VEGA ET1035 Datasheet

Centre for Development of Advanced Computing (C-DAC)
A Scientific Society of Ministry of Electronics & IT,

Government of India
Vellayambalam, Thiruvananthapuram
Kerala - 695033
Phone: 0471- 2725897, 2723333 (Ext 347),
www.vegaprocessors.in, vega@cdac.in

Doc. Name	**VEGA ET1035 Datasheet**		
Project	**Digital India RISC-V (DIR-V) Program**		
Doc. No.	DS HD 083 0004	Group/ Section	HDG
Control Status*	Controlled	Copy No.**	01

Functional designation of approving authority	**Group Head**	Functional designation of issuing authority	**Project Leader**
For any clarifications/corrections/amendments in this document please contact issuing authority			

Issue History

Issue No.	Effective From(Date)	Reason for Re-issue	Total Doc. Pages	Date of consolidation of review outputs
1	11/ 09/2024	NA	30	11/ 09/2024

Approved By:__GH____________
(Functional designation)   Date of approval: 11/ 09/2024

Contents

1 Introduction

VEGA ET1035 is a 32-bit CPU IP core based on RISC-V Instruction Set Architecture. Primarily tailored for low-power embedded applications, the core has a highly optimized 3-stage in-order pipeline with machine mode support. The pipeline is configurable and accommodates the RISC-V RV32 IM extensions.

1.1 Name of the product

32-bit VEGA Processor IP core

1.2 Product Mnemonic

VEGA ET1035

1.3 Definitions

RISC-V - RISC-V is an open standard instruction set architecture (ISA) based on established Reduced Instruction Set Computer (RISC) principles. Unlike most other ISA designs, the RISC-V ISA is provided under open-source licenses that do not require fees to use.

1.4 Acronyms & Abbreviations

PC	-	Program Counter
SoC	-	System on Chip
WFI	-	Wait For Interrupt

1.5 References

1. The RISC-V Instruction Set Manual Volume I: User-Level ISA
2. The RISC-V Instruction Set Manual Volume II: Privileged Architecture

2 Processor Pipeline

ET1035 supports a configurable set of RISC-V RV32 IM extensions. Among these, RV32-I is the default setting and all other extensions can be configured according to user needs.

An overall block diagram of the processor is shown below.

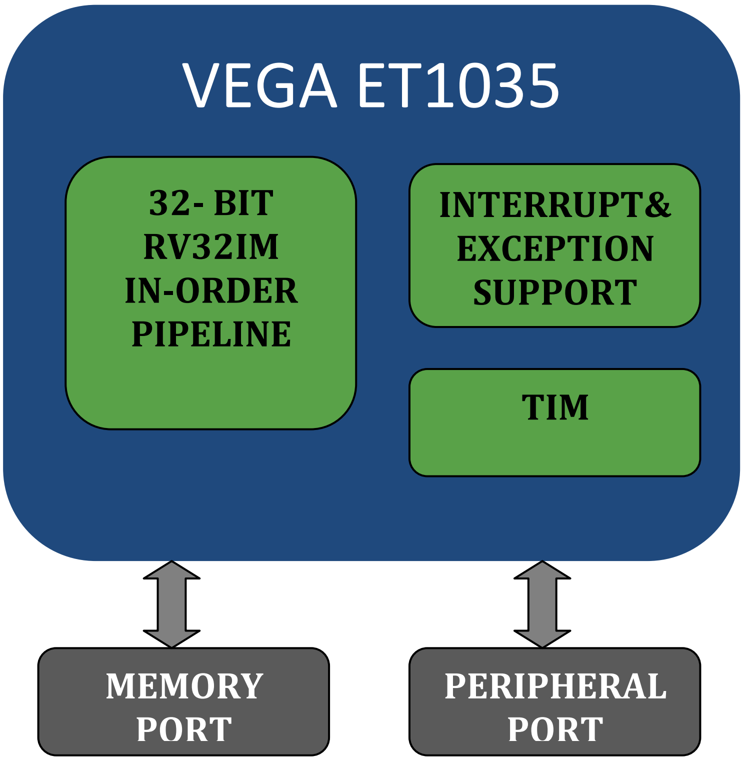
VEGA ET1035
32- BIT RV32IM IN-ORDER PIPELINE
INTERRUPT& EXCEPTION SUPPORT
TIM
MEMORY PORT
PERIPHERAL PORT

Figure 1 : ET1035 BLOCK DIAGRAM

2.1 Features

- The 32-bit RISC-V(RV32IM)Instruction Set Architecture
  - User level ISA Version 2.2
  - Privileged Architecture Version 1.10
- 3 stage In-Order pipeline
- Support for 55 instructions
- Support for privileged Instruction set
- 32-bit load/store architecture
- Pipelined Harvard architecture
- Byte, half-word and word memory access
- One cycle bubble for Jump and Branch Instructions
- Software Interrupt support
- Timer Interrupt support
- External Interrupt support
- Vectored Interrupts
- Processor Modes– Machine
- VHDL Implementation

2.2 Processor Internals

The ET1035 pipeline line is divided into three stages. The pipeline has a peak execution rate of one instruction per clock cycle and is fully bypassed so that most instructions have single cycle latency.

2.2.1 Fetch Stage & Instruction Memory System

In the Instruction Fetch stage instructions are fetched at every cycle from the instruction memory whose address is pointed by the program counter (PC). The address is presented to instruction memory at the start of a cycle. Then during the cycle, the instruction is read out of instruction memory, and at the same time, a calculation is done to determine the next PC. The next PC is calculated by incrementing the PC by 4, and by choosing whether to take that as the next PC or to take the result of a branch/jump calculation as the next PC. The instruction fetch unit is capable of supplying one instruction to the decode stage per cycle. The instruction address must be word-aligned.

2.2.2 Decode & Execute Stage

The Instruction Decode unit decodes the instruction and ensures the operands for the execution units are available. This stage is where the control unit determines what values the control lines must be set to depending on the instruction. During the decode stage, the registers are read from the register file and the opcode is passed to the control unit which asserts the required control signals. Sign extension is also done for the calculation of effective address. It also checks for illegal opcodes. It generates a packet for the execution of instruction.

The execute stage performs the required operation on the data provided by the decode stage. In the execute stage, for Register type instructions, the ALU operations are performed according to the ALU operation control signals and for load and store instructions, effective address calculation is done. The Multiplier unit calculates signed/unsigned multiplications. The Divider unit calculates signed/unsigned division and remainder. The Load-Store Unit accesses the data memory. The Branch Unit calculates jump and branch addresses. Only one operation can be executed per clock cycle. Most operations complete in one clock cycle.

2.2.3 Memory Access & Write back stage

The Memory Access stage waits for a memory read access to complete. Address, data, and control signals to access the memory are calculated during the Execute stage. For instructions that involve memory operations (e.g., load and store instructions), this stage performs the memory read or write. This is where the actual data is read from or written to memory.

In Write back stage the result from the Execute stage or the data read from memory is written back to the register file or other destination as specified by the instruction. Write back stage checks for interrupt/exception/WFI after receiving a packet from the Memory stage. If it is present, this stage handles that interrupt/exception/WFI and a redirection PC will be sent to the fetch stage.

2.3 Pipeline

The figure below shows the pipeline diagram of ET1035. The different pipeline stages along with their pipeline registers, the feedback and control paths are depicted in the diagram.

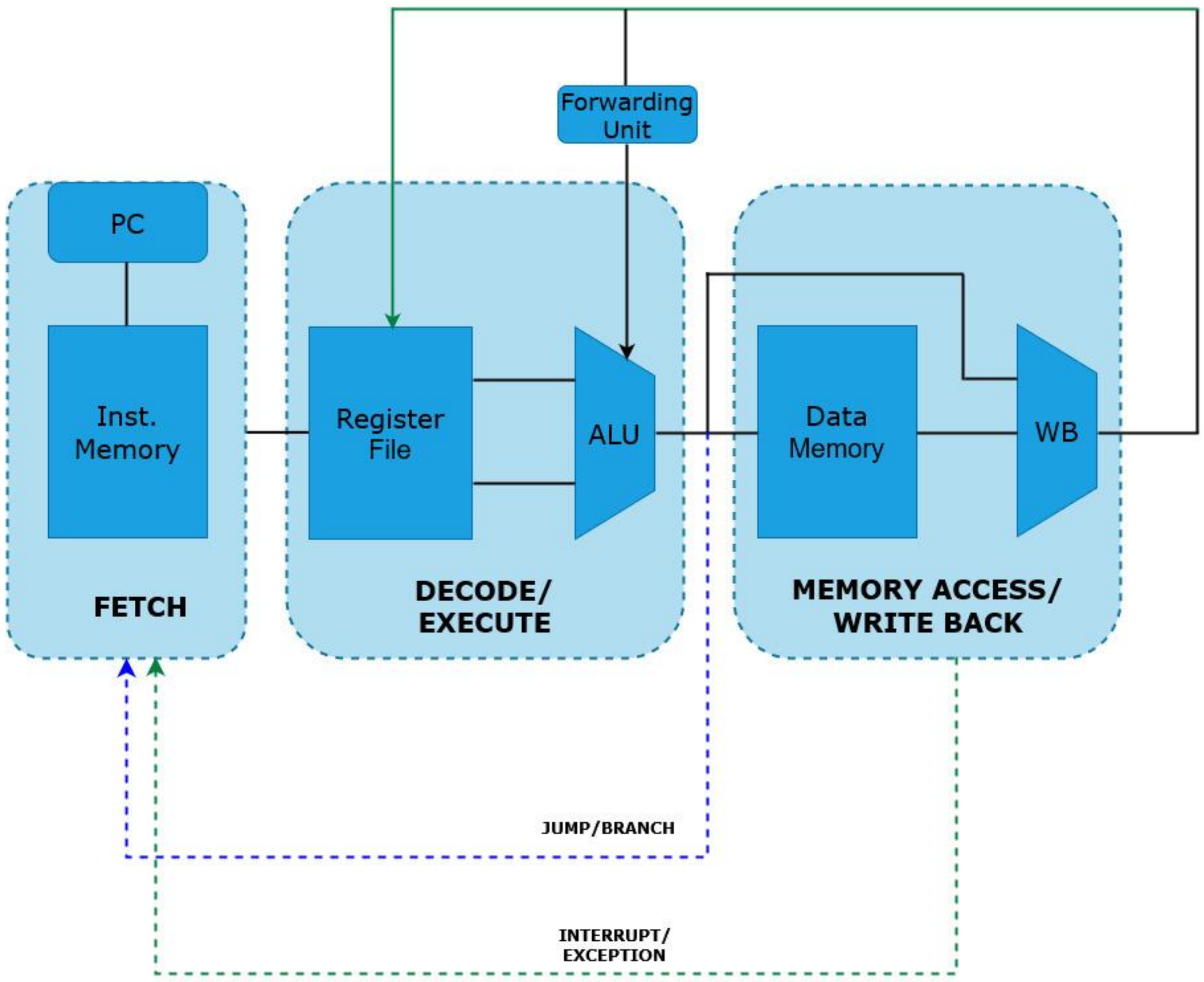

Figure 2 : ET1035 Pipeline Diagram

2.3.1 Interrupt detection logic

Interrupts can only be enabled/ disabled on a global basis and not individually. There is an event/interrupt controller outside of the core that performs masking and buffering of the interrupt lines. The global interrupt enable is done via the CSR register MSTATUS.

Interrupt is detected based on the control and status registers. Interrupt is processed only if interrupt is enabled from the memory access and write back stage. Interrupt enable from memory access and write back stage indicates that there is no memory instruction initiated from those pipeline stages. This is done to implement the precise interrupt processing.

2.3.2 Interrupt processing logic

Interrupt detected will be processed in Machine mode based on the control information from CSR registers. Interrupt processing logic redirect fetch pipeline to fetch instruction from the interrupt handler. CSR registers are updated as part of the interrupt processing. All pipeline stages are flushed after interrupt processing.

2.3.3 Exception in non-debug mode

Exception is processed in machine mode as done for interrupt processing and is controlled by CSR registers. Exception processing logic redirect fetch pipeline to fetch instruction from the exception handler. CSR registers are updated as part of the exception processing. All pipeline stages are flushed after exception processing. Privileged instructions like MRET (Machine Return), and ECALL (Environment call) are executed as part of exception processing logic.

3 Instruction Set Supported by ET1035

The instructions supported by ET1035 and their basic description are given below.

Serial No.	Instruction	Description
RV32 I BASE INSTRUCTION SET		
1	LUI	Used to build 32-bit constants
2	AUIPC	Used to build pc-relative addresses
Control Transfer Instructions		
3	JAL	Plain unconditional jumps
4	JALR	Indirect jump instruction (jump and link register)
5	BEQ	Takes branch if register contents are equal
6	BNE	Takes branch if register contents are not equal
7	BLT	Take the branch if rs1 is less than rs2, using signed comparison
8	BGE	Take the branch if rs1 is greater than rs2, using signed comparison
9	BLTU	Take the branch if rs1 is less than rs2, using unsigned comparison
10	BGEU	Take the branch if rs1 is greater than rs2, using unsigned comparison
Memory Instructions		
11	LB	Loads a 8-bit value from memory (signed)
12	LH	Loads a 16-bit value from memory(signed)
13	LW	Loads a 32-bit value from memory(signed)
14	LBU	Loads a 8-bit value from memory (unsigned)
15	LHU	Loads a 16-bit value from memory(unsigned)

16	SB	Stores 8-bit value to memory
17	SH	Stores 16-bit value to memory
18	SW	Stores 32-bit value to memory
Integer Register-Immediate Instructions		
19	ADDI	Addition of immediate and register content
20	SLTI	Set less than immediate using signed operands
21	SLTIU	Set less than immediate using unsigned operands
22	XORI	Bitwise XOR operation
23	ORI	Bitwise OR operation
24	ANDI	Bitwise AND operation
25	SLLI	Logical left shift
26	SRLI	Logical right shift
27	SRAI	Arithmetic right shift
Integer Register-Register Instructions		
28	ADD	Addition of two register contents
29	SUB	Subtraction of two register contents
30	SLL	Logical left shift
31	SLT	Signed comparison of two register contents
32	SLTU	Unsigned comparison of two register contents
33	XOR	Bitwise XOR operation
34	SRL	Logical right shift
35	SRA	Arithmetic right shift
36	OR	Bitwise OR operation
37	AND	Bitwise AND operation

38	FENCE	Used to order device I/O and memory accesses
39	FENCE.I	Used to synchronize the instruction and data streams
40	ECALL	Used to make a request to the supporting execution environment
41	MRET	Used to return from traps in machine mode
CSR Instructions		
42	CSRRW	Swaps values in the CSRs and integer registers
43	CSRRS	Reads the value of the CSR, zero extends the value ,writes it to integer register
44	CSRRC	Reads the value of the CSR, zero extends the value, writes it to integer register
45	CSRRWI	Immediate variant of CSRRW
46	CSRRSI	Immediate variant of CSRRS
47	CSRRCI	Immediate variant of CSRRC
RV32M STANDARD EXTENSION		
63	MUL	Performs an XLEN-bit×XLEN-bit multiplication and product is lower XLEN bits
64	MULH	signed×signed multiplication and product is upper XLEN bits
65	MULHSU	signed×unsigned multiplication and product is upper XLEN bits
66	MULHU	unsigned×unsigned multiplication and product is upper XLEN bits
67	DIV	signed integer division of XLEN bits by XLEN bits and result is quotient

68	DIVU	unsigned integer division of XLEN bits by XLEN bits and result is quotient
69	REM	signed integer division of XLEN bits by XLEN bits and result is remainder
70	REMU	unsigned integer division of XLEN bits by XLEN bits and result is remainder

Table 1: Instruction Set

4 Register Map

This chapter provides the list of registers used in ET1035 as specified in RISC-V ISA. There are two sets of registers- an integer register file and a control & status register file.

4.1 Integer Registers

There are 32 integer registers specified by RISC-V ISA, each of width 32 bits. All registers have read and write permissions. There are 31 general-purpose registers x1-x31, which are designed to hold integer values. Register x0 is hardwired to the constant 0 and can be used as a source of constant zero or as a don't care destination register. Don't care destination x0 is used to ignore the result of instruction execution provided that destination register is mandatory for instruction structure.

The list of integer registers and their ABI name is specified in the table below.

Serial No:	Register Name	ABI Name	Description
1	X0	zero	Hard-wired zero
2	X1	ra	Return address
3	X2	sp	Stack pointer
4	X3	gp	Global pointer
5	X4	tp	Thread pointer
6	X5	t0	Temporary/alternate link register
7	X6	t1	Temporaries
8	X7	t2	Temporaries
9	X8	s0/fp	Saved register/frame pointer
10	X9	s1	Saved register
11	X10	a0	Function arguments/return values
12	X11	a1	Function arguments/return values
13	X12	a2	Function arguments
14	X13	a3	Function arguments

15	X14	a4	Function arguments
16	X15	a5	Function arguments
17	X16	a6	Function arguments
18	X17	a7	Function arguments
19	X18	s2	Saved registers
20	X19	s3	Saved registers
21	X20	s4	Saved registers
22	X21	s5	Saved registers
23	X22	s6	Saved registers
24	X23	s7	Saved registers
25	X24	s8	Saved registers
26	X25	s9	Saved registers
27	X26	s10	Saved registers
28	X27	s11	Saved registers
29	X28	t3	Temporaries
30	X29	t4	Temporaries
31	X30	t5	Temporaries
32	X31	t6	Temporaries

Table 2: Integer Registers

4.2 Control & Status Registers

There are different control & status registers as specified in RISC-V privilege specification according to processor modes. ET1035 supports machine mode of operation. The registers supported in the machine mode are listed below.

Serial No:	CSR Name	CSR Address	R/W Privilege	Description
MACHINE MODE REGISTERS				
1	MVENDORID	0xF11	MRO	Vendor ID
2	MARCHID	0xF12	MRO	Architecture ID
3	MIMPID	0xF13	MRO	Implementation ID
4	MHARTID	0xF14	MRO	Hardware thread ID
5	MSTATUS	0x300	MRW	Machine status register
6	MISA	0x301	MRW	ISA and extensions
7	MIE	0x304	MRW	Machine interrupt-enable register
8	MTVEC	0x305	MRW	Machine trap-handler base address.
9	MSCRATCH	0x340	MRW	Scratch register for machine trap handlers
10	MEPC	0x341	MRW	Machine exception program counter
11	MCAUSE	0x342	MRW	Machine trap cause
12	MTVAL	0x343	MRW	Machine bad address or instruction
13	MIP	0x344	MRW	Machine interrupt pending
14	MCYCLE	0xB00	MRW	Machine cycle counter
15	MINSTRET	0xB02	MRW	Machine instructions-retired counter

Table 3: Control & Status Registers

5 Exceptions & Interrupts

The ET1035 supports Machine mode of operation. Code run in machine-mode (M-mode) is usually inherently trusted, as it has low-level access to the machine implementation. M-mode can be used to manage secure execution environments on RISC-V.

5.1 Exceptions

The various exceptions taken by the processor are listed below. The illegal instruction exception and ECALL instruction exceptions cannot be disabled and are always active.

Sl No:	Description	Interrupt Code	Exception Code
1	Instruction access fault	0	1
2	Illegal instruction	0	2
3	Load address misaligned	0	4
4	Load access fault	0	5
5	Store address misaligned	0	6
6	Store access fault	0	7
7	Environment call from M-mode	0	11

Table 4: Supported Exceptions

When entering an interrupt/exception handler, the core sets MEPC to the current program counter and jumps to a new PC according to value in MTVEC register. Upon encountering MRET instruction, core jumps back to PC value in MEPC register.

5.2 Interrupts

The ET1035 also facilitates various interrupts based on the values in CSR registers MIP and MIE, along with the control information in MSTATUS. Interrupts can only be enabled/ disabled on a global basis and not individually. There is an event/interrupt controller outside of the core that performs masking and buffering of the interrupt lines. The global interrupt enable is done via the CSR register MSTATUS.

The various interrupts supported by ET1035 is listed below.

Sl No:	Description	Interrupt Code	Exception Code
1	Machine software interrupt	1	3
2	Machine timer interrupt	1	7
3	Machine external interrupt	1	11

Table 5: Supported Interrupts

5.3 CSRs related to TRAP handling

The control and status registers involved in interrupt/exception handling are explained below.

5.3.1 Machine Status Register (MSTATUS)

The MSTATUS register keeps track of and controls the hart’s current operating state. An interrupt-based view of MSTATUS is shown in the below figure.

31 -13	12-11	10-9	8	7	6	5	4	3	2	1	0
	-	-	-	**MPIE**	-	-	-	**MIE**	-	-	-

Figure 3: Bit mapping of MSTATUS register

Bit #	R/W	Name	Description
3	RW	MIE	Machine mode Interrupt Enable: Set this bit to enable the interrupt
7	RW	MPIE	Previous Interrupt Enable: When an exception is encountered, MPIE will be set to MIE. When the MRET instruction is executed, the value of MPIE will be restored to MIE.

The MSTATUS register contains the global interrupt enable bits for each mode. When interrupt is encountered, MPIE will be set to MIE. When the MRET instruction is executed, the value of MPIE will be restored to MIE.

5.3.2 Machine Trap-Vector Base-Address Register (MTVEC)

The MTVEC register is an XLEN-bit read/write register that holds trap vector configuration, consisting of a vector base address (BASE) and a vector mode (MODE).The value in the BASE field must always be aligned on a 4-byte boundary, and the MODE setting may impose additional alignment constraints on the value in the BASE field. When MODE=Direct, all traps into machine mode cause the PC to be set to the address in the BASE field. When MODE=Vectored, all synchronous exceptions into machine mode cause the PC to be set to the address in the BASE field, whereas interrupts cause the PC to be set to the address in the BASE field plus four times the interrupt cause number.

The register bits are shown in figure below.

31	1 0
BASE[31:2]	**MODE**

Figure 4: Bit mappings of MTVEC register

Bit #	R/W	Name	Description
1-0	RW	MODE	Vector mode ’’00”- All exceptions set PC to BASE. ’’01”- Asynchronous interrupts set PC to BASE+4×cause “1X”- RESERVED
31-2	RW	BASE	Trap vector Base address

5.3.3 Machine Interrupt Pending (MIP)

The MIP register is a 32-bit read/write register containing information on pending interrupts.

The register bits are shown in below figure.

31-12	11	10	9	8	7	6	5	4	3	2	1	0
-	MEIP	-	-	-	MTIP	-	-	-	MSIP	-	-	-

Bit #	R/W	Name	Description
3	RW	MSIP	Read-only bit that indicates Machine software interrupt pending cleared by writing zero to this bits
7	R	MTIP	Machine timer interrupt pending MTIP bit is read-only and is cleared by writing to the memory-mapped machine-mode timer compare register
11	R	MEIP	Machine External interrupt pending This read-only bit that indicates a machine-mode external interrupt is Pending. MEIP is set and cleared by a platform-specific interrupt controller.

Figure 5: Bit mapping of MIP register

5.3.4 Machine Interrupt Enable (MIE)

The MIE register is a 32-bit read/write register containing interrupt enable bits corresponding to MIP bits.

The register bits are shown in below figure.

31-12	11	10	9	8	7	6	5	4	3	2	1	0
-	MEIE	-	-	-	MTIE	-	-	-	MSIE	-	-	-

Figure 6: Bit mapping of MIE register

Bit #	R/W	Name	Description
3	RW	MSIE	Machine software interrupt Enable
7	RW	MTIE	Machine timer interrupt Enable
11	RW	MEIE	Machine External interrupt Enable

5.3.5 Machine Cause Register (MCAUSE)

When a trap is taken into M-mode, MCAUSE is written with a code indicating the event that caused the trap. Otherwise, MCAUSE is never written by the implementation, though it may be explicitly written by software.

31	30 0
Interrupt	**Exception Code**

Interrupt	Exception Code	Description
1	3	Machine software interrupt
1	7	Machine timer interrupt
1	11	Machine external interrupt
1	>11	Reserved for future use
0	1	Instruction access fault
0	2	Illegal instruction
0	4	Load address misaligned
0	5	Load access fault
0	6	Store address misaligned
0	7	Store access fault
0	11	Environment call from M-mode
0	>11	Reserved for future use

Figure 7: MCAUSE values after taking a trap

5.3.6 Machine Trap Value (MTVAL) Register

When a trap is taken into M-mode, MTVAL is written with exception-specific information to assist software in handling the trap. Otherwise, MTVAL is never written by the implementation, though it may be explicitly written by software. When a hardware breakpoint is triggered, or an instruction-fetch, load, or store address-misaligned, access or page-fault exception occurs, MTVAL is written with the faulting effective address. On an illegal instruction trap, MTVAL is written with the first XLEN bits of the faulting instruction as described below. For other exceptions, MTVAL is set to zero, but a future standard may redefine MTVAL’s setting for other exceptions.

31 0
MTVAL

5.3.7 Machine Scratch Register (MSCRATCH)

The mscratch register is a 32-bit read/write register dedicated for use by machine mode. Typically, it is used to hold a pointer to a machine-mode hart-local context space and swapped with a user register upon entry to an M-mode trap handler.

31	0
MSCRATCH	

5.3.8 Machine Exception Program Counter (MEPC)

When an exception is encountered, the current program counter is saved in MEPC, and the core jumps to the exception address. When an MRET instruction is executed, the value from MEPC replaces the current program counter. MEPC is written with the virtual address of the instruction that was interrupted or that encountered the exception.

31	0
MEPC	

6 Interface description

This section provides the interface details of ET1035. Instruction Memory interface manages how instructions are fetched from memory and delivered to the instruction pipeline. The Data Memory interface allows the processor to handle data during instruction execution, particularly for load and store operations. The ET1035 is connected to a global interrupt controller which will provide external, software and timer interrupts.

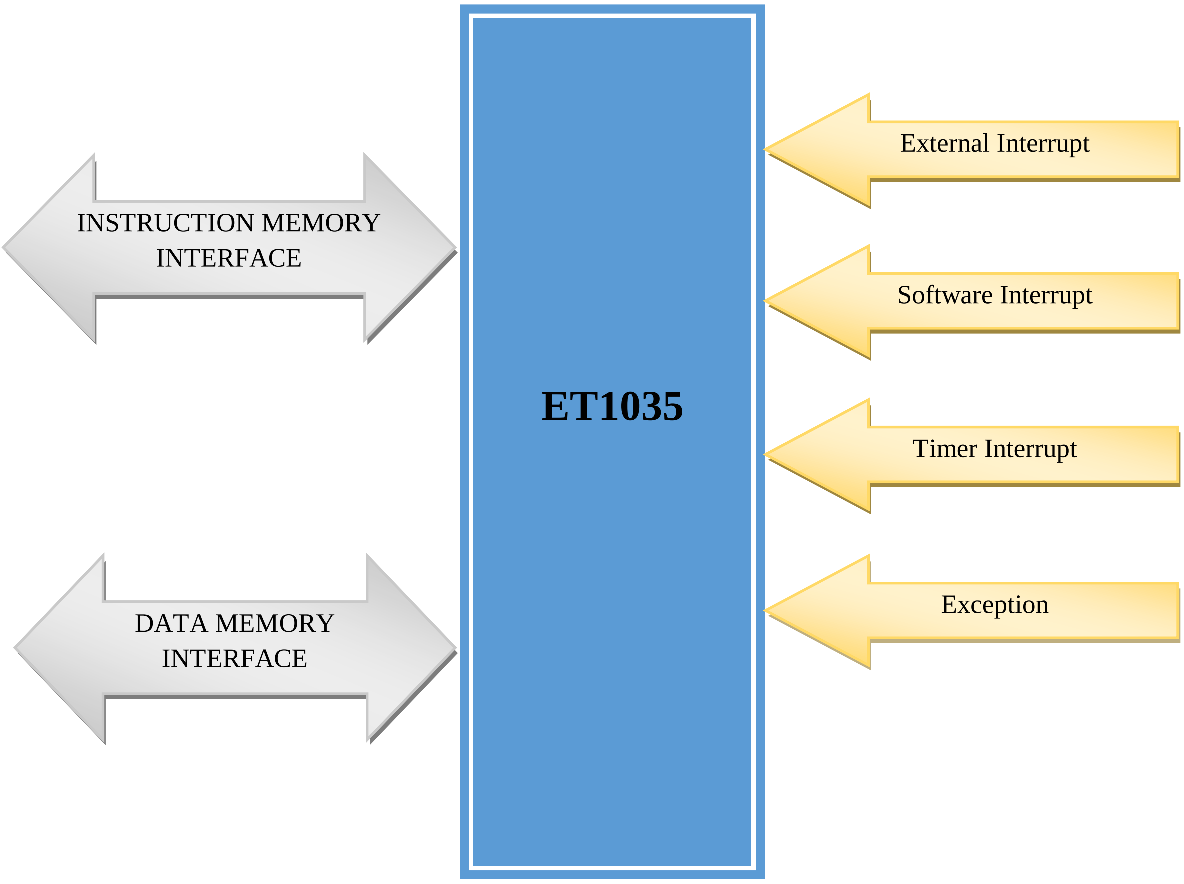

Figure 8: Interface Diagram

6.1 Instruction Memory Interface

The instruction fetch unit is capable of supplying one instruction to the decode stage per cycle if the instruction cache / the instruction memory is able to supply one instruction per cycle. The instruction address must be word-aligned due to the support of compressed instructions. It is not possible to jump to instruction addresses that have the LSB bit set.

6.2 Data Memory Interface

Data memory interface will duplicate the write data on the dmem_data_out port for 8-bit and 16-bit write operation. For read operation data must be provided on the appropriate byte locations, ie for an 8-bit load with address ending with 11 data read from the data in port (31:24) and update the destination register.

6.3 Signal List

SERIAL NO.	SIGNAL NAME	INTERFACE	SIZE (BITS)	DIRECTION	DESCRIPTION
1	imem_req	Instruction Memory Interface	1	OUT	Active high Request signal for instruction memory access
2	imem_seq		1	OUT	High indicates sequential memory access
3	imem_addr		32	OUT	Address
4	imem_data_in		32	IN	Instruction read from memory
5	reset_addr		32	IN	Reset address must be given here
6	imem_access_fault		1	IN	Instruction memory access fault, active high
7	dmem_req	Data Memory Interface	1	OUT	Request signal for data memory access
8	dmem_rw		1	OUT	High for Write, Low for read
9	dmem_size		3	OUT	Memory access size , Byte=00 , HW=01,W=10, reserved=11
10	dmem_addr		32	OUT	Address
11	dmem_data_out		32	OUT	Data for Write operation
12	dmem_data_in		32	IN	Input data for read operation
13	load_access_fault		1	IN	Load access fault, active high
14	load_addr_mis_align		1	IN	Load address misalignment, active high
15	store_access_fault		1	IN	Store access fault, active high

16	store_addr_mis_align		1	IN	Store address misalignment, active high
17	clk	Core signals	1	IN	Clock input
18	rst		1	IN	Active High reset
19	wait_n		1	IN	High for Normal operation Low to extent Memory access cycle
20	timer_interrupt		1	IN	Active High timer interrupt
21	ext_interrupt		1	IN	Active High external interrupt