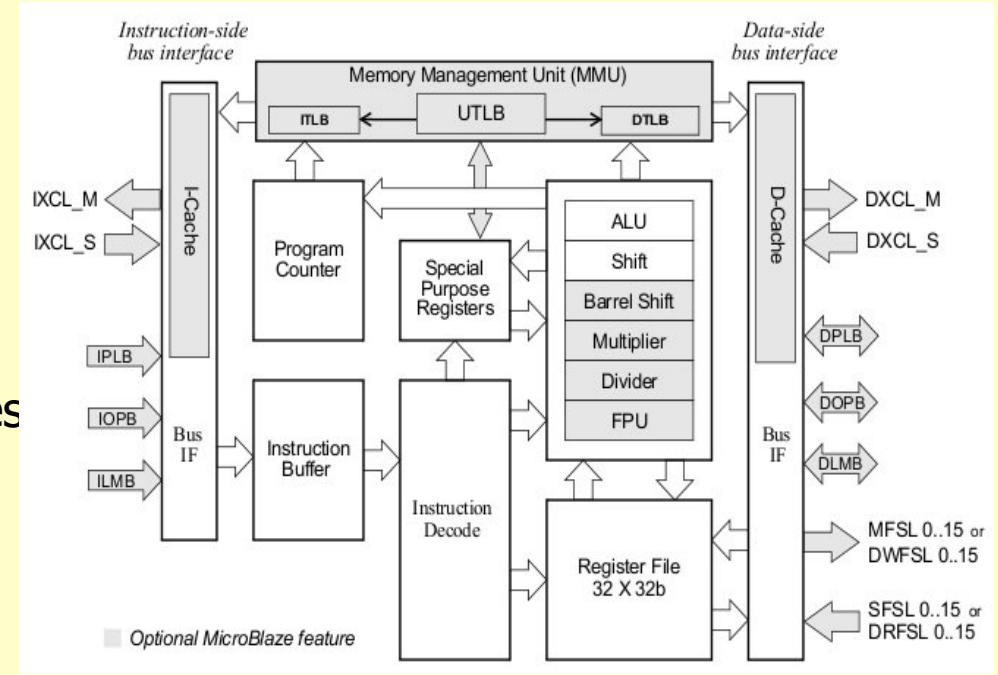
MicroBlaze Overview

Forrest Brewer

Core

- RISC Architecture
  - 3/5 stage single-issue pipe
  - Separate Data and Ins
- 32 32-bit GP registers
- 32-bit instructions
  - 3-operand/2-address modes
- Optional MMU
- Optional Busses:
  - LMB (local memory)
  - OPB (on-chip peripheral)
  - PLB (Processor Local Bus)
    - PLB from IBM PowerPC



Core Options

- OPB (Data or Ins)
- LMB (Data or Ins)
- PLB (Data or Ins)
- Divider/Barrel Shifter
- HW Debug
- FSL links (Multi-processor)
- Data and Ins Caches
- Exception Support
- FPU
- HW Floating Point Convert
- MMU

- Each option adds to the processor footprint on the FPGA
- Special Registers:
  - MSR (Machine Status) (1)
  - EAR (Exception Address) (3)
  - ESR (Exception Status) (5)
  - PC (Program Counter) (0)
  - FSR (FPU Status) (7)
  - BTR (Branch Target) (11)
  - All via SPR[x]
    - E.g. PC is SPR[0]

Data Layout

- Word
  - Bit-reversed big-endian
- Half Word
- Byte

Byte n	Byte n+1	Byte n+2	Byte n+3
MSByte			LSByte
0			31
MSBit			LSBit

Instruction Format

- 3-operand Instructions (5-bit field)
- 16-bit Immedate Operands
- Load/Store *(Ra+Rb) and *(Ra+Immediate)

Type A	0-5	6-10	11-15	16-20	21-31	Semantics
Type B	0-5	6-10	11-15	16-31		
ADD Rd,Ra,Rb	000000	Rd	Ra	Rb	00000000000	Rd := Rb + Ra
RSUB Rd,Ra,Rb	000001	Rd	Ra	Rb	00000000000	Rd := Rb + $\overline{Ra}$ + 1
ADDC Rd,Ra,Rb	000010	Rd	Ra	Rb	00000000000	Rd := Rb + Ra + C
RSUBC Rd,Ra,Rb	000011	Rd	Ra	Rb	00000000000	Rd := Rb + $\overline{Ra}$ + C
ADDK Rd,Ra,Rb	000100	Rd	Ra	Rb	00000000000	Rd := Rb + Ra
RSUBK Rd,Ra,Rb	000101	Rd	Ra	Rb	00000000000	Rd := Rb + $\overline{Ra}$ + 1
ADDKC Rd,Ra,Rb	000110	Rd	Ra	Rb	00000000000	Rd := Rb + Ra + C
RSUBKC Rd,Ra,Rb	000111	Rd	Ra	Rb	00000000000	Rd := Rb + $\overline{Ra}$ + C

GP Registers

Bits	Name	Description	Reset Value
0:31	R0	Always has a value of zero. Anything written to R0 is discarded	0x00000000
0:31	R1 through R13	32-bit general purpose registers	-
0:31	R14	32-bit register used to store return addresses for interrupts.	-
0:31	R15	32-bit general purpose register. Recommended for storing return addresses for user vectors.	-
0:31	R16	32-bit register used to store return addresses for breaks.	-
0:31	R17	If MicroBlaze is configured to support hardware exceptions, this register is loaded with the address of the instruction following the instruction causing the HW exception, except for exceptions in delay slots that use BTR instead (see "Branch Target Register (BTR)"); if not, it is a general purpose register.	-
0:31	R18 through R31	R18 through R31 are 32-bit general purpose registers.	-

Processor Version Reg

- 11 32-bit status registers describing the processor options and a unique identifier as well as cache sizes TLB options and target FPGA design.
- Required because there are dozens of optional processor components– allowing software to configure for hardware options

3 or 5 state Pipeline

	cycle 1	cycle 2	cycle 3	cycle4	cycle5	cycle6	cycle7
instruction 1	Fetch	Decode	Execute				
instruction 2		Fetch	Decode	Execute	Execute	Execute	
instruction 3			Fetch	Decode	Stall	Stall	Execute

	cycle 1	cycle 2	cycle 3	cycle 4	cycle 5	cycle 6	cycle 7	cycle 8	cycle 9
instruction 1	IF	OF	EX	MEM	WB				
instruction 2		IF	OF	EX	MEM	MEM	MEM	WB	
instruction 3			IF	OF	EX	Stall	Stall	MEM	WB

- Choice of pipeline depth
  - 5-stage offers faster clock, but longer latency
  - Branch requires 3-cycles in the Execution step
- Delay Slots
  - Like the MIPS design, only flush the fetch on taken branch
  - Decode stage instruction will complete (branch delay slot)
    - Cannot have IMM, branch or break ins in delay slot.
    - Recoverable exceptions are allowed in Branch Delay Slot

Harvard Memory Architecture

- Separate Data and Memory interfaces and address spaces
  - Can overlap if desired (debug: user modifiable code)
- All I/O is memory Mapped
  - Bus selection is mapped into address ranges
- Cache line is 4 or 8 words

Privileged Instructions

- GET, PUT, NGET, NPUT.. MTS, MSRCLR, MSRSET, BRK, RTID… are all privileged.
- Will raise protection exception in user code
  - Exception: BRKI 0x8, or BRKI 0x18 perform user vector exception
- Hardware Exceptions, Interrupts and Software Breaks cause entry to privileged mode.
  - Need Prolog and Epilog code to protect user mode registers
  - RTED (Return from Exception or Interrupt) goes back to user or virtual mode.

Exceptions

1. Reset
2. Hardware Exception
3. NMI
4. Break
5. Interrupt
6. User Vector (exception)

- Exceptions are prioritized from top
- Vectors in low address space
- Register File Return Addresses

Event	Vector Address	Register File Return Address
Reset	0x00000000 - 0x00000004	-
User Vector (Exception)	0x00000008 - 0x0000000C	Rx
Interrupt	0x00000010 - 0x00000014	R14
Break: Non-maskable hardware	0x00000018 - 0x0000001C	R16
Break: Hardware		
Break: Software		
Hardware Exception	0x00000020 - 0x00000024	R17 or BTR
Reserved by Xilinx for future use	0x00000028 - 0x0000004F	-

Reset

PC <- 0x00000000

MSR <- C_RESET_MSR (configurable)

EAR, ESR, FSR, PID, ZPR, TLBX <- 0

- Code starts executing from 0x0 (RESET Vector)
- Reset Needs to be asserted for 16 cycles Minimum!

Breaks

- Hardware and Software Breaks both supported
  - Ext_BRK and Ext_NM_BRK are the signals
  - Code vector is 0x18
  - Return Address stored in R16
  - BIP (Break in Progress) bit set in MSR
  - RTBD instruction clears BIP, returns to PC <- *(R16)
- Software
  - BRK and BRKI instructions invoke software breaks

MicroBlaze Interrupt

- One source supported
  - Interrupt signal port
  - PIC Programmable Interrupt Controllers available
- IE bit of MSR needs to be set to allow interrupts
- Execution stage completes
- Decode stage overwritten by branch to 0x10
- PC address of instrcution that was in Decode stage is the return address -> R14
- IE bit is MSR is cleared, reset by RTID (return)
- Interrupts are ignored if BIP or EIP bits of MSR (branch or exception routines in progress) are set.
- Latency determined by instruction in progress and vector memory delay -> Hardware_Divide if present has huge latency…

Caches

- Optional hardware caches for Instructions or Data
  - 1-way direct mapped cache
  - Cachable address range is user settable
  - Variable Size (set during configuration) 64B-64kB
  - Disable bits in MSR (ACE and DCE)
  - WIC, WDC instructions to allow software invalidation of cache lines
  - Cache lines 4 or 8 words (configurable)
- Caches use BRAM of Spartan for both cache and tags
  - Be wary of physical memory constraints!

FPU

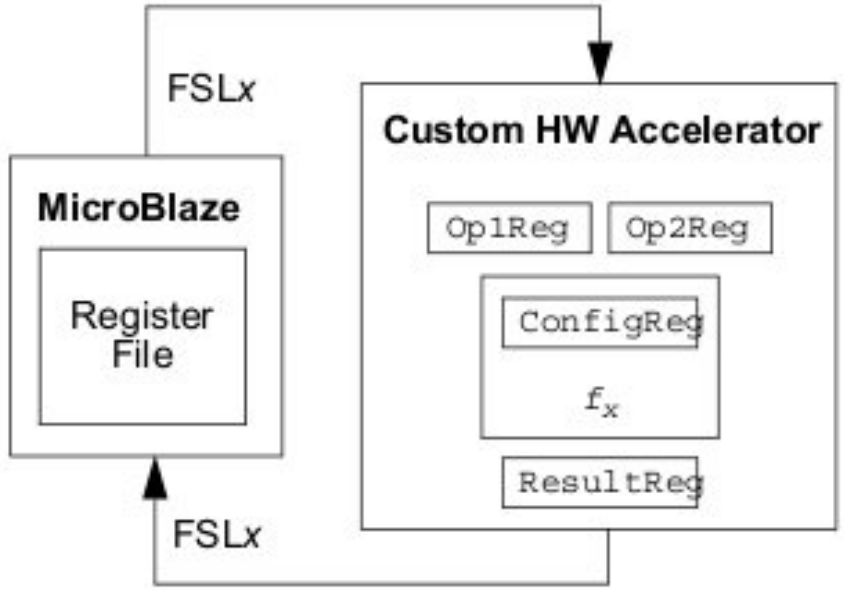
- IEEE 754 Standard Single-Precision Floating Point
- ADD, SUB, MUL, DIV, Comp, Conv, SQRT
- Nan is supported (quiet exception)
- Overflow returns signed ∞
- 32-bit float 8-bit exponent, 23-bit mantissa
- Vaules from and returned to GP register set of processor
- Exceptions (when enabled) are regular Hardware Exceptions (FSR keeps the bits)
  - result register not overwritten if exception

Fast Simplex Links

- 16 FSL interfaces allow custom hardware accelerators
- Use GET and PUT instructions

Example code:

```
// Configure f_x
cput Rc,RFSLx
// Store operands
put Ra, RFSLx // op 1
put Rb, RFSLx // op 2
// Load result
get Rt, RFSLx
```



Debugging and Tracing

- JTAG based Software Debug
  - Background Debug Mode
  - Uses MDM (Xilinx Microprocessor Debug Module)
- Supports
  - Configuralble hardware breakpoints, watchpoints
  - Run/Stop/Step processor
  - Read and Write GP regs and most special purpose registers
  - Multiple processors (chained JTAG)

MicroBlaze ABI

Data Types

- Byte 8-bit, Short 16-bit, and Long 32-bit
- C-types:
  - ‘char’ 8-bit
  - ‘short’ 16-bit
  - ‘long’ or ‘int’ 32-bit
  - ‘float’ 32-bit
  - ‘enum’ 32-bit
  - Pointers can be 16 or 32 bit, depending on data area size

Register Conventions (GCC)

Register	Type	Enforcement	Purpose
R0	Dedicated	HW	Value 0
R1	Dedicated	SW	Stack Pointer
R2	Dedicated	SW	Read-only small data area anchor
R3-R4	Volatile	SW	Return Values/Temporaries
R5-R10	Volatile	SW	Passing parameters/Temporaries
R11-R12	Volatile	SW	Temporaries
R13	Dedicated	SW	Read-write small data area anchor
R14	Dedicated	HW	Return address for Interrupt
R15	Dedicated	SW	Return address for Sub-routine
R16	Dedicated	HW	Return address for Trap (Debugger)
R17	Dedicated	HW, if configured to support HW exceptions, else SW	Return address for Exceptions
R18	Dedicated	SW	Reserved for Assembler
R19-R31	Non-volatile	SW	Must be saved across function calls. Callee-save

Register Conventions II

Register	Type	Enforcement	Purpose
RTLBSX	Special	HW	Translation Look-Aside Buffer Search Index
RPVR0-RPVR11	Special	HW	Processor Version Register 0 through 11
RPC	Special	HW	Program counter
RMSR	Special	HW	Machine Status Register
REAR	Special	HW	Exception Address Register
RESR	Special	HW	Exception Status Register
RFSR	Special	HW	Floating Point Status Register
RBTR	Special	HW	Branch Target Register
REDR	Special	HW	Exception Data Register
RPID	Special	HW	Process Identifier Register
RZPR	Special	HW	Zone Protection Register
RTLBLO	Special	HW	Translation Look-Aside Buffer Low Register
RTLBHI	Special	HW	Translation Look-Aside Buffer High Register
RTLBX	Special	HW	Translation Look-Aside Buffer Index Register

Register Use Notes

- R3-R12 are volatile – not retained in over function calls
  - R3, R4 are function return values
  - R5-R10 used to pass parameters
- R19-R31 are stable across function calls (non-volatile)
  - Called function needs to save these to stack in prologue and return them in epilogue code
  - R14-R17 store return addresses from interrupts, subroutines, traps, exceptions
  - Subroutine Call: BRL (Branch and Link) – saves PC at R15
  - Short pointers (SDA) use R2 and R13 as address anchors for read-only and read/write small data areas respectively
  - R1 is the stack pointer
  - R18 is the assembler operation temporary register

Stack Convention

- Stack grows toward lower addresses
- Caller passes parameters using R5-R10 or by adding a stack frame
- Callee Returns values via R3-R4 or by writing to caller stack frame

High Address	
	Function Parameters for called sub-routine (Arg n .. Arg1) (Optional: Maximum number of arguments required for any called procedure from the current procedure).
Old Stack Pointer	Link Register (R15)
	Callee Saved Register (R31....R19) (Optional: Only those registers which are used by the current procedure are saved)
	Local Variables for Current Procedure (Optional: Present only if Locals defined in the procedure)
	Functional Parameters (Arg n .. Arg 1) (Optional: Maximum number of arguments required for any called procedure from the current procedure)
New Stack Pointer	Link Register
Low Address	

Memory

Types: SDA (small data area), Data Area, Common Area, Literals (Constants)

- SDA
  - Globally initiallized variables
  - Max size object threshold in mbgcc: 8-bytes
  - R13 + 16-bit immediate offset, also absolute (32-bit address)
- Data Area
  - Larger initialized variables (also could be SDA access < 64kB)
- Common
  - Uninitialized global space
- Literals
  - R2 Read-Only Data anchor (hardware enforced)
  - Could be overwritten by absolute address…

Interrupt and Exception Handlers

crt0.o as usual is the initialization for main()

Xilinx provides a compiler option: -x1-mode-xmdstub which allows overriding the default handlers by linking synbolic addresses:

On	Hardware jumps to	Software Labels
Start / Reset	0x0	_start
User exception	0x8	_exception_handler
Interrupt	0x10	_interrupt_handler
Break (HW/SW)	0x18	-
Hardware exception	0x20	_hw_exception_handler
Reserved by Xilinx for future use	0x28 - 0x4F	-

Exception Handler Dispatch

- The compiler writes the user-specified addresses to the vector area as follows:
- You can override the default routines by using the GNU function attribute: 'interrupt_handler'
- This attribute adds the appropriate prologue and epilog code and passes the link symbol to crt0.o

```
0x00:   bri     _start1
0x04:   nop
0x08:   imm     high bits of address (user exception handler)
0x0c:   bri     _exception_handler
0x10:   imm     high bits of address (interrupt handler)
0x14:   bri     _interrupt_handler
0x20:   imm     high bits of address (HW exception handler
0x24:   bri     _hw_exception_handler
```