Processor DLX Integer Specification

PEAS Project

April 6, 2003

This document is the specification of the processor named "DLX integer" for tutorial. This specification is for the design by ASIP Meister.

1 Design Goal

area	delay	power consumption
30000 gates	13 ns	45000 uW/MHz

In this design, area is the first priority.

2 Pipeline structure

- 5 stages pipeline
  1. instruction fetch
  2. instruction decode / register read
  3. execution
  4. memory access
  5. register write back
- delay branch use
  The number of delay slots is 1.
- instruction bit width
  32-bit
- data bit width
  32-bit

3 Resources (modules)

In the design of ASIP Meister, following resources are indispensable. Program counter, Instruction register, Instruction memory access unit, Data memory access unit, and Register-file. Hence, these resources have to be selected.

For data operations, Arithmetic logic unit(henceforth ALU), Multiplier(henceforth MUL), Divider(henceforth DIV), and Shifter(henceforth SFT) are chosen. For sign extension, Extender(henceforth EXT) is chosen. In addition, one more EXT is added for unconditional branch instructions.

Data width of Each resource is 32-bit.

use	selected FHM	instantiated FHM name	selected parameter
Program counter	pcu	PC	incremental step = 4 add algorithm = cla (carry-lookahead)
Instruction register	register	IR	
Instruction memory access unit	imau	IMAU	
Data memory access unit	dmau	DMAU	
Register-file	register-file	GPR	register number = 32 number of read ports = 2 number of write ports = 1
ALU	alu	ALU0	adder algorithm=cla
EXT	extender	EXT0	input data bit width = 16-bit
MUL	multiplier	MUL0	multiply algorithm = seq (sequential) add algorithm = cla data type = 2's complement
DIV	divider	DIV0	divide algorithm = seq add algorithm = cla data type = 2's complement
SFT	shifter	SFT0	shift amount = variable
EXT	extender	EXT1	input data bit width = 28-bit

4 Storages

In storage specification, usage and bit format of each storage such as program counter, register file and dedicated registers are described. The storage specification of DLX integer is described as follows.

Storage Name	Resource	bit width	usage	location	binary
GPR0	GPR	32	zero reg.	original	00000
GPR28	GPR	32	frame pointer	original	11100
GPR29	GPR	32	stack pointer	original	11101
GPR30	GPR	32	link register	original	11110
GPR31	GPR	32	return register	original	11111
PC	PC	32	program counter	original	
IR	IR	32	inst. register	original	
IMEM	IMAU	32	inst. memory	original	
DMEM	DMAU	32	data memory	original	

5 I/O ports

In the design of ASIP Meister, a processor core needs Clock and Reset signal, Hence, these were already declared in the design window.

Selected DMAU obtained from FHM-DB needs an address bus, a data bus, an acknowledge signal port for handshake protocol to memory, a request signal port also for handshake, and a mode select signal port.

As for IMAU (from FHM-DB) needs an address bus and a data bus.

port name	signal direction	bit width	use
CLK	in	1	clock signal
Reset	in	1	reset signal
instAB	out	32	address bus for Instruction memory
instDB	in	32	data bus for Instruction memory
DataAB	out	32	address bus for data memory
DataDB	inout	32	data bus for data memory
DataAck	in	1	acknowledge signal for data memory
DataReq	out	1	request signal for data memory
dataWm	out	4	write mode select signal for data memory

6 Instruction Set

In this section, 4 instruction format types in DLX integer are shown. After that, instruction set overview and encoding are shown by tables.

6.1 Instruction types

Following 4 types are prepared.

1. register-register type(RR type)

   Two data are read from registers and operated. The result is written to a register.

2. register-immediate type(RI type)

   One data read from a register, and another immediate value are operated. The result is written to a register.

3. Load/Store type (LS type)

   Memory access Instructions are this type. One data read from a register is used as a base address of a memory, and another immediate value is used as a displacement address. If load instruction, data is read from memory using the specified address and write to the specified register. If store instruction, data is write to memory.

4. branch with condition type (B type)

   1 data (read from register) is compared with zero. If the result is true(1), branch is executed. Address to be jumped is determined by 16-bit PC-relative addressing.

5. unconditional branch type (J type)

   Address to be jumped is determined by 26-bit PC-relative addressing.

name	msb	lsb	field type addr mode	field attr operand name	name/value element	reg class
R_R	31	26	OP-code	name	opecode	
	25	21	Operand	name	rs0	
			Reg Direct	rs0	Resource	GPR
	20	16	Operand	name	rs1	
			Reg Direct	rs1	Resource	GPR
	15	11	Operand	name	rd	
			Reg Direct	rd	Resource	GPR
	10	0	OP-code	name	func	
R_I	31	26	OP-code	name	opecode	
	25	21	Operand	name	rs0	
			Reg Direct	rs0	Resource	GPR
	20	16	Operand	name	rs1	
			Reg Direct	rs1	Resource	GPR
	15	0	Operand	name	const	
			Immediate data	const	Immediate	
L_S	31	26	OP-code	name	opecode	
	25	21	Operand	name	rs0	
			Reg Indirect with Disp.	rs0	Resource	GPR
	20	16	Operand	name	rd	
			Reg Direct	rd	Resource	GPR
	15	0	Operand	name	const	
			Reg Indirect with Disp.	const	Displacement	
B	31	26	OP-code	name	opecode	
	25	21	Operand	name	rs0	
			Reg Direct	rs0	Resource	GPR
	20	16	Operand	name	rs1	
			Reg Direct	rs1	Resource	GPR
	15	0	Operand	name	const	
			PC relative address	const	Symbol	
J	31	26	OP-code	name	opecode	
	25	0	Operand	name	const	
			PC relative address	const	Symbol	

6.2 Instruction Set Overview

name	meaning of Instruction	Instruction type	format
ADD	signed add	RR type	“ADD” rs0 “,” rs1 “,” rd
ADDU	unsigned add	RR type	“ADDU” rs0 “,” rs1 “,” rd
ADDI	signed immediate add	RI type	“ADDI” rs0 “,” const “,” rd
ADDUI	unsigned immediate add	RI type	“ADDUI” rs0 “,” const “,” rd
SUB	signed subtarct	RR type	“SUB” rs0 “,” rs1 “,” rd
SUBU	unsigned subtract	RR type	“SUBU” rs0 “,” rs1 “,” rd
SUBI	signed immediate subtract	RI type	“SUBI” rs0 “,” const “,” rd
SUBUI	unsigned immediate subtract	RI type	“SUBUI” rs0 “,” const “,” rd
MULT	signed multiply	RR type	“MULT” rs0 “,” rs1 “,” rd
MULTU	unsigned multiply	RR type	“MULTU” rs0 “,” rs1 “,” rd
DIV	signed divide	RR type	“DIV” rs0 “,” rs1 “,” rd
DIVU	unsigned divide	RR type	“DIVU” rs0 “,” rs1 “,” rd

Table 1: Instruction Set overview 1

name	meaning of Instruction	Instruction type	format
AND	bit and	RR type	“AND” rs0 “,” rs1 “,” rd
ANDI	immediate bit and	RI type	“ANDI” rs0 “,” const “,” rd
OR	bit or	RR type	“OR” rs0 “,” rs1 “,” rd
ORI	immediate bit or	RI type	“ORI” rs0 “,” const “,” rd
XOR	bit exclusive or	RR type	“XOR” rs0 “,” rs1 “,” rd
XORI	immediate bit exclusive or	RI type	“XORI” rs0 “,” const “,” rd
SLL	shift left logical	RR type	“SLL” rs0 “,” rs1 “,” rd
SRL	shift right logical	RR type	“SRL” rs0 “,” rs1 “,” rd
SRA	shift right arithmetic	RR type	“SRA” rs0 “,” rs1 “,” rd
SLLI	shift left logical immediate	RI type	“SLLI” rs0 “,” const “,” rd
SRLI	shift right logical immediate	RI type	“SRLI” rs0 “,” const “,” rd
SRAI	shift right arithmetic immediate	RI type	“SRAI” rs0 “,” const “,” rd
SLT	set less than	RR type	“SLT” rs0 “,” rs1 “,” rd
SGT	set greater than	RR type	“SGT” rs0 “,” rs1 “,” rd
SLE	set less equal	RR type	“SLE” rs0 “,” rs1 “,” rd
SGE	set greater equal	RR type	“SGE” rs0 “,” rs1 “,” rd
SEQ	set equal	RR type	“SEQ” rs0 “,” rs1 “,” rd
SNE	set not equal	RR type	“SNE” rs0 “,” rs1 “,” rd
SLTI	set less than immediate	RI type	“SLTI” rs0 “,” const “,” rd
SGTI	set greater than immediate	RI type	“SGTI” rs0 “,” const “,” rd
SLEI	set less equal immediate	RI type	“SLEI” rs0 “,” const “,” rd
SGEI	set greater equal immediate	RI type	“SGEI” rs0 “,” const “,” rd
SEQI	set equal immediate	RI type	“SEQI” rs0 “,” const “,” rd
SNEI	set not equal immediate	RI type	“SNEI” rs0 “,” const “,” rd

Table 2: Instruction Set overview 2

name	meaning of Instruction	Instruction type	format
LHI	load high immediate	RI type	“LHI” const “,” rd
LB	load byte	LS type	“LB” addr “,” rd
LBU	load byte unsigned	LS type	“LBU” addr “,” rd
LH	load halfword	LS type	“LH” addr “,” rd
LHU	load halfword unsigned	LS type	“LHU” addr “,” rd
LW	load word	LS type	“LW” addr “,” rd
SB	store byte	LS type	“SB” rd “,” addr
SH	store halfword	LS type	“SH” rd “,” addr
SW	store word	LS type	“SW” rd “,” addr
BEQZ	branch equal zero	B type	“BEQZ” rs0 “,” rs1 “,” rd
BNEZ	branch not equal zero	B type	“BNEZ” rs0 “,” rs1 “,” rd
J	jump (unconditional branch)	J type	“J” const
JAL	jump (unconditional branch) and link register	J type	“JAL” const
JR	jump (unconditional branch) register	RR type	“JR” rs0
JALR	jump (unconditional branch) and link register	RR type	“JALR” rs0

Table 3: Instruction Set overview 3

6.3 Instruction encoding

name	ope-code 31..26	register 1 25..21	register 2 20..16	register 3 15..11	ope-code 10..0
SLL	000000	rs0	rs1	rd	00000000000
SRL	000000	rs0	rs1	rd	00000000010
SRA	000000	rs0	rs1	rd	00000000011
JR	000000	rs0	rs1	rd	00000001000
JALR	000000	rs0	rs1	rd	00000001001
MULT	000000	rs0	rs1	rd	00000011000
MULTU	000000	rs0	rs1	rd	00000011001
DIV	000000	rs0	rs1	rd	00000011010
DIVU	000000	rs0	rs1	rd	00000011011
ADD	000000	rs0	rs1	rd	00000100000
ADDU	000000	rs0	rs1	rd	00000100001
SUB	000000	rs0	rs1	rd	00000100010
SUBU	000000	rs0	rs1	rd	00000100011
AND	000000	rs0	rs1	rd	00000100100
OR	000000	rs0	rs1	rd	00000100101
XOR	000000	rs0	rs1	rd	00000100110
SLT	000000	rs0	rs1	rd	00000101010
SGT	000000	rs0	rs1	rd	00000101011
SLE	000000	rs0	rs1	rd	00000101100
SGE	000000	rs0	rs1	rd	00000101101
SEQ	000000	rs0	rs1	rd	00000101110
SNE	000000	rs0	rs1	rd	00000101111

Table 4: RR type

name	ope-code 31..26	immediate value 25..0
J	000010	const
JAL	000011	const

Table 5: J type

name	ope-code 31..26	register 1 25..21	register 2 20..16	immediate value 15..0
BEQZ	000100	rs0	rs1	const
BNEZ	000101	rs0	rs1	const
ADDI	001000	rs0	rs1	const
ADDUI	001001	rs0	rs1	const
SUBI	001010	rs0	rs1	const
SUBUI	001011	rs0	rs1	const
ANDI	001100	rs0	rs1	const
ORI	001101	rs0	rs1	const
XORI	001110	rs0	rs1	const
LHI	001111	rs0	rs1	const
SLLI	010000	rs0	rs1	const
SRLI	010001	rs0	rs1	const
SRAI	010010	rs0	rs1	const
SLTI	011010	rs0	rs1	const
SGTI	011011	rs0	rs1	const
SLEI	011100	rs0	rs1	const
SGEI	011101	rs0	rs1	const
SEQI	011110	rs0	rs1	const
SNEI	011111	rs0	rs1	const
LB	100000	rs0	rs1	const
LH	100001	rs0	rs1	const
LW	100011	rs0	rs1	const
LBU	100100	rs0	rs1	const
LHU	100101	rs0	rs1	const
SB	101000	rs0	rs1	const
SH	101001	rs0	rs1	const
SW	101011	rs0	rs1	const

Table 6: RI/B/LS type

7 Reset Interrupt

The DLX integer processor has the reset interrupt. (Under current version of ASIP Meister, only reset interrupt is supported.)

The implemented reset interrupt is shown below.

- occurrence condition

  This interrupt occurs when reset port received signal “1”. (This condition is the only supported condition under the current version.)

- reset operation

  PC, IR and GPR are cleared.

- number of execution cycles

  1 cycle

8 Behavior

name	meaning of Instruction	Behavior
ADD	signed add	rd = rs1 + rs2
ADDU	unsigned add	rd = rs1 + rs2
ADDI	signed immediate add	rd = rs1 + const
ADDUI	unsigned immediate add	rd = rs1 + const
SUB	signed subtarct	rd = rs1 - rs2
SUBU	unsigned subtract	rd = rs1 - rs2
SUBI	signed immediate subtract	rd = rs1 - const
SUBUI	unsigned immediate subtract	rd = rs1 - const
MULT	signed multiply	rd = rs1 * rs2
MULTU	unsigned multiply	rd = rs1 * rs2
DIV	signed divide	rd = rs1 / rs2
DIVU	unsigned divide	rd = rs1 / rs2

Table 7: Instruction Behavior 1

name	meaning of Instruction	Behavior
AND	bit and	rd = rs1 & rs2
ANDI	immediate bit and	rd = rs1 & const
OR	bit or	rd = rs1 | rs2
ORI	immediate bit or	rd = rs1 | const
XOR	bit exclusive or	rd = rs1 ^ rs2
XORI	immediate bit exclusive or	rd = rs1 ^ const
SLL	shift left logical	rd = rs1 << rs2
SRL	shift right logical	rd = rs1 >> rs2
SRA	shift right arithmetic	rd = rs1 >>> rs2
SLLI	shift left logical immediate	rd = rs1 << const
SRLI	shift right logical immediate	rd = rs1 >> const
SRAI	shift right arithmetic immediate	rd = rs1 >>> const
SLT	set less than	rd = rs1 < rs2
SGT	set greater than	rd = rs1 > rs2
SLE	set less equal	rd = rs1 <= rs2
SGE	set greater equal	rd = rs1 >= rs2
SEQ	set equal	rd = rs1 == rs2
SNE	set not equal	rd = rs1 != rs2
SLTI	set less than immediate	rd = rs1 < const
SGTI	set greater than immediate	rd = rs1 > const
SLEI	set less equal immediate	rd = rs1 <= const
SGEI	set greater equal immediate	rd = rs1 >= const
SEQI	set equal immediate	rd = rs1 == const
SNEI	set not equal immediate	rd = rs1 != const

Table 8: Instruction Behavior 2

name	meaning of Instruction	Instruction type
LHI	load high immediate	rd = rs1 ¡¡ 16
LB	load byte	rd = *addr[7:0]
LBU	load byte unsigned	rd = *addr[7:0]
LH	load halfword	rd = *addr[15:0]
LHU	load halfword unsigned	rd = *addr[15:0]
LW	load word	rd = *addr
SB	store byte	*addr = rd
SH	store halfword	*addr = rd
SW	store word	*addr = rd
BEQZ	branch equal zero	if (rs1 == 0) PC = PC + const;
BNEZ	branch not equal zero	if (rs1 != 0) PC = PC + const;
J	jump (unconditional branch)	PC = PC + const
JAL	jump (unconditional branch) and link register	LINK = Next(PC); PC = PC + const;
JR	jump (unconditional branch) register	PC = rs1;
JALR	jump (unconditional branch) and link register	LINK = Next(PC); PC = rs1;

Table 9: Instruction Behavior 3