Ingenic®

XBurst Microprocessor Core

User Manual

Revision: 1.0
Date: Apr. 2007

北京君正集成电路有限公司
Ingenic Semiconductor Co. Ltd

Ingenic XBurst Microprocessor Core

User Manual

Release history

Date	Revision	Change
Apr. 2006	1.0	First release

Disclaimer

Ingenic Semiconductor Co., Ltd.

Room 801C, Power Creative E, No.1 B/D ShangDi East Road, Haidian District,
Beijing 100085, China
Tel: 86-10-58851008
Fax: 86-10-58851005
Http: //www.ingenic.cn

1 Overview

XBurst microprocessor core is a high performance and low power implementation of an industry standard instructions set architecture. XBurst deploys an innovative 8-stage pipeline micro-architecture and provides superior performance, die size and power consumption comparing with existent industry RISC cores.

1.1 Block Diagram

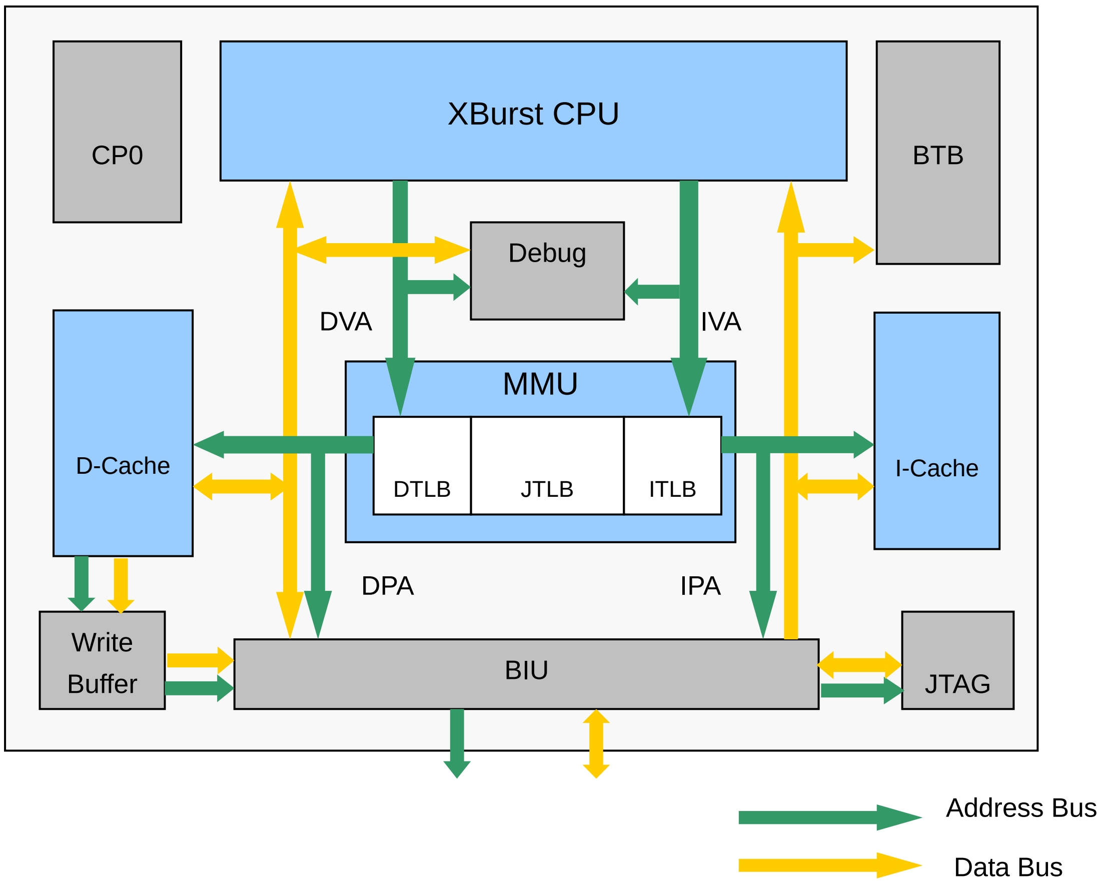


Figure 1-1-1 XBurst Diagram

1.2 Features

Table 1-1 XBurst Core Features

Item	Features
XBurst CPU	• Industry standard RISC instruction set • 32 32-bit general purpose registers, no shadow GPR • 8-stage pipeline • Interlocked implementation • Virtual address space: 4 G-Bytes
Multiply-Divide Unit (MDU)	• Maximum issue rate of one 32x16 multiply every clock • Maximum issue rate of one 32x32 multiply every other clock • Minimum 2 clock cycles, maximum 34 clock cycles for divide
Branch Target Buffer (BTB)	• Virtually-tagged • Up to 64 entry direct mapped • 2-bit branch history maintained
Memory Management Unit (MMU)	• 4 G-Bytes of address space • 32 dual-entry full associative joint TLB • 4 entry ITLB • 4 entry DTLB • 7 different page size from 4Kb to 16MB supported in any entry • Support entry lock • Space identifier ASID: 8 bits • Small (1K) page not implemented
Data Cache	• Virtually-indexed, physically-tagged • 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes • LRU replacement algorithm • Write-back, write-through • 16-word depth write buffer • Cache line lock not implemented
Instruction Cache	• Virtually-indexed, physically-tagged • 4 way, 8-word line, alterable size: 4K, 8K, 16K bytes • LRU replacement algorithm • Cache line lock not implemented
Debug&JTAG	• JTAG interface to host machine • ACC mode to accelerate JTAG memory access • Two instruction and one data breakpoint
Internal Timer	• N/A
Bus Interface	• Compliance with AHB protocol

2 CPU

2.1 CPU Registers

The CPU provides 32 general purpose 32-bit registers, a 32-bit Program Counter (PC), and two 32-bit registers that hold the result of integer multiply and divide operations.

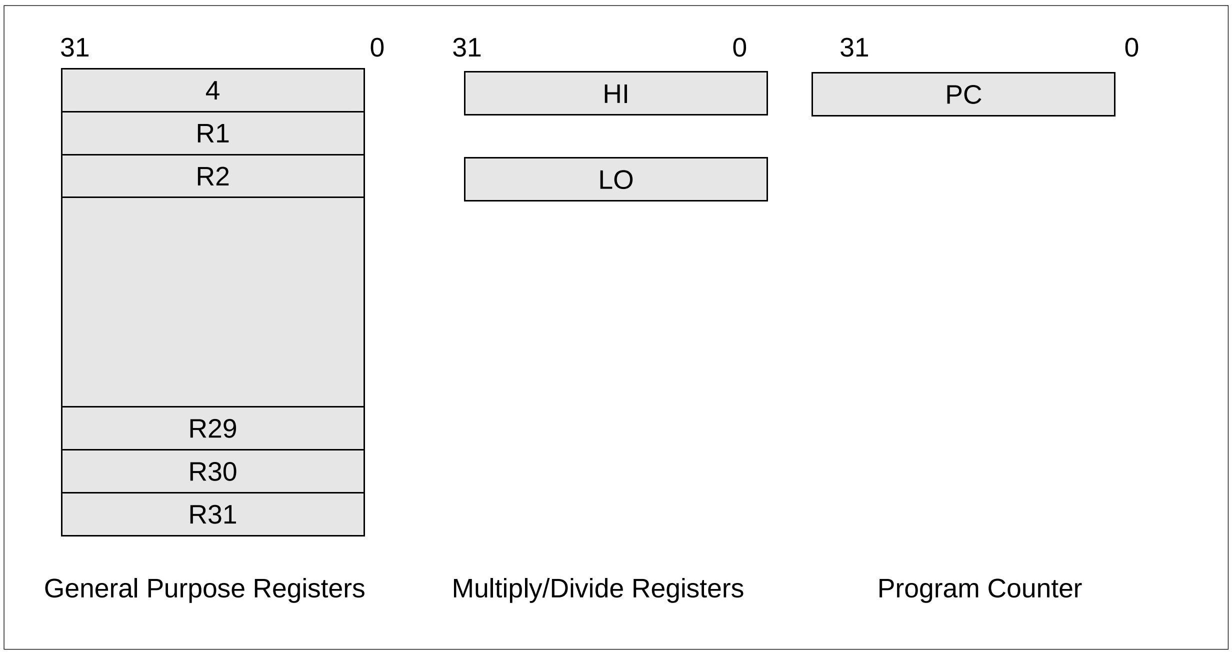


Figure 2-1 CPU Registers

A Program Control & Status registers does not exist; its functions are provided by the Status and Casue register incorporated in CP0 . CP0 registers are described in later sections.

2.2 Instruction Format

Each CPU instruction is 32 bits long. The instructions are divided into 3 format type as shown in Figure 2-2

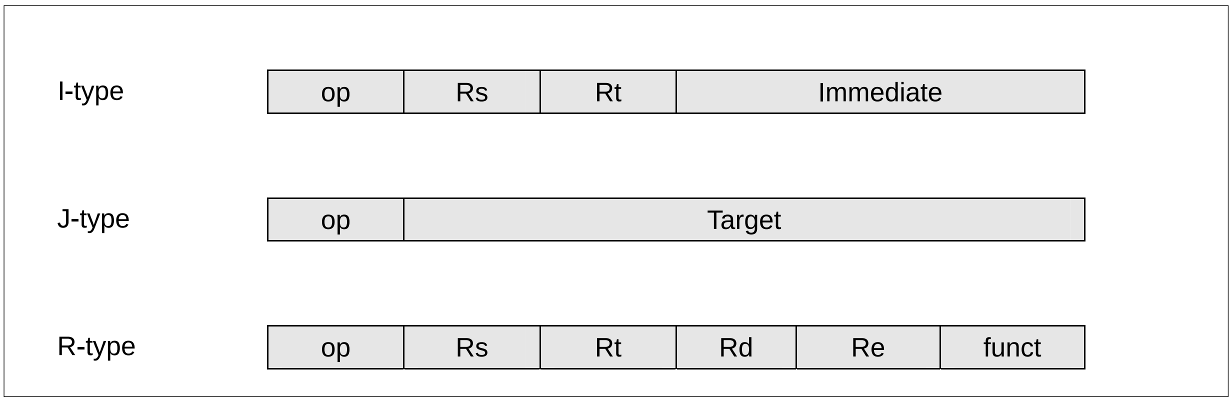


Figure 2-2CPU Instruction Format

2.3 CPU Instruction Set

Table 2-1 Arithmetic Instructions

Mnemonic	Format	Description
ADD	ADD Rd, Rs, Rt	Add Word
ADDI	ADDI Rt, Rs, Immediate	Add Immediate Word
ADDIU	ADDI Rt, Rs, Immediate	Add Immediate Unsigned Word
ADDU	ADDU Rd, Rs, Rt	Add Unsigned Word
CLO	CLO Rd, Rs	Count Leading Ones in Word
CLZ	CLZ Rd, Rs	Count Leading Zeros in Word
DIV	DIV Rs, Rt	Divide Word
DIVU	DIVU Rs, Rt	Divide Unsigned Word
MADD	MADD Rs, Rt	Multiply and Add Word to Hi, Lo
MADDU	MADDU Rs, Rt	Multiply and Add Unsigned Word to Hi, Lo
MSUB	MSUB Rs, Rt	Multiply and Subtract Word to Hi, Lo
MSUBU	MSUBU Rs, Rt	Multiply and Subtract Unsigned Word to Hi, Lo
MUL	MUL Rd, Rs, Rt	Multiply Word to GPR
MULT	MULT Rs, Rt	Multiply Word
MULTU	MULTU Rs, Rt	Multiply Unsigned Word
SLT	SLT Rd, Rs, Rt	Set on Less Than
SLTI	SLTI Rt, Rs, immediate	Set on Less Than Immediate
SLTIU	SLTIU Rt, Rs, immediate	Set on Less Than Immediate Unsigned
SLTU	SLTU Rd, Rs, Rt	Set on Less Than Unsigned
SUB	SUB Rd, Rs, Rt	Subtract Word
SUBU	SUBU Rd, Rs, Rt	Subtract Unsigned Word

Table 2-2 Logical Instructions

Mnemonic	Format	Description
AND	AND rd, rs, rt	And
ANDI	ANDI rt, rs, immediate	And Immediate
LUI	LUI rt, immediate	Load Upper Immediate
NOR	NOR rd, rs, rt	Not Or
OR	OR rd, rs, rt	Or
ORI	ORI rt, rs, immediate	Or Immediate
XOR	XOR rd, rs, rt	Exclusive Or
XORI	XORI rt, rs, immediate	Exclusive Or Immediate

Table 2-3 Shift Instructions

Mnemonic	Format	Description
SLL	SLL rd, rt, sa	Shift Word Left Logical
SLLV	SLLV rd, rt, rs	Shift Word Left Logical Variable
SRA	SRA rd, rt, sa	Shift Word Right Arithmetic
SRAV	SRAV rd, rt, rs	Shift Word Right Arithmetic Variable
SRL	SRL rd, rt, sa	Shift Word Right Logical
SRLV	SRLV rd, rt, rs	Shift Word Right Logical Variable

Table 2-4 Branch and Jump

Mnemonic	Format	Description
B	B offset	Unconditional Branch
BAL	BAL rs, offset	Branch and Link
BEQ	BEQ rs, rt, offset	Branch on Equal
BGEZ	BGEZ rs, offset	Branch on Greater Than or Equal to Zero
BGEZAL	BGEZAL rs, offset	Branch on Greater Than or Equal to Zero and Link
BGTZ	BGTZ rs, offset	Branch on Greater Than Zero
BLEZ	BLEZ rs, offset	Branch on Less Than or Equal to Zero
BLTZ	BLTZ rs, offset	Branch on Less Than Zero
BLTZAL	BLTZAL rs, offset	Branch on Less Than Zero and Link
BNE	BNE rs, rt, offset	Branch on Not Equal
J	J target	Jump
JAL	JAL target	Jump and Link
JALR	JALR rd, rs	Jump and Link Register
JR	JR rs	Jump Register

Table 2-5 Instruction Control

Mnemonic	Format	Description
NOP	NOP	No Operation
SSNOP	SSNOP	Superscalar No Operation

Table 2-6 Load, Store and Memory Control

Mnemonic	Format	Description
LB	LB rt, offset(base)	Load Byte
LBU	LBU rt, offset(base)	Load Byte Unsigned
LH	LH rt, offset(base)	Load Halfword
LHU	LHU rt, offset(base)	Load Halfword Unsigned
LL	LL rt, offset(base)	Load Linked Word
LW	LW rt, offset(base)	Load Word
LWL	LWL rt, offset(base)	Load Word Left
LWR	LWR rt, offset(base)	Load Word Right
PREF	PREF hint,offset(base)	Prefetch
SB	SB rt, offset(base)	Store Byte
SC	SC rt, offset(base)	Store Conditional Word
SH	SH rt, offset(base)	Store Halfword
SW	SW rt, offset(base)	Store Word
SWL	SWL rt, offset(base)	Store Word Left
SWR	SWR rt, offset(base)	Store Word Right
SYNC	SYNC	Synchronize Shared Memory

Table 2-7 Data Move Instructions

Mnemonic	Format	Description
MFHI	MFHI rd	Move From HI Register
MFLO	MFLO rd	Move From LO Register
MOVN	MOVN rd, rs, rt	Move Conditional on Not Zero
MOVZ	MOVZ rd, rs, rt	Move Conditional on Zero
MTHI	MTHI rs	Move To HI Register
MTLO	MTLO rs	Move To LO Register

Table 2-8 Trap Instructions

Mnemonic	Format	Description
BREAK	BREAK	Breakpoint
SYSCALL	SYSCALL	System Call
TEQ	TEQ rs, rt	Trap if Equal
TEQI	TEQI rs, immediate	Trap if Equal Immediate
TGE	TGE rs, rt	Trap if Greater or Equal
TGEI	TGEI rs, immediate	Trap if Greater of Equal Immediate
TGEIU	TGEIU rs, immediate	Trap if Greater or Equal Immediate Unsigned
TGEU	TGEU rs, rt	Trap if Greater or Equal Unsigned
TLT	TLT rs, rt	Trap if Less Than
TLTI	TLTI rs, immediate	Trap if Less Than Immediate
TLTIU	TLTIU rs, immediate	Trap if Less Than Immediate Unsigned
TLTU	TLTU rs, rt	Trap if Less Than Unsigned
TNE	TNE rs, rt	Trap if Not Equal
TNEI	TNEI rs, immediate	Trap if Not Equal Immediate

Table 2-9 Privileged Instructions

Mnemonic	Format	Description
CACHE	CACHE op, offset(base)	Perform the cache operation specified by op.
ERET	ERET	Exception Return
MFC0	MFC0 rt, rd MFC0 rt, rd, sel	Move from Coprocessor 0
MTC0	MTC0 rt, rd MTC0 rt, rd, sel	Move to Coprocessor 0
TLBP	TLBP	Probe TLB for Matching Entry
TLBR	TLBR	Read Indexed TLB Entry
TLBWI	TLBWI	Write Indexed TLB Entry
TLBWR	TLBWR	Write Random TLB Entry
WAIT	WAIT	Enter Standby Mode

Table 2-10 JTAG Debug Instructions

Mnemonic	Format	Description
DERET	DERET	Debug Exception Return
SDBBP	SDBBP code	Software Debug Breakpoint

Table 2-11 Obsolete CPU Branch Instructions

Mnemonic	Format	Description
BEQL	BEQL rs, rt, offset	Branch on Equal Likely
BGEZALL	BGEZALL rs, offset	Branch on Greater Than or Equal to Zero and Link Likely
BGEZL	BGEZL rs, offset	Branch on Greater Than or Equal to Zero Likely
BGTZL	BGTZL rs, offset	Branch on Greater Than Zero Likely
BLEZL	BLEZL rs, offset	Branch on Less Than or Equal to Zero Likely
BLTZALL	BLTZALL rs, offset	Branch on Less Than Zero and Link Likely
BLTZL	BLTZL rs, offset	Branch on Less Than Zero Likely
BNEL	BNEL rs, rt, offset	Branch on Not Equal Likely

2.4 Instruction Cycles

Most instructions in XBurst are one cycle pass, that is, when the pipeline is fully filled, there is one instruction issued in each cycle. However, some particular instructions or execution sequences require extra execution cycles. Following table lists cycle consumption of all instructions.

Instruction	Cycles	Description
WAIT	-	WAIT instruction will be repeatedly executed until an interrupt arise
MTC0 **CACHE(I)** **TLBWI/R** **EXT/INS**	4	3 extra interlock cycles
ROTR/ROTRV	3	2 extra interlock cycles
CACHE(D) **SEB/SEH**	2	1 extra interlock cycles
JALR/BCC	4/1	Zero penalty when BTB predict correctly and the branch is taken, otherwise, 3 cycles penalty
BCCL	5/4/2/1	Zero penalty when BTB taken and real taken, otherwise: 1. BTB miss, branch is taken, 3 penalties 2. BTB miss, branch is untaken, 1 penalty 3. BTB predict taken, branch is untaken, 4 cycles penalty 4. BTB predict untaken, branch is taken, 3 cycles penalty
MUL **MULT/MULTU** **MADD/MADDU** **MSUB/MSUBU**	2/1	No extra interlock cycle for 32x16 case 1 extra interlock cycle for 32x32 case
DIV/DIVU	2~34	Determined by characteristic value of divider
Others	1	

3 CP0

CP0 functions as System Control Coprocessor which provides the register interface to the XBurst processor core and supports memory management, address translation, exception handling, and other privileged operations. This section describes XBurst core's definition and implementation of CP0 registers.

Table 3-1 CP0 Registers

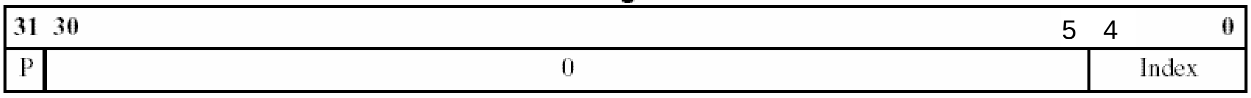
Register Number	Register Name	Function
0	Index	Index into the TLB array.
1	Random	Randomly generated index into the TLB array.
2	EntryLo0	Low-order portion of the TLB entry for even-numbered virtual pages.
3	EntryLo1	Low-order portion of the TLB entry for odd-numbered virtual pages.
4	Context	Pointer to page table entry in memory.
5	PageMask	Controls the variable page sizes in TLB entries.
6	Wired	Controls the number of fixed ("wired") TLB entries.
7	Reserved	N/A
8	BadVaddr	Reports the address for the most recent address related.
9	Count	N/A
10	EntryHi	High-order portion of the TLB entry.
11	Compare	N/A
12	Status IntCtl	Processor status and control Controls expanded interrupted mode
13	Cause	Cause of last exception
14	EPC	Program counter at last exception
15	PRId EBase	Processor identification and revision CPUNum is set to zero
16	Config Config1 Config2 Config3 Config7	Configurations registers; ***config7 is added to configure BTB***
17	LLAddr	Load linked address
18	WathLo	Watchpoint address (low order)
19	WatchHi	Watchpoint address (high order) and mask
20-22	Reserved	Reserved
23	Debug	Debug control and exception status
24	DEPC	Program counter at last debug exception
26	ErrCtl	Controls access to data and SPRAM arrays for CACHE instruction
27	Reserved	Reserved
28	TagLo/	Low-order portion of cache tag interface

	DataLo	
29	Reserved	Reserverd
30	ErrorEPC	Program counter at last error
31	DESAVE	Debug hander scratchpad register

3.1 Index Register (CP0 Register 0, Select 0)

A 32-bit read/write register that contains the index used to access the TLB for TLBP, TLBR, and TLBWI instructions. The width of the index field is 5.

Index Register Format



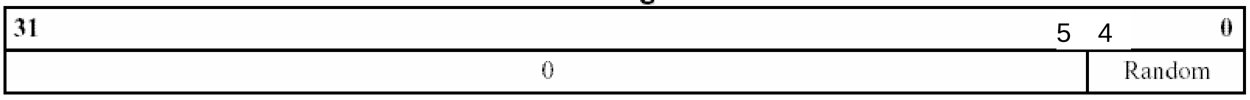
Name	Bits	Description	Read/Write	Reset state
P	31	Probe Failure bit. Set by a failed TLBP.	R	Undefined
Index	4:0	Index to a TLB entry for a TLBR or TLBWI instruction.	R/W	Undefined
0	30~5	Reserved bits	0	0

3.2 Random Register (CP0 Register 1, Select 0)

A read-only register used to index the TLB for a TLBWR instruction. The width of the index field is 5.

The processor initializes the Random register to the upper bound (0x1F) on a Reset exception and when the wired register is written.

Random Register Format

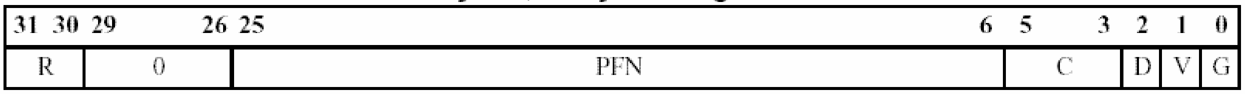


Name	Bits	Description	Read/Write	Reset state
Random	4:0	Random index for TLBWR.	R	5'0x1F
0	31:4	Reserved bits	0	0

3.3 EntryLo0, EntryLo1 Register (CP0 Register 2, 3, Select 0)

The pair of *EntryLo* registers acts as the interface between the TLB and the TLBR, TLBWI, and TLBWR instructions. The contents of the *EntryLo0* and *EntryLo1* registers are undefined after an address error, TLB invalid, TLB modified, or TLB refill exceptions.

EntryLo0, EntryLo1 **Register Format**



Name	Bits	Description	Read/Write	Reset state
PFN	25:6	Page Frame Number. PA[31:12]	R/W	Undefined
C	5:3	Cache attribute of the page.		
D	2	Dirty attribute of the page		
V	1	Valid attribute of the page		
G	0	Global attribute of the page		
R, 0	~	Reserved bits	0	0

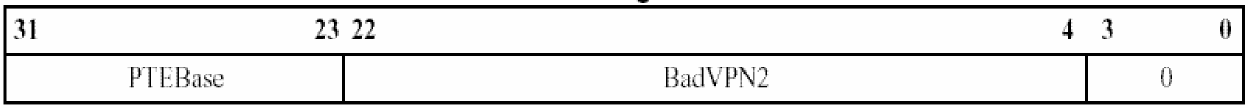
3.4 Context Register (CP0 Register 4, Select 0)

The *Context* register is a read/write register containing a pointer to an entry in the page table entry (PTE) array.

A TLB exception (TLB Refill, TLB Invalid, or TLB Modified) causes bits VA31:13 of the virtual address to be written into the BadVPN2 field of the *Context* register. The PTEBase field is written and used by the operating system.

The BadVPN2 field of the *Context* register is not defined after an address error exception.

Context **Register Format**



Name	Bits	Description	Read/Write	Reset state
PTEBase	31:23	A pointer to the PTE array in memory	R/W	undefined
BadVPN2	22:4	TLB missed VA[31:13], loaded by hardware when a TLB miss occurs.	R	Undefined
Reserved	~	Assumed to be 0.	0	0

3.5 PageMask Register (CP0 Register 5, Select 0)

The *PageMask* register is a read/write register used for reading from and writing to the TLB. It holds a comparison mask that sets the variable page size for each TLB entry. Behavior is **UNDEFINED** if a value other than those listed is used.

PageMask Register Format

31 25	24 13	12 0
0	Mask	0

Name	Bits	Description	Read/Write	Reset state
Mask	24:13	Mask bits for varying page size 0000_0000_0000: 4KB 0000_0000_0011: 16KB 0000_0000_1111: 64KB 0000_0011_1111: 256KB 0000_1111_1111: 1MB 0011_1111_1111: 4MB 1111_1111_1111: 16MB 1K page size is not implemented	R/W	Undefined
Reserved	31:25 12:0	Assumed as 0.	0	0

3.6 Wired Register (CP0 Register 6, Select 0)

The *Wired* register is a read/write register that specifies the boundary between the wired and random entries in the TLB. The width of the Wired field is 5. Wired entries are fixed, non-replaceable entries that are not overwritten by a TLBWR instruction. Wired entries can be overwritten by a TLBWI instruction. The *Wired* register is set to zero by a Reset exception. Writing the *Wired* register causes the *Random* register to reset to its upper bound.

Wired Register Format

31 5	4 0
0	Wired

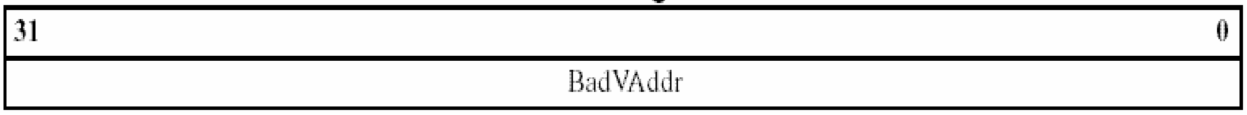
Name	Bits	Description	Read/Write	Reset state
0	31:5			
Wired	4:0	Lock pointer for TLB entries	R/W	0

3.7 BadVAddr Register (CP0 Register 8, Select 0)

The *BadVAddr* register is a read only register that captures the most recent virtual address that caused one of the following exceptions:

- Address error (AdEL or AdES)
- TLB Refill
- TLB Invalid
- TLB Modified

***BadVAddr* Register Format**



31 ... 0
BadVAddr

Name	Bits	Description	Read/Write	Reset state
BadVAddr	31:0	Failed virtual address in Address Error or TLB Faults.	R	Undefined

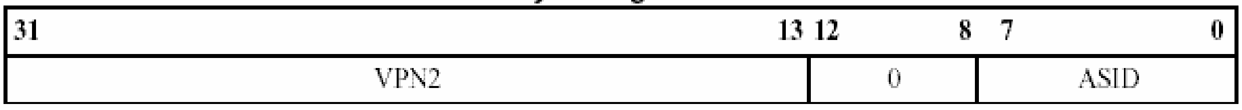
3.8 EntryHi Register (CP0 Register 10, Select 0)

The *EntryHi* register contains the virtual address match information used for TLB read, write, and access operations.

A TLB exception (TLB Refill, TLB Invalid, or TLB Modified) causes bits $VA_{31:13}$ of the virtual address to be written into the VPN2 field of the *EntryHi* register. The ASID field is written by software with the current address space identifier value and is used during the TLB comparison process to determine TLB match.

The VPN2 field of the *EntryHi* register is not defined after an address error exception.

***EntryHi* Register Format**



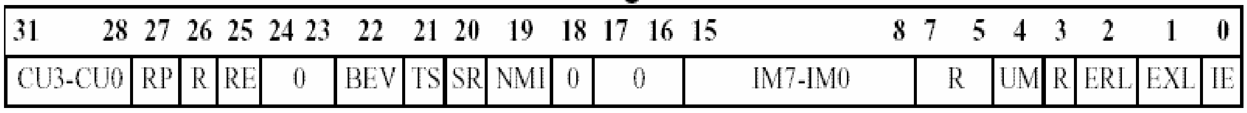
31 ... 13	12 ... 8	7 ... 0
VPN2	0	ASID

Name	Bits	Description	Read/Write	Reset state
VPN2	31:13	VA[31:13], updated when: TLB exception, TLB read, TLB write (by software)	R/W	Undefined
ASID	7:0	Current process ID. Updated by software for TLB write and address map, and by hardware on a TLB read.	R/W	Undefined
Reserved	12:8	Assumed as 0.	0	0

3.9 Status Register (CP0 Register 12, Select 0)

The Status register (SR) is a read/write register that contains the operating mode, interrupt enabling and the diagnostic states of the processor.

Status Register Format



31 28	27	26	25	24 23	22	21	20	19	18	17 16	15 8	7 5	4	3	2	1	0
CU3-CU0	RP	R	RE	0	BEV	TS	SR	NMI	0	0	IM7-IM0	R	UM	R	ERL	EXL	IE

Name	Bits	Description	Read/Write	Reset state
CU3-CU0	31-28	CU3~CU1 are not supported, always read as 0 CP0 is always enabled in kernel mode, regardless of the setting of CU0 bit	R/W	Undefined
RP	27	Enables reduced power mode. The state of the RP bit is available on the core interface.	R/W	0
RE	25	Support?	R/W	Undefined
BEV	22	Controls the location of exception vectors 0: Normal 1: Bootstrap	R/W	1
TS	21	TLB shut down	R/W	0
SR	20	Support?	0	0
NMI	19	Support?	R/W	1 for NMI
IM[7:0]	15:8	Interrupt Mask 0: Interrupt request disabled 1: Interrupt request enabled	R/W	Undefined
UM	4	Indicates that the processor is in User Mode 0: kernel mode 1: user mode	R/W	Undefined
ERL	2	Error Level. Set by the processor when a reset exception is taken. 0: normal level 1: error level	R/W	1
EXL	1	Exception Level. Set by the processor when any exception other than a Reset exception is taken. 0: normal level 1: exception level	R/W	Undefined
IE	0	Interrupt Enable 0: disable interrupts 1: enable interrupts	R/W	Undefined
Reserved	~		0	0

3.10 IntCtl Register (CP0 Register 12, Select 1)

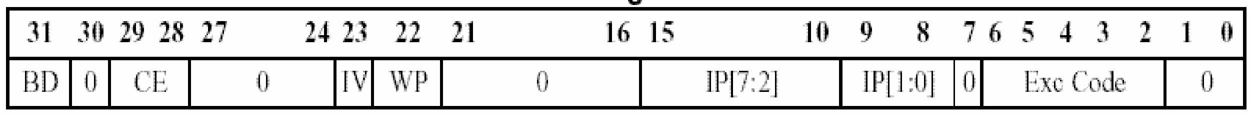
31 29	28 26	25 10	9 5	4 0
IPTI	IPPCI	0	VS	0

Name	Bits	Description	Read/Write	Reset state
IPTI	31:29	N/A	0	0
IPPCI	28:26	N/A	0	0
VS	9:5		R/W	0

3.11 Cause Register (CP0 Register 13, Select 0)

The *Cause* register primarily describes the cause of the most recent exception. In addition, fields also control software interrupt requests and the vector through which interrupts are dispatched.

Cause Register Format



Name	Bits	Description	Read/Write	Reset state
BD	31	Indicates whether the last exception taken is occurred in a delay slot. 0: Not in delay slot 1: In delay slot BD bit is not updated on a new exception if the EXL bit is set	R	Undefined
CE	29:28	Coprocessor unit number referenced when a Coprocessor unusable exception is taken.	R	Undefined
DC		N/A		
PCI		N/A		
IV	23	Interrupt exception vector 0: use the general exception vector (0x180) 1: use the special exception vector (0x200)	R/W	Undefined
WP	22	Watch exception deferred.	R/W	Undefined
IP[7:2]	15:10	Interrupt pending	R	Undefined
IP[1:0]	9:8	Controls the request for software interrupt	R/W	Undefined
Exc Code	6:2	Exception code	R	Undefined
Reserved	~		0	0

Table1-2 Cause Register ExcCode Field Descriptions

Exception Code Value	Mnemonic	Description
0	Int	Interrupt
1	Mod	TLB modification exception
2	TLBL	TLB exception (load or instruction fetch)
3	TLBS	TLB exception (store)
4	AdEL	Address error exception (load or instruction fetch)
5	AdES	Address error exception (store)
6	N/A	
7	N/A	
8	Sys	Syscall exception
9	Bp	Breakpoint exception

10	RI	Reservered instruction exception
11	CpU	Coprocessor unusable exception
12	Ov	Integer overflow exception
13	Tr	Trap exception
14-22	N/A	
23	WATCH	Reference to WatchHi/WatchLo address
24	Mcheck	Machine Check
25-31	-	Reservered

3.12 Exception Program Counter(CP0 Register 14, Select 0)

The Exception Program Counter (*EPC*) is a read/write register that contains the address at which processing resumes after an exception has been serviced.

For synchronous (precise) exceptions, the *EPC* contains one of the following:

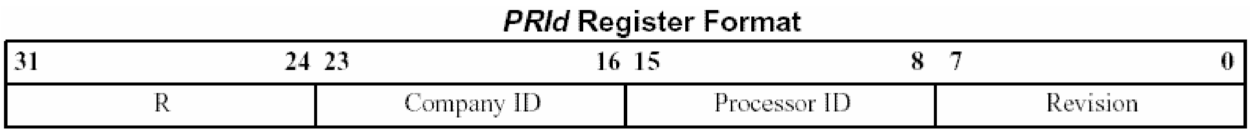
- The virtual address of the instruction that was the direct cause of the exception
- The virtual address of the immediately preceding branch or jump instruction, when the exception causing instruction is in a branch delay slot and the Branch Delay bit in the Cause register is set.

On new exceptions, the processor does not write to the *EPC* register when the EXL bit in the *Status* register is set. However, the register can still be written via the MTC0 instruction.

31   0
EPC

Name	Bits	Description	Read/Write	Reset state
EPC	31:0	Exception Program Counter	R/W	Undefined

3.13 Processor Identification (CP0 Register 15, Select 0)


PRId Register Format

31 24	23 16	15 8	7 0
R	Company ID	Processor ID	Revision

Name	Bits	Description	Read/Write	Reset state
Company ID	23:16			0x2
Processor ID	15:8		R	0x80
Revision	7:0			0x11
R	~		0	0

3.14 Ebase Register (CP0 Register 15, Select 1)

31	30	29 … 12	11 10	9 … 0
1	0	Exception Base	0 0	CPUNum

Name	Bits	Description	Read/Write	Reset state
CPUNum	9:0		R	0
Exception Base	29:12		R/W	0
R	~		0	0

3.15 Config Register (CP0 register 16, Select 0)

The *Config* register specifies various configuration and capabilities information. Most of the fields in the *Config* register are initialized by hardware during the Reset exception process,

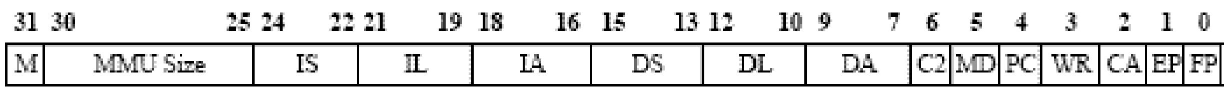
***Config* Register Format — Select 0**

31	30 28	27 25	24 21	20	19	18 17	16	15	14 13	12 10	9 7	6 3	2 0
M	K23	KU	R	MDU	R	MM	BM	BE	AT	AR	MT	0	K0

Name	Bits	Description	Read/Write	Reset state
M	31	Hardwired to 1 to indicates the presence of the Config1 register	R	1
MDU	20	Indicates MDU type. This bit is hardwired to 0 to indicate fast multiplier.	R	0
MM	18:17	Merge mode for write buffer. This bit is hardwired to 0 to indicate no merging.	0	0
BM	16	Burst order. This bit is hardwired to 0 to indicate sequential burst order	R	0
BE	15	Endian mode	R	Externally set
AT	14:13	Architecture type. This field is hardwired to 2’b00.	R	0
AR	12:10	Architecture revision level. This field is hardwired to 3’b000 to indicate revision 1.	R	0
MT	9:7	MMU type. This field is hardwired to 3’b001 to indicate TLB type MMU.	R	3’b001
K0	2:0	Kseg0 cache attributes 0~1: Cacheable, noncoherent, write-through, no write allocate 2, 7: Uncacheable 3~6: Cacheable, nonchoherent, write-back, write allocate	R/W	2
K23, KU, R, 0	~		0	0

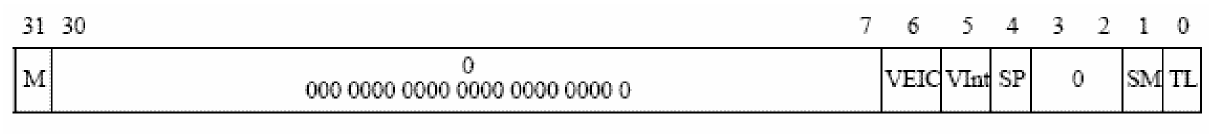
3.16 Config1 Register (CP0 Register 16, Select 1)

The *Config1* register is an adjunct to the *Config* register and encodes additional capabilities information. All fields in the *Config1* register are read-only.



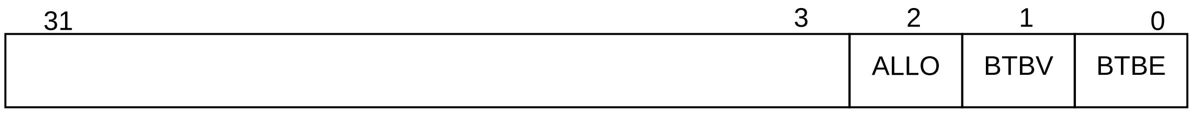
Name	Bits	Description	Read/Write	Reset state
M		Hardwired to 0 to indicate the absence of the Config2 register	R	1
MMU size	30:25	0x1F: 32 entries JTLB	R	0x1F
IS	24:22	Number of instruction cache sets per way: 0x0: 64 0x1: 128 0x2 - 0x6: Reserved 0x7: 32	R	preset
IL	21:19	Instruction cache line size: 0x0: No I-cache present 0x3: 16 bytes; 0x4: 32 bytes 0x1, 0x2, 0x5 - 0x7: Reserved	R	0x4
IA	18:16	Instruction cache associativity 0x3: 4 way	R	0x3
DS	15:13	Number of data cache sets per way: 0x0: 64 0x1: 128 0x2 - 0x6: Reserved 0x7: 32	R	preset
DL	12:10	Data cache line size. 0x0: No D-cache present 0x3: 16 bytes; 0x4: 32 bytes 0x1, 0x2, 0x5 - 0x7: Reserved	R	0x4
DA	9:7	Data cache associativity 0x3: 4 way	R	0x3
PC	4	Performance Counter register implemented	R	0
WR	3	Watch registers implemented.	R	1
CA	2	Code compression	R	0
EP	1	Debug JTAG present	R	1
FP	0	FPU implemented	R	0
R, 0	~		0	0

3.17 Config3 Register (CP0 Register 16, Select 3)



Name	Bits	Description	Read/Write	Reset state
M	31		R	0
30~7, 3~2	30:7, 3:2		0	0
VEIC	6		R	0
Vint	5		R	1
SP	4		R	0
SM	1		R	0
TL	0		R	0

3.18 Config7 Register (CP0 Register 16, Select 7)

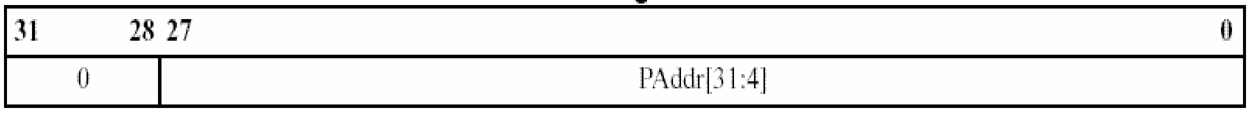


Name	Bits	Description	Read/Write	Reset state
Reserved	31:3	Reserved bits	0	0
ALLOC	2	Allocate hint of PREF instruction 0: enabled (default); 1: disabled	R/W	0
BTBV	1	BTB invalid. Writing 1 to this bit to invalidates BTB;	W	0
BTBE	0	BTB enable. 0: enabled (default); 1: disabled	R/W	0

3.19 Load Linked Address (CP0 Register 17, Select 0)

The *LLAddr* register contains the physical address read by the most recent Load Linked (LL) instruction. This register is for diagnostic purposes only, and serves no function during normal operation.

***LLAddr* Register Format**



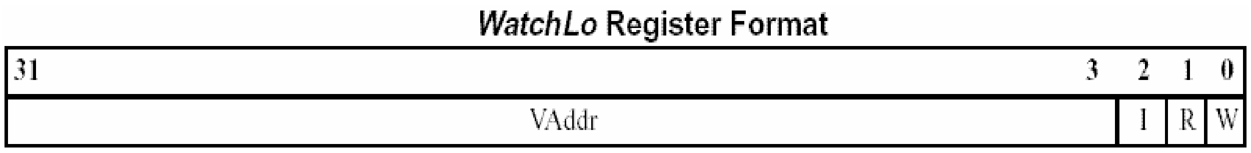
Name	Bits	Description	Read/Write	Reset state
Paddr	27:0	This field encodes the physical	R	Undefined

		address read by the most recent Load Linked instruction.		
Reserved:	31:28		0	0

3.20 WatchLo Register (CP0 Register 18, Select 0)

The *WatchLo* and *WatchHi* registers together provide the interface to a watchpoint debug facility that initiates a watch exception if an instruction or data access matches the address specified in the registers. Watch exceptions are taken only if the EXL and ERL bits are zero in the *Status* register. If either bit is a one, the WP bit is set in the *Cause* register, and the watch exception is deferred until both the EXL and ERL bits are zero.

The *WatchLo* register specifies the base virtual address and the type of reference (instruction fetch, load, store) to match.

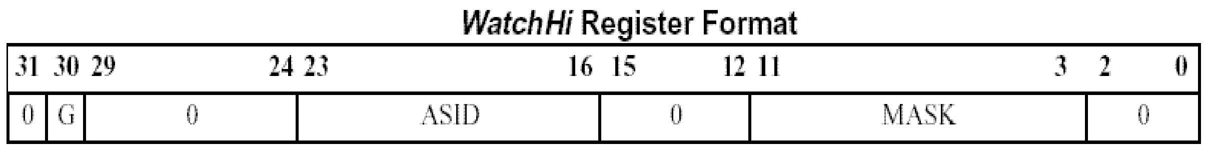

WatchLo Register Format

31 ... 3	2	1	0
VAddr	I	R	W

Name	Bits	Description	Read/Write	Reset state
Vaddr	31:3	This field specifies the virtual address to match. Note that this is a double word address, since bits [2:0] are used to control the type of match.	R/W	Undefined
I	2	If this bit is set, watch exceptions are enabled for instruction fetches that match the address.	R/W	0 for Cold Reset only
R	1	If this bit is set, watch exceptions are enabled for loads that match the address.	R/W	
W	0	If this bit is set, watch exceptions are enabled for stores that match the address.	R/W	

3.21 WatchHi Register (CP0 Register 19, Select 0)

The *WatchHi* register contains information that qualifies the virtual address specified in the *WatchLo* register: an ASID, a Global (G) bit, and an optional address mask. If the G bit is 1, any virtual address reference that matches the specified address will cause a watch exception. If the G bit is a 0, only those virtual address references for which the ASID value in the*WatchHi* register matches the ASID value in the *EntryHi* register cause a watch exception. The optional mask field provides address masking to qualify the address specified in *WatchLo*.


WatchHi Register Format

31	30	29 ... 24	23 ... 16	15 ... 12	11 ... 3	2 ... 0
0	G	0	ASID	0	MASK	0

Name	Bits	Description	Read/Write	Reset state
G	31	If this bit is one, any address that matches that specified in the *WatchLo* register causes a watch exception. If this bit is zero, the ASID field of the *WatchHi* register must match the ASID field of the *EntryHi* register to cause a watch exception.	R	Undefined
ASID	23:16	ASID value which is required to match that in the *EntryHi* register if the G bit is zero in the *WatchHi* register.	R	0 for Cold Reset only
Mask	11:3	Bit mask that qualifies the address in the*WatchLo* register. Any bit in this field that is a set inhibits the corresponding address bit from participating in the address match.	R	
Reserved	~		0	0

3.22 Debug Register (CP0 Register 23, Select 0)

The *Debug* register is used to control the debug exception and provide information about the cause of the debug exception and when re-entering at the debug exception vector due to a normal exception in debug mode. The R information bits are updated every time the debug exception is taken or when a normal exception is taken when already in debug mode.

Only the DM bit and the EJTAGver field are valid when read from non-debug mode; the value of all other bits and fields is UNPREDICTABLE. Operation of the processor is UNDEFINED if the *Debug* register is written from non-debug mode.

Some of the bits and fields are only updated on debug exceptions and/or exceptions in debug mode, as shown below:

- DSS, DBp, DDBL, DDBS, DIB, DINT are updated on both debug exceptions and on exceptions in debug modes
- DExcCode is updated on exceptions in debug mode, and is undefined after a debug exception
- Halt and Doze are updated on a debug exception, and is undefined after an exception in debug mode
- DBD is updated on both debug and on exceptions in debug modes

All bits and fields are undefined when read from normal mode, except those explicitly described to be defined, e.g. EJTAGver and DM.

3.23 Debug Exception Program Counter Register (CP0 Register 24, Select 0)

The Debug Exception Program Counter (*DEPC*) register is a read/write register that contains the address at which processing resumes after a debug exception or debug mode exception has been serviced.

***DePC* Register Format**

31 ... 0
DePC

Name	Bits	Description	Read/Write	Reset state
DEPC	31:0	Debug exception point.	R/W	Undefined

3.24 ErrCtl Register (CP0 Register 26, Select 0)

The ErrCtl register provides a mechanism for enabling software testing of the way-select and data RAM arrays for both the ICache and DCache. The way-selection RAM test mode is enabled by setting the WST bit. It modifies the functionality of the CACHE Index Load Tag and Index Store Tag operations so that they modify the way-selection RAM and leave the Tag RAMs untouched. When this bit is set, the lower 6 bits of the PA field in the TagLo register are used as the source and destination for Index Load Tag and Index Store Tag CACHE operations.

The WST bit also enables the data RAM test mode. When this bit is set, the Index Store Data CACHE instruction is enabled. This CACHE operation writes the contents of the DataLo register to the word in the data array that is indicated by the index and byte address.

The SPR bit enables CACHE accesses to the optional Scratchpad RAMs. When this bit is set, Index Load Tag, Index Store Tag, and Index Store Data CACHE instructions will send reads or writes to the Scratchpad RAM port. The effects of these operations are dependent on the particular Scratchpad implementation.

***ErrCtl* Register Format**

31 30	29	28	27 ... 0
R	WST	SPR	R

Name	Bits	Description	Read/Write	Reset state
WST	29	Indicates whether the tag array or the way-select array should be read/written on Index Load/Store Tag CACHE instructions. Also enables the Index Store Data CACHE instruction that writes the contents of DataLo to the data array.	R/W	0
SPR	28	This bit is reserved.	0.	0
Reserved	~		0	0

3.25 Taglo Register (CP0 Reigster 28, Select 0)

The TagLo register acts as the interface to the cache tag array. The Index Store Tag and Index Load Tag operations of the CACHE instruction use the TagLo register as the source and destination of tag information, respectively.

31 12	11 4	3 2	1	0
PAddr[31:12]	reserved	U[1:0]	L	V

Name	Bits	Description	Read/Write	Reset state
PA	31:12	Physical address of the indexed cache line	R/W	Undefined
U	3:2	Dirty bits of data cache. Each bit for half of a cache line	R	Undefined
L	1	Lock bit of the cache line (Not implemented)	R/W	Undefined
V	0	Valid bit of the cache line	R/W	Undefined
R	9~4	Reserved	0	0

3.26 DataLo Register (CP0 number 28, Select 1)

The *DataLo* register acts as the interface to the cache data array. The Index Load Tag operation of the CACHE instruction reads the corresponding data values into the *DataLo* register.

***DataLo* Register Format**

31 0
DataLo

Name	Bits	Description	Read/Write	Reset state
DataLo	31:0	Low-order data read from cache	R/W	Undefined

3.27 ErrorEPC Register (CP0 Register 30, Select 0)

The *ErrorEPC* register is a read-write register, similar to the *EPC* register, except that *ErrorEPC* is used on error exceptions. It is also used to store the program counter on Reset exceptions.

Unlike the *EPC* register, there is no corresponding branch delay slot indication for the *ErrorEPC* register.

***ErrorEPC* Register Format**

31 0
ErrorEPC

Name	Bits	Description	Read/Write	Reset state

ErrorEPC	31:0	Last error exception point.	R/W	Undefined

3.28 DeSave Register (CP0 Register 31, Select 0)

The Debug Exception Save (*DeSave*) register is a read/write register that functions as a simple memory location.

DeSave Register Format

31 … 0
DeSave

Name	Bits	Description	Read/Write	Reset state
DeSave	31:0	Debug exception save contents.	R/W	Undefined

4 Exception

4.1 Overview

This section describes how XBurst processor handles exceptions.

4.2 Exception Priorities

More than one exception request could occur simultaneously, where the exception with the highest priority will be accepted by CPU. The exception priorities are fixed as illustrated by the table below:

Table 4-1 XBurst Exception Priorities

Exception types	Exception Events	Exception Priorities
Reset	Reset or NMI	0 (highest)
Dfault	AdEL AdES TLBL TLBS TLB Mod MCheck	1
Dbrk	Debug data break	2
Trap Ov	Trap Overflow	3
CpU RI Sys DBp Brk	Co-processor unusable Reserved instruction SYSCALL instruction BREAK instruction Debug SDBBP instruction	4
DSS	Debug single step	5
Ifault	AdEL TLBL	6
Ibrk Dint	Debug Instruction Breakpoint or watchpoint Debug interrupt	7
INT	Hardware Interrupt or software interrupt	8 (lowest)

The exceptions supported by XBurst CPU Core are described in below:

- **RESET**: Reset exception request by a system controller outside of the CPU core. Reset exception can be induced by a power-on reset or non-masking interrupt (NMI) from external input pin. Check SR.nmi to identify NMI. Soft reset is not supported.
- **INT**: Interrupt exception induced by hardware interrupt request issued by an interrupt controller (INTC) or soft interrupt by writing CAUSE.IP directly.

- **Dfault**: Data access fault exception induced by a data access request from CPU. Check CAUSE.exccode for specific causes such as TLB refill, address error, initial write and machine check.
- **Ifault**: Instruction fetch fault induced by an instruction fetch request from CPU. Check CAUSE.exccode for specific causes such as TLB refill or address error.

- **CpU**: Co-processor unusable exception caused by executing a reserved or illegal co-processor instruction.
- **RI**: Reserved instruction exception caused by executing a normal reserved instruction.
- **Brk**: Break exception caused by executing BREAK instruction or SDBBP in debug mode.
- **DBp**: Debug break exception caused by executing SDBBP instruction in non-debug mode.
- **Sys**: Syscall exception caused by executing SYSCALL instruction.
- **Trap**: Trap exception caused by executing one of those conditional trap instructions with the condition met.
- **Ov**: Overflow exception caused by executing one of ADD/ADDI/SUB with the result overflow.
- **Dss:** Debug single step exception striken on each instruction when it is enabled.
- **Ibrk**: Debug instruction break exception induced by one of three causes, the first, when an instruction reference matches the address information stored in the *WatchHi* and *WatchLo* registers; the second, when an instruction hardware breakpoint matches an executed instruction; the third, the debug interrupt is asserted..
- **Dbrk**: Debug data break exception occurs when a data hardware breakpoint matches the load/store transaction of an executed load/store instruction or an data reference matches the address information stored in the *WatchHi* and *WatchLo* registers.

Note the specific causes are recorded into CP0 register CAUSE and DEBUG. Refer to MIPS32-4K processor manual for details.

4.3 Exception Cause

Table 6-3 Cause.ExcCode Field Descriptions

Exception Code Value	Mnemonic	Descrption
0	Int	Interrupt
1	Mod	TLB modification exception
2	TLBL	TLB exception (load or instruction fetch)
3	TLBS	TLB exception (store)
4	AdEL	Address error exception (load or instruction fetch)
5	AdES	Address error exception (store)
6	-	Reservered
7	-	Reservered
8	Sys	Syscall exception
9	Bp	Breakpoint exception

10	RI	Reservered instruction exception
11	CpU	Coprocessor unusable exception
12	Ov	Integer overflow exception
13	Tr	Trap exception
14-22	-	Reservered
23	WATCH	Reference to WatchHi/WatchLo address
24	Mcheck	Mchine Check
25-31	-	Reservered

4.4 Exception Vector Table

Table 4-2 XBurst Exception Vector Table

Exp Type	Status.BEV	Status.EXL	Cause.IV	DCR.Prob	vector
Reset, NMI	-	-	-	-	**0Xbfc0_0000**
Debug exception	-	-	-	0	**0xbfc0_0480**
				1	**0xff20_0200**
TLB Refill	0	0	-	-	**0x8000_0000**
		1			**0x8000_0180**
	1	0	-	-	**0xbfc0_0200**
		1			**0xbfc0_0380**
Interupt	0	0	0	-	**0x8000_0180**
			1		**0x8000_0200**
	1		0	-	**0xbfc0_0380**
			1		**0xbfc0_0400**
Other Exceptions	0	-	-	-	**0x8000_0180**
	1				**0xbfc0_0380**

4.5 Exception Handling Process

4.5.1 Exception Acknowledgement Process

It takes several cycles for XBurst CPU to switch from the current program flow to the exception routine.

The CPU exception acknowledgement process fulfills the following jobs:

- Set the correct exception cause;
- Compute the correct return address and saved to ErrPC for Reset exception, EPC for normal exceptions, and DEPC for debug exceptions;

- Fix the correct exception vector according to the specific exception type and jump to the handler;

- Enter debug mode by setting DEBIG.dm if a debug exception is acknowledged.

4.5.2 Return from Exception Routine

Return from exception routine is implementedby executing instruction ER**ET for non-debug exceptions or DERET for debug exceptions**

5 BTB

6 MMU

6.1 Overview

XBurst Processor Core has an on-chip memory management unit (MMU) that implements address translation. The MMU features a resident translation look-aside buffer (TLB) that caches information for user-created address translation tables located in external memory. It enables high-speed translation of virtual addresses into physical addresses. Address translation uses the paging system and supports 7 page sizes. The access right to virtual address space can be set for privileged and user modes to provide memory protection.

6.2 Virtual Memory Map

XBurst Core uses 32-bit virtual addresses to access a 4-Gbyte virtual address space that is divided into several areas according to different operation mode. Address space mapping is shown in figure 3.2.

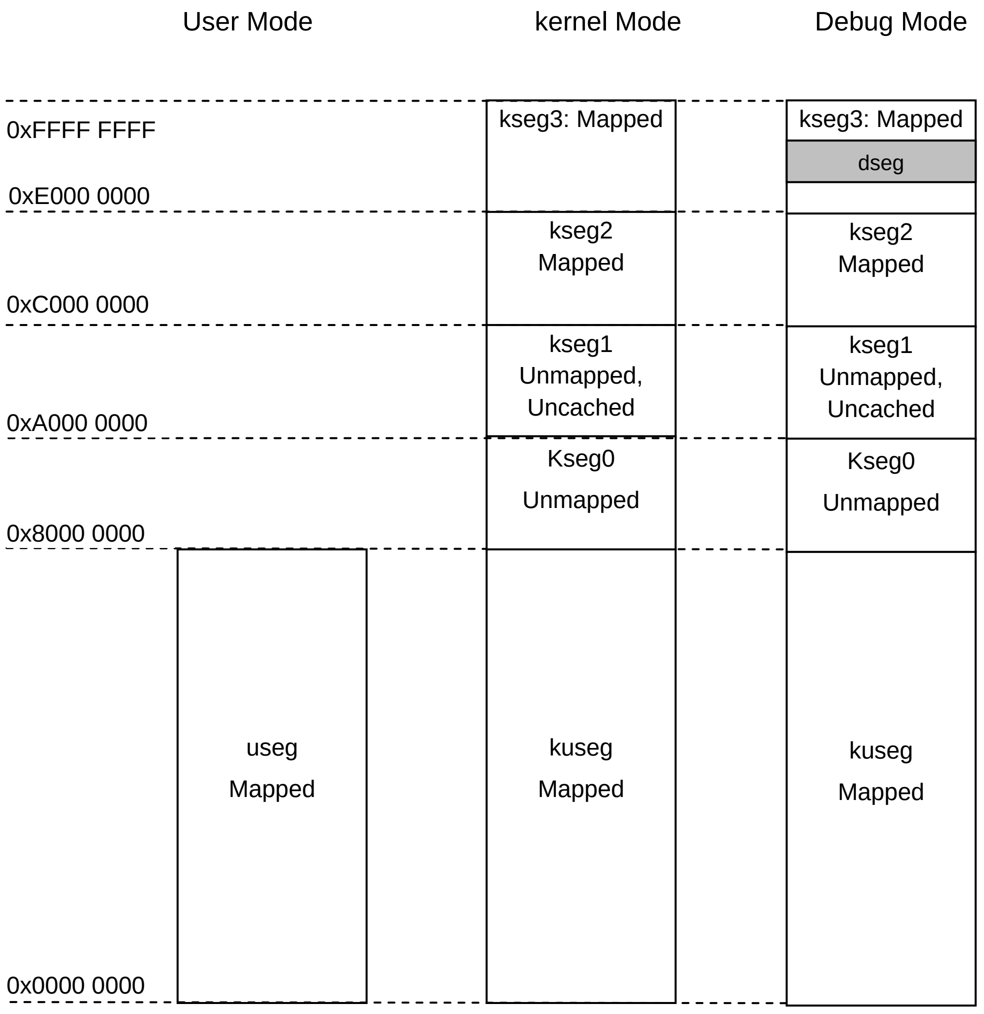


Figure 6-1 Virtual Memory Map

6.2.1 User Mode

The processor operates in User mode when the DM bit in the *Debug* register is 0 and the *Status* register contains the following bit values:

- UM = 1
- EXL = 0
- ERL = 0

The user segment useg starts at address 0x0000_0000 and ends at address 0x7FFF_FFFF. Accesses to all other addresses cause an address error exception.

The system maps all references to *useg* through the TLB. For XBurst core, the virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address before translation. Bit settings within the TLB entry for the page determine the cacheability of a reference.

6.2.2 Kernel Mode

The processor operates in Kernel mode when the DM bit in the *Debug* register is 0 and the *Status* register contains one or more of the following values:

- UM = 0
- ERL = 1
- EXL = 1

When a non-debug exception is detected, EXL or ERL will be set and the processor will enter Kernel mode. At the end of the exception handler routine, an Exception Return (ERET) instruction is generally executed. The ERET instruction jumps to the Exception PC, clears ERL, and clears EXL if ERL=0. This may return the processor to User mode.

Kernel mode virtual address space is divided into regions differentiated by the high-order bits of the virtual addres.

6.2.2.1 kuseg

kuseg address range is 0x0000_0000 - 0x7FFF_FFFF (2G byte). The virtual address is extended with the contents of the 8-bit ASID field to form a unique virtual address.

When ERL = 1 in the *Status* register, the user address region becomes a 2^{31}-byte unmapped and uncached address space. While in this setting, the kuseg virtual address maps directly to the same physical address, and does not include the ASID field.

6.2.2.2 kseg0

In Kernel mode, when the most-significant three bits of the virtual address are 100_2, 32-bit kseg0 virtual address space is selected; it is the 2^{29}-byte (512-MByte) kernel virtual space located at addresses 0x8000_0000 - 0x9FFF_FFFF. References to kseg0 are unmapped; the physical address selected is defined by subtracting 0x8000_0000 from the virtual address. The K0 field of the *Config* register controls cacheability.

6.2.2.3 kseg1

In Kernel mode, when the most-significant three bits of the 32-bit virtual address are 101_2, 32-bit kseg1 virtual address space is selected. kseg1 is the 2^{29}-byte (512-MByte) kernel virtual space located at addresses 0xA000_0000 - 0xBFFF_FFFF. References to kseg1 are unmapped; the physical address selected is defined by subtracting 0xA000_0000 from the virtual address. Caches are disabled for accesses to these addresses, and physical memory (or memory-mapped I/O device registers) are accessed directly.

6.2.2.4 kseg2

In Kernel mode, when UM = 0, ERL = 1, or EXL = 1 in the Status register, and DM = 0 in the Debug register, and the most-significant three bits of the 32-bit virtual address are 1102, 32-bit kseg2 virtual address space is selected.

6.2.2.5 kseg3

In Kernel mode, when the most-significant three bits of the 32-bit virtual address are 111_2, the kseg3 virtual address space is selected.

6.3 TLB

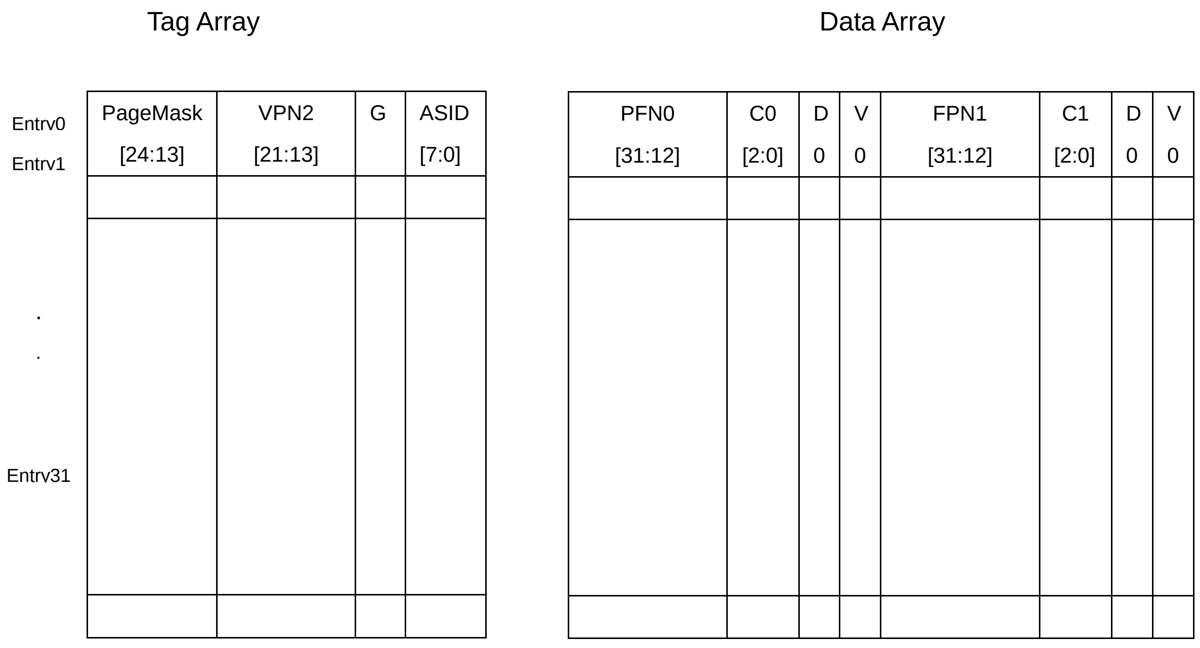
The following subsections discuss the TLB memory management scheme used in XBurst processor core. The TLB consists of one joint and two micro address translation buffers:

- 32 dual-entry fully associative Joint TLB (JTLB)
- 4-entry fully associative Instruction micro TLB (ITLB)
- 4-entry fully associative Data micro TLB (DTLB)

6.3.1 Joint TLB

XBurst core implements a 32 dual-entry, fully associative Joint TLB that maps 64 virtual pages to their corresponding physical addresses. The JTLB is organized as 32 pairs of even and odd entries containing pages that range in size from 4-KBytes to 16-MBytes into the 4-GByte physical address space. The purpose of the TLB is to translate virtual addresses and their corresponding Address Space Identifier (ASID) into a physical memory address.. Because this structure is used to translate both instruction and data virtual addresses, it is referred to as a "joint" TLB.

The JTLB is organized in page pairs to minimize its overall size. Each virtual *tag* entry corresponds to two physical data entries, an even page entry and an odd page entry. The highest order virtual address bit not participating in the tag comparison is used to determine which of the two data entries is used. Since page size can vary on a page-pair basis, the determination of which address bits participate in the comparison and which bit is used to make the even-odd determination must be determined dynamically during the TLB lookup.



Structure of Joint TLB

Table 7-6 TLB Tag Entry Fields

Field Name	Description
PageMask[24:13]	Mask bits for varying page size 0000_0000_0000: 4KB 0000_0000_0011: 16KB 0000_0000_1111: 64KB 0000_0011_1111: 256KB 0000_1111_1111: 1MB 0011_1111_1111: 4MB 1111_1111_1111: 16MB
VPN2[31:13]	Virtual Page Number divided by 2. This field contains the upper bits of the virtual page number. Because it represents a pair of TLB pages, it is divided by 2. Bits 31:25 are always included in the TLB lookup comparison. Bits 24:13 are included depending on the page size, defined by PageMask.
G	Global Bit. When set, indicates that this entry is global to all processes and/or threads and thus disables inclusion of the ASID in the comparison.
ASID	Address Space Identifier. Identifies which process or thread this TLB entry is associated with.

Table 7-7 TLB Data Entry Fields

Field Name	Description
PFN0[31:12], PFN1[31:12]	Physical Frame Number. Defines the upper bits of the physical address. For page sizes larger than 4 KBytes, only a subset of these bits is actually used.
C0[2:0], C1[2:0]	Cacheability. Contains an encoded value of the cacheability attributes and determines whether the page should be placed in the cache or not. The field is encoded as following: 000: Cacheable, write through, no write-allocate. 001: Cacheable, write through, no write-allocate. 010: Uncacheable. 011: Cacheable, write-back, write-allocate 100: Cacheable, write-back, write-allocate 101: Cacheable, write-back, write-allocate 110: Cacheable, write-back, write-allocate 111: Uncacheable.
D0, D1	“Dirty” or Write-enable Bit. Indicates that the page has been written, and/or is writable. If this bit is set, stores to the page are permitted. If the bit is cleared, stores to the page cause a TLB Modified exception.
V0, V1	Valid Bit. Indicates that the TLB entry and, thus, the virtual page mapping are valid. If this bit is set, accesses to the page are permitted. If the bit is cleared, accesses to the page cause a TLB Invalid exception.

In order to fill an entry in the JTLB, software executes a TLBWI or TLBWR instruction. Prior to invoking one of these instructions, several CP0 registers must be updated with the information to be written to a TLB entry.

- PageMask is set in the CP0 *PageMask* register.
- VPN2 and ASID are set in the CP0 *EntryHi* register.
- PFN0, C0, D0, V0 and G bit are set in the CP0 *EntryLo0* register.
- PFN1, C1, D1, V1 and G bit are set in the CP0 *EntryLo1* register.

Note that the global bit “G” is part of both *EntryLo0* and *EntryLo1*. The resulting “G” bit in the JTLB entry is the logical AND between the two fields in *EntryLo0* and *EntryLo1*.

The address space identifier (ASID) helps to reduce the frequency of TLB flushing on a context switch. The existence of the ASID allows multiple processes to exist in both the TLB and instruction caches. The ASID value is stored in the *EntryHi* register and is compared to the ASID value of each entry.

6.3.2 Instruction TLB

The ITLB is a small 4-entry, fully associative TLB dedicated to performing translations for the instruction stream. The ITLB only maps 4-Kbyte pages/sub-pages.

The ITLB is managed by hardware and is transparent to software. If a fetch address cannot be translated by the ITLB, the JTLB is accessed to attempt to translate it in the following clock cycle. If successful, the translation information is copied into the ITLB. The ITLB is then re-accessed and the address will be successfully translated. It results in an ITLB miss penalty of at least 3 cycles (if the JTLB is busy with other operations, it may take additional cycles).

6.3.3 Data TLB

The DTLB is a small 4-entry, fully associative TLB, which provides a faster translation for Load/Store addresses than is possible with the JTLB. The DTLB only maps 4-Kbyte pages/sub-pages.

Like the ITLB, the DTLB is managed by hardware and is transparent to software. Unlike the ITLB, when translating Load/Store addresses, the JTLB is accessed in parallel with the DTLB. If there is a DTLB miss and a JTLB hit, the DTLB can be reloaded that cycle. The DTLB is then re-accessed and the translation will be successful. This parallel access reduces the DTLB miss penalty to 1 cycle.

6.4 Virtual to Physical Address Translation

During virtual-to-physical address translation, XBurst CPU core compares ASID and, depending on pages size, the highest 8-to-20 bits (VPN) of the virtual address to the contents of the TLB. The following figure illustrates the TLB address translation process.

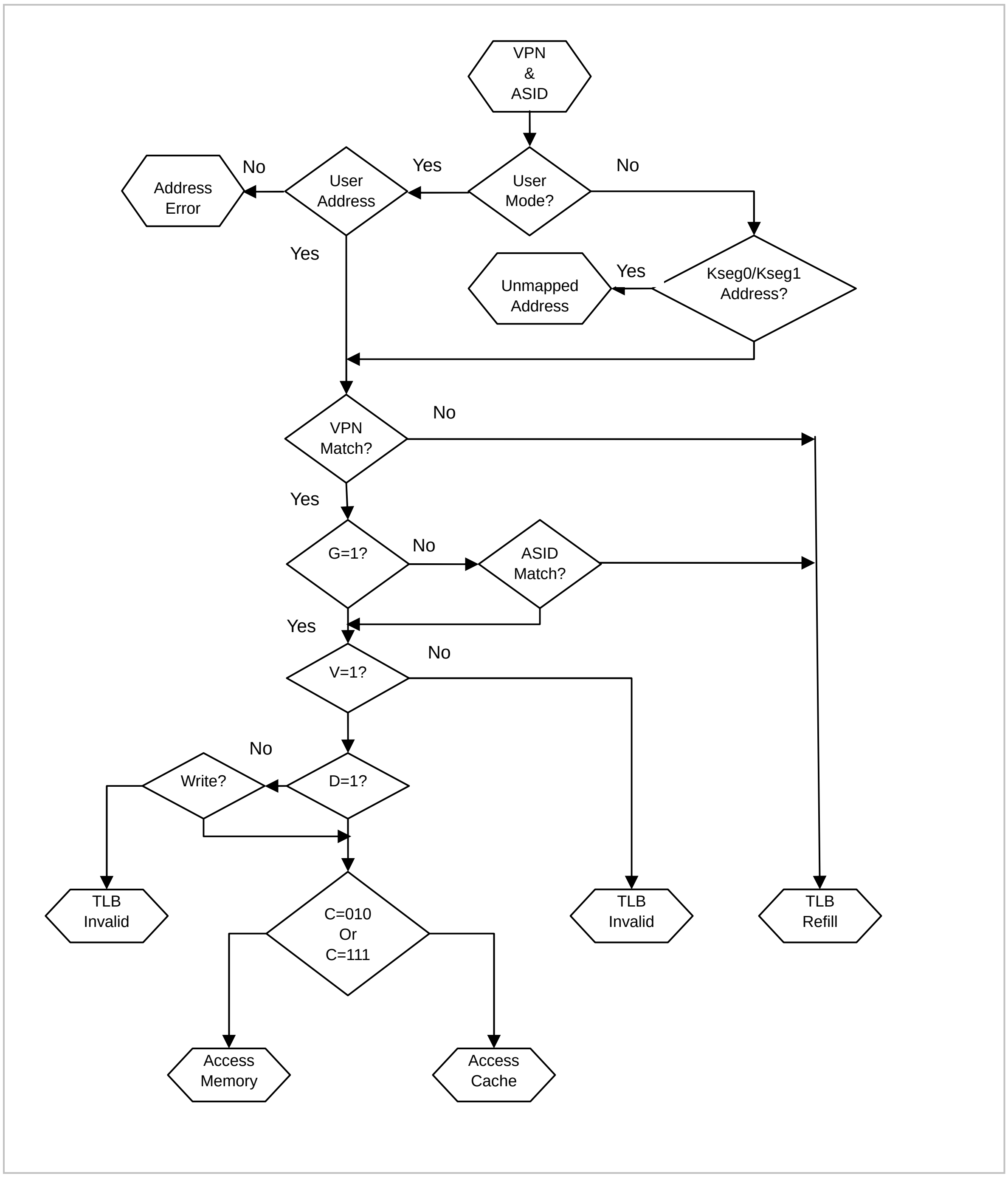


Figure 6-2 XBurst TLB Address Translation Flow

A virtual address matches a TLB entry when the VPN field of the virtual address equals the VPN field of the entry, and either G bit of the TLB entry is set or ASID field if the virtual address (as held in the *EntryHi* register) matches the ASID field of the TLB entry.

6.5 MMU Exceptions

6.6 MMU CP0 Registers

CP0 contains a group of registers dedicated for MMU manipulation, configuration or TLB related exceptions. Table 6-1 lists these registers. See Section CP0 for details of these registers.

Table 6-1 CP0 Registers for MMU

Register Number	Register Name	Function
0	Index	Index into the TLB array.
1	Random	Randomly generated index into the TLB array.
2	EntryLo0	Low-order portion of the TLB entry for even-numbered virtual pages.
3	EntryLo1	Low-order portion of the TLB entry for odd-numbered virtual pages.
4	Context	Pointer to page table entry in memory.
5	PageMask	Controls the variable page sizes in TLB entries.
6	Wired	Controls the number of fixed (“wired”) TLB entries.
8	BadVaddr	Reports the address for the most recent address related.
10	EntryHi	High-order portion of the TLB entry.

6.7 TLB Instructions

This section describes the instructions defined for manipulating TLB contents.

Op Code	Description
TLBP	TLB Probe
TLBR	TLB Read
TLBWI	TLB Write Index
TLBWR	TLB Write Random

6.7.1 TLBP

To find a matching entry in the TLB. The *Index* register is loaded with the address of the TLB entry whose contents match the contents of the *EntryHi* register. If no TLB entry matches, the high-order bit of the *Index* register is set.

6.7.2 TLBR

To read an entry from the TLB. The *EntryHi*, *EntryLo0*, *EntryLo1*, and *PageMask* registers are loaded with the contents of the TLB entry pointed to by the Index register.

6.7.3 TLBWI

To write a TLB entry indexed by the *Index* register. The TLB entry pointed to by the Index register is written from the contents of the *EntryHi*, *EntryLo0*, *EntryLo1*, and *PageMask* registers.

6.7.4 TLBWR

To write a TLB entry indexed by the *Random* register. The TLB entry pointed to by the *Random* register is written from the contents of the *EntryHi*, *EntryLo0*, *EntryLo1*, and *PageMask* registers.

7 Cache

7.1 Overview

XBurst processor core has separate instruction cache (I-Cache) and data cache (D-Cache) which allows instruction and data references to proceed simultaneously. Key features of caches are as following:

Table 7-1 Cache Features

Parameter	I-Cache	D-Cache
Size	16K Bytes	16k Bytes
Line Size	32 Byte	32 Byte
Numbe of Sets	128	128
Associativity	4 way	4 way
Lookup policy	Virtually indexed Physically tagged	Virtually indexed Physically tagged
Replace policy	LRU	LRU
Lock	N/A	N/A
Others	-	16-word deep write buffer

7.2 Cache Coherency Attribute

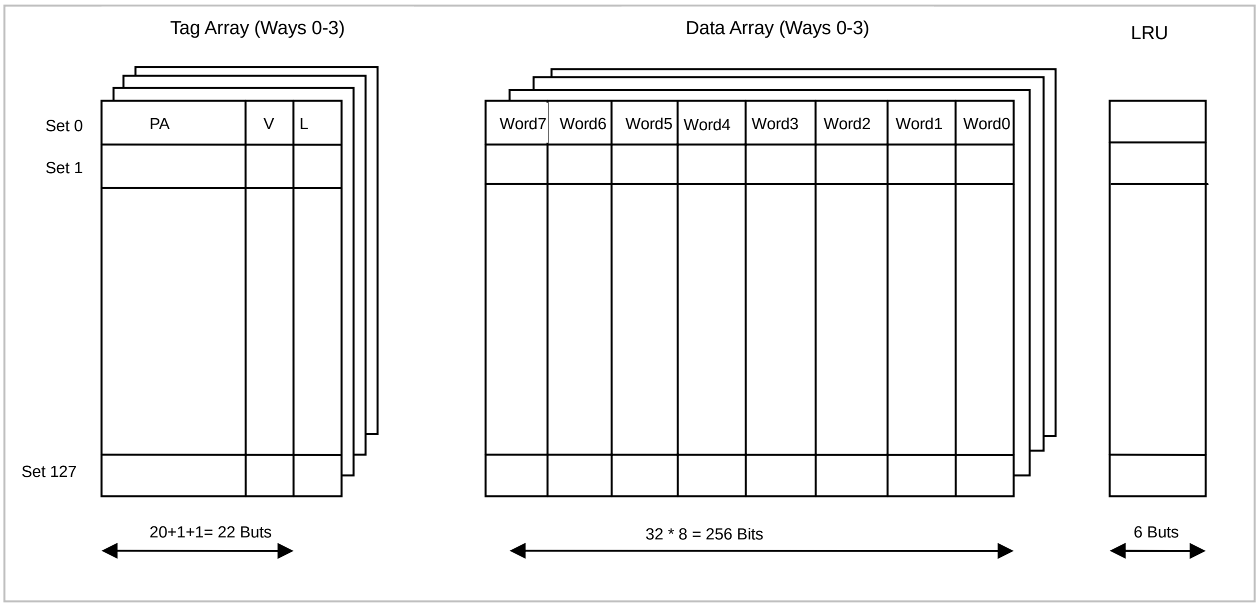
Cache coherency attribute is specified by the C[2:0] field in EntryLo0 and EntryLo1 entry of the TLB table for mapped address regions useg/kuseg,kseg2 and kseg3. For unmapped segment kseg0, Config.K0[2:0] field specifies the cache attribute. Unmapped segment kseg1 is not cacheable. The cache attribute is defined as following:

Table 1-2 Cache Coherency Attributes

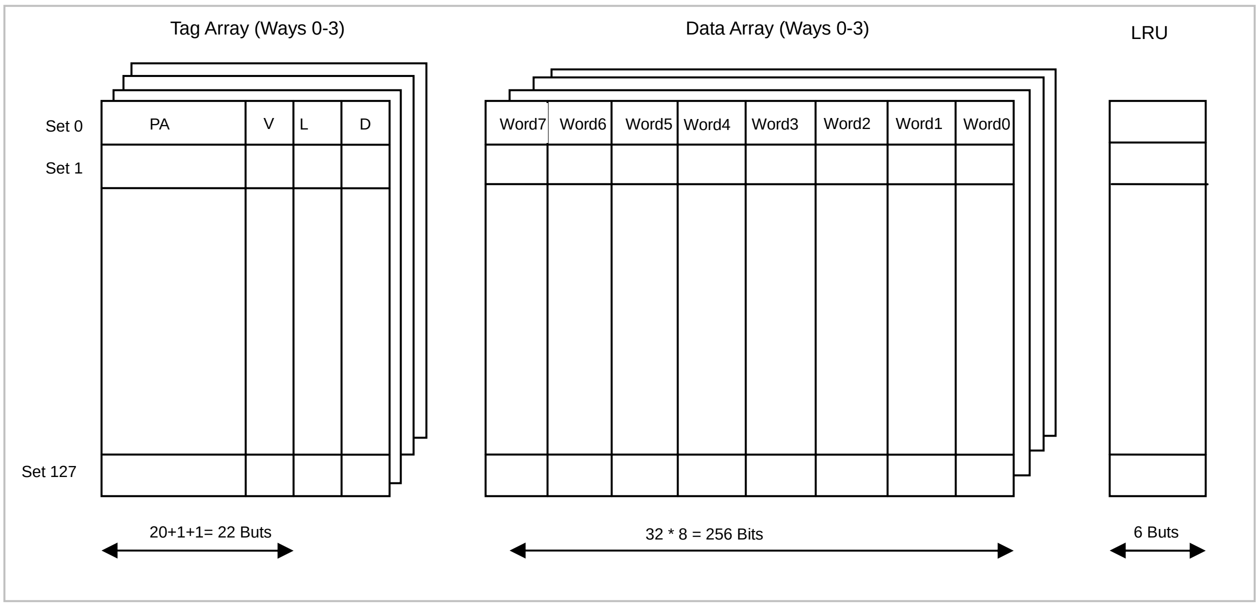
CCA	Encoding	**Description**
0	000	Cacheable, write-through, no write-allocate.
1	001	Cacheable, write-through, no write-allocate.
2	010	Uncacheable.
3	011	Cacheable, Write-back, write-allocate.
4	100	Cacheable, Write-back, write-allocate.
5	101	Cacheable, Write-back, write-allocate.
6	110	Cacheable, Write-back, write-allocate.
7	111	Uncacheable.

7.3 Cache Structure

Both I-Cache and D-Cache uses a 4-way set associative system. It is composed of four ways (banks), each of which is divided into a tag section and a data section. Each of the tag and data sections is divided into 128 entries. The data section of the entry is called a line. Each line consists of 32 bytes (4 bytes ∗8). The capacity per way is 4 k bytes (32 bytes ∗ 128 entries), with a total of 16 k bytes in the cache as a whole (4 ways).



I-Cache Structure



D-Cache Structure

As shown in the above figures, the structure of I-Cache and D-Cache is similar, with the exception that the Tag entry of D-Cache contains a 2 bits dirty field D.

TAG Array: The address tag PA holds the physical address used in the external memory access. It is composed of 22 bits (address bits 31–10) used for comparison during cache searches. The V bit indicates whether the entry data is valid. When the V bit is 1, data is valid; when 0, data is not valid. L bit indicates whether the entry is locked (L bits is not implemented in this version). D[1:0] is only implemented in D-Cache. D[0] indicates whether the lower half of the cache line (word0-word3) is dirty. D[1] indicates whether the higher half of the cache line (word4-word7) is dirty.

Data Array: Each entry contains 256 bits for 8 instructions. Entries are registered in the cache in line units (32 bytes). The data array is not initialized by a power-on or manual reset.

LRU: With the 4-way set associative system, up to four instructions or data with the same entry address (address bits 11–5) can be registered in the cache. When an entry is registered, the LRU shows which of the four ways it is recorded in. There are six LRU bits, controlled by hardware. A least-recently-used (LRU) algorithm is used to select the way.

The six LRU bits indicate the way to be replaced as shown in (table 5.2). If a bit pattern other than those listed in table 5.2 is set in the LRU bits by software, the cache will not function correctly. When modifying the LRU bits by software, set one of the patterns listed in table 5.2.

7.4 Replacement

7.5 Write Buffer

7.6 Cache Registers

Table 6-1 CP0 Registers

Register Number	Register Name	Function
17	LLAddr	Load linked address
26	ErrCtl	Controls access to data and SPRAM arrays for CACHE instruction
28	TagLo/ DataLo	Low-order portion of cache tag interface

7.7 Cache Instructions

This section describes the instructions defined for manipulating Cache contents.

7.7.1 CACHE instruction

CACHE instruction format is shown below:

31 26	25 21	20 16	15 0
CACHE 101111	Base	op	offset

The 16-bit offset is sign-extended and added to the contents of the base register to form a virtual address. The virtual address is used in one of the following ways based on the operation to be performed and the type of cache as described in the following table.

- The virtual address is translated by MMU to a physical address. The physical address is then used to address the cache
- The effective address is used to index the sets and ways of the cache, as shown below

31 14	13 12	11 5	4 0
unused	Way index	Set Index	Byte offset

op[17:16] of the Cache instruction specifies the cache on which to perform the operation.

00 – I-Cache
01 – D-Cache
10 – Reserved.
11 – Reserved

Table 7-2 list all the functions CACHE instruction can perform on I-cache.

Table 7-2 I-Cache Operations (op[17:16] = 2'b00)

Op[20:18]	Operation	Function description
3'b 000	Index Invalidate	Invalidate a I-cache specified by the virtual address. Virtual address is used to directly index the specified way in specified set.Software. This function can be used by software to invalidate the entire instruction cache by stepping through all valid indices.
3'b001	Index Load Tag	Read the tag for the cache block at the specified index into the *TagLo* register. Also read the word corresponding to the word offset(ignore least significant two bits of the address) into the *DataLo* register.
3'b010	Index Store Tag	Write the tag for the cache block at the specified index from the *TagLo* register.
3'b011	Index Store data	Write the DataLo contents to the way and word index as specified. Note that: This op is valid only when Errctl[WST] is 1.
3'b100	Hit Invalidate	If the virtual address hits I-cache, the hit line is invalidated; otherwise, nothing is done.
3'b101	Refill a line	Fetch the cache line containing the virtual address from memory and fill it into the I-cache. Note : The cache line is re-fetched even if it is already in the cache.
3'b110	Reserved	
3'b111	Prefecth and lock	If the virtual address misses cache, the line containing the address is fetched from memory.

Table 7-2 list all the functions CACHE instruction can perform on D-cache.

Table 7-5　D-Cache Operations (op[17:16] = 2’b01)

Op[20:18]	Operation	Function description
3’b 000	Index write back Invalidate	Invalidate a D-cache specified by the virtual address. Virtual address is used to directly index the specified way in specified set. If the cache line is dirty, write back the dirty data and set it invalid. This function can be used by software to invalidate the entire D-cache by stepping through all valid indices.
3’b001	Index Load Tag	Read the tag for the cache block at the specified index into the *TagLo* register. Also read the word corresponding to the word index into the *DataLo* register (ignore VA[1:0])
3’b010	Index Store Tag	Write the tag for the cache block at the specified index from the *TagLo* register. This encoding may be used by software to initialize the entire D-cache by stepping through all valid indices. Doing so requires that the *TagLo* and *TagHi* registers associated with the cache be initialized first.
3’b011	Index Store data	Write the DataLo contents to the way and word address as specified Note that: This op is valid only when Errctl[WST] is 1.
3’b100	Hit Invalidate	If the virtual address hits D-cache, the hit line is invalidated; otherwise, nothing is done.
3’b101	Hit write back Invalidate	If the virtual address hits D-cache, invalidate the hit cache line. If the cache line is dirty, write back the dirty data and set it invalid.
3’b110	Hit write back	If D-cache hit and it is dirty, write back dirty data and leave it still valid, but clear the dirty bits. Otherwise, treat as NOP.
3’b111	Prefecth and lock	If the virtual address misses cache, the line containing the address is fetched from memory,　and if the line to be replaced is dirty, write back the dirty data.

Note:

- For index operation, software should use unmapped address to avoid TLB exceptions, nor data watch exceptions.
- The operation of this instruction is UNDEFINED for any operation/cache combination that is not implemented.
- The operation of this instruction is UNDEFINED if the operation requires an address, and that address is uncacheable.

7.7.2 PREF instruction

PREF instruction format is shown below:

31 … 26	25 … 21	20 … 16	15 … 0
PREF 110011	Base	hint	offset

The 16-bit offset is sign-extended and added to the contents of the base register to form a virtual address.

PREF does not cause addressing-related exceptions. If the address specified would cause an addressing exception, the exception condition is ignored and no data movement occurs. PREF never generates a memory operation for a location with an uncached memory access type.

The hint field supplies information about the way that the data is expected to be used. PREF is an advisory instruction that may change the performance of the program. However, for all hint values and all virtual addresses, it neither changes the architecturally visible state nor does it alter the meaning of the program.

Any of the following conditions causes the processor core to treat a PREF instruction as a NOP.

- A reserved hint value is used
- Writeback-invalidate (25) hint value is used
- The address has a translation error
- The address maps to an uncacheable page
- The data is already in the cache
- There is already another load/prefetch outstanding

PREF is a non-blocking operation and does not cause the pipeline to stall while waiting for the data to be returned.

Table 1-6 Values of the *hint* Field for the PREF Instruction

Hint	Action	Description
0,1, 6, 7	Pefetch	Prefetched data in the same way as cacheable load.miss
30	Allocate	Allocate a cache line, without the overhead involved in filling the data from memory.
Others	NOP	Nothing to be done, treat as NOP

7.7.3 SYNC instruction

31 — 26	25 — 11	10 — 6	5 — 0
SPECIAL 000000	0	stype	SYNC 001111

SYNC is used to synchronize shared memory. SYNC affects only uncached and cached coherent loads and stores. The loads and stores that occur before the SYNC must be completed before the loads and stores after the SYNC are allowed to start.

Executing the SYNC instruction causes the write buffer to be flushed. The SYNC instruction stalls until all loads and stores are completed.

8 JTAG Debug

ACC mode is used to accelerate processor access to dmseg. In the mode, another access protocol is adopted instead of standard EJTAG one to realize fast dmseg access.

8.1.1 ACC Mode Flag

AM: 1 – ACC mode; 0 – MIPS mode. Reset value is 0. It is invisible for EJTAG probe and software. The bit is set by expanded instruction EJTAGBOOTA, cleared by NORMALBOOT or TAP reset.

8.1.2 EJTAG Control Register in ACC mode (ECR_A)

It is connected between TDI and TDO by instruction CONTROL in ACC mode. Probe polls this 2-bit register to service the processor access to EJTAG memory.

1	0
PRW	PA

Field	BITS	Description	Read/write	Reset value
PA	0	Processor Access (PA) 0: No pending processor access 1: Pending processor access	R	0
PRW	1	Processor access is read or write 0: read access 1: write access	R	Undefined

8.1.3 Processor Access Address Register in ACC mode (ADDRESS_A)

It is connected between TDI and TDO by instruction ADDRESS or ALL in ACC mode. This register is used to provide the address of the processor access to EJTAG non-drseg region.

36 35	34	33 32	31 0
BST	R	SZ	PAA

Field	BITS	Description	R/W	Reset value
PAA	31: 0	Processor Access address	R	Undefined
SZ	33:32	Processor Access size 00: byte 01: half word 10: word 11: reserved	R	Undefined
R	34	Reserved		
BST	36:35	Processor Access burst pattern 00: single 01: 4-beat wrapping burst 10: 8-beat wrapping burst 11: reserved Note: 4-beat wrapping burst will never occur in this implementation. And 8-beat wrapping burst may occur only for burst read access.	R	Undefined

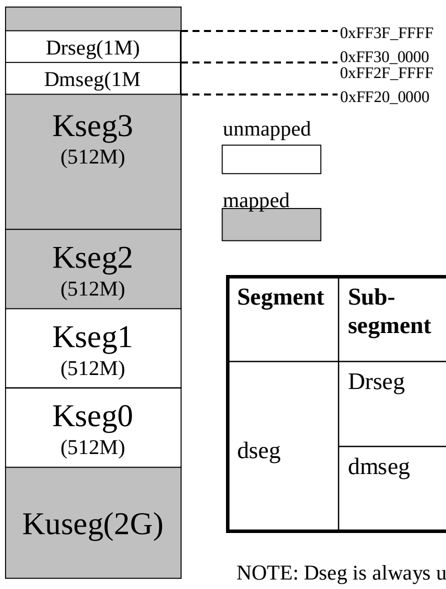
8.1.4 Processor Access Data Register in ACC mode (DATA_A)

It is connected between TDI and TDO by instruction DATA or ALL in ACC mode. This register is similar to PAD in normal MIPS mode, except it has one more RDY bit.

32	31 … 0
RDY	PAD

Field	BITS	Description	R/W	Reset value
PAD	31:0	Processor Access data The register has the written value for a processor access write to the dmseg. And it is also used to provide the data value for load data or fetch instruction from the dmseg.	R/W	Undefined
RDY	1	Pipeline lock label. 1- processor can proceed due to processor access to dmseg done 0- processor should be locked due to unfinished processor access to dmseg	W	Undefined

8.1.5 Address space in Debug mode (AM = 0)



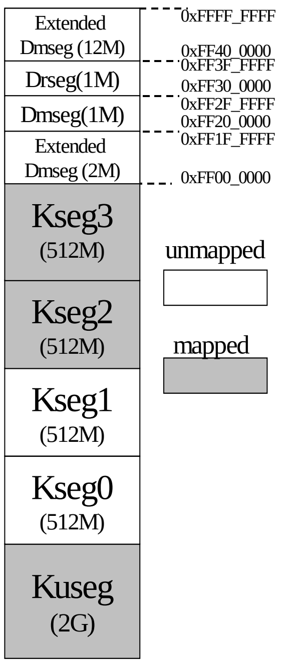
Dseg Address Space and Cache Attributes

Segment	Sub-segment	Virtual address (unmapped *)	Physical address	Cache attribute
dseg	Drseg	0xFF30_0000 – 0xFF3F_FFFF	0xFF30_0000-0xFF3F_FFFF	2
	dmseg	0xFF20_0000 – 0xFF2F_FFFF	0xFF20_0000-0xFF2F_FFFF	2

NOTE: Dseg is always unmapped in debug mode despite of attribute of Kseg3 .

Dseg space in MIPS mode

8.1.6 Address space in Debug mode (AM = 1)



Dseg Address Space and Cache Attributes

Segment	**Sub segment**	**Virtual address**	**Physical address**	**Cache attribute**
Dseg	extended Dmseg (2M)	0xFF00_0000-0xFF1F_FFFF	0xFF00_0000-0xFF1F_FFFF	0
	Dmseg (1M)	**0xFF20_0000-0xFF2F_FFFF**	**0xFF20_0000-0xFF2F_FFFF**	**2**
	Drseg (1M)	**0xFF30_0000-0xFF3F_FFFF**	**0xFF30_0000-0xFF3F_FFFF**	**2**
	extended Dmseg (12M)	0xFF40_0000-0xFFFF_FFFF	0xFF40_0000-0xFFFF_FFFF	0

NOTE: Dseg is always unmapped in debug mode despite of attribute of Kseg3.

Extended Dseg space in ACC mode

8.1.7 Supported EJTAG Instructions

Value	**Instruction**	**Function**
0x01	IDCODE	Select Chip Identification data register
0x03	IMPCODE	Select implementation register
0x08	ADDRESS	ADDRESS register is selected in MIPS mode while ADDRESS_A register is selected in ACC mode.
0x09	DATA	DATA register is selected in MIPS mode while DATA_A register is selected in ACC mode.
0x0A	CONTROL	ECR register is selected in MIPS mode while ECR_A register is selected in ACC mode
0x0B	ALL	In MIPS mode, Selects the ADDRESS, DATA and ECR register. The scan sequence is TDI-> ADDRESS-> DATA->ECR->TDO. In ACC mode, Selects the DATA_A, ADDRESS_A, and ECR_A register. The scan sequence is TDI-> DATA_A ->ADDRESS_A ->ECR_A->TDO.
0x0C	EJTAGBOOT	Boot from probe host in MIPS mode by setting ECR.Ejtagbrk, ECR.ProbEn and ECR.ProbTrap when reset. Bypass register is selected.
0x0D	NORMALBOOT	Boot in normal way by clearing Ejtagbrk, ProbEn and ProbTrap when reset. Bypass register is selected.
0x1C	EJTAGBOOTA	Boot from probe host in ACC mode by setting ECR.Ejtagbrk, ECR.ProbEn, ECR.ProbTrap and AM when reset. Bypass register is selected.
0x1F	BYPASS	Select Bypass register

8.1.8 Fetch/Load and Store From/to the EJTAG Probe through dmseg in MIPS mode

Fetch/load from EJTAG memory

1. The internal hardware latches the requested address into the Address register (ADDRESS)
2. The internal hardware sets the following bits in the EJTAG Control register:
   PrAcc = 1;
   PRnW = 0;
   Psz = Value depending on the transfer size
3. The EJTAG Probe selects the EJTAG Control register, shifts out its content and tests the PrAcc bit: when the PrAcc bit is found 1, it means that pending processor access need be serviced.
4. The EJTAG Probe checks the PRnW bit to determine the required access.
5. The EJTAG Probe selects the ADDRESS register and shifts out its content.
6. The EJTAG Probe selects the DATA register and shifts in the required instruction/data.
7. The EJTAG Probe selects the EJTAG Control register again, and shifts a PrAcc = 0 bit into this register to indicate that the instruction/data is available and the processor can proceed.

Store to EJTAG memory

1. The internal hardware latches the requested address into the Address register (ADDRESS)
2. The internal hardware sets the following bits in the EJTAG Control register:
   PrAcc = 1;
   PRnW = 1;
   Psz = Value depending on the transfer size
3. The EJTAG Probe selects the EJTAG Control register, shifts out its content and tests the PrAcc bit: when the PrAcc bit is found 1, it means that pending processor access need be serviced.
4. The EJTAG Probe checks the PRnW bit to determine the required access.
5. The EJTAG Probe selects the ADDRESS register and shifts out its content.
6. The EJTAG Probe selects the DATA register and shifts out its content to location determined by ADDRESS.
7. The EJTAG Probe selects the EJTAG Control register again and shifts a PrAcc = 0 bit into this register to indicate that the write has been done and the processor can proceed.