果の最上位ビット	計算結果の全てのビットが0	最上位ビットからの桁上げ	正＋正＝負、負＋負＝正
1	-	SUB,CMP			最上位ビットからの桁上げ	正-負＝負、負-正＝正
2	&	AND			0	0
3	|	OR			0	0
4	^	XOR			0	0
5		MOV			0	0
6	<<	SLL			最後にシフトアウトされたビットの値	0
7	<<	SLR			0	0
8	>>	SRL			最後にシフトアウトされたビットの値	0
9	>>>	SRA			最後にシフトアウトされたビットの値	0
10	+	LDST,B			0	正＋正＝負、負＋負＝正
11		LI			0	0
12	+	B			0	正＋正＝負、負＋負＝正
13	+	BLT			0	正＋正＝負、負＋負＝正
14	+	BLE			0	正＋正＝負、負＋負＝正
15	+	BNE			0	正＋正＝負、負＋負＝正

図 6: opin による演算の種類と条件コード

機能設計仕様書

3.4 register

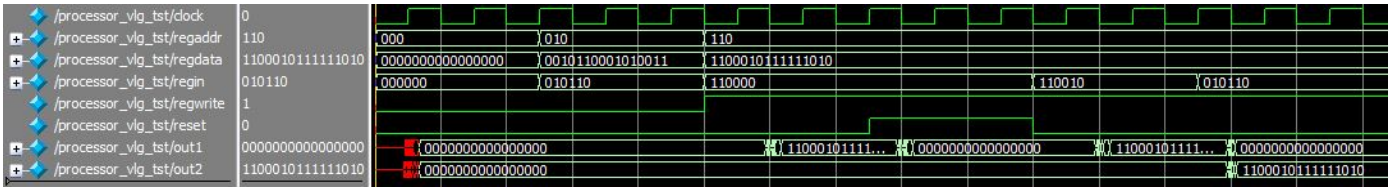

図 7: register のシミュレーション結果

register のシミュレーション結果を図 7 に示す。regwrite が 0 のときはレジスタに書き込むを行わず、0 のままであることが分かる。そして regwrite が 1 でクロックの立ち上がりの場合に regaddr のアドレスのレジスタに regdata が代入され、出力 out1 と out2 に格納されている。また、使用した論理要素、レジスタ、ピンの数はそれぞれ 30,128,60 個であった。

3.5 signextention

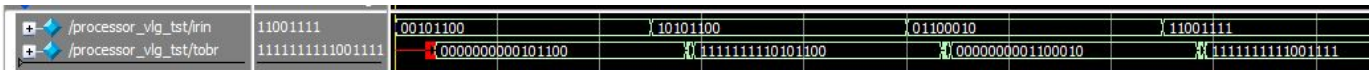

図 8: signextention のシミュレーション結果

signextention のシミュレーション結果を図 8 に示す。符号付き 8 ビットの入力が 16 ビットに符号拡張されているが分かる。符号付きの符号拡張には 8 ビットの入力の左側に入力の最上位ビットを 8 つ並べることで実現した。また、使用した論理要素、レジスタ、ピンの数はそれぞれ 0,0,24 個であった。

3.6 decodefrom2to16

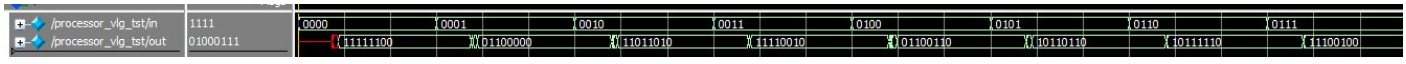

図 9: decodefrom2to16 のシミュレーション結果

機能設計仕様書

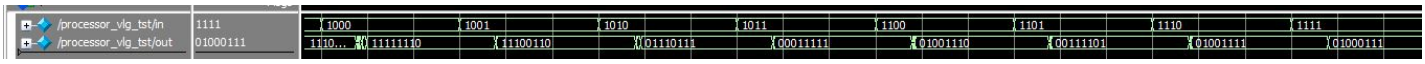

図 10: decodefrom2to16 のシミュレーション結果

decodefrom2to16 のシミュレーション結果を図 9,10 に示す。4 ビットの入力から LED の表示のための 8 ビットを出力する。課題 2 で用いたデコーダーと方針は同じである。また、使用した論理要素、レジスタ、ピンの数はそれぞれ 8,0,12 個であった。

4 Netlist Viewer での回路図

図 11 から 16 にそれぞれのモジュールの回路図を示す。

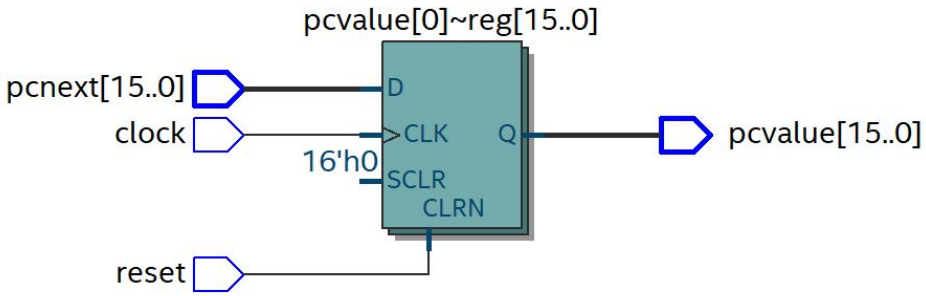

図 11: PC の回路図

機能設計仕様書

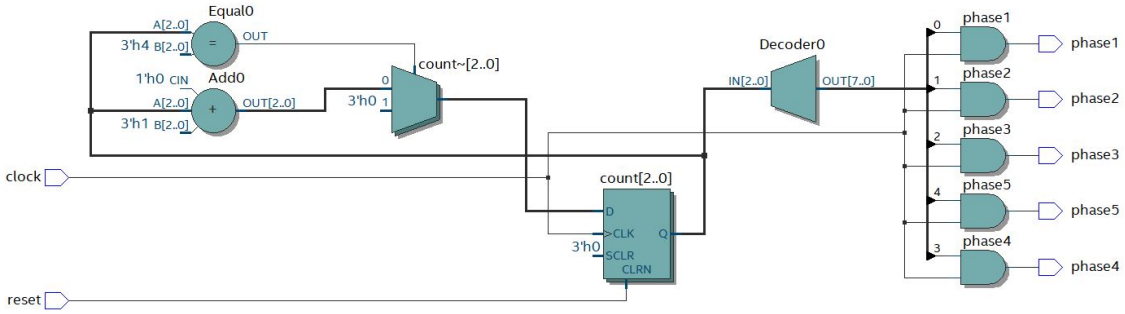

図 12: phasecounter の回路図

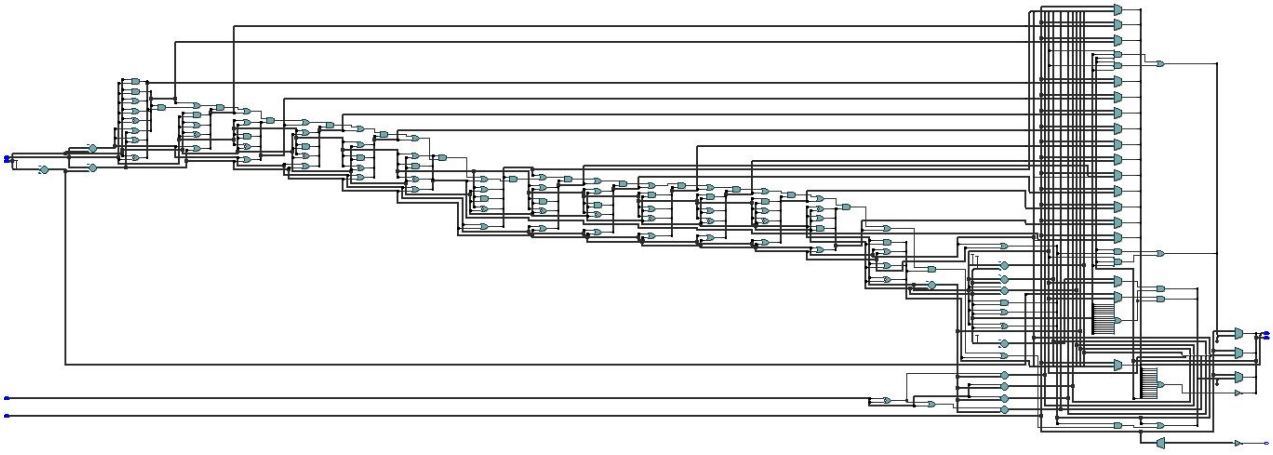

図 13: ALU の回路図

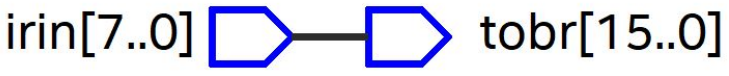

図 14: signextention の回路図

機能設計仕様書

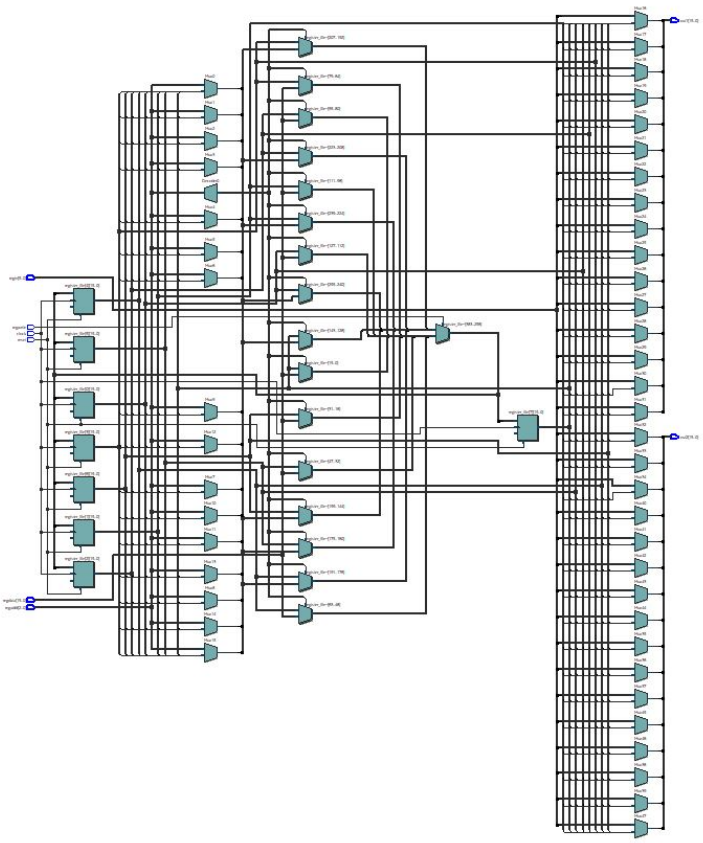

図 15: register の回路図

機能設計仕様書

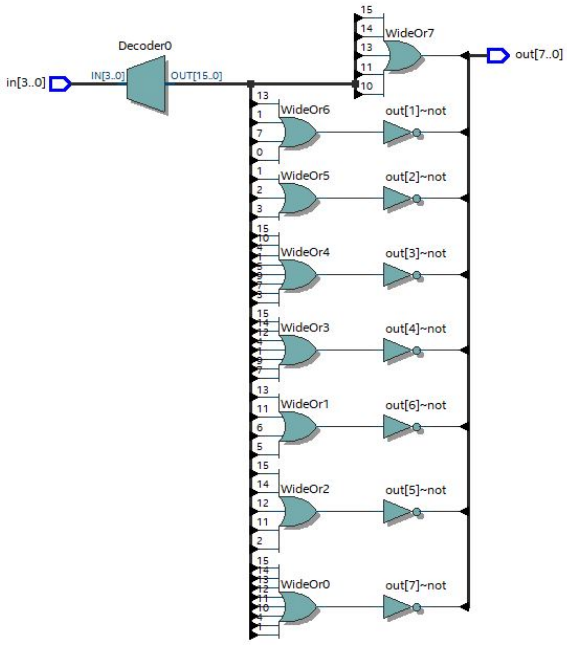

図 16: decodefrom2to16 の回路図