

microCore (μ Core)

Scalable

Dual Stack

Computing Engine
for Real-Time Control

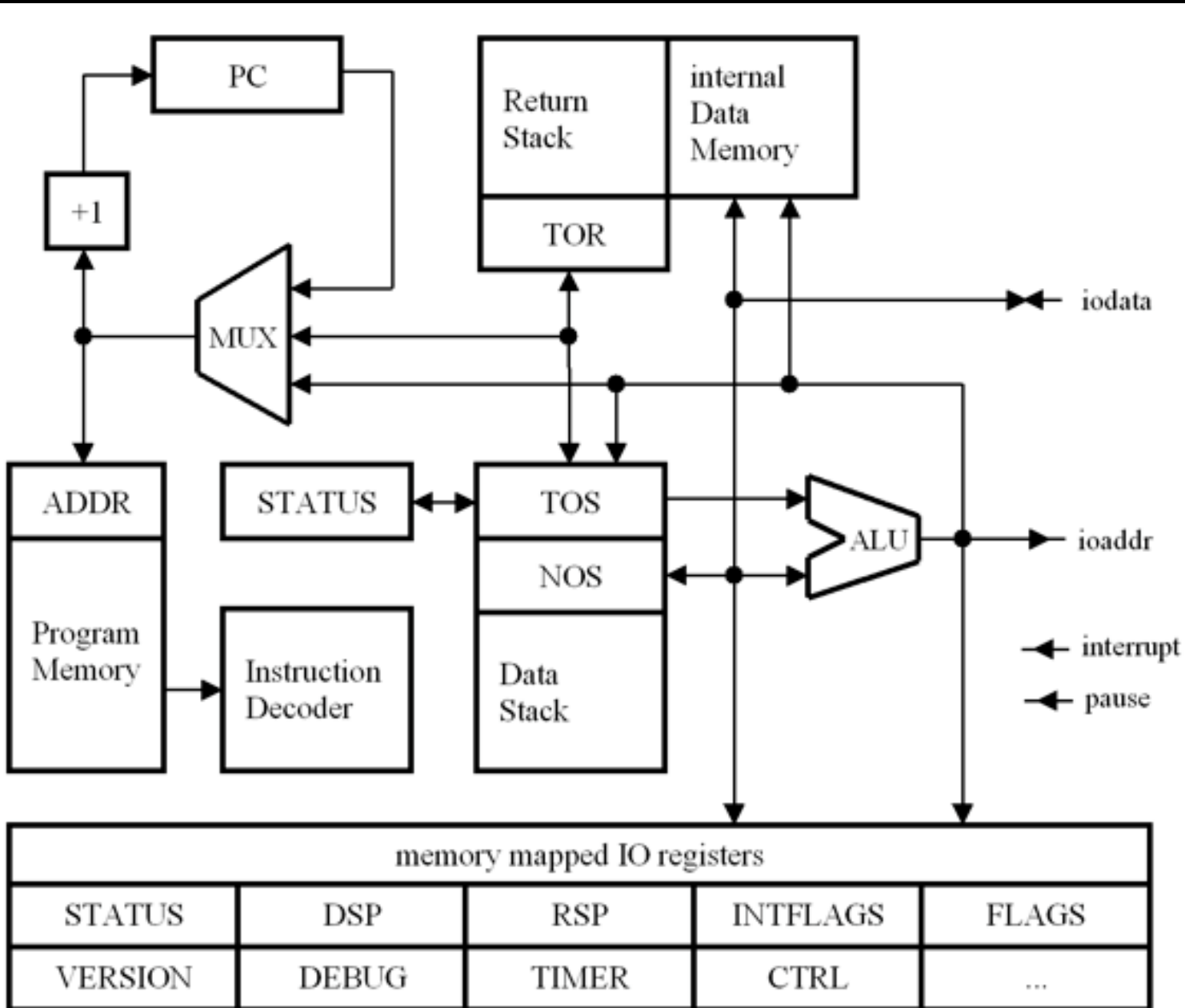
Klaus Schleisiek

kschleisiek at freenet.de

Characteristics

- Computing engine written in VHDL for FPGAs
- Dual stack (data stack & return stack)
- Harvard architecture (8 bit program, independent data memory)
- Configurable word width for data memory and stacks
- Postfix instruction set architecture (literal operands/addresses precede rather than follow instructions consuming them)
- Single cycle execution, no pipelining
- External events: Interrupt & Pause traps
- Multiple data and return stack areas for efficient multitasking
- Deterministic program execution
- Hardware/Software co-design environment
- Malleable instruction set

Architecture



Stack versus Registers

- Parameter passing

Both architectures suffer from 'misplaced' arguments that need to be re-ordered.

- Interrupt processing

Stack: No registers need to be saved.

- Code optimization

'Register allocation' is well known, 'stack allocation' is an esoteric topic.

- Instruction set impact

Stack: No need to have a register-address-field in the instruction.

Looking at the technical merits, the stack architecture has several advantages, especially for real time applications.

Its major disadvantage is its **unfamiliarity**.

Harvard Architecture

- μ Core's instructions and the program memory are 8 bits wide.
- Data memory is an independent memory area that has a **configurable data width**.
- μ Core has no **bytes**. Each data memory cell is data width wide.

Postfix Instruction Set

- Enabling technology for the **configurable data width**.
- **Literal instructions** have the MSB set and are concatenated on the stack to any magnitude from 7 bit nibbles.

lit flag	MSB	interpretation
0	0	An opcode that does not need a literal.
0	1	A literal following an opcode. The least significant 7 bits are pushed on the stack as a 2s-complement number in the range -64 .. 63, and the lit flag is set.
1	0	An opcode following a literal. If it is just a noop, the literal remains on the stack as a number. The lit flag is reset.
1	1	A literal following another literal. The literal in TOS is shifted 7 bit positions to the left and the 7 least significant bits of the instruction are appended. This covers the following number ranges: 1 lit -64 .. 63 2 lits -8192 .. 8191 3 lits -1048576 .. 1048575 etc. This way numbers of any magnitude can be constructed.

Single Cycle Execution

- Each instruction needs one active clock transition.
- μ Core's instruction cycle may be several system clock periods long by way of a clock enable signal. It may even be 'frozen' externally.
- Read access to **FPGA blockRAMs** need two un-interruptible cycles (1: set address, 2: read). A general **chaining mechanism** allows to define un-interruptible multi-cycle code sequences also e.g. for read-modify-write instructions like +!.

External Events

- μ Core has two different input signals to react to external events:

Interrupt

An event did happen that was **not** expected by the software.

Pause

An event did **not** happen that was expected by the software.

- Each instruction is self contained and therefore, μ Core may be interrupted between any two instructions.
(Chained instructions count as 'one' instruction.)

Pause

- In a multi-tasking system, the **pause trap** allows to delegate **mutual exclusion** of resources to the hardware.
- Then an ADC with an 8 channel input multiplexer can be controlled on the software level as follows:

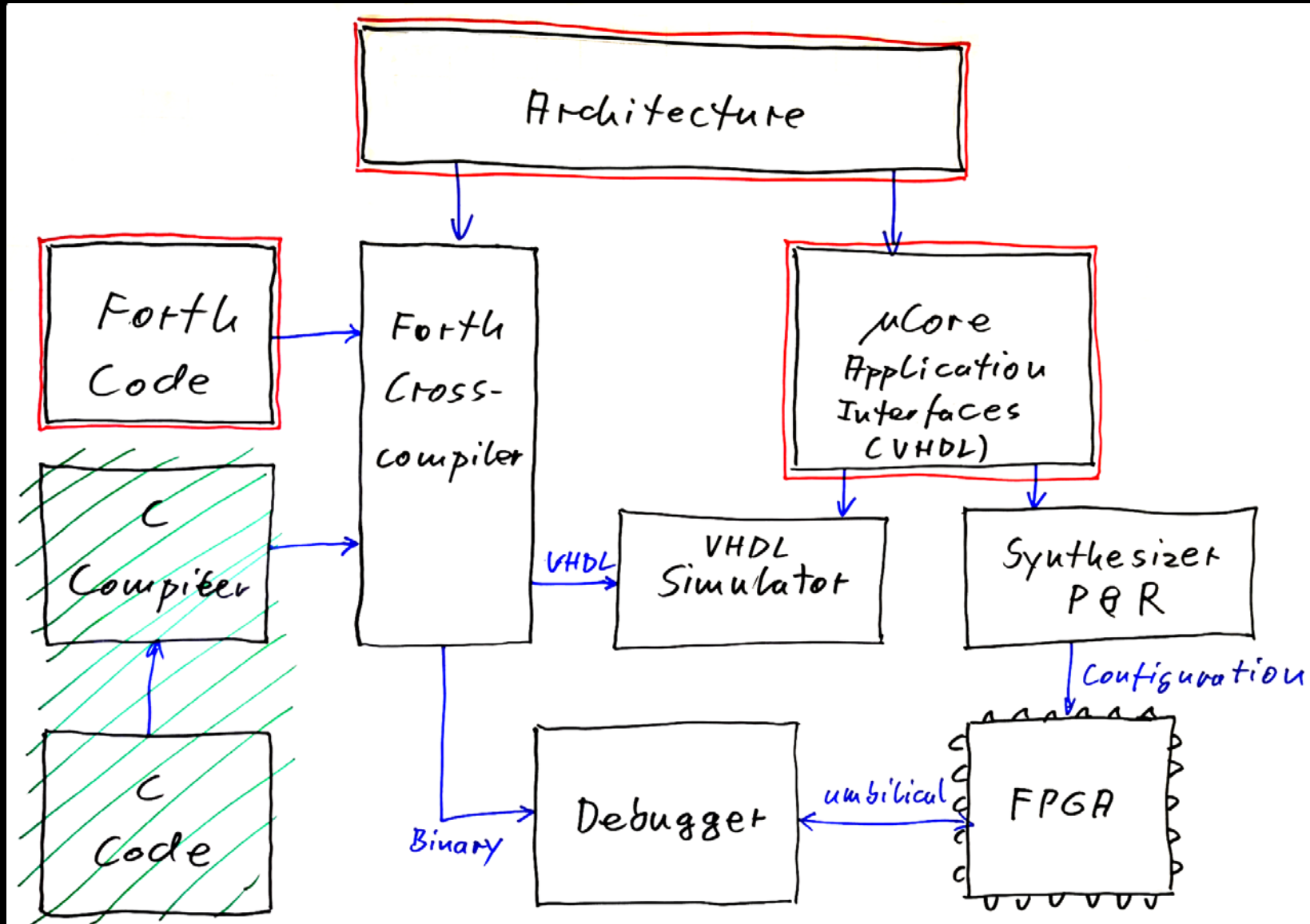
`<channel#> ADC !` \ start conversion
`ADC @ Sample !` \ fetch result

- `ADC !` will raise **pause** until the ADC is **no longer in use**.
- `ADC @` will raise **pause** until **conversion has finished**.

Hardware/Software co-design

- VHDL code for μ Core on a target system.
- μ Forth cross-compiler and debugger on a host PC. It loads on top of the `microcore/gforth_62` docker image.
- Two wire RS232 **umbilical link** between host and target.
- Interactive Forth command line interface mapped into the target.
- A disassembler.
- A single-step tracer. (Indispensable for stack checking!)
- Both μ Core itself and the μ Forth cross-compiler are based on the `architecture_pkg.vhd` file.

Design Flow



Malleable Instruction Set

- 47 'core', 24 'extended', and 5 'float' instructions.
- 42 unused opcodes for application specific instructions.
- Adding a new instruction is a 3-step procedure:
 1. Define the **binary code** for <new-op> in `architecture_pkg.vhd`.
 2. Add a WHEN clause to the instruction decoder's CASE statement for the **semantics** of <new-op> in `uCntrl.vhd`.
 3. Define a **name** for <new-op> in `opcodes.fs` to make it known to μ Forth.

Instantiations

- μ Core has been ported to Xilinx (XC2S), Lattice (XP2), Altera (EP2), and Actel/Microsemi (A3PE) FPGAs.
- Reference instantiations using an LFXP2-8:

Instruction set	word width	SLICES	data memory	program memory	maximum clock
core	16	988	6k	8k	33 MHz
extended	16	1199	6k	8k	30 MHz
core	27	1259	4k	8k	33 MHz
extended	27	1608	4k	8k	28 MHz
extended and floating point	27	1808	4k	8k	26 MHz
core	32	1432	3k	8k	33 MHz

