ipPROCESS

ARCHITECTURE SPECIFICATION

32-bit μDLX Core Processor

Universidade Federal da Bahia

Version: 1.0

GNU LGPL License

Revision History

Date	Description	Author(s)
04/27/2014	Conception	João Carlos Bittencourt
04/30/2014	Instruction layout description	João Carlos Bittencourt
05/09/2014	• Text revision; • Update diagrams and instruction layout; • Update instruction fetch I/O definitions; • Missing pictures inclusion; • Include memory access and write back pin/port definitions;	João Carlos Bittencourt
04/13/2014	Missing pictures	João Carlos Bittencourt
04/15/2014	• Fix instruction fetch pin definitions; • Include instruction fetch datapath; • Include pipeline registers definitions; • Fix memory access stage pin/port definitions; • Include write back data path;	João Carlos Bittencourt
04/15/2014	Add execute block diagram	Igo Amauri Luz
04/16/2014	• Add branch prediction signal to ID and IF blocks and pin definitions; • Add table for pin definitions in Execute stage • Add branch prediction signal in pipeline registers definitions; • Include architecture interface figure;	João Carlos Bittencourt
04/18/2014	Add execute datapath diagram	Igo Amauri Luz
04/23/2014	Formating; Stakeholders listing	João Carlos Bittencourt
05/23/2014	• Fixing and formatting; Add stakeholders listing; • Fix tables, spacing and maths; • Update document with news blocks, datapath, architecture;	Igo Amauri Luz João Carlos Bittencourt

05/24/2014	• Add Execute and memory datapath; • Update memory Pin/Port Definitions; • Update document with news blocks, datapath, architecture;	Igo Amauri Luz
05/26/2014	• Add Top Architecture; missing signals description; • Update top Figure;	Igo Amauri Luz
06/27/2014	• Update Instruction Fetch table; • Update pipeline registers;	João Carlos Bittencourt
06/29/2014	• Pipeline Registers update and Control Signals; • Update MEM/WB pipeline registers; • Update WB pin/port definitions; • Add core control signals and descriptions; • Minnor changes; • Right positioning micro-instructions control signals;	João Carlos Bittencourt

CONTENTS

1. Introduction

1.1. Purpose

The main purpose of this document is to define specifications of a uDLX implementation and to provide a full overview of the design. This specifications defines all implementation parameters that composes the general uDLX requirements and specification. This definitions include processor operation modes, instruction set (ISA) and internal registers characteristics. This document also include detailed information of pipeline stages architecture, buses and other supplemental units.

1.2. Stakeholders

Name	Roles/Responsibilities
Igo Amauri Luz	TBD
João Carlos Bittencourt	TBD
Lauê Rami Costa	TBD
Linton Thiago Esteves	TBD
Victor Valente	TBD

1.3. Document Outline Description

This document is outlined as follow:

- Section **??**: This section presents the core processor block diagram, Pin/Port definitions and global parameters and configuration directives.
- Section **??**: This section presents the μDLX instruction layout and specifications.
- Section **??**: This section presents a description of each pipeline stage block, including pin definitions, signals and internal datapath.

1.4. Acronyms and Abbreviations

Along this and other documents part of this project, it will be recurrent the usage of some acronyms and abbreviations. In order to keep track of this elements the Table **??** presents a set of abbreviations used and its corresponding meaning.

Table 2: Acronym and descriptions of elements in this document.

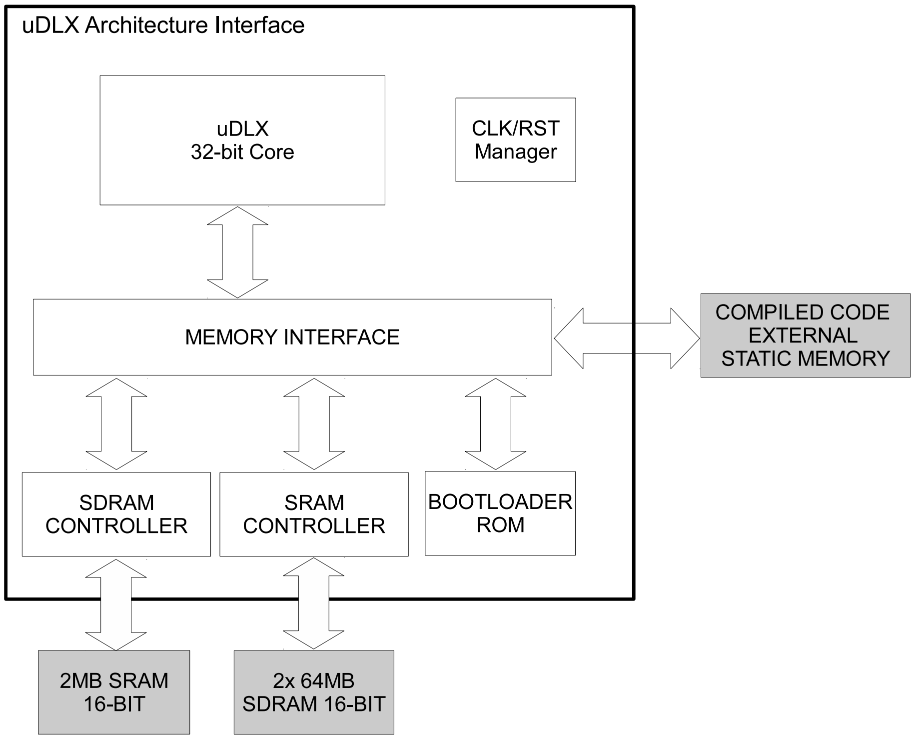
Acronym	Description
RISC	Reduced Instruction Set Computer
GPR	General Purpose Registers
FPGA	Field Gate Programmable Array
GPPU	General Purpose Processing Unit
SDRAM	Synchronous Dynamic Random Access Memory
HDL	Hardware Description Language
RAW	Read After Write
CPU	Central Processing Unit
ISA	Instruction Set Architecture
ALU	Arithmetic and Logic Unit
PC	Program Counter
RFlags	Flags Register
Const	Constant
BPM	Branch Prediction Buffer

2. Architecture Overview

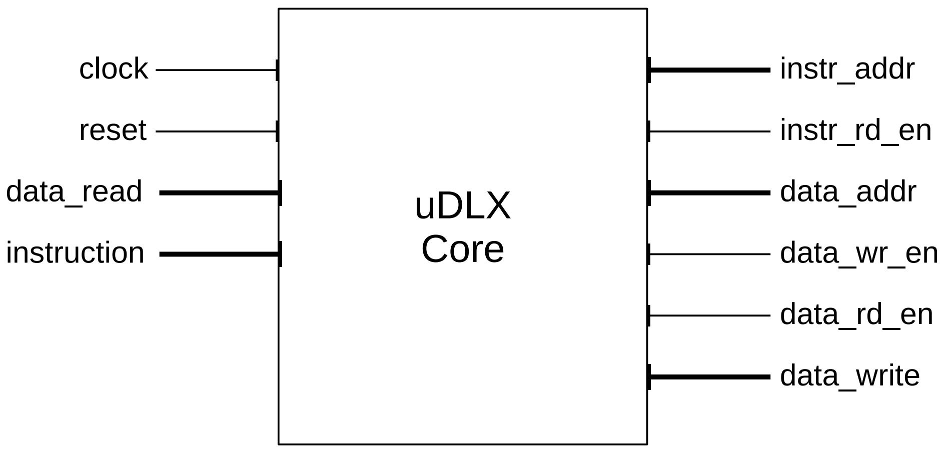
2.1. Interface Architecture

The μDLX architecture interface is composed by the following components.

- **μDLX 32-bit Core:** The core four-deep pipeline processor.
- **Memory Interface:** Provides a middle layer between the core processor and the external memories. This interface also controls the bootloader process.
- **SDRAM Controller:** Provides the interface for controlling the external SDRAM.
- **SRAM Controller:** Provides the interface for controlling the external SRAM.



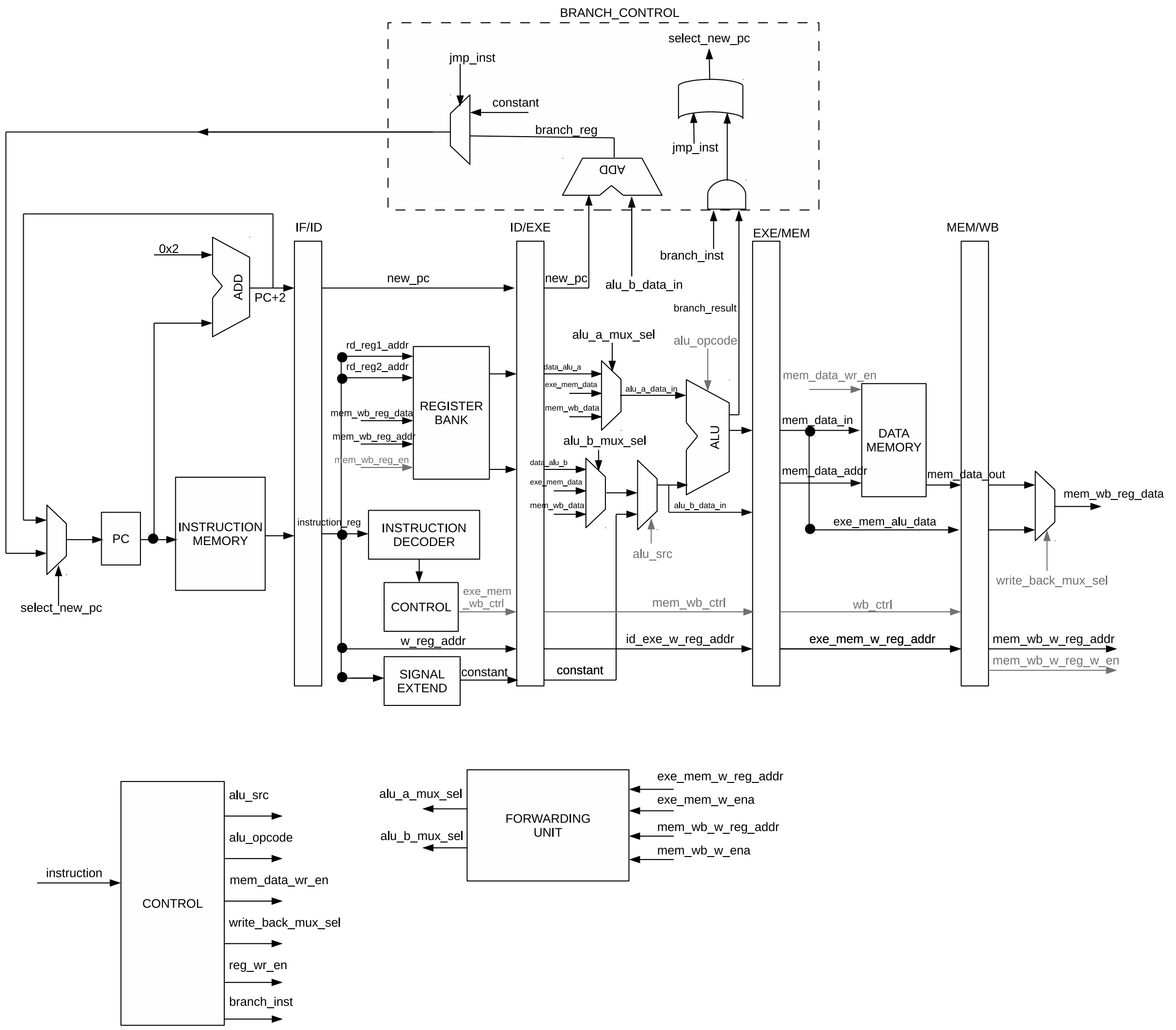
2.2. Block Diagram



2.3. Pin/Port Definitions

Name	Length	Direction	Description
clock	1	input	CPU core clock
reset	1	input	CPU core reset
instruction	32	input	SRAM instruction data
data_read	32	input	SDRAM read data
instr_addr	20	output	SRAM address
instr_rd_en	1	output	SRAM read enable
data_addr	32	output	SDRAM address
data_wr_en	1	output	SDRAM write enable
data_rd_en	1	output	SDRAM read enable
data_write	32	output	SDRAM write data

2.4. Top Architecture (Deprecated: *needs a review*)

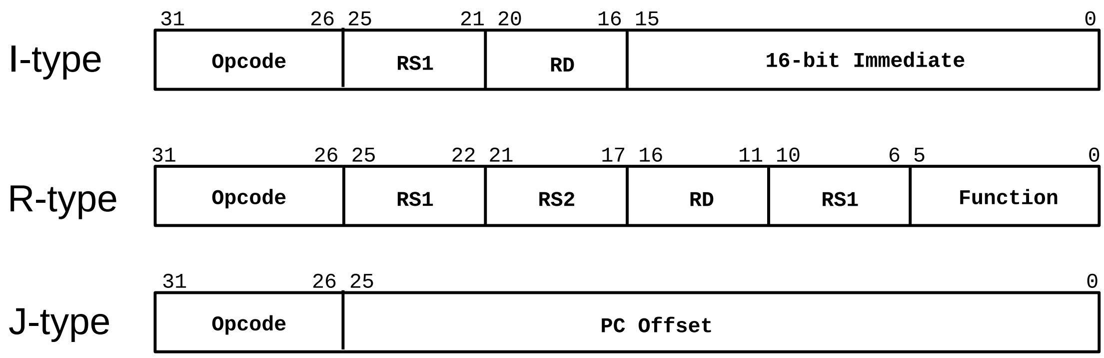


3. Instructions Layout

In order to be the most close to the DLX and also be as compatible as possible to MIPS R2000 and R3000 architecture few instructions were added to the instruction set. Instructions that are not present in DLX and MIPS architecture are added in not used OPCODES.

3.1. Instructions Set

DLX instruction structure has 5 types of instructions, the floating point instructions were removed. The 3 DLX instructions types supported in the project are shown in figure below.



NOP instruction has zero in all bits and will not be mapped to an instruction type.

3.2. I-type Instruction

Immediate instructions use the immediate data to address a load and store operation, make arithmetic operations and make brachs using Some arithmetic immediate instructions were added because they make easier assembly programming. The conditional branch operations BEQZ and BNEQZ were added to enable some compiler compatibility and future core improvement.

OPCODE	Mneumonic	Operation
100011/0x23	lw	$R_D = mem$
101011/0x2b	sw	$mem = R_D$
101010/0x09	brfl	TBD
001000/0x08	addi	$R_D = R_S1 + Sext(imm)$
001010/0x10	subi	$R_D = R_S1 - Sext(imm)$
001101/0x0c	andi	$R_D = R_S1 \wedge Sext(imm)$
001101/0x13	ori	$R_D = R_S1 \vee Sext(imm)$
000100/0x04	beqz	$PC = PC + 4 + (R_S1 = 0\ ?\ SExt(imm) : 0)$
000101/0x05	bnez	$Fixme$
000111/0x16	jr	$PC = R_S1$

Only LW, SW, BRFL, and JR instructions are in the project requirements, the other were added to make compatibility and some codes easier.

3.3. R-type Instruction

This instructions realize registers operations, most operations are arithmetic. All arithmetic opcodes are zero, differing only in the function value.

FUNCTION	Mneumonic	Operation
100000/0x20	add	$R_D = R_S1 + R_S2$
100010/0x22	sub	$R_D = R_S1 - R_S2$
100100/0x24	and	$R_D = R_S1 \wedge R_S2$
100101/0x25	or	$R_D = R_S1 \vee R_S2$
011000/0x18	mult	$R_D = R_S1\ .\ R_S2$
011010/0x1A	div	$R_D = R_S1\ /\ R_S2$
011100/0x1C	cmp	$R_D = R_S1\ cmp\ R_S2$
011101/0x1D	not	$R_D = \neg R_S2$

The MULT, DIV, CMP, and NOT instructions were added in DLX instruction set

3.4. J-type Instruction

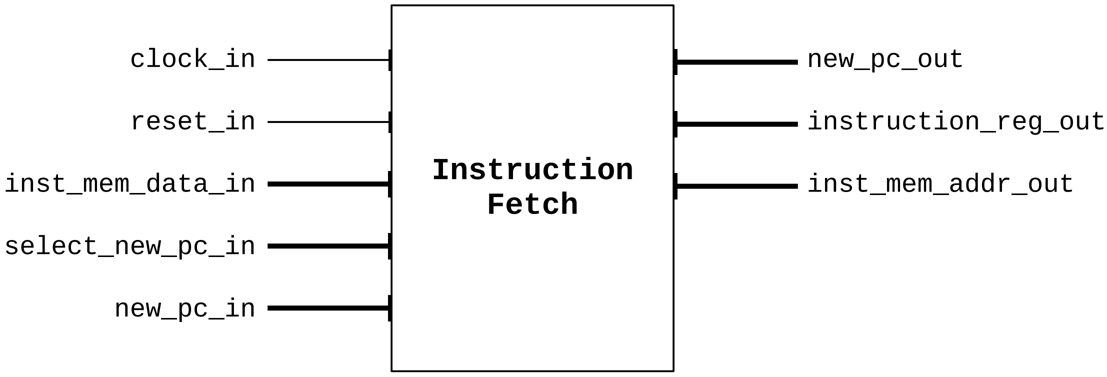
Instructions related with instruction memory jump. The instructions are JPC (J),RET (RFE), CALL (TRAP).

OPCODE	Mneumonic	Operation
000010/0x02	jpc(j)	PC = PC + 4
111110/0x3e	call(trap)	trap = 1; EPC = PC; PC = SISR; ESR = SR; ECA = masked CA; SR = 0; EDATA = SExt(imm); Clear CA but catch new interrupt events
111111/0x3f	ret(rfe)	SR = ESR; PC = EPC

4. Architecture Description

4.1. Instruction Fetch

4.1.1. Block Diagram



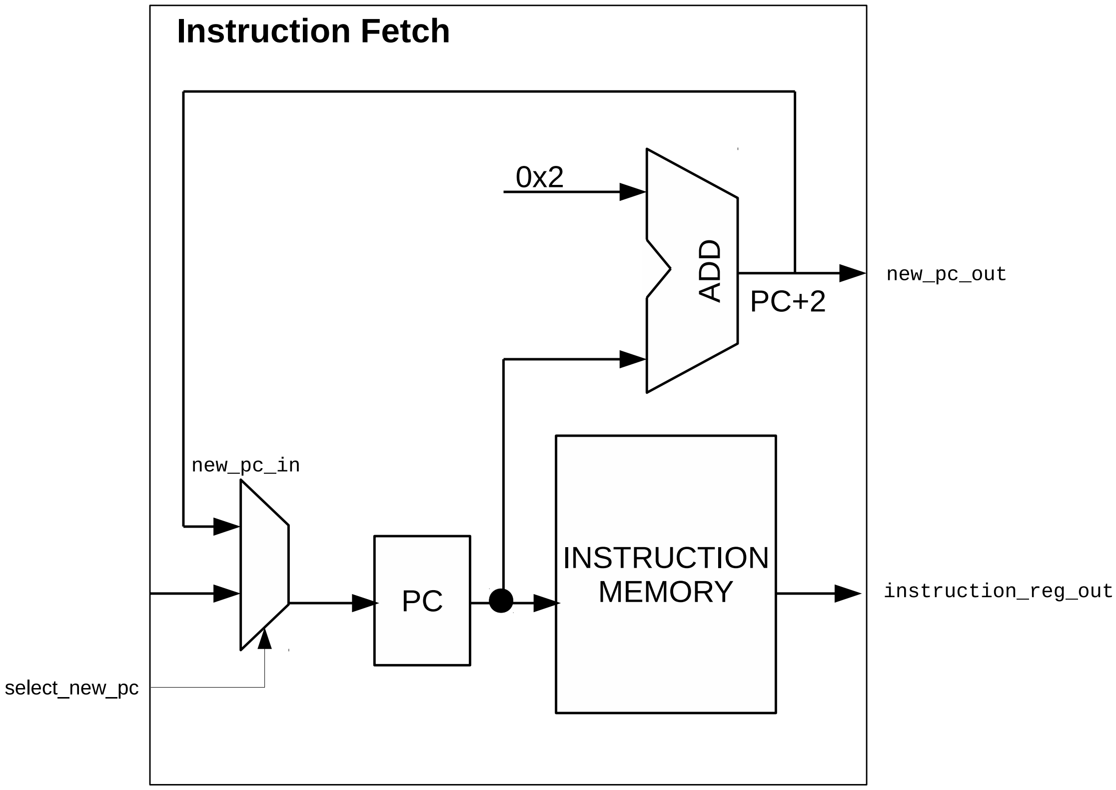
4.1.2. Pin/Port Definitions

Name	Length	Direction	Description
clock_in	1	input	CPU core clock
reset_in	1	input	CPU core reset
inst_mem_data_in	TBD	input	SRAM data
select_new_pc_in	1	input	Signal used for branch not taken
new_pc_in	20	input	New value of PC
new_pc_out	20	output	Updated value of PC
instruction_reg_out	32	output	CPU core instruction
inst_mem_addr_out	20	output	SRAM address

4.1.3. Internal Datapath

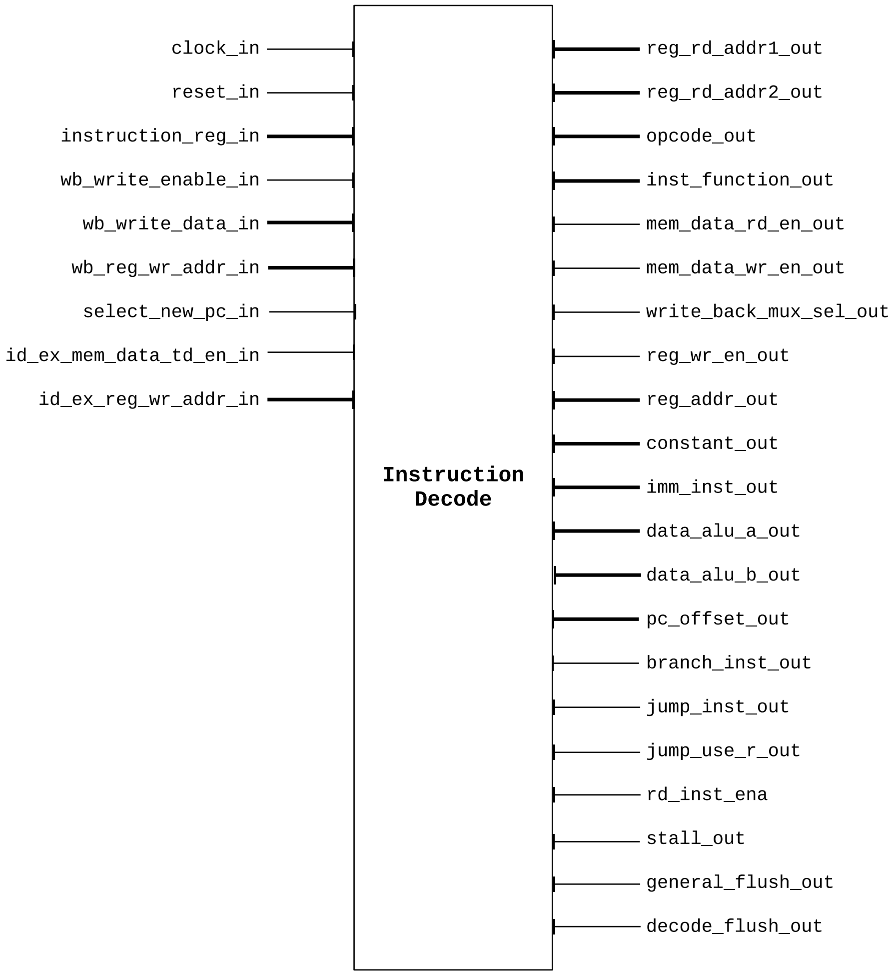
The internal data path is composed by the following components.

Program Counter : During the instruction time of an instruction this is the address of the instruction word. The address of the instruction that occurs during the next instruction time is determined by assigning a value to PC during an instruction time. If no value is assigned to PC during an instruction time by any pseudocode statement, it is automatically incremented by 2 before the next instruction time.


Instruction Fetch
0x2
ADD
PC+2
new_pc_out
new_pc_in
PC
INSTRUCTION
MEMORY
instruction_reg_out
select_new_pc

4.2. Instruction Decode/Register Fetch

4.2.1. Block Diagram



4.2.2. Pin/Port Definitions

Name	Length	Direction	Description
clock_in	1	input	CPU core clock
reset_in	1	input	CPU core reset
instruction_reg_in	32	input	CPU core instruction
wb_write_enable_in	1	input	GPR write enable signal from WB
wb_write_data_in	32	input	GPR data to be writen
select_new_pc_in	1	input	Stall and flush generation in jumps
id_ex_mem_data_rd_en_in	1	input	Forwarding singal for hazzard control
id_ex_reg_wr_addr_in	6	input	Forwarding address for hazzard control

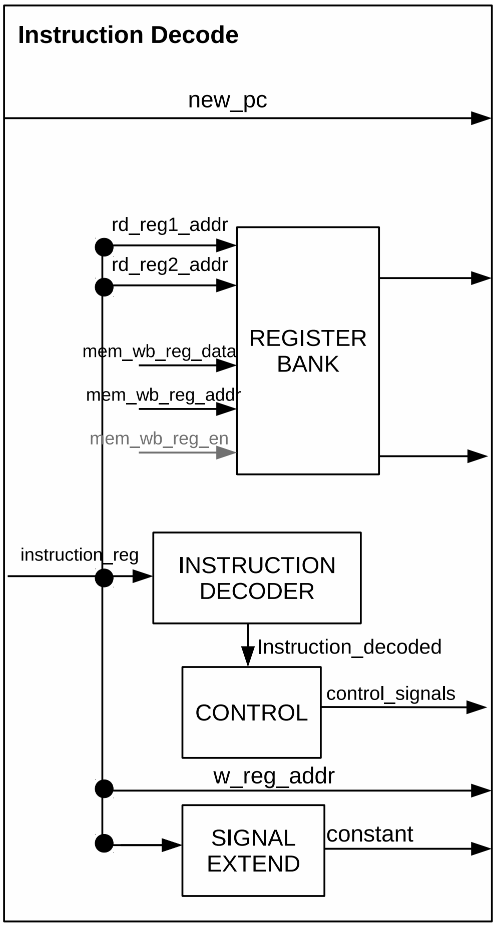
continued on next page

continued from previous page

Name	Length	Direction	Description
reg_rd_addr1_out	5	output	Register file source address A
reg_rd_addr2_out	5	output	Register file source address B
opcode_out	6	output	Instruction OpCode
inst_function_out	6	output	ALU function
mem_data_rd_en_out	1	output	Data memory read enable
mem_data_wr_en_out	1	output	Data memory write enable
write_back_mux_sel_out	1	output	Write back mux selector
reg_wr_en_out	1	output	GPR bank write enable signal
reg_addr_out	5	output	GPR bank destiny address
constant_out	32	output	32-bit Sign-extended constant
imm_inst_out	1	output	TBD
data_alu_a_out	32	output	ALU input A data
data_alu_b_out	32	output	ALU input B data
new_pc_out	20	output	Updated value of PC delayed
pc_offset_out	26	output	TBD
branch_inst_out	1	output	Conditional branch instruction
jmp_inst_out	1	output	Incoditional branch instruction
jump_use_r_out	1	output	TBD
rd_inst_ena	1	output	TBD
stall_out	1	output	Insert a stall on the datapath
general_flush	1	output	Flush pipeline registers

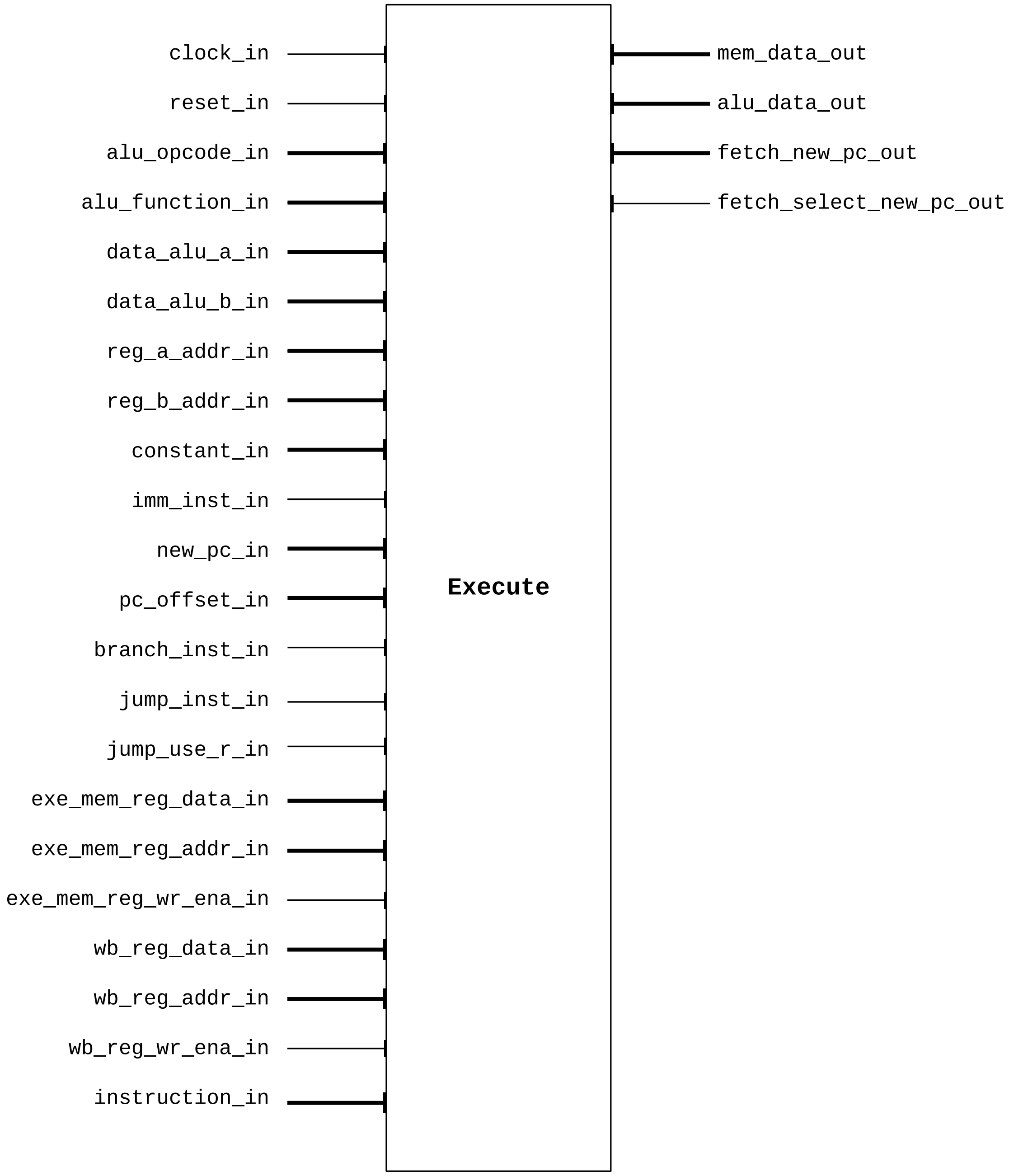
4.2.3. *Internal Datapath*

The internal data path is composed by the following components.



4.3. Execute/Address Calculate

4.3.1. Block Diagram



4.3.2. Pin/Port Definitions

Name	Length	Direction	Description
clock_in	1	input	CPU core clock

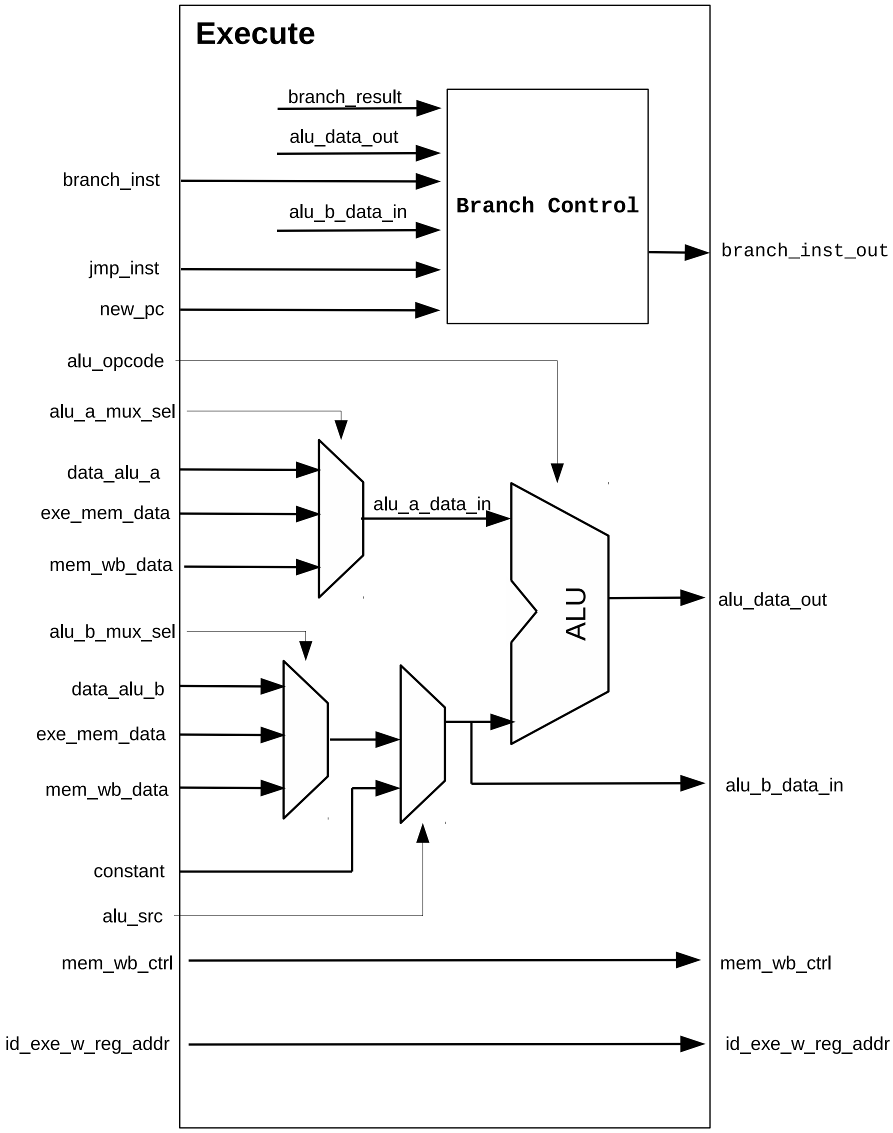
continued on next page

continued from previous page

Name	Length	Direction	Description
reset_in	1	input	CPU core reset
alu_opcode_in	6	input	ALU opperation code
alu_function_in	6	input	Selects ALU function
data_alu_a_in	32	input	ALU input A data
data_alu_b_in	32	input	ALU input B data
reg_a_addr_in	5	input	ALU Port-A addr in Register File
reg_b_addr_in	5	input	ALU Port-B addr in Register File
constant_in	32	input	32-bit constant
imm_inst__in	1	input	TBD
new_pc_in	20	input	Updated value of PC
pc_offset_in	26	input	Constant offset in J-Instructions
branch_inst_in	1	input	Branch instruction control signal
jmp_inst_in	1	input	Incoditional branch control signal
jmp_use_r_in	1	input	TBD
exe_mem_reg_data_in	32	input	Forwarding data from the MEM stage data register
exe_mem_reg_addr_in	5	input	Forwarding address from the MEM stage
exe_mem_reg_wr_ena_in	1	input	Forwarding write enable signal from the MEM stage
w_reg_data_in	32	input	GPR bank data
w_reg_addr_in	5	input	GPR bank destiny address
w_reg_wr_en_in	1	input	GPR bank write enable
instruction_in	1	input	CPU core instruction
mem_data_out	1	output	Forwarding output data to be writen in data memory
alu_data_out	32	output	ALU output data
fetch_new_pc_out	32	output	TBD
fetch_select_new_pc_out	32	output	TBD

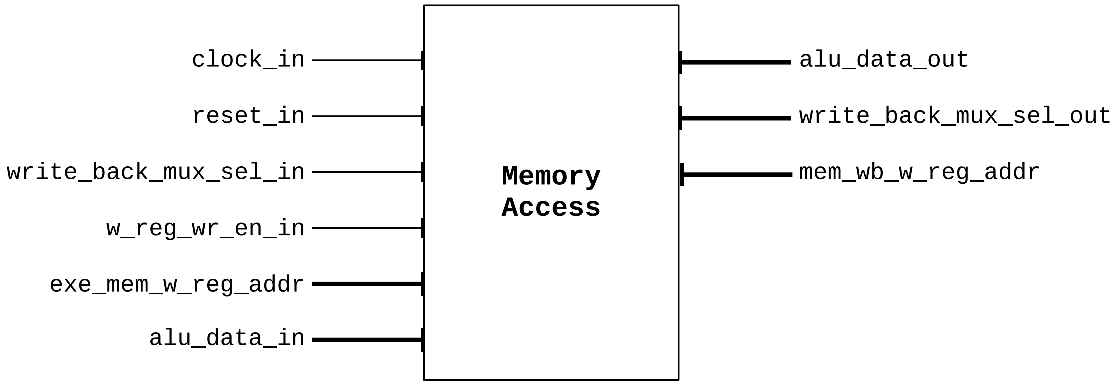
4.3.3. *Internal Datapath*

The internal data path is composed by the following components.


Execute
branch_result
alu_data_out
branch_inst
alu_b_data_in
Branch Control
branch_inst_out
jmp_inst
new_pc
alu_opcode
alu_a_mux_sel
data_alu_a
exe_mem_data
mem_wb_data
alu_a_data_in
ALU
alu_data_out
alu_b_mux_sel
data_alu_b
exe_mem_data
mem_wb_data
alu_b_data_in
constant
alu_src
mem_wb_ctrl
mem_wb_ctrl
id_exe_w_reg_addr
id_exe_w_reg_addr

4.4. Memory Access

4.4.1. Block Diagram

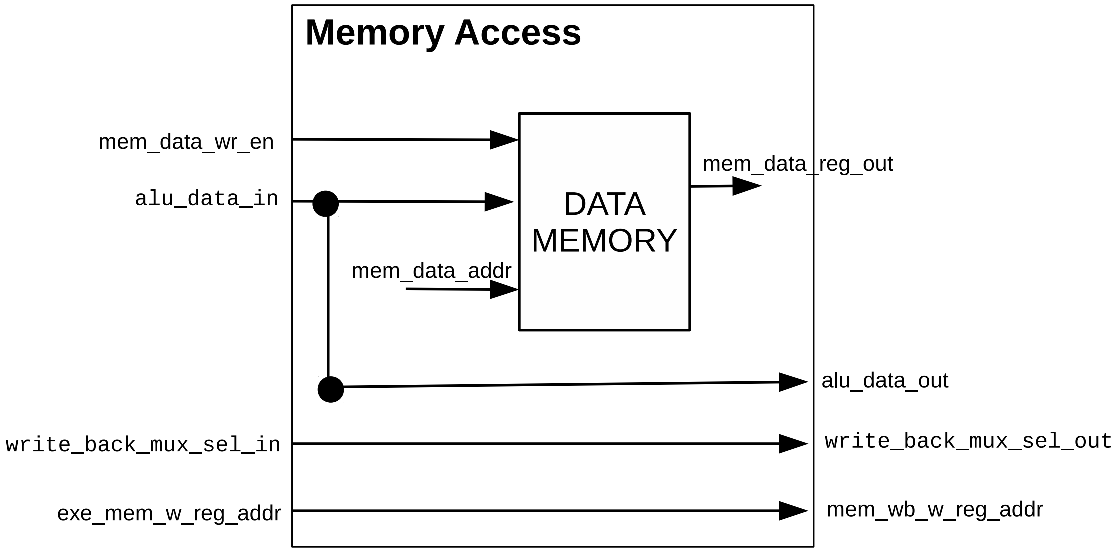


4.4.2. Pin/Port Definitions

Name	Length	Direction	Description
clock_in	1	input	CPU core clock
reset_in	1	input	CPU core reset
write_back_mux_sel_in	1	input	Write back mux select
w_reg_wr_en_in	1	input	GPR bank write enable signal
exe_mem_w_reg_addr	TBD	input	GPR bank destiny address
alu_data_in	32	input	ALU data output
alu_data_out	32	output	ALU data output
write_back_mux_sel_out	TBD	output	Write back mux select
mem_wb_w_reg_addr	TBD	output	GPR bank destiny address

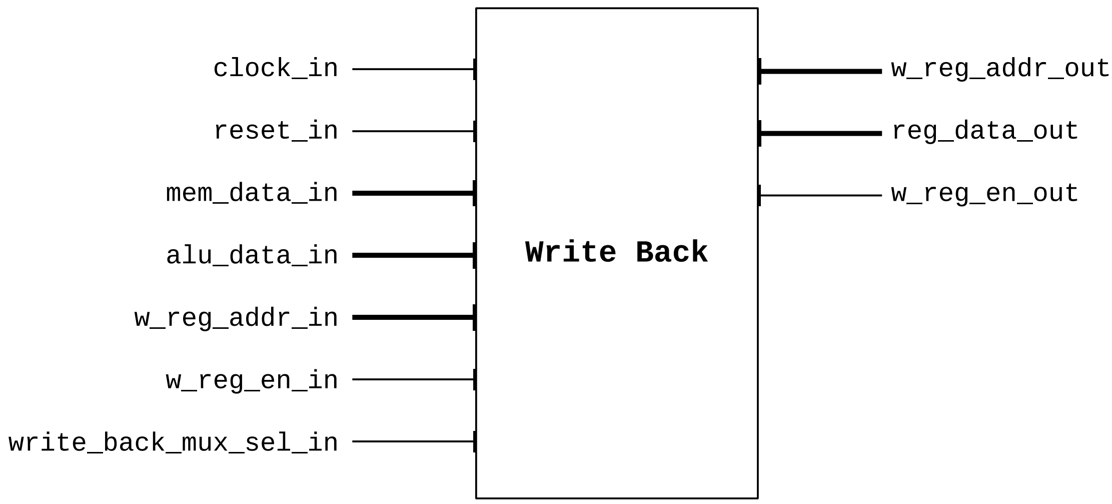
4.4.3. Internal Datapath

The internal data path is composed by the following components.


Memory Access
mem_data_wr_en
alu_data_in
DATA
MEMORY
mem_data_reg_out
mem_data_addr
alu_data_out
write_back_mux_sel_in
write_back_mux_sel_out
exe_mem_w_reg_addr
mem_wb_w_reg_addr

4.5. Write Back

4.5.1. Block Diagram

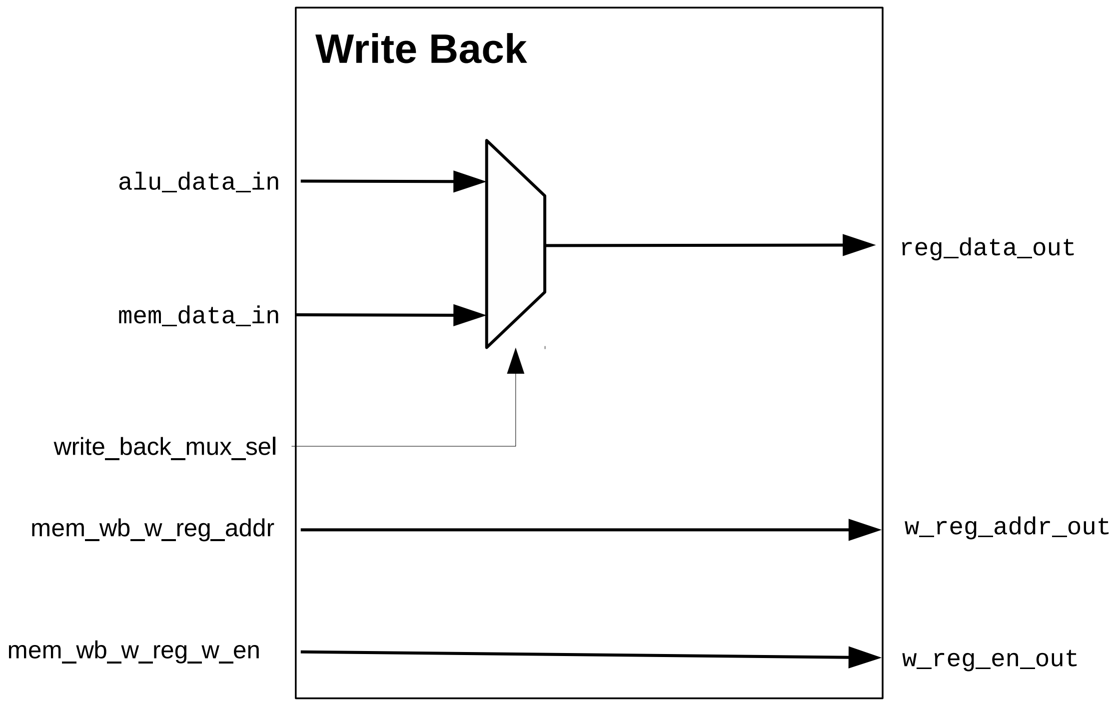


4.5.2. Pin/Port Definitions

Name	Length	Direction	Description
clock_in	1	input	CPU core clock
reset_in	1	input	CPU core reset
mem_data_in	32	input	SDRAM data output
alu_data_in	32	input	ALU data output
w_reg_en_in	4	input	GPR bank write enable signal
w_reg_addr_in	1	input	GPR bank destiny address
write_back_mux_sel	TBD	input	Write back mux select
w_reg_addr_out	4	output	GPR bank destiny address
reg_data_out	32	output	GPR bank write data
w_reg_en_out	1	output	GPR bank write enable signal

4.5.3. Internal Datapath

The internal data path is composed by the following components.


Write Back
alu_data_in
mem_data_in
write_back_mux_sel
mem_wb_w_reg_addr
mem_wb_w_reg_w_en
reg_data_out
w_reg_addr_out
w_reg_en_out

4.6. Pipeline Register Description

The following desciptions hightlights the datapath pipeline registers. For general purpose, the clock and reset signals was supressed. Althought, flush signal was also removed from IF/ID, ID/EX and EX/MEM pipeline registers.

4.6.1. *Instruction Fetch/Instruction Decode*

Name	Length	Description
clk	1	CPU core clock
rst	1	CPU core reset
stall	1	Indicates a stall insertion on the datapath
flush	1	Force flush in pipeline registers
pc	20	Stores the next program counter value.
inst_mem_data	32	Stores the intruction word.

4.6.2. *Instruction Decode/Execute*

Name	Length	Description
clk	1	CPU core clock
rst	1	CPU core reset
flush	1	Force flush in pipeline registers
data_alu_a	32	Stores the value of ALU input port A.
data_alu_b	32	Stores the value of ALU input port B.
new_pc	20	Stores the next program counter value.
instruction	32	Stores the intruction word.
opcode	3	Stores the instruction opperation code.
inst_function	6	Stores ALU function
reg_rd_addr1	6	Stores GPR source A address
reg_rd_addr2	6	Stores GPR source B address
reg_wr_addr	6	Stores GPR destination address
reg_wr_en	1	Stores the signal to enable GPR write back.
constant	32	Stores the signed extended integer constant.

continued on next page

continued from previous page

Name	Length	Description
immst	1	TBD
pc_offset	26	Stores the PC offset in a J-Instruction
mem_data_rd_en	1	Stores read enable for data memory
mem_data_rw_en	1	Stores write enable for data memory
write_back_mux_sel	1	Stores the select signal for write back mux.
branchst	1	TBD
jump_inst	1	TBD
jump_use_r	1	TBD

4.6.3. *Execute/Memory Access*

Name	Length	Description
mem_data_rd_en	1	TBD
mem_data_wr_en	1	TBD
mem_data	32	TBD
alu_data	32	Stores the output ALU Data
reg_wr_en	1	Stores the write register enable signal
reg_wr_addr	5	Stores the write register address
write_back_mux_sel	1	Stores the select signal for write back Multiplexer.
select_new_pc	1	TBD
instruction	32	Stores the intruction word.

4.6.4. *Memory Access/Write Back*

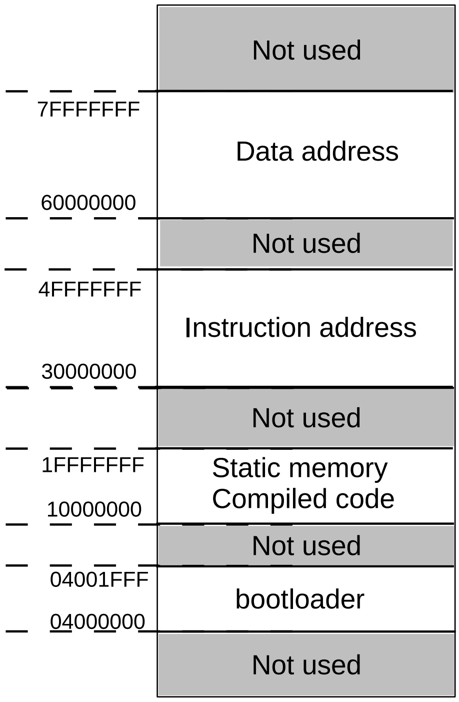
Name	Length	Description
write_back_mux_sel	1	Stores the select signal for write back Multiplexer.
alu_data	32	Stores the ALU output data.
reg_wr_en	1	Stores the signal to enable GPR write back.
reg_wr_addr	5	Stores the GPR write address.

continued on next page

continued from previous page		
Name	**Length**	**Description**
instruction	32	Stores the intruction word.

4.7. Memory and Device Interface

The memory and device interface is responsible for memory mapping to the processor. Depending on the address accessed a specific memory will be accessed by the memory device interface. If it is a read operation in the next clock cycle the read data will be directed to the DLX input read data port.

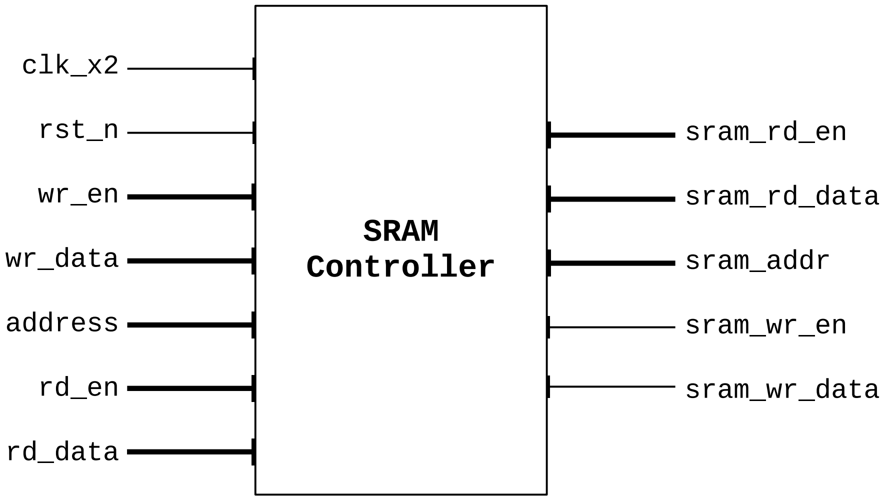


The memory map shown was created with separate address to show that devices have no relation between them and to show that the memory used in this project will be a small fraction for the 4GBytes possible.

4.8. SRAM Controller

The SRAM will store the instructions, as the SRAM has only 16 bits of word and an instruction has 32 bits the read and write operation must be done in a faster clock domain to avoid stopping the microprocessor. The SRAM has instruction code this way this memory is not written in normal operation. The SRAM will be written by the processor just during the bootloader code is running and during this operation no read will be done. Not having read and write simultaneously make easier the control not needing to deal with write and read operations at the same time.

4.8.1. Block Diagram



4.8.2. Pin/Port Definitions

Name	Length	Direction	Description
clk_x2	1	input	SRAM clock that is twice the DLX clock
rst_n	1	input	System asynchronous reset
wr_en	1	input	Data write enable
wr_data	32	input	Data to be written in the memory
address	24	input	Memory address from the DLX
rd_en	1	input	Data read enable
rd_data	32	output	Read data from SRAM to the DLX
sram_rd_en	1	output	TBD
sram_rd_data	16	output	TBD
sram_addr	10	output	Memory address that goes to the SRAM
sram_wr_en	1	output	TBD

continued on next page

Name	Length	Direction	Description
sram_wr_data	16	output	TBD

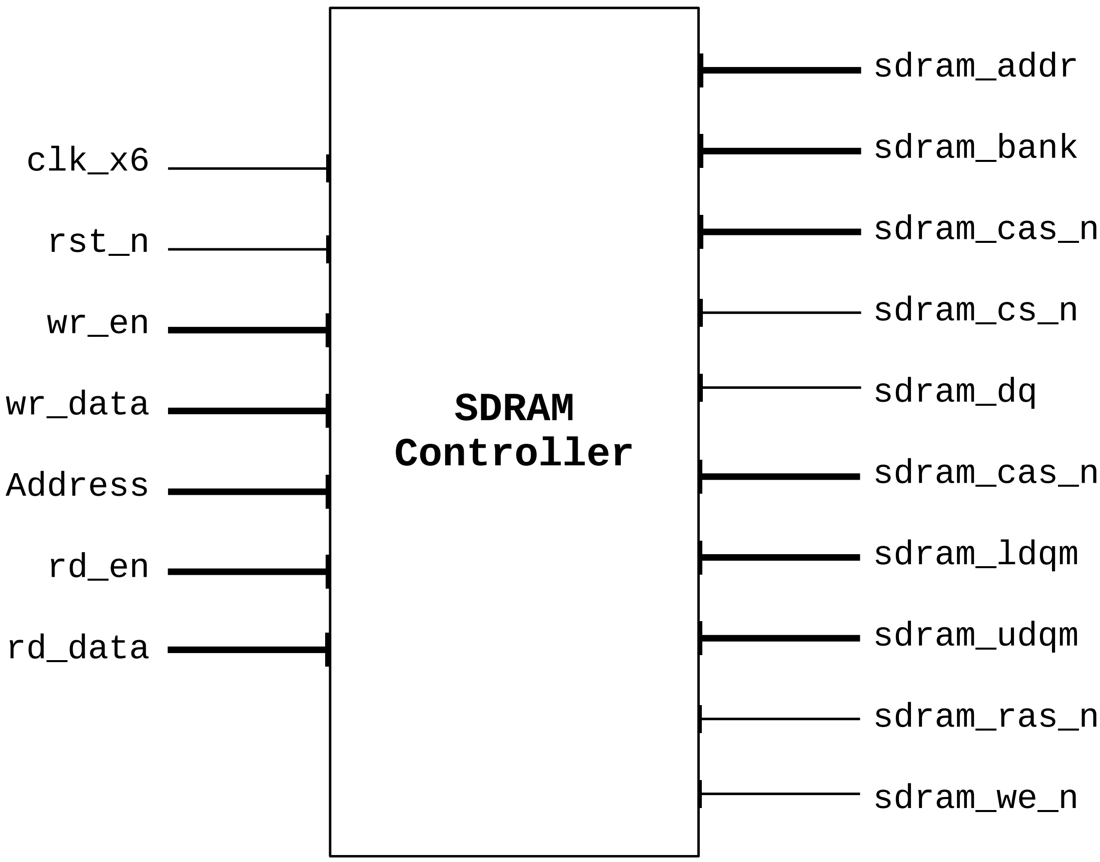
continued from previous page

As the SRAM has half word the controller works with a clock twice faster enabling read two address, concatenating and sending to the DLX in time.

4.9. SDRAM Controller

The SDRAM controller will be very simple and optimized to read or write one word that will be requested by the processor. There will be no burst, just one word write or read operation.

4.9.1. Block Diagram



4.9.2. Pin/Port Definitions

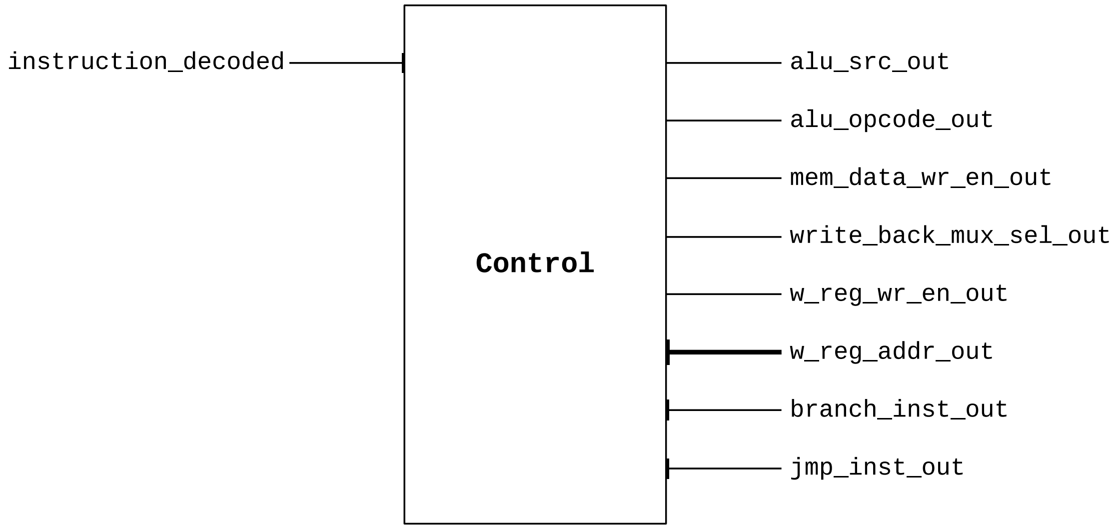
Name	Length	Direction	Description
clk	1	input	SDRAM controller input clock. 6 times faster than DLX clock
rst_n	1	input	System asynchronous reset
wr_en	1	input	Data write enable
wr_data_in	32	input	Data to be written in the memory
address	24	input	Address of write or read operation
rd_en	1	input	Data read enable

continued on next page

continued from previous page			
Name	**Length**	**Direction**	**Description**
rd_data	32	output	Data read from the SDRAM to the memory interface
sdram_addr	12	output	SDRAM data address
sdram_bank	2	output	The SDRAM selected Bank
sdram_cas_n	1	output	SDRAM CAS
sdram_cs_n	1	output	SDRAM chip select
sdram_dq	32	output	SDRAM read data, 2 bus of 16 bits, one for each memory
sdram_ldqm	1	output	TBD
sdram_udqm	1	output	TBD
sdram_ras_n	1	output	SRAM RAS
sdram_we_n	1	output	SDRAM write enable

4.10. Control Micro-instructions Description

4.10.1. Block Diagram



4.10.2. Pin/Port Definitions

Name	Length	Direction	Description
instruction_decoded	1	input	TBD
alu_src_out_n	1	output	TBD
alu_opcode_out	1	output	TBD
mem_data_wr_en_out	1	output	TBD
write_back_mux_sel_out	1	output	TBD
w_reg_wr_en_out	1	output	TBD
w_reg_addr_out	1	output	TBD
branch_inst_out	12	output	TBD
jmp_inst_out	1	output	TBD

4.10.3. *Micro-instructions*

Signal	R-Type	I-Type					J-PC
		Arith.	Load	Store	Branch	J-R	
inst_function	I[5:0]	N/A	N/A	N/A	N/A	N/A	N/A
reg_rd_addr1	I[25:21]	I[25:21]	I[25:21]	I[25:21]	I[25:21]	I[25:21]	0
reg_rd_addr2	I[20:16]	0	0	I[20:16]	0	0	0
reg_rd_en1	1	1	1	1	1	1	0
reg_rd_en2	1	0	0	1	0	0	0
reg_wr_addr	I[15:11]	I[20:16]	I[20:16]	0	I[20:16]	N/A	0
reg_wr_en	1	1	1	0	0	0	0
immediate	0	I[15:0]	I[15:0]	I[15:0]	I[15:0]	N/A	0
imm_inst	0	1	1	1	1	1	0
pc_offset	0	0	0	0	0	0	I[25:0]
mem_data_rd_en	0	0	1	0	0	0	0
mem_data_wr_en	0	0	0	1	0	0	0
write_back_mux_sel	0	0	1	0	0	0	0
branch_inst	0	0	0	0	1	0	0
jump_inst	0	0	0	0	0	1	1
jump_use_r	0	0	0	0	0	1	0

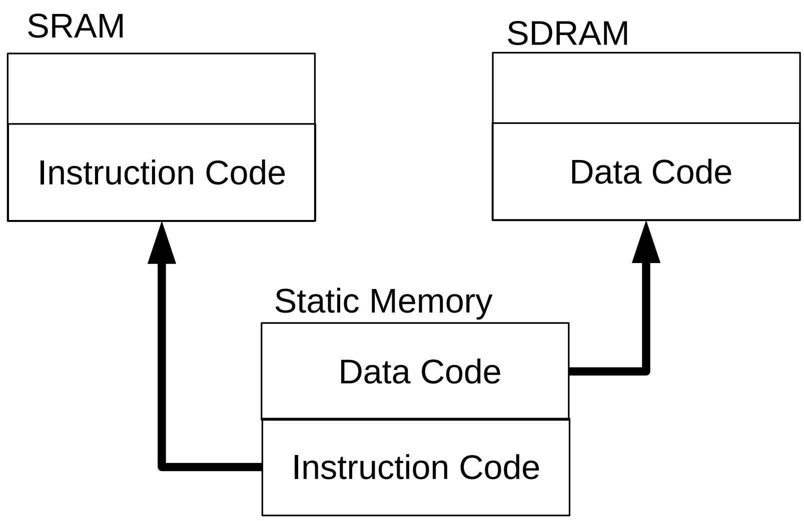
4.10.4. *Signals Descriptions*

Signal	Description
inst_function	Instruction ALU function.
reg_rd_addr1	Register file read address of Port-A.
reg_rd_addr2	Register file read address of Port-B.
reg_rd_en1	Register file read enable of Port-A.
reg_rd_en2	Register file read enable of Port-B.
reg_wr_addr	Register file write address.
reg_wr_en	Register file write enable.
immediate	Immediate halfword.
imm_inst	Indicates a Immediate instruction type.
pc_offset	Determines the PC offset lenght in a J-type instruction.
mem_data_rd_en	Data memory read enable.
mem_data_wr_en	Data memory write enable.
write_back_mux_sel	Write back mux control signal.
branch_inst	Indicates a branch instruction.
jump_inst	Indicates a jump instruction, either Jump-R or Jump-PC
jump_use_r	Indicates a Jump-R instruction.

4.11. Bootloader

The bootloader is the ROM memory has the code responsible to initialize the processor and write the code that will be executed in the correct address and consequently each memory device.

The ROM must have the DLX reset PC register address. When the system is reseted the processor will read instructions from the ROM address. The code inside the ROM will copy the instruction and data code from a static memory to the instruction memory SRAM and the data memory SDRAM.



Usually data in code is read-only data and it is common in compiled C code. Probably there will be only instruction code to be loaded during bootloader operation.