BlueJEP: A Flexible and High-Performance Java Embedded Processor

Flavius Gruian[1] Mark Westmijze[2]

[1]Lund University, Sweden
flavius.gruian@cs.lth.se

[2]University of Twente, The Netherlands
m.westmijze@student.utwente.nl

Java Technologies for Real-time and Embedded Systems, 2007

Outline

1 Introduction

2 Design

3 Implementation and Experiments

4 Discussion

5 Summary and Continuing Work

What are we trying to do?

1. Design a Java processor starting from JOP [M. Schöberl]
2. Evaluate BlueSpec System Verilog as a design language

BlueSpec System Verilog (BSV)

Rule based, strongly-typed, declarative hardware specification language, making use of Term Rewriting Systems to describe computations as atomic state changes.

3. Outperform other existing Java processors in terms of
   - design time
   - flexibility
   - execution speed
   - device area

BlueJEP

BlueSpec System Verilog **J**ava **E**mbedded **P**rocessor

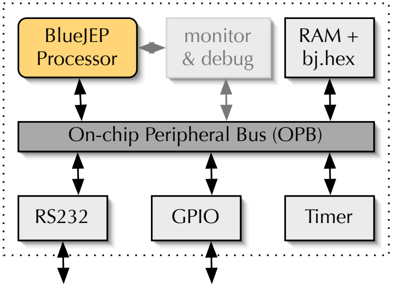
Design features and constraints

Many design features shared between BlueJEP and JOP (VHDL):

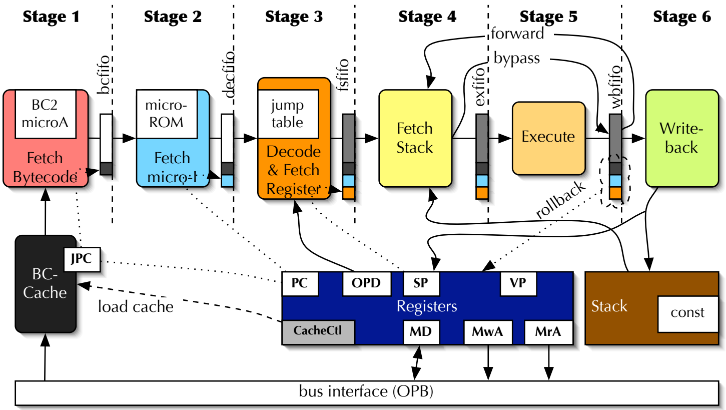
- micro-programmed, stack machine core
- predictable rather than high-performance (RT systems)
- given instruction set (bytecodes)
- fixed micro-instruction set (for ease of programming)
- identical executable image (loaded classes)
- same back-end (synthesis) tools
- same implementation platform (FPGA)

Complete system overview

- small: system on a FPGA
- flexible: support exploration
- real-time: easily predictable timing
- portable: standard interfaces for fast integration (OPB, LMB for Xilinx EDK)



Six Stages Micro-Programmed Pipeline



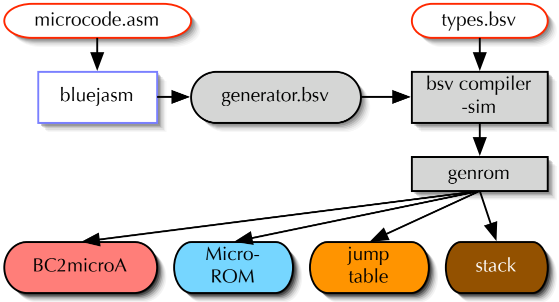
Handling data

- Data dependencies cause stalls (stages 3,4,5):
  - searchable FIFOs are used to look for specific destinations
  - stages do not fire if the required sources are destinations in any of the following SFIFOs
- Improved performance through forwarding stack words (from the write-back FIFO – stage 6)
- Register forwarding seems to yield marginal improvements only at the expense of more hardware (therefore not used)
- Bypass *Execute* (stage 5) for data moving operations
- External memory accessed via registers (MWA, MRA, MD)

Handling control

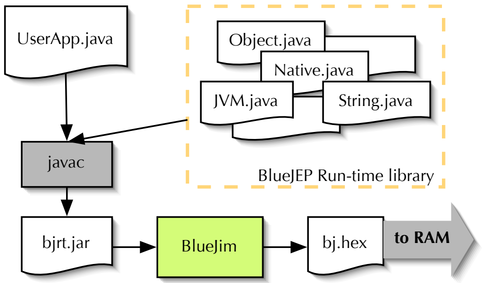
- Micro-code branches: BZ, BNZ, BP, BNP, BM, BNM, GOTO affect PC
- Java branches are combinations of comparison operations, JPC load/store and micro-branches.
- Speculative execution of micro-branches – always "not taken":
  - no need for SFIFOs, no need to stall when PC or JPC changes $\rightarrow$ simpler hardware
  - context (JPC, PC, SP) must be passed along and restored when needed in the *Writeback* stage $\rightarrow$ wider FIFOs
  - flushing FIFOs and restoring context is easy $\rightarrow$ simpler code (hard to debug though...)
- Special register for controlling the load of the method cache (CACHECTL) on INVOKES and RETURNS.

From assembly to micro-ROM



- The encoding of the micro-instructions does not affect the assembler (bluejasm)!
- The actual encoding is interesting for optimization purposes only.

From application to run-time environment



BlueJim image generator

- offline class loading and linking
- replaces native calls with custom bytecodes
- throws away unused methods and fields
- adds GC information

JVM.java	Java implemented bytecodes.
Native.java	Java-hardware interface.
*.java	Reduced JRE library.

Target System and Tools

Target FPGA

- Xilinx Virtex-II (XC2V1000, fg456-4)

Tools

- BSV compiler 2006.11, *BSV* $\longrightarrow$ *Verilog*
- Xilinx EDK 9.1i, *Verilog* + *IPs* $\longrightarrow$ *System*
- Xilinx ISE 9.1i, *System* $\longrightarrow$ *FPGA*
- Chipscope, to monitor and debug

Device Area

Synthesis parameters: optimized for speed, distributed RAM.

Resources	Taken	Available	Percentage
Slices	3460	5120	68%
Flip-Flops	756	10240	7%
4LUTs	6858	10240	66%
	2422	used as logic	
	4436	used as RAM	

Observations, compared to JOP

- Logic takes around the same amount of resources
- RAM takes around five times more resources (the BSV RegFiles are memories with 5 read ports and 1 write port)

Clock Speed

Maximum clock speeds for BlueJEP and JOP (with OPB):

	Virtex-II	**Spartan3**	**Virtex5**
	XC2V-4	XS3-5	XC5VLX30-3
JOP (OPB)	60 MHz	66 MHz	200 MHz
BlueJEP	85 MHz	76 MHz	221 MHz
ϕ	1.42	1.15	1.10

Clock factor

$$\phi = f_{\text{BlueJEP}} / f_{JOP}$$

- BlueJEP running faster than JOP is partly a consequence of increasing the number of stages from 4 to 6 !

Bytecode Execution Speed

Bytecode(s)	JOP	BlueJEP	
	CC	CC	$RS_{1.42}$
iload iadd	2	3	0.95
iinc	11	13	1.20
ldc	9	12	1.06
if_icmplt taken	6	23	0.37
if_icmplt n/taken	6	8	1.06
getfield	23	38	0.86
getstatic	15	18	1.18
iaload	29	45	0.92
invoke	126	166	1.08
invoke static	100	111	1.28

Clock factor

$$\phi = \frac{f_{\text{BlueJEP}}}{f_{JOP}}$$

Relative speedup

$$RS_{\phi} = \phi \frac{CC_{JOP}}{CC_{\text{BlueJEP}}}$$

- some bytecodes are executed faster, some slower than on JOP
- speculative execution takes its toll on taken branches

BlueSpec System Verilog issues

Coding compared to a VHDL design:

- shorter development time (1/2)
- fewer lines (1/3)
- more readable, maintainable, flexible

Test & Debug along with the classic Verilog/VHDL ways:

- easy, software-like test-benches (*StmtFSM*)
- standalone BSV high-level executable
- probes, asserts, debug messages...

Results are as expected:

- larger area (needs efficient synthesis tools)
- OK performance (timing is harder to control)

The Design Rationale

Follow the classic JOP design (loosely) in order to compare BSV and VHDL design flows, but exploration led to...

- Six pipeline stages instead of four
  - simpler stages
  - shorter critical path
- Speculative execution
  - simpler control
  - no stalls on success
- OPB bus interface
  - easy integration with other OPB cores in the Xilinx EDK
  - easily replaceable
- Micro-instruction set
  - adapted for our architecture and folding
  - custom micro-assembler back-end

Finally...

Summary We introduced BlueJEP, which:

- is a native Java embedded processor
- is specified in BlueSpec System Verilog
- has similar performance to existing solutions
- proves that BSV is perfect for fast prototyping

Extensions

- Micro-instruction Folding [under evaluation]
- Memory Management Support [completed]