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

<table>
<thead>
<tr>
<th>Byte n</th>
<th>Byte n+1</th>
<th>Byte n+2</th>
<th>Byte n+3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSByte</td>
<td></td>
<td>LSBYTE</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>MSBit</td>
<td></td>
<td>LSBIT</td>
<td></td>
</tr>
</tbody>
</table>
Instruction Format

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

<table>
<thead>
<tr>
<th>Type A</th>
<th>0-5</th>
<th>6-10</th>
<th>11-15</th>
<th>16-20</th>
<th>21-31</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type B</td>
<td>0-5</td>
<td>6-10</td>
<td>11-15</td>
<td>16-31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD Rd,Ra,Rb</td>
<td>000000</td>
<td>Rd</td>
<td>Ra</td>
<td>Rb</td>
<td>000000000000</td>
<td>Rd := Rb + Ra</td>
</tr>
<tr>
<td>RSUB Rd,Ra,Rb</td>
<td>000001</td>
<td>Rd</td>
<td>Ra</td>
<td>Rb</td>
<td>000000000000</td>
<td>Rd := Rb + Ra + 1</td>
</tr>
<tr>
<td>ADDC Rd,Ra,Rb</td>
<td>000010</td>
<td>Rd</td>
<td>Ra</td>
<td>Rb</td>
<td>000000000000</td>
<td>Rd := Rb + Ra + C</td>
</tr>
<tr>
<td>RSUBC Rd,Ra,Rb</td>
<td>000011</td>
<td>Rd</td>
<td>Ra</td>
<td>Rb</td>
<td>000000000000</td>
<td>Rd := Rb + Ra + C</td>
</tr>
<tr>
<td>ADDK Rd,Ra,Rb</td>
<td>000100</td>
<td>Rd</td>
<td>Ra</td>
<td>Rb</td>
<td>000000000000</td>
<td>Rd := Rb + Ra</td>
</tr>
<tr>
<td>RSUBK Rd,Ra,Rb</td>
<td>000101</td>
<td>Rd</td>
<td>Ra</td>
<td>Rb</td>
<td>000000000000</td>
<td>Rd := Rb + Ra + 1</td>
</tr>
<tr>
<td>ADDKC Rd,Ra,Rb</td>
<td>000110</td>
<td>Rd</td>
<td>Ra</td>
<td>Rb</td>
<td>000000000000</td>
<td>Rd := Rb + Ra + C</td>
</tr>
<tr>
<td>RSUBKBC Rd,Ra,Rb</td>
<td>000111</td>
<td>Rd</td>
<td>Ra</td>
<td>Rb</td>
<td>000000000000</td>
<td>Rd := Rb + Ra + C</td>
</tr>
</tbody>
</table>
## GP Registers

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:31</td>
<td>R0</td>
<td>Always has a value of zero. Anything written to R0 is discarded</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0:31</td>
<td>R1 through R13</td>
<td>32-bit general purpose registers</td>
<td>-</td>
</tr>
<tr>
<td>0:31</td>
<td>R14</td>
<td>32-bit register used to store return addresses for interrupts.</td>
<td>-</td>
</tr>
<tr>
<td>0:31</td>
<td>R15</td>
<td>32-bit general purpose register. Recommended for storing return addresses for user vectors.</td>
<td>-</td>
</tr>
<tr>
<td>0:31</td>
<td>R16</td>
<td>32-bit register used to store return addresses for breaks.</td>
<td>-</td>
</tr>
<tr>
<td>0:31</td>
<td>R17</td>
<td>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 &quot;Branch Target Register (BTR)&quot;); if not, it is a general purpose register.</td>
<td>-</td>
</tr>
<tr>
<td>0:31</td>
<td>R18 through R31</td>
<td>R18 through R31 are 32-bit general purpose registers.</td>
<td>-</td>
</tr>
</tbody>
</table>
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

- **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

<table>
<thead>
<tr>
<th>Event</th>
<th>Vector Address</th>
<th>Register File Return Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>0x00000000 - 0x00000004</td>
<td>-</td>
</tr>
<tr>
<td>User Vector (Exception)</td>
<td>0x00000008 - 0x0000000C</td>
<td>Rx</td>
</tr>
<tr>
<td>Interrupt</td>
<td>0x00000010 - 0x00000014</td>
<td>R14</td>
</tr>
<tr>
<td>Break: Non-maskable hardware</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Break: Hardware</td>
<td>0x00000018 - 0x0000001C</td>
<td>R16</td>
</tr>
<tr>
<td>Break: Software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Exception</td>
<td>0x00000020 - 0x00000024</td>
<td>R17 or BTR</td>
</tr>
<tr>
<td>Reserved by Xilinx for future use</td>
<td>0x00000028 - 0x0000004F</td>
<td>-</td>
</tr>
</tbody>
</table>
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 instruction 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 $\infty$
- 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
ecut 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**
  - Configurable 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)

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

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

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

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

- SDA
  - Globally initialized 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 symbolic addresses:

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