機能設計仕様書

作成者：Pyii Phyo Maung
Student ID: 1029322149

May 11, 2023

1　構成要素の分割

Simpleは、フェーズカウンタ、ALU、シフタ、全体のシンプルファイル、レジスタファイル、レジスタ、ブランチ、ctl、removechattering、pc（プログラムカウンター）、ラム（主記憶）というコンポーネントに分割された。今回の担当は以下の通りである。

名前	担当の部分
加藤利梓	phasecounter, ALU, shifter, 全体
神事倫紀	RegisterFile, register, Branch, ctl, removechattering
オースティン	pc, ram
全員	各モジュールのデバッグと編集

Table 1: 担当の分割

この分割により、より良い組織、より高い保守性、より容易なテスト、各コンポーネントの固有の機能の効率的な活用が可能になる。

2　外部仕様とコード

私が担当した部品、pc（プログラムカウンター）とram（主記憶）の外部仕様は以下の通りです：

2.1　PC

PCモジュールには、以下の信号があります。ここでは、各信号について簡単に説明します：

2.1.1　Inputs

1. clock： クロック回路の同期を駆動するクロック信号である。この信号の立ち上がりエッジ（つまり0から1に遷移するとき）で、回路はプログラムカウンタの値を更新するなどの特定のアクションを実行する。

2. reset: リセットはプログラムカウンタを初期状態（この場合は0）に戻すための信号である。通常、プログラムの開始時や割り込みなどのイベント発生時に発生する。

3. branchFlag: ブランチフラグはプログラム中の分岐を行うかとうかを示すためのフラグである。ブランチフラグが設定されている場合（すなわち1）、プログラムカウンタはData Registerの値に更新され、設定さ

れていない場合（すなわち0）、プログラムカウンタは1だけ増分される。

4. ce(ChangeEnable): Change Enableはプログラムカウンターの更新を有効または無効にするための信号である。Change Enableが1の場合、プログラムカウンタはBranch FlagとData Registerに基づいて更新される。Change Enableが0の場合、プログラムカウンタは更新されない。

5. dr(dataregister): Data Registerはブランチフラグが設定された場合に、プログラムカウンタが更新される値である。通常、プログラムで次に実行される命令のアドレスとなる。

2.1.2 Outputs

1. pc（プログラム・カウンター）: プログラムカウンターは現在の値である。プログラム内で現在実行されている命令のアドレスを表す。

2. pcPlusOne:PC Plus Oneはプログラムカウンターの値に1を足した値である。分岐がない場合（すなわち、Branch Flagが0の場合）、次に実行される命令のアドレスを表す。

要約すると、これらの信号はプログラムの流れを制御するために使用される。プログラムカウンタは、現在との命令が実行されているかを記録していて、制御信号に基づいてプログラムの異なる部分にジャンプするために更新することができる。

2.1.3 コード

```
module PC(input clock,reset,branchFlag,ce,//changeEnable
input [15:0] dr,
output reg [15:0] pc,pcPlusOne);

reg [15:0] p2_master,p2_slave,p3_master,p3_slave,p4_master,p4_slave,p5;
reg pcStoped;

always @(*)begin
pcPlusOne <= pc + 1;
end

always @(posedge clock or negedge reset) begin
```

```
if(!reset)begin
pc <= 0;
end else begin
if(ce == 1'b1)begin
if(branchFlag == 1'b1) begin
pc <= dr;
end else  begin
pc <= pc+1;
end
end
end
end
```

2.2 RAM

RAM (Random Access Memory)モジュールは、様々な信号を持つ。以下に、それらの信号の詳細を説明する。

2.2.1 信号

- データ入力 (Data In)：この信号は、RAMに書き込むデータを持つ。データは、RAMの特定のアドレスに書き込まれる。
- データ出力 (Data Out)：この信号は、RAMから読み出されるデータを持つ。データは、RAMの特定のアドレスから読み出される。
- アドレス (Address)：この信号は、RAM内の特定の位置を指定する。データ入力信号によってデータが書き込まれる場所、またはデータ出力信号によってデータが読み出される場所を決定する。
- 書き込み許可 (Write Enable)：この信号が1である場合、データ入力信号のデータがアドレス信号で指定されたRAMの位置に書き込まれる。この信号が0である場合、RAMへの書き込みは行われない。
- 読み出し許可 (Read Enable)：この信号が1である場合、アドレス信号で指定されたRAMの位置からデータが読み出され、データ出力信号に送られる。この信号が0である場合、RAMからの読み出しは行われない。

以上のように、これらの信号はRAMモジュールの動作を制御する。データの書き込みと読み出し、それぞれの操作が可能なアドレスの指定、そしてそれらの操作の許可や禁止を決定する。

2.2.2 コード

```
// megafunction wizard: %RAM: 1-PORT%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altsyncram

// ============================================================
// File Name: ram.v
// Megafunction Name(s):
//  altsyncram
//
// Simulation Library Files(s):
//  altera_mf
// ============================================================
// ************************************************************
// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
//
// 20.1.0 Build 711 06/05/2020 SJ Lite Edition
// ************************************************************


//Copyright (C) 2020  Intel Corporation. All rights reserved.
//Your use of Intel Corporation's design tools, logic functions
//and other software and tools, and any partner logic
//functions, and any output files from any of the foregoing
//(including device programming or simulation files), and any
//associated documentation or information are expressly subject
//to the terms and conditions of the Intel Program License
//Subscription Agreement, the Intel Quartus Prime License Agreement,
//the Intel FPGA IP License Agreement, or other applicable license
//agreement, including, without limitation, that your use is for
//the sole purpose of programming logic devices manufactured by
```

```
//Intel and sold by Intel or its authorized distributors.  Please
//refer to the applicable agreement for further details, at
//https://fpgasoftware.intel.com/eula.


// synopsys translate_off
`timescale 1 ps / 1 ps
// synopsys translate_on
module ram (
address,
clock,
data,
wren,
q);

input [11:0]  address;
input   clock;
input [15:0]  data;
input   wren;
output [15:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
tri1   clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

wire [15:0] sub_wire0;
wire [15:0] q = sub_wire0[15:0];

altsyncram altsyncram_component (
.address_a (address),
.clock0 (clock),
.data_a (data),
.wren_a (wren),
.q_a (sub_wire0),
```

```
.aclr0 (1'b0),
.aclr1 (1'b0),
.address_b (1'b1),
.addressstall_a (1'b0),
.addressstall_b (1'b0),
.byteena_a (1'b1),
.byteena_b (1'b1),
.clock1 (1'b1),
.clocken0 (1'b1),
.clocken1 (1'b1),
.clocken2 (1'b1),
.clocken3 (1'b1),
.data_b (1'b1),
.eccstatus (),
.q_b (),
.rden_a (1'b1),
.rden_b (1'b1),
.wren_b (1'b0));
defparam
altsyncram_component.clock_enable_input_a = "BYPASS",
altsyncram_component.clock_enable_output_a = "BYPASS",
altsyncram_component.init_file = "BubbleSort.mif",
altsyncram_component.intended_device_family = "Cyclone IV E",
altsyncram_component.lpm_hint = "ENABLE_RUNTIME_MOD=YES,INSTANCE_NAME=NONE",
altsyncram_component.lpm_type = "altsyncram",
altsyncram_component.numwords_a = 4096,
altsyncram_component.operation_mode = "SINGLE_PORT",
altsyncram_component.outdata_aclr_a = "NONE",
altsyncram_component.outdata_reg_a = "UNREGISTERED",
altsyncram_component.power_up_uninitialized = "FALSE",
altsyncram_component.read_during_write_mode_port_a = "NEW_DATA_NO_NBE_READ",
altsyncram_component.widthad_a = 12,
altsyncram_component.width_a = 16,
altsyncram_component.width_byteena_a = 1;


endmodule
```

```
// ============================================================
// CNX file retrieval info
// ============================================================
// Retrieval info: PRIVATE: ADDRESSSTALL_A NUMERIC "0"
// Retrieval info: PRIVATE: AclrAddr NUMERIC "0"
// Retrieval info: PRIVATE: AclrByte NUMERIC "0"
// Retrieval info: PRIVATE: AclrData NUMERIC "0"
// Retrieval info: PRIVATE: AclrOutput NUMERIC "0"
// Retrieval info: PRIVATE: BYTE_ENABLE NUMERIC "0"
// Retrieval info: PRIVATE: BYTE_SIZE NUMERIC "8"
// Retrieval info: PRIVATE: BlankMemory NUMERIC "1"
// Retrieval info: PRIVATE: CLOCK_ENABLE_INPUT_A NUMERIC "0"
// Retrieval info: PRIVATE: CLOCK_ENABLE_OUTPUT_A NUMERIC "0"
// Retrieval info: PRIVATE: Clken NUMERIC "0"
// Retrieval info: PRIVATE: DataBusSeparated NUMERIC "1"
// Retrieval info: PRIVATE: IMPLEMENT_IN_LES NUMERIC "0"
// Retrieval info: PRIVATE: INIT_FILE_LAYOUT STRING "PORT_A"
// Retrieval info: PRIVATE: INIT_TO_SIM_X NUMERIC "0"
// Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "Cyclone IV E"
// Retrieval info: PRIVATE: JTAG_ENABLED NUMERIC "1"
// Retrieval info: PRIVATE: JTAG_ID STRING "NONE"
// Retrieval info: PRIVATE: MAXIMUM_DEPTH NUMERIC "0"
// Retrieval info: PRIVATE: MIFfilename STRING ""
// Retrieval info: PRIVATE: NUMWORDS_A NUMERIC "4096"
// Retrieval info: PRIVATE: RAM_BLOCK_TYPE NUMERIC "0"
// Retrieval info: PRIVATE: READ_DURING_WRITE_MODE_PORT_A NUMERIC "3"
// Retrieval info: PRIVATE: RegAddr NUMERIC "1"
// Retrieval info: PRIVATE: RegData NUMERIC "1"
// Retrieval info: PRIVATE: RegOutput NUMERIC "0"
// Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "0"
// Retrieval info: PRIVATE: SingleClock NUMERIC "1"
// Retrieval info: PRIVATE: UseDQRAM NUMERIC "1"
// Retrieval info: PRIVATE: WRCONTROL_ACLR_A NUMERIC "0"
// Retrieval info: PRIVATE: WidthAddr NUMERIC "12"
// Retrieval info: PRIVATE: WidthData NUMERIC "16"
// Retrieval info: PRIVATE: rden NUMERIC "0"
```

```
// Retrieval info: LIBRARY: altera_mf altera_mf.altera_mf_components.all
// Retrieval info: CONSTANT: CLOCK_ENABLE_INPUT_A STRING "BYPASS"
// Retrieval info: CONSTANT: CLOCK_ENABLE_OUTPUT_A STRING "BYPASS"
// Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING "Cyclone IV E"
// Retrieval info: CONSTANT: LPM_HINT STRING "ENABLE_RUNTIME_MOD=YES,INSTANCE_NAME=NONE"
// Retrieval info: CONSTANT: LPM_TYPE STRING "altsyncram"
// Retrieval info: CONSTANT: NUMWORDS_A NUMERIC "4096"
// Retrieval info: CONSTANT: OPERATION_MODE STRING "SINGLE_PORT"
// Retrieval info: CONSTANT: OUTDATA_ACLR_A STRING "NONE"
// Retrieval info: CONSTANT: OUTDATA_REG_A STRING "UNREGISTERED"
// Retrieval info: CONSTANT: POWER_UP_UNINITIALIZED STRING "FALSE"
// Retrieval info: CONSTANT: READ_DURING_WRITE_MODE_PORT_A STRING "NEW_DATA_NO_NBE_READ"
// Retrieval info: CONSTANT: WIDTHAD_A NUMERIC "12"
// Retrieval info: CONSTANT: WIDTH_A NUMERIC "16"
// Retrieval info: CONSTANT: WIDTH_BYTEENA_A NUMERIC "1"
// Retrieval info: USED_PORT: address 0 0 12 0 INPUT NODEFVAL "address[11..0]"
// Retrieval info: USED_PORT: clock 0 0 0 0 INPUT VCC "clock"
// Retrieval info: USED_PORT: data 0 0 16 0 INPUT NODEFVAL "data[15..0]"
// Retrieval info: USED_PORT: q 0 0 16 0 OUTPUT NODEFVAL "q[15..0]"
// Retrieval info: USED_PORT: wren 0 0 0 0 INPUT NODEFVAL "wren"
// Retrieval info: CONNECT: @address_a 0 0 12 0 address 0 0 12 0
// Retrieval info: CONNECT: @clock0 0 0 0 0 clock 0 0 0 0
// Retrieval info: CONNECT: @data_a 0 0 16 0 data 0 0 16 0
// Retrieval info: CONNECT: @wren_a 0 0 0 0 wren 0 0 0 0
// Retrieval info: CONNECT: q 0 0 16 0 @q_a 0 0 16 0
// Retrieval info: GEN_FILE: TYPE_NORMAL ram.v TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL ram.inc FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL ram.cmp TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL ram.bsf TRUE
// Retrieval info: GEN_FILE: TYPE_NORMAL ram_inst.v FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL ram_bb.v FALSE
// Retrieval info: LIB_FILE: altera_mf
```

BubbleSort.mifのファイルを使えるようにinit_fileの部分も追加した。

3　内部仕様

私が担当した部品であるpc（プログラムカウンター）とram（主記憶）の内部仕様は、以下の通りである：

3.1　PC

PCモジュールには、p2_master、p2_slave、p3_master、p3_slave、p4_master、p4_slave、p5という5つの内部レジスタがある。これはパイプライン化する時に役に立つ公式である。また、プログラムカウンタが停止したことを示すpcStoped信号も含まれている。 PCモジュールの主な役割は、現在または次に実行する命令のアドレスを追跡することである。内部的には、このモジュールはクロック信号に同期して動作し、各クロックサイクルで新たな命令アドレスを生成する。

3.2　RAM

RAMモジュールは、Intel FPGAのaltsyncramメガファンクションをインスタンス化したものである。動作にはシングルポートを使用し、アドレス幅は12ビット、データ幅は16ビットである。RAMは最大4096ワードを保持することができる。また、このモジュールには、内部データと出力データ信号qを接続するためのsub_wire0と名付けられた出力ワイヤーが含まれている。RAMモジュールの主な役割は、データの一時的な保存と取り出しを行うことである。内部的には、このモジュールはデータを格納するためのメモリセルの配列と、データの書き込みと読み出しを制御するロジックから成り立っている。

4　機能設計仕様

4.1　PC

PCモジュールは、ce（チェンジイネーブル）信号がハイで、リセット信号がローのとき、各クロックサイクルでプログラムカウンタを1つずつインクリメントする。branchFlag信号がHighの場合、プログラムカウンタをインクリメントする代わりに、データレジスタの値をプログラムカウンタにロードする。機能的には、PCモジュールは以下の設計仕様を持つ:

- クロック信号の立ち上がりエッジで動作する。

- リセット信号があると、PCの値は0にリセットされる。
- branchFlagが立っていて、ce (Change Enable)が有効な場合、PCの値はデータレジスタ（dr）の値に更新される。
- branchFlagが立っていなくて、ceが有効な場合、PCの値は現在の値に1を加えたものに更新される。
- ceが無効な場合、PCの値は更新されない。

4.2 RAM

RAMモジュールは、4096ワードのメモリを提供し、各ワードは16ビット幅である。読み出しと書き込みをサポートしている。これらの操作のためのアドレスは、12ビットのアドレス入力で指定される。メモリに書き込むデータは16ビットのデータ入力から供給され、ライトイネーブル信号wrenが書き込み動作を制御する。読み出しデータは16ビットのq出力で利用可能である。モジュールは、同期化のためにクロック信号を使用する。機能的には、RAMモジュールは以下の設計仕様を持つ:

- データ入力は、書き込み許可信号が立っているときにメモリに書き込まれる。
- データ出力は、読み出し許可信号が立っているときにメモリから読み出される。
- アドレス信号は、読み出しまたは書き込みが行われるメモリセルの位置を指定する。
- 書き込み許可信号が立っているときにのみ、データ入力は指定されたアドレスに書き込まれる。
- 読み出し許可信号が立っているときにのみ、指定されたアドレスからデータが読み出される。