Implementation of a Simple Queue Processor on a FPGA

Specification

Created by Hiroki Hoshino

Date: 2009/08/06

Contents

I. Fetch Unit (QP_FU)

I/O & Register

Input

FU_I_Jump	Jump	DU
FU_I_Branch	Branch	EU
FU_I_JumpAddress	Target address of Jump	DU
FU_I_BranchAddress	Target address of Branch	EU
FU_I_IntReq	Interrupt request	Int Controller
FU_I_IntAddress	Interrupt routine address	Int Controller
FU_I_IntEnable	Interrupt enable	DU
FU_I_RFI	Return from interrupt	DU

Output

FU_O_Instruction	Instruction	DU
FU_O_PC	PC	DU

Register

FU_PC_reg	Program counter
FU_IAR_reg	Interrupt address register for saving PC

Behavior

- Instruction fetch using PC address from instruction memory
- Renew PC
  - ➢ Check branch
    - ● Branch is taken when either Jump or Branch is 1
    - ● Taken: New address is target adders of jump or branck
    - ● Not taken: New address is incremented PC
  - ➢ Set PC to interrupt routine address when the interrupt is requested
    - ✧ Save the PC address
      - ● Branch taken: FU_IAR_reg <= Target Address
      - ● Not taken: FU_IAR_reg <= FU_PC_reg
- IRQ priority
  1. Interrupt
  2. Return from interrupt
  3. No conditional branch
  4. Conditional branch

II. Decode Unit (QP_DU)

I/O & Register

Input

DU_I_Instruction	Fetched data	FU
DU_I_PC	PC	FU
DU_I_ARegValue	Value of SPR, for jump instruction	IU

Output

DU_O_PN	Produced number	QCU
DU_O_CN	Consumed number	QCU
DU_O_RegSel	Select SPR	QCU
DU_O_RegNum	Select SPR address	QCU
DU_O_Operand	Offset or immediate value	QCU
DU_O_ControlExe	Execution unit control signals	QCU
DU_O_ExeOp	Execution unit operation signals	QCU
DU_O_ControlMem	Memory unit control signals	QCU
DU_O_ControlWB	Write back unit control signals	QCU
DU_O_PC	PC	QCU
DU_O_ARegAddress	SPR address, for jump instruction	IU
DU_O_Jump	Jump	FU
DU_O_JumpAddress	Target address of jump	FU
DU_O_IntEnable	Interrupt enable	FU
DU_O_RFI	Return from interrupt	FU

Register

DU_O_IntEnable_reg	Interrupt enable

Behavior

- Separate instruction to 8-bit and 8-bit and analysis them
- Recognize the type of instruction from upper 8-bit
  - Decide PN, CN
    - In ALU instruction, CN = 2 when operand is 0
  - Generate control signals
    - SPR_Sel
    - Control_Exe
    - Exe_op
    - Control_Mem
    - Control_WB
    - Reg_Num
      - In Setd,seta instruction, this value is set by operand
      - In Move instruction, this value is set by operand
- Lower 8-bit is operand, immediate or offset.
- No conditional branch（When previous instruction is a_register manipulate instruction, the correct value cannot calculate in pipeline architecture）
  - Jump <= 1
  - “b” instruction
    - Jump_address <= PC + operand (sign extended and 1 bit left shifted)
  - “jump*” instruction
    - Jump_address <= a_reg_value + operand (sign extended and 1 bit left shifted)
- Interrupt handling
  - Enable interrupt instruction (eint) : DU_O_IntEnable <= 1
  - Disable interrupt instruction (dint) : DU_O_IntEnable <= 0
- Return from interrupt instruction (rfi) : Enable interrupt (DU_O_IntEnable<=1), then refill the save PC

III. Queue Computation Unit (QP_QCU)

I/O & Register

Input

QCU_I_PN	Produced number	DU
QCU_I_CN	Consumed number	DU
QCU_I_RegSel	Select SPR	DU
QCU_I_RegNum	Select SPR address	QCU
QCU_I_Operand	Offset or immediate value	DU
QCU_I_ControlExe	Execution unit control signals	DU
QCU_I_ExeOp	Execution unit operation signals	DU
QCU_I_ControlMem	Memory unit control signals	DU
QCU_I_ControlWB	Write back unit control signals	DU
QCU_I_PC	PC	DU
QCU_I_Branch	Branch instruction is taken or not	EU
QCU_I_RenewQH	New QH address, for branch	EU
QCU_I_RenewQT	New QT address, for branch	EU
QCU_I_IntReq	Interrupt enable	Int Controller
QCU_I_IntEnable	Return from interrupt	DU
QCU_I_RFI	Execution unit control signals	DU

Output

QCU_O_Src1Adderss	1st operand address	IU
QCU_O_Src2Address	2nd operand address	IU
QCU_O_DstAddress	Destination address	IU
QCU_O_Operand	Operand or immediate value	IU
QCU_O_ControlExe	Execution unit control signals	IU
QCU_O_ExeOp	Execution unit operation signals	IU
QCU_O_ControlMem	Memory unit control signals	IU
QCU_O_ControlWB	Write back unit control signals	IU
QCU_O_PC	PC	IU

Register

QCU_QH_reg	Queue head
QCU_QT_reg	Queue tail
QCU_IQHR_reg	QH save register for interrupt
QCU_IQTR_reg	QT save register for interrupt

<u>Behavior</u>

- Reg_Sel[2],[1],[0] represent that Src1, Src2, Dst use the SPR or not
  - “Reg_Sel == 3’b101” means Src1 and Dst use SPR
- Calculate Src1
  - Output just QH value
  - When Src1 wants to use SPR, Src1 address of MSB will be 1, and other bit is set by Reg_Num value
- Calculate Src2
  - Calculated by QH + Operand
- Calculate Dst
  - Output just QT value
  - When Src1 wants to use SPR, Src1 address of MSB will be 1, and other bit is set by Reg_Num value
- Calculate next QH
  - Check PCSel
    - Branch taken: QH <= RenewQH
    - Not taken: QH <= QH + CN
- Calculate next QT
  - Check PCSel
    - Taken: QT <= RenewQT
    - Not taken: QT <= QT + PN
- Interrupt
  - Save QH, QT
  - Return QH, QT using saved QH, QT when RFI is executed

IV. Issue Unit (QP_IU)

I/O & Register

Input

IU_I_Src1Address	1st operand address	QCU
IU_I_Src2Address	2nd operand address	QCU
IU_I_DstAddress	Destination address	QCU
IU_I_Operand	Operand or immediate value	QCU
IU_I_ControlExe	Execution unit control signals	QCU
IU_I_ExeOp	Execution unit operation signals	QCU
IU_I_ControlMem	Memory unit control signals	QCU
IU_I_ControlWB	Write back unit control signals	QCU
IU_I_PC	PC	QCU
IU_I_RegWrite	Write enable for QREG, SPR	EU
IU_I_WriteData	Write data	EU
IU_I_WriteAddress	Address of write data, destination	EU
IU_I_ARegAddress	"a reg" address, for no conditional branch	DU
IU_I_IntReq	Interrupt request	Int Controller
IU_I_IntEnable	Interrupt enable	DU
IU_I_RFI	Return from interrupt	DU

Output

IU_O_Src1Address	1st operand address	EU
IU_O_Src1	1st operand data	EU
IU_O_Src2	2nd operand data	EU
IU_O_DstAddress	Destination address	EU
IU_O_Operand	Offset or immediate value	EU
IU_O_ControlExe	Execution unit control signals	EU
IU_O_ExeOp	Execution unit operation signals	EU
IU_O_ControlMem	Memory unit control signals	EU
IU_O_ControlWB	Write back unit control signals	EU
IU_O_PC	PC	EU
IU_O_ARegValue	Base address, for no conditional branch	DU

Register

QREG	32 32-bit Queue Register
SPR	4 32-bit d_reg, 4 32-bit a _reg, 8 32-bit Random access register

IQREG	Save register for QREG, size is same as QREG
ISPR	Save register for SPR, size is same as SPR

Behavior

- Register fetching stage
  - Need to check dependency and do scheduling in super scalar architecture
- Fetch data using Src1_address, Src2_address
- Access to SPR when MSB of Src_address is 1
- Accessing SPR, there is possibility to happen dependency （only pipeline）
  - Example: Instruction sequence is like “Setd0LL -> Setd0LH”, then the processor accesses same address register, so that write back data is wrong
  - Solve in EU
    - Send Src1_address to EU
    - EU calculates and hold result and address
    - EU compares current Src address with previous Src address, if match, EU uses held register value
- Check Reg_Write
  - 1: Write back to QREG or SPR
  - 0: Nothing
- A_Reg fetch
  - In DU, in order to solve no conditional branch, the address of SPR is sent to DU
- Interrupt
  - Save QREG, SPR to IQREG, ISPR

V. Execution Unit (QP_EU)

I/O & Register

Input

EU_I_Src1Address	1st operand address	IU
EU_I_Src1	1st operand data	IU
EU_I_Src2	2nd operand data	IU
EU_I_DstAddress	Destination address	IU
EU_I_Operand	Offset or immediate value	IU
EU_I_ControlExe	Execution unit control signals	IU
EU_I_ExeOp	Execution unit operation signals	IU
EU_I_PC	PC	IU
EU_I_ControlMem	Memory unit control signals	IU
EU_I_ControlWB	Write back unit control signals	IU
EU_I_IntReq	Interrupt request	Int Controller
EU_I_IntEnable	Interrupt enable	DU
EU_I_RFI	Return from interrupt	DU

Output

EU_O_Result	Execution result	MU
EU_O_WriteDataToMem	Writing data for Store instruction	MU
EU_O_DstAddress	Destination address	MU
EU_O_ControlMem	Memory unit control signals	MU
EU_O_ControlWB	Write back unit control signals	MU
EU_O_BranchAddress	Target address for branch	FU
EU_O_Branch	Branch is taken or not taken	FU, QCU
EU_O_RenewQH	New QH for branch	QCU
EU_O_RenewQT	New QT for branch	QCU

Register

EU_CC_reg	Conditional code ([0]:zero, [1]:negative)
EU_SPRReg_reg	Value of previous set instruction
EU_SPRAddress_reg	Register address of previous set instruction
EU_ICC_reg	Save register for CC
EU_ISPRReg_reg	Save register for SPRReg
EU_ISPRAddress_reg	Save register for SPRAddress

Behavior

- Select a result data by Control_Exe
- Select ALU function signals and branch function signals by Exe_op
- Set instruction
  - SPR_selected selects SPR_reg if Src1_address is same as SPR_address
    - Result <= SPR_selected[HH, HL, LH, LL] is replaced by operand
  - When move instruction, return SPR1 value
    - Result <= SPR1
  - SPR_reg <= Result
  - SPR_address <= Src1_address
- LD/ST instruction
  - Generate memory address
    - Using ALU
  - Result (Memory_address) <= Src1 (d) + operand
  - LD
    - Result <= MEM[Memory_address]
  - ST
    - MEM[Memory_address] <= Src1
- ALU instruction
  - Select 2nd operand from Src2 or Operand by Immediate signal
  - Exe_op（8 types） decide func signals
    - Result <= Src1 func Src2_selected
  - Shift instruction shifts the value by amount of operand
  - ALU functions

Exe_op	Function
0	Addition
1	Subtraction
2	Or
3	And
4	Compare
5	Shift left
6	Shift right logical
7	Shift right arithmetic
8	Not

  - Comp type, set C.C
    - Set C.C_Write. Write enable.
    - Set MSB of (Src1 – Src2_Selected) to cc[1]
    - Set Nor of (Src1 – Src2_Selected) to cc[0]

      - ✧ Cc[0] represents zero or not, cc[1] represents negative or not

- Mult instruction
  - ➢ Implement multiplier
  - ➢ Signed only
- Branch instruction
  - ➢ Exe_op（2 types） decides Jump or Branch
    - ✧ Need to flush pipeline register when taken
  - ➢ Branch prediction technique is needed to avoid branch penalty (future work)
  - ➢ Branch type
    - ✧ Check C.C.
      - ● Match Condition：Branch_address <= PC + operand (sign extended) << 1
        Branch <= 1
        RenewQH <= Src1_address
        RenewQT <= Dst_address
      - ● Not match：Branch_address, RenewQH, RenewQT are not renewed
        Branch <= 0
    - ✧ Conditions table

Instruction	Meaning	C.C.		Exe_op		
		Negative	Equal	[2]	[1]	[0]
Beq	A == B	*	1	0	1	0
Bnq	A != B	*	0	0	0	1
Blt	A < B	1	0	1	0	1
Bgt	A > B	0	0	1	0	0
Ble	A <= B	1	*	0	1	1
Bge	A >= B	0	*	0	0	1

  - ➢ Jump type（resolved in DU）
    - ✧ Target_address <= Src1 + operand (sign extended) << 1
    - ✧ PCSel <= 1
    - ✧ RenewQH <= Src1_address
    - ✧ RenewQT <= Dst_address
  - ➢ Interrupt
    - ✧ Save C.C., SPR_Reg and SPR_Address

VI. Memory Unit (QP_MU)

I/O & Register

Input

MU_I_Result	Execution result	EU
MU_I_WriteDataToMem	Writing data for store instruction	EU
MU_I_DstAddress	Destination address	EU
MU_I_ControlMem	Memory unit control signals	EU
MU_I_ControlWB	Write back unit control signals	EU
MU_I_ReadDataFromPeripheral	Data from peripheral	Peripheral

Output

MU_O_Result	Execution result	WBU
MU_O_ReadDataFromMem	Data from peripheral	WBU
MU_O_DstAddress	Destination address	WBU
MU_O_ControlWB	Write back unit control signals	WBU
MU_O_ControlPeripheral	Read/Write signal for peripheral	Peripheral
MU_O_PeripheralAddress	Address bus for peripheral	Peripheral
MU_O_WriteDataToPeripheral	Data bus for peripheral	Peripheral

Register

Behavior

- Decides to access to Memory or not by Control_Mem signals
  - Mem_Read
  - Mem_Write
- Memory mapped I/O is employed then other peripherals are connected in same address bus
  - LED
  - 7 Segment LED
  - Push Button
  - Slide Switch

VII. Write Back Unit (QP_WBU)

I/O & Register

Input

WBU_I_Result	Execution result	MU
WBU_I_ReadDataFromMem	Data from peripheral	MU
WBU_I_DstAddress	Destination address	MU
WBU_I_ControlWB	Write back unit control signals	MU

Output

WBU_O_WriteDataToReg	Writing data to QREG or SPR	IU
WBU_O_DstAddress	Destination address	IU
WBU_O_RegWrite	Register write enable	IU

Register

Behavior

- Select output from ALU result or memory data by Control_WB signal
  - Result_Sel
- Output Reg_Write signal

VIII. Instruction Set Architecture

The detail is appendix.

- Set Type
    - Set immediate to SPR
    - Move SPR value to QREG
    - Move QREG value to SPR
- Ld/St Type
    - Load value to memory or access peripheral
    - Store value to memory or access peripheral
- ALU Type
    - Calculate QREG value arithmetically or logically, and write back the result to QREG
- ALU immediate Type
    - Calculate QREG value and immediate
- Branch Type
    - No conditional branch instruction (b, jmp* instruction) is executed in DU, renew the PC
    - Branch instruction
        - Decide branch taken or not by C.C.
        - Use the PC as a base address, generate instruction address by adding it by operand
    - Jump instruction
        - Use the A_reg(SPR) as a base address, generate instruction address by adding it by operand

IX. Interrupt

- Interruption type
  - Interruption by Timer (int_req0)
  - Interruption by Push Switch (int_req1)
- Interrupt routine address
  - Int_req1: 0x00000200
  - Int_req0: 0x00000280
- Interrupt handling
  - Synchronous interruption
  - Interrupt is prohibited when IntEnable is 0
  - Renew the PC to interrupt routine address from next instruction fetch
  - ~~Save PC of instruction which is executed in EU to IAR~~
  - ~~Save QH, QT of instruction which is executed in EU~~
  - ~~Reset QH, QT~~
  - ~~Pipeline register reset (flushing)~~
  - Just one interruption is allowed
- Interrupt control instruction
  - Eint: Enable interrupt. Manipulate IntEnable register.
  - Dint: Disable interrupt. Manipulate IntEnable register.
  - RFI: Return from interrupt. Set PC using Interrupt Address Register(IAR).
- Before interruption
  - Save registers
    - QREG, SPR
    - QH, QT, PC
    - C.C.
  - ~~Flushing the pipeline registers~~

X. I/O

- Memory mapped I/O
  - Memory
  - 7-segment LED
  - Slide Switch
  - Timer
  - Push Switch
- Data Memory
  - 32x1024 = 4KB
  - Address range is from 1024 to 2047 (0x00000400-0x000007FF)
- 7-Segment LED
  - 0x80000000
  - Separate each 4-bit, control 8 HEX (Parallel IO, PIO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HEX7				HEX6				HEX5				HEX4				HEX3				HEX2				HEX1				HEX0			

- Timer
  - 0x80000010(Timer Command)
  - 0x80000011(Timer Counter Value)
  - Set the initial counter value by storing the value in Timer counter value address
  - Set the timer counter by storing the command in Timer command address
  - Start the timer by storing the command in Timer command address
  - Timer requests the interruption when the timer value is 0
  - Table of Timer command bit meaning

7	6	5	4	3	2	1	0
-	-	-	-	-	Number Set	-	Start/ Stop

- Slide Switch
  - 0x80000018
  - Control all slide switches. PIO
  - Each 1bit means 1 slide switch.
- Push Switch (KEY)
  - 0x80000020
  - Interrupt happens when the push switch is pushed.

Note: Implementation Rule

- Naming Rule
  - Capital letter is used
    - Src1_address => Src1Address
  - Input port
    - UnitName_I_Name
  - Output port
    - UnitName_O_Name
  - Wire
    - UnitName_Name_wire
  - Register
    - UnitName_Name_reg
- Strongly recommended to use “Case” to avoid warning