A Parametrizable Hybrid Stack-Register Processor as Soft Intellectual Property Module

Peter Lüthi, Thomas Röwer, Manfred Stadler, Daniel Forrer, Stefan Moscibroda, Norbert Felber, Hubert Kaeslin, Wolfgang Fichtner

Integrated Systems Laboratory, Swiss Federal Institute of Technology, Zürich, Switzerland

September, 14th 2000

Abstract

We wanted to build a parametrizable processor IP-Module, which had to be capable of managing high interrupt loads.

Being able to perform fast context saves, the key concept was a register bank implemented as top of stack.

Overview

- IP requirements
- Processor architecture & parameterization
- Functional verification flow
- Test integration
- Conclusions & outlook

IP Requirements

Highly adaptable processor IP

- Qualitative customization : instruction set
- Quantitative customization : data width, ...
- Separation of core and interfaces

Convenient functional verification flow

- Automated test vector generation based on current configuration

Architecture Comparison

	Stack architecture	Register based architecture
Compiler efficiency	–	+
Fast IRQ launch	+	–

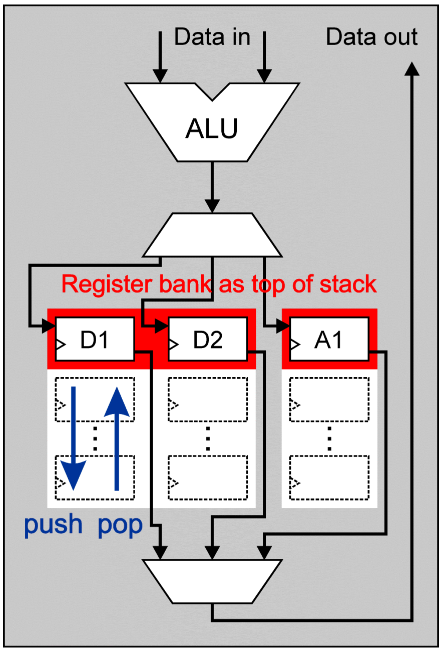
Get best of both worlds:

⟹ Register bank implemented as top of stack

Processor Architecture

Advantages

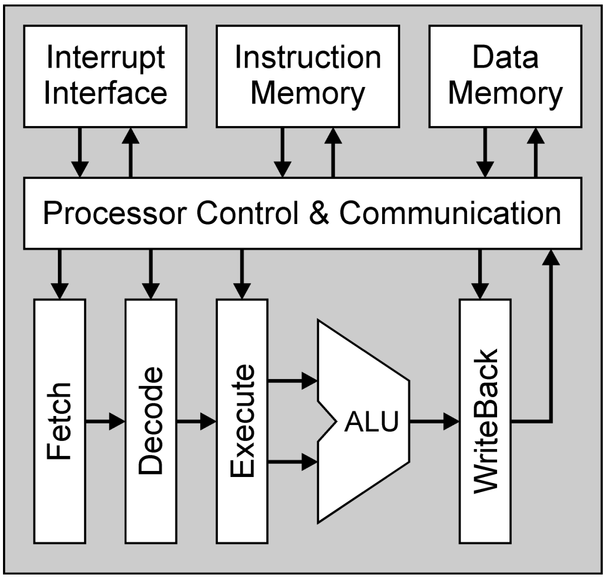
- Quick context save on interrupt request by „push“ operation
- Context restore by „pop“ operation
- Ability to launch interrupts with 2 clock cycles latency
- No pipeline flush



Processor Architecture

Features

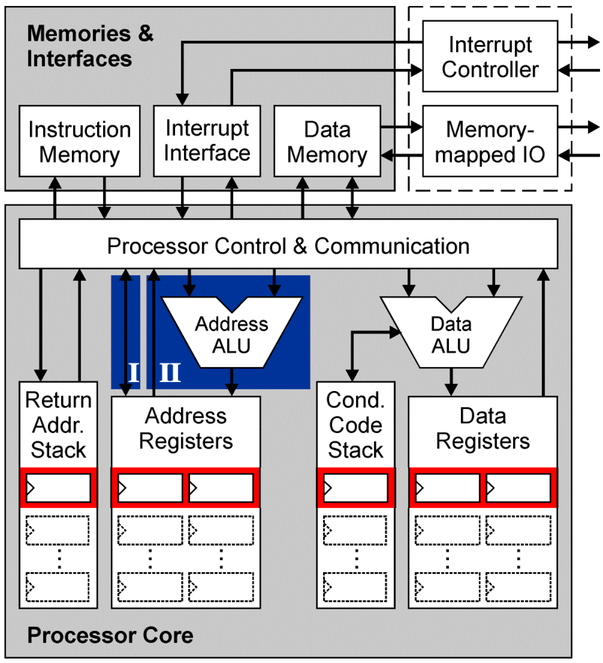
- Harvard architecture
- Customizable RISC instruction set
- 4 stage pipeline
- Read-after-write allowed
- Single cycle execution



Processor Architecture

Features

- Separation of core and interfaces
- Param. data memory interface with FIFO buffer
- Cond. code & return address stack
- Optional address ALU



Parameterization

Qualitative customization

- Deactivation of unwanted instructions
- Optional address ALU

Quantitative customization

- Data width & data memory address range
- Instruction memory address range
- No. of directly accessible data & address registers
- Depth of data / address stack & return address stack

Functional Verification Flow

Problem

Functional test vectors are strongly dependent on chosen parameter setting

Solution

Generation of verification vectors compatible to current parameter set

⟹ Flow based on behavioral model

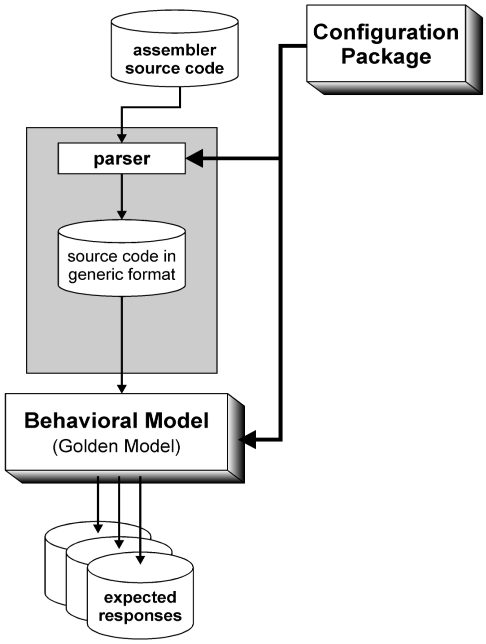
Functional Verification Flow

Step 1

Translation of assembler source into generic format

Step 2

Generation of expected responses using behavioral model



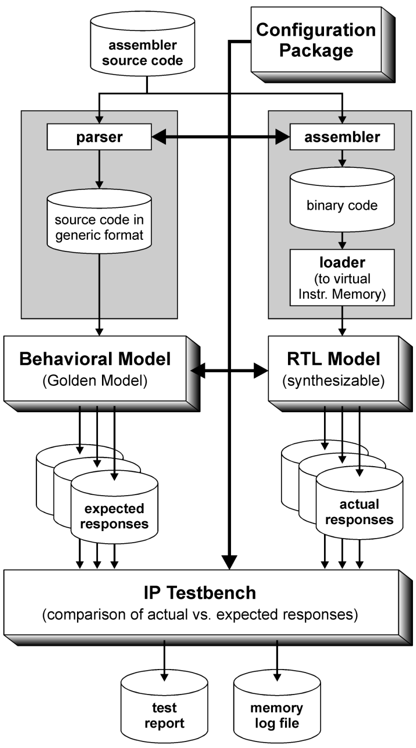
Functional Verification Flow

Step 3

Translation of assembler source into binary code

Step 4

Functional verfication of RTL model

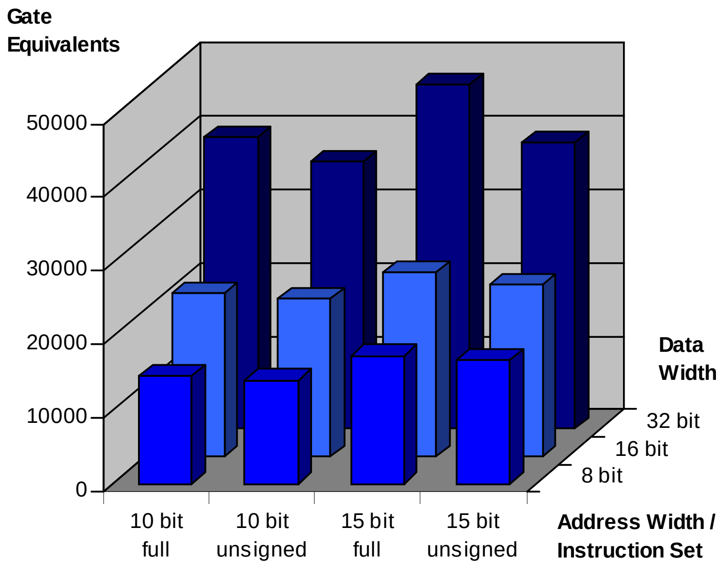


Various Test Synthesis Runs

Common parameters

Top of stack data register:	8
Top of stack addr. register:	1
Data / addr. stack depth:	3
Return addr. stack depth:	8

Same timing constraints for all synthesis runs used



Test Integration

Process:	0.6 µm CMOS 3LM
Data registers:	12 x 16 bit
Address registers:	4 x 10 bit
Stack depth:	4
Data memory size:	1k x 16 bit
Core (w/o. mem):	104‘000 trans.
Max. operating frequency:	121 MHz
Supply voltage:	5 Volt

Power Consumption

Operating current :

288 mA @ 121.5 MHz, 5 V

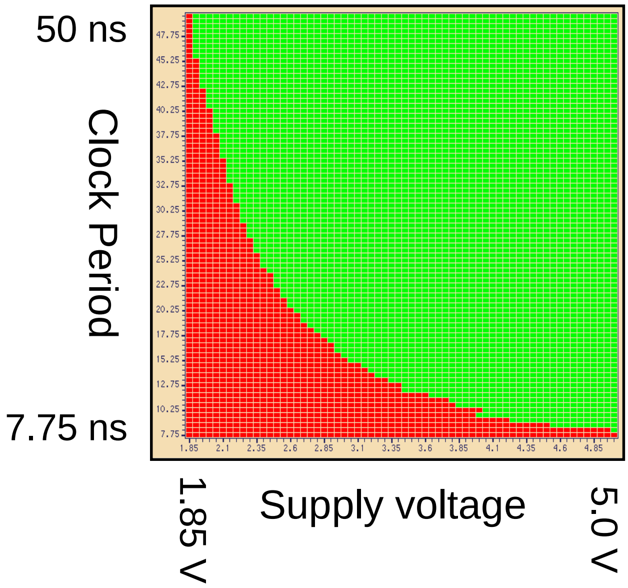
19.7 mA @ 21.9 MHz, 1.9 V

Power / MIPS :

11.85 mW / MIPS
@ 121,5 MHz, 5 V

1.7 mW / MIPS
@ 21.9 MHz, 1.9 V

Shmoo Plot



Conclusions

- Highly parametrizable RISC processor Soft IP-Module
- Hybrid stack-register architecture
- Very fast interrupt launch
  Max. 2 clock cycles latency : 16.46 ns @ 121.5 MHz
- Convenient functional verification flow
  Covering automatically the current parameter setting

Outlook

- High-level programming language compiler
- For parameter settings requiring a large chip area :

  Reduction of silicon area by moving the data / address stack to an embedded memory