A SMALL HIGH PERFORMANCE MICROPROCESSOR CORE SIRIUS FOR EMBEDDED LOW POWER DESIGNS, DEMONSTRATED IN A MEDICAL MASS APPLICATION OF AN ELECTRONIC PILL (EPILLE®)

Dirk Jansen, Nidal Fawaz, Daniel Bau, Marc Durrenberger
Institut für Angewandte Forschung, Hochschule Offenburg
Badstrasse 24, 77652 Offenburg, Germany
email: d.jansen@fh-offenburg.de

Abstract: A new, small, and optimized for low power processor core named SIRIUS has been developed, simulated, synthesized to a netlist and verified. From this netlist, containing only primitives like gates and flip-flops, a mapping to an ASIC - or FPGA technology can easily be done with existing synthesizer tools, allowing very complex SOC designs with several blocks. Emulation via FPGA can be done on already simple setups and cheap hardware because of the small core size. The performance is estimated 50 MIPS on Cyclone II FPGA and about 100 MIPS on a 0.35 CMOS 5M2P technology with 4197 primitives used for the core, including a 16 x 16 multiplier. An example design of the ASIC for an electronic ePille device currently in development is shown.

Key words: Processor core, System-on-Chip, embedded systems, ASIC, electronic capsule, and medical application

1. INTRODUCTION

Small processor soft cores, well suited to mobile applications with low power consumption, made from few logic cells, and easy to program are still not easily available on the market, although there are many offers of IP cores

Jansen, D., Fawaz, N., Bau, D., Durrenberger, M, 2007, in IFIP International Federation for Information Processing, Volume 231, Embedded System Design: Topics, Techniques and Trends, eds. A. Rettberg, Zanella, M., Dömer, R., Gerstlauer, A., Rammig, F., (Boston: Springer), pp. 363–372.

from different suppliers. Soft cores shall be defined here either as a HDL description in VHDL or Verilog, or as a RTL- net list of primitive instances as there are logic gates, flip-flops and memory blocks, available in every library for ASIC/FPGA design or other common user reconfigurable technology.

The open cores website [1] offers several clone processor cores based on the well known legacy 8048/8051 8bit concept, some few on Z80 and 6800 basis and some 16bit and 32bit high level cores like MIPS. A remarkable high level core from Gaisler Research ltd. is LEON 3 [2], basically a SPARC 32 with many features and extensions. There are several IP companies in the market which are licensing processor cores, most important ARM ltd. with its ARM 7 thumb family and ARM 9xx, which are both high performance 32 bit cores. Other companies are ARC ltd, with a scalable, reconfigurable core design, and the free available OpenRISC_1200 with several implementations. Besides of the license costs, full performance is only available after a more or less expensive porting process to a target technology.

There will be more and more devices containing programmable architectures (FPGA). Now with the availability of low power devices (IGLOO-series, MAX II-series, and PolarPro-series), designers will use soft processor cores with these devices for battery powered designs too, gaining a bigger slice from the electronic IC market.

Here we report on the application of a soft processor core used in a (potential) mass application for an ePille®, an electronically controlled disguisable medical drug delivery device, which may replace in part the actual chemical drug release mechanisms used today. First dominant application will be non bloody delivery of insulin to patients with diabetes mellitus disease.

The paper is organized in the following way: Chapter 1 describes the main specification and goals of the design. Chapter 2 shows instruction set architecture and block schematics. Chapter 3 gives synthesis results and performance data. Chapter 4 reports how a concrete SOC design of an ePille was made with this soft core. Chapter 5 gives achieved performance results.

2. DESIGN GOALS

The SIRIUS core (acronym for Small Imprint RISC for Ubiquitous Systems) stands in performance somewhere between the very successful architectures of the ATMEL AVR (ATmega 8bit) , the TI MSP 430, the PicoBlaze - and well below the ARM 7/9– class of 32bit machines, the LEON (SPARC), Motorola 68xxx and other 32bit architectures (NIOS II,

MicroBlaze). Although benchmarking still has to be done, the goals are easy to summarize:

- Load-Store architecture with 16bit external data bus.
- RISC architecture with 3 stage pipeline and 1 Instruction/clock cycle and an average of 0.8 MIPS/MHz performance.
- 16bit/32bit internal bus structure with 32bit ALU and 16x16 MPY,
- Orthogonal register set of 16 registers, 12 x 16bit, 4 x 32bit, the 16bit universal registers may be combined to double registers and handled as 6 x32bit registers.
- Instruction pointer 32bit and stack pointer 32bit are part of the register set.
- Stack oriented architecture with unlimited nesting levels.
- Pointers 16 bit as well as 32bit supported.
- Multiplex bus system, separated 8bit IO bus and fast memory bus, all IO is connected via a separate 8bit bus with own address space.
- Address space 64k or 4G depending on pointer addressing,
- Vectored hardware interrupt and software interrupt (exception)
- Compact 16 bit instruction format with only three modes.
- Instruction set architecture with 56 instructions optimized for compact C-Compilation.
- Net list version made from gate primitives, able to be mapped on every existing technology without using macros.

In the basic version, a J. v. Neumann architecture with common program/data memory and 16bit bus is used. Fig 1 shows a block diagram of the core. One of the main ideas was to create a scalable architecture, which can be fitted to every application. SIRIUS can be used for basic one chip designs, containing everything, as well as for Linux based designs with addressing capability of hundreds of Megabytes, using external DRAM. The design is further optimized for setting up a virtual machine for JAVA processing. Any NOP instructions and any limitations in using instructions are avoided, allowing still assembler programming where it makes sense.

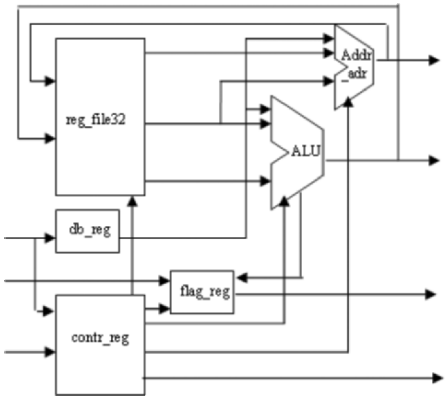

Figure 1. Block diagram of the SIRIUS soft core

The goal of the SIRIUS development is a core with similar or improved features than the existing 16bit class of processors, but with better programmability and a high efficient C-Compiler for comfortable programming. Most of all instructions run in 1 cycle with a 3 phase pipeline structure. There is a 32bit infrastructure for internal data handling and ALU processing. As a result, 32bit performance is not worse than 16bit performance, beside loads and stores need additional cycles. Instructions for 32bit and 16bit are using the same mnemonics, so there are only few codes to remember.

3. INSTRUCTION SET ARCHITECTURE

The programmer's model is shown in Fig. 2. The instruction set is a nearly direct replacement of the DAG operators, defined by Fraser and Hanson [3] with their re-targetable Little C – Compiler (LCC) [4]. ISA is a result of optimization for C-Compilation. So the compiler was defined first and the instruction set then deviated. Fig 3 shows the code space for a 6bit operation code. From 64 available different codes, 56 are actually used for instructions. The instruction set is near minimum, but with the immediate instructions added. These instructions use a constant as one argument, which makes these instructions 32/48 bit long. See Fig 4 for the instruction formats. Load/Store instructions are always based on a pointer, which is hold in a register, normally called indexed addressing. This pointer can be 16bit or 32bit, depending on attr0, the data can be 16bit or 32bit long, depending on attr1. Calls are again based on pointers, 16bit or 32bit, the return address pushed on the stack is always 32bit, allowing a large memory model used. Both addressing modes can be mixed, allowing flexible memory assignment.

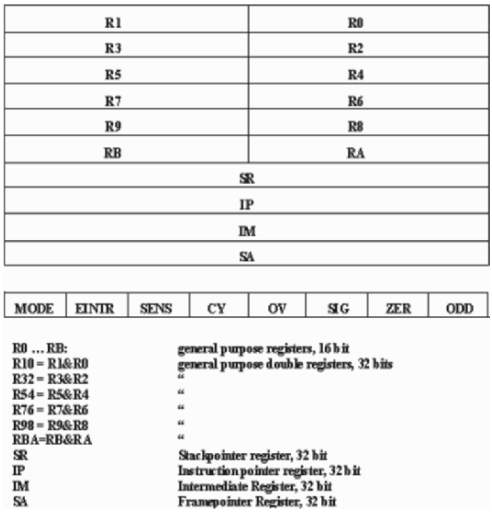

Figure 2. Programmer's Model

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	SAL	SAR	SLL	SLR	CLU	CUL	CLI	CIL	MPU	ADD	SUB	CMP	ANA	ORA	XRA
1	MOV	SALc	SARc	SLLc	SLRc	INV	INC	DEC	LXI	LDI	ADI	SBI	CPI	ANI	ORI	XRI
2	POP	PSH	PIN	POT	JMP	CAL	RET	SWI	BOV	MPS	--	--	--	--	--	--
3	LDB	STB	LDX	STX	BCY	BEQ	BGT	BGE	BNE	ITR	PSF	PPF	RST	ENI	DIS	HLT

Figure 3. Code space for 6bit operation code, RST is reset and ITR is interrupt, not usable as instructions, 6 codes are still free for extension

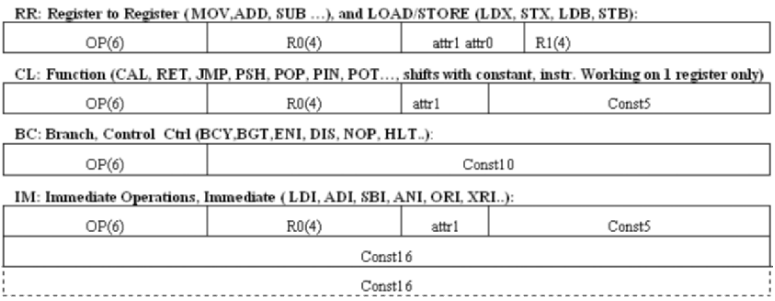

Figure 4. Instruction Formats

Branches are always relative branches in relation to the instruction pointer; the branch-offset is limited by the size of the constant to +/- 1023

bytes or 511 Instructions, normally sufficient. Branches with larger offsets have to be replaced by Jumps, which may have every size, which is done by the assembler. All these instructions are of 16bit format.

The instructions are pipelined during execution with the basic 3 stage pipeline scheme

FETCH – DECODE – EXECUTE.

Write back is included in EXEC for register to register instructions. The flip-flop based register set structure allows to read a register, to process the data in the ALU, and to write into the register in only one clock cycle. The only exception is the MPY instruction, which is pipelined itself and uses 2 EXEC cycles, and the load/store instructions, which need 3 cycles, because of the 16bit limited data bus size and the J. v. Neumann architecture. This gives a better balancing of delays and allows higher clock frequencies for register to register processing. Fig 5 shows an example for pipelining. In general, the pipeline depth is adapted to the delay of a memory read, so memory acquisition delay of the target technology mainly defines the maximal frequency of the core.

AdMM	Addr0	Addr1	Addr2	Addr3	Addr4	Addr5	Addr6	Adrdat	Addr6	Addr8	Addr9	Adrdat	Addr9		
db_in		Instr0	Instr1	Instr2	Data2	Instr4	Instr5	Instr6	Data5	Instr6	Instr8	Instr9	Data8	Instr9	
Nmoncs			MOV	ADD	LDI1	LDI2	ADD	LDX1	LDX2	LDX5	SUB	STX	STX	STX	MOV
ireg			Instr0	Instr1	Instr2	Instr2	Instr4	Instr5	Instr5	Instr5	Instr6	Instr 8	Instr8	Instr8	Instr9
db_reg			Instr0	Instr1	Instr2	Data2	Instr4	Instr6	Instr6	Data5	Instr6	Instr8	Instr9	Data8	
regs				Res0	Res1		Res2	Res4			Res5	Res6			

Figure 5. Pipelining of a program sequence, showing 1 cycle MOV, ADD, SUB, 3 cycle LDX, STX and 2 cycle LDI

4. SYNTHESIS RESULTS AND PERFORMANCE

The core was coded in VHDL and synthesized to a VERILOG net list, using the design compiler DC of Synopsys Inc. As a target technology, a selected and generalized 0.35µm technology library was used. This implicates the following advantages as seen in Fig 6:

- Timing is optimized like for an ASIC design, but without concrete mapping to an existing technology.

- The netlist can easily be remapped with DC to any CMOS target technology without significant changes and risks, e.g. to 0.35µm AMI, or 0.13µm UMC or other technologies.
- A complete timing based simulation (SDF File from Synopsys DC) with MODELSIM is possible, same as a static timing analysis.
- The netlist can be loaded to any FPGA design IDE like ALTERA-QUARTUS together with a basic (behavior)-library of the primitives used in the netlist. This allows controlling the detailed behavior via emulation. The tool normally remaps and optimizes the netlist to the used FPGA target technology. The only disadvantage is that available special cells like multipliers in the FPGA are not used; because they are dissolved to primitives (This can be overcome with a special version of the net list for FPGA synthesis).

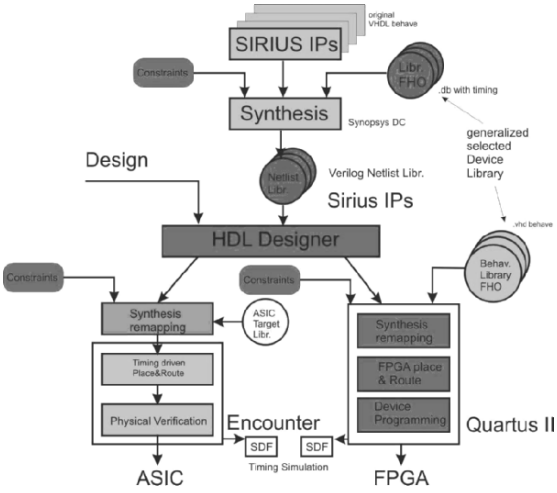

Figure 6. Netlist based IP development and dissemination

The only disadvantage in the above method is the higher effort in simulation because of the fine grain structure of the netlist, which leads to longer simulation times (not really significant on fast computers). For larger software simulations, a software instruction set simulator or the emulation on the FPGA is more effective. This is again true for driver development for new IO-blocks.

Low power consumption is based on the following methods: Only registers are clocked, which are enabled and have to be written. Circuitry, which is not used, is disabled (e.g. the multiplier is logically disconnected when not used). All busses are made in form of multiplexer architectures, no tristates used. Using a low power target library.

In normal operation mode, only a small part of the logic is active. In addition, power can be optimized by the DC, using power constrains. Real consumption can only be discussed after defining the target process, the supply voltage, clock frequency and the temperature range [5]. The HLD mode, which is entered by the hold instruction and can only be left by reset or an interrupt, keeps only very few gates active and is a natural low power mode. With the achieved low number of cells for the core, static power is in the below microampere range, which is important for power down or sleep modes.

5. PUTTING THE SIRIUS CORE IN AN EPILL APPLICATION

The first project which shall demonstrate the performance of the core is a medical application, where a disguisable drug container (ePille®) Fig 7 shall release its medicament by remote commands via a near field communication (NFC) interface.

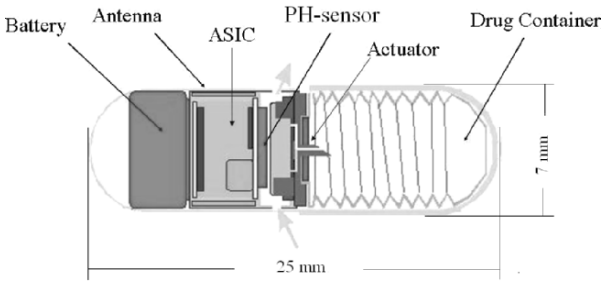

Figure 7. ePille concept of an electronic drug delivering device

Due to some additional requirements, sensor processing etc, which shall not further discussed here, a complex controller ASIC is set up, see Fig 8. The core is the heart of the circuit; all peripheries are connected via the peripheral 8bit bus. There is a parallel port as well as a serial IO, SPI and I²C Interface, a modem block for DQPSK communication [6] and a NFC block for ISO 14443a RFID communication [7], depending on the requirements. Watchdog and timer blocks are classic. Power down mode with clock switching allows together with PLL based main clock oscillator extreme low power consumption. Event handling and wake up is made by a special programmable wake up manager WUM.

Memory in this prototype is made of SRAM, which takes a lot of silicon area, much more than all other circuitry (see Fig 9). It will be replaced in a mass product by ROM or even Flash-ROM which both is much smaller. Booting is done by a small boot-ROM, synthesized from gates, containing a loader, which loads the actual software via the SPI – Interface from an external serial flash memory, which can store additional software and collected data during the usage process. Some of the above mentioned blocks may be spared in a production type of the chip. Fig 9 shows a first routing of the chip, giving an impression on the small size of the SIRIUS core. The routing does not contain all of the analogue blocks, which are partly still under development. The ASIC is designed in 0.35µm CMOS from AMI with 5 metal and 2 poly-layers and has an area of about 11 mm^2, mostly SRAM. The SIRIUS core alone is below 1 mm^2 in size.

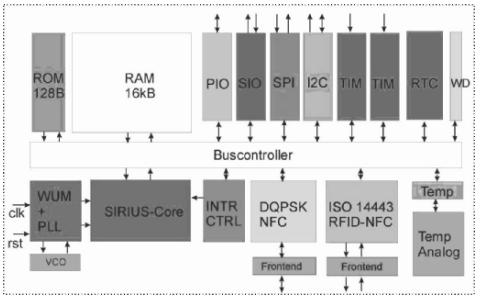

Figure 8. Block diagram of the ePille-controller with SIRIUS processor core

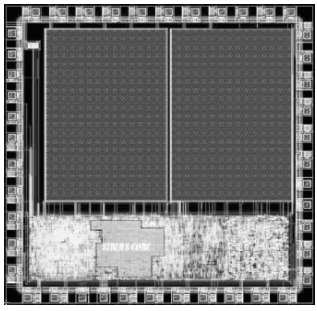

Figure 9. Prototype ePille- SOC-ASIC layout in 0.35 AMI CMOS technology, (not complete), Size is 11mm^2

6. RESULTS

A new, small and for low power optimized processor core SIRIUS has been developed, simulated, synthesized to a netlist and verified via FPGA emulation. The basic concept was already presented in 2005 [8]. Table 1 summarizes the achieved results in synthesis for a 0.35µm AMI library as well as for the low entry FPGA Cyclone II device. Table 2 shows estimated performance from the actual available data, measured results will be presented on the conference.

Table 1. Results of synthesizing for 0.35 ASIC Library and for ALTERA Cyclone II

Block	Prim. (ASIC)	LE (FPGA)
SIRIUS core	4197	2870
ROM	392	106
Bus controller	92	335
PIO	52	20

Block	Prim. (ASIC)	LE (FPGA)
SIO	388	132
SPI	241	123
I2C	343	191
Watchdog	102	50
Intr. Contrl.	773	346
ISO-14443 Interf.	1407	617
QPSK Interface	3186	1252
Timer	214	88

Table 2. Estimated performance data of a SIRIUS based SOC

Performance	Value	Comment
MIPS average/MHz	0.8	Depending on J. v. Neumann architecture and 16b ext. Bus
Max clock frequency (ASIC)	120 MHz	For 0.35 CMOS technology and 3.3 V supply
Max clock frequency FPGA	60 MHz	For Altera Cyclone II and optimized placement
Power in HOLD mode	<20uW	Main clock running, all periphery shut off, ASIC
Power in P D mode	<3 µW	Only RTC running, main clock off
Code density	16bit	High, All instructions 16 bit, near to C-statements
Architecture	RISC	J.v. Neumann, pipelined execution, 3 stage pipeline
Address space	64k / 4G	Depending on pointer size used, both hardware supported
OS feasible	LINUX	Together with external DRAM and large memory model
Registers	12 + 4	16 bit registers can be combined to 32 bit registers
ALU	32 bit	Instructions use 16/32 bit ALU
MPY	16 x 16	Signed/unsigned Multiply in Hardware, 2 cycles
Interrupt	128	Vectored Interrupt with interrupt controller, periphery unit
Instruction set	56	Instructions mostly identical for 16 bit and for 32 bits
High level Language support	LCC	C-Compiler and IDE available, JVM

REFERENCES

[1] Opencores: http://www.opencores.org/
[2] LEON3: http://www.gaisler.com/
[3] Fraser, Christoph and Handson, David: A Retargetable C Compiler: Design and Implementation. Addison Wesley P.C., Redwood City, Ca ISBN 0-8053-1670-1, 1995
[4] LCC Compiler: http://www.cs.princeton.edu/software/lcc/
[5] Jansen, Dirk ed., et. alt.: Electronic Design Automation Handbook, Verlag Kluwer, NL, 2003.
[6] Jansen, Dirk, Fawaz, N.: DQPSK Modulator for Inductive Data Transmission, MPC-Workshop, Reutlingen, June 2002.
[7] Jansen, Dirk, Baier, F.: Induktive bidirektionale Schnittstelle ähnlich ISO/IEC 14443-A, MPC-Workshop, Reutlingen, June 2002.
[8] Jansen, Dirk: Systematic Design of a Small Processor Core with C-Capability for SOC Designs; Presentation at the CECS, University of California, Irvine on July 9, 2005.