Qupls4_copro

Overview

The Qupls4 copro is co-processor that handles MMU and display controller tasks. It is capable of updating the display controller's register set at specific points during the display generation. The copro has a small instruction set. Copro instructions are 32-bit words. All data values are 64-bit.

If enabled, during every vertical reset of the display the copro's instruction pointer is set to point at location 10h and the copro begin executing instructions.

For TLB misses the copro begins executing instructions at location 008h.

Interrupts

The copro supports two sources of interrupts, the video frame interrupt and a TLB miss interrupt. Copro interrupts are very high speed. It takes three clock cycles to re-awaken from low power mode. Registers are swapped in one of those three cycles.

Operation

Normal operations is to execute a WAIT instruction which will put the copro in a lower power mode, then process the interrupt when it occurs.

When the interrupt occurs the instruction pointer for the instruction is stored on an internal stack. Then registers r1 to r7 are swapped with interrupt versions of the registers. On interrupt return the registers are swapped back and the instruction pointer is loaded from the internal stack.

Vectors

The copro vectors to the following locations on reset or interrupt.

Vector	Value
Reset	$0000
TLB Miss	$0008
Vertical Sync	$0010

Priority

TLB miss interrupts have a higher priority than display interrupts. TLB interrupts are allowed to interrupt the display interrupt processing routine.

Copro Programming Model

There are fourteen 64-bit general purpose registers r1 to r14. R0 is always zero, and r15 refers to the instruction pointer.

Copro Control Registers

Copro address registers are used to store addresses for copro programs that include the restart address (address register zero), and subroutine linkage addresses.

	63		0	
$CC0	~32	Offset31..0		Address register #0
....				14 more registers
$CF8	~32	Offset31..0		Address register #15
$CB0		F	E	Control Register

Copro Control Register

This register has bits to enable the copro. It also has a bit indicating the restart rate for the copro. The copro program may be automatically restarted at the beginning of every video frame (the default configuration) or it may restart every 16th frame.

WAITcc

Opcode: 1

Description:

The wait instruction waits for a signal combination or interrupt to occur. This could be the display generation scan to reaching a specific horizontal, vertical and frame position. The copro is not active while waiting and other devices may freely access the display register set.

While waiting the copro is placed into a lower power mode. Three clock cycles are required to exit the mode.

Instruction Format:

Bits	Field	Description
17:31	Imm	Constant
13:16	Rs2	Signal conditions mask
9:12	Rs1	Signal conditions value
5:8	Cond	
4:0	1	WAIT opcode

Cond – condition and operation to perform

$Cond_4$	cc	Jump operation
0 to 7		Reserved
9	GEP	Scan position after {Rs1, Rs2}
10		Unconditional WAIT
9 to 14		reserved

Signal Conditions:

Rs1:

Bits	Field	Description
63:60	~	reserved
48	B	blitter wait
37:32	F	frame. to wait for
27:16	V	vert. pos. to wait for
11:0	H	horiz. pos. to wait for

Rs2

Bits	Field	Description
11:0	H	horiz. pos. to wait for
37:32	MF	mask: frame
27:16	MV	mask: vertical pos
11:0	MH	mask: horizontal pos

B – indicates that the copro should wait for an outstanding blit operation to complete before continuing.

F – the frame number that the copro should wait for. This may be masked off by the frame mask (MF) field.

V – the vertical position that the copro should wait for. This may be masked by the vertical position mask field (MV)

H – the horizontal position that the copro should wait for. this may be masked by the horizontal position mask field (MH).

LOAD

Opcode: 16

Description:

The LOAD instruction moves a value from a register or memory to Rd. The load address is specified as the sum of Rs1 and a sign extended displacement constant.

Instruction Format:

Bits	Field	Description
17:31	Disp	Displacement
13:16	~	reserved
9:12	Rs1	Base Address
5:8	Rd	Destination register (not used)
4:0	16	STORE opcode

STORE

Opcode: 17

Description:

The STORE instruction moves a value from Rs2 into one of the display controller or MMU registers specified as the sum of Rs1 and a sign extended displacement constant. This allows the copro to do things like initiate a blitter operation or trigger an interrupt.

Instruction Format:

Instruction Format:

Bits	Field	Description
17:31	Disp	Displacement
13:16	Rs2	Value to store
9:12	Rs1	Base Address
5:8	Rd	Destination register (not used)
4:0	17	STORE opcode

STOREI

Opcode: 18

Description:

The STORE instruction moves an immediate value into one of the display controller or MMU registers specified as the sum of Rs1 and a sign extended displacement constant.. This allows the copro to do things like initiate a blitter operation or trigger an interrupt.

This instruction must be aligned at an odd address for correct operation. The immediate constant follows the instruction.

Instruction Format:

Bits	Field	Description
17:31	Disp	Displacement
13:16	Rs2	Value to store
9:12	Rs1	Base Address
5:8	Rd	Destination register (not used)
4:0	18	STORE opcode

Jcc

JEQ / JNE / JLT / JLE / JGE / JGT / JLEP / JGEP / DJNE

Opcode: 4

Description:

This instruction conditionally jumps to a destination address if the relationship between Rs1 and Rs2 is true.

The DJNE instruction decrements the Rs1 register then jumps if the decremented value is not equal to Rs2.

Instruction Format:

Bits	Field	Description
17:31	Address	
13:16	Rs2	Test register (mask)
9:12	Rs1	Test register (value)
5:8	Cond	Branch condition
4:0	4	Jcc opcode

Cond – condition and operation to perform

$Cond_4$	cc	Jump operation
0	EQ	Rs1 = Rs2
1	NE	Rs1 <> Rs2
2	LT	Rs1 < Rs2
3	LE	Rs1 <= Rs2
4	GE	Rs1 >= Rs2
5	GT	Rs1 > Rs2
6	DJNE	Rs1 = Rs1 -1, Rs1 != Rs2
8	LEP	Scan position before {Rs1, Rs2}
9	GEP	Scan position after {Rs1, Rs2}
10	GQE	Graphics que empty
11	GQNE	Graphics que not empty
12 to 15		reserved

JUMP

Opcode: 9

Description:

This instruction unconditionally jumps to a destination address. The destination address is the sum of Rs1 and a sign extended constant value.

Instruction Format:

Bits	Field	Description
17:31	address	
13:16	~	Reserved
9:12	Rs1	Base address
5:8	0	Op
4:0	9	JMP opcode

JSR

Opcode: 9

Description:

This instruction unconditionally jumps to a subroutine by pushing the return address on an internal stack then changing program flow to the destination address. The destination address is the sum of Rs1 and a sign extended constant value.

Instruction Format:

Bits	Field	Description
17:31	address	
13:16	~	Reserved
9:12	Rs1	Base address
5:8	1	Op
4:0	9	JMP opcode

RET

Opcode: 11

Description:

This instruction unconditionally returns from a subroutine by popping the return address off an internal stack.

Instruction Format:

Bits	Field	Description
17:31	~	reserved
13:16	~	Reserved
9:12	~	Reserved
5:8	~	Reserved
4:0	11	RET opcode

CALC_INDEX

Opcode: 12

Description:

This instruction computes the page table index or the PTE for the current miss address using the page size from the page table attributes register and the level supplied by register Rs1 and places the value in register Rd. Note the calculated value still needs to be masked according to the number of entries in the page table. This is done by CALC_ADR.

This instruction replaces about four operations with an operation executed in a single cycle.

Calculation:

Index = miss_address >> ((log2(page size)– 3) * level + log2(page size))

Instruction Format:

Bits	Field	Description
17:31	~	reserved
13:16	~	Reserved
9:12	Rs1	Table level – three LSB bits
5:8	Rd	Index value
4:0	12	opcode

CALC_ADR

Opcode: 13

Description:

This instruction computes the address of the PTE for the current miss address using the page size from the page table attributes register, the index into the page table supplied by CALC_INDEX in register Rs2 and the base address of the table supplied by register Rs1 and places the value in register Rd.

This instruction replaces about four operations with an operation executed in a single cycle.

Calculation:

Mask = (1 << (log2(page size)) - 1

Address = Rs1 | (Rs2 & mask)

Instruction Format:

Bits	Field	Description
17:31	~	reserved
13:16	Rs2	Table index (from CALC_INDEX)
9:12	Rs1	Page table address
5:8	Rd	Address of PTE
4:0	13	opcode

BUILD_ENTRYNO

Opcode: 14

Description:

This instruction builds the entry_no argument required to access the TLB. It takes the PTE index in Rs1, the way in Rs2 and a constant used to determine whether to read or write the PTE.

This instruction replaces about three operations with an operation executed in a single cycle.

Calculation:

Entry_no = Rs1[15:0] | (Rs2[7:0] << 16) | (1 << const)

Instruction Format:

Bits	Field	Description
17:31	Const	Only the lower five bits used
13:16	Rs2	way
9:12	Rs1	PTE index
5:8	Rd	Entry_no
4:0	14	opcode

BUILD_VPN

Opcode: 15

Description:

This instruction builds the VPN portion of a TLB entry required to update the TLB. It takes the current miss address, asid, and current count and builds it into a single value.

This instruction replaces about four operations with an operation executed in a single cycle.

Calculation:

VPN = (miss address) >> (log2(page size) + log2(TLB entries)) | (asid << 48) | (count << 42)

Instruction Format:

Bits	Field	Description
17:31	~	Reserved
13:16	~	Reserved
9:12	~	Reserved
5:8	Rd	VPN, ASID, COUNT
4:0	15	Opcode

ADD

Opcode: 22

Description:

This instruction adds Rs1, Rs2 and a sign extended constant.

Calculation:

Rd = Rs1 + Rs2 + Imm

Instruction Format:

Bits	Field	Description
17:31	Imm	constant
13:16	Rs2	2nd operand
9:12	Rs1	1st operand
5:8	Rd	Result value
4:0	22	Opcode

AND64

Opcode: 24

Description:

This instruction bitwise ‘ands’ Rs1, Rs2 and a 64-bit constant. The instruction must be aligned at an odd address for proper operation.

Calculation:

Rd = Rs1 & Rs2 & Imm

Instruction Format:

Bits	Field	Description
17:31	~	reserved
13:16	Rs2	2nd operand
9:12	Rs1	1st operand
5:8	Rd	Result value
4:0	24	Opcode

AND

Opcode: 24

Description:

This instruction bitwise ‘ands’ Rs1, Rs2 and a sign extended constant.

Calculation:

Rd = Rs1 & Rs2 & Imm

Instruction Format:

Bits	Field	Description
17:31	Imm	constant
13:16	Rs2	2nd operand
9:12	Rs1	1st operand
5:8	Rd	Result value
4:0	24	Opcode

OR

Opcode: 25

Description:

This instruction bitwise ‘ors’ Rs1, Rs2 and a sign extended constant.

Calculation:

Rd = Rs1 | Rs2 | Imm

Instruction Format:

Bits	Field	Description
17:31	Imm	constant
13:16	Rs2	2nd operand
9:12	Rs1	1st operand
5:8	Rd	Result value
4:0	25	Opcode

XOR

Opcode: 26

Description:

This instruction bitwise exclusive ‘ors’ Rs1, Rs2 and a sign extended constant.

Calculation:

Rd = Rs1 ^ Rs2 ^ Imm

Instruction Format:

Bits	Field	Description
17:31	Imm	constant
13:16	Rs2	2nd operand
9:12	Rs1	1st operand
5:8	Rd	Result value
4:0	26	Opcode

SHL

Opcode: 20

Description:

This instruction performs a left shift operation on Rs1 using a count which is the bitwise or of the value in Rs2 and the immediate. Usually either the count or Rs2 would be zero. The shift count is limited to five bits allowing shifts of up to 31 bits.

Calculation:

Rd = Rs1 << (Rs2 | imm)

Instruction Format:

Bits	Field	Description
17:31	Imm	Five LSBs only
13:16	Rs2	Count
9:12	Rs1	Operand to shift
5:8	Rd	Result value
4:0	20	Opcode

SHR

Opcode: 21

Description:

This instruction performs a right shift operation on Rs1 using a count which is the bitwise or of the value in Rs2 and the immediate. Usually either the count or Rs2 would be zero. The shift count is limited to five bits allowing shifts of up to 31 bits.

Calculation:

Rd = Rs1 >> (Rs2 | imm)

Instruction Format:

Bits	Field	Description
17:31	Imm	Five LSBs only
13:16	Rs2	Count
9:12	Rs1	Operand to shift
5:8	Rd	Result value
4:0	21	Opcode

IRET

Opcode: 5

Description:

This instruction conditionally returns from a subroutine by popping the return address off an internal stack. Registers r1 to r7 are swapped with the interrupt set of registers.

Instruction Format:

Bits	Field	Description
17:31	~	reserved
13:16	~	Reserved
9:12	~	Reserved
5:8	~	Reserved
4:0	5	RET opcode