VHDL Synthesis of XZ8 – an 8-bit RISC processor

Eng., M.Sc. Seraphim Dimitrov Tabakov – TU Sofia
George Ivanov Radulov – TU Sofia
Eng., M.Sc. Rossen Ivanov Radonov – TU Sofia
radulov@ecad.vmei.acad.bg

ECAD Laboratory is a leading educational center in Technical University of Sofia for digital, analog and mixed-signal ICs. The Laboratory is specialized in testing new software solutions, developing new design flows and providing different instruction courses that prepare our students for the real-life problems in ASIC and FPGA domain.

One of the latest success stories in ECAD Laboratory is the XZ8 - an 8-bit RISC processor developed exclusively for educational goals. It has very light and understandable hardware structure, but in the same time it is a RISC processor and uses a pure Harvard architecture. Its reduced, in 18 instructions, instruction set makes the XZ8 very convenient for research and a starting point in the area of professional micro Controllers, micro Processors and DSPs. Although the chip was developed for education, it is a general-purpose processor and can be used in many different applications.

VHDL was an integral part of the design methodology we used to create the ZX8. Front-end design and behavioral-level simulation were done with VHDL. Multiple test benches were developed with VHDL. After the RTL description was synthesized, the back end of the design process was finished through XILINX implementation using XNF gate-level schematic regeneration. The chip had difficult design goals; we needed to create a real solution within the university conditions; we needed to clarify and structure the design flow for educational purposes; we had to create a real understandable RISC with a real Harvard architecture.

VHDL was primary chosen because it provides a rich set of semantic constructs, and it allows fast exploration of alternate architectures. Some of the semantic features we use include procedures, user defined types, packages (for data/types/subprogram sharing), constants and so on. VHDL has powerful looping capabilities, superior time resolution (femto-second accuracy), fast behavioral simulation run times, and allows design reuse through the creation of our own libraries. Finally, VHDL is unmatched in the benefits it brings to test bench development. This is significant because a lot of effort goes into generation of the test bench to verify design functionality and to create test vectors.

After the system and performance requirements were determined, the system was partitioned into several functional blocks using Synopsys

Graphical Environment (SGE) Schematics. When drawn properly, schematics provide a graphical representation of the design hierarchy in a manner that allows the functional data flow to be visible. A nice feature of the Synopsys tool kit is that after the block diagram is drawn and interconnect, it then generates a RTL VHDL file and for all functional blocks it generates VHDL skeleton files that must be filled with the particular behavior descriptions.

When writing hardware descriptions using any HDL, careful attention must be paid to the coding structure and semantic constructs that are used. Although VHDL provides target device independence through the use of synthesis, an optimized RTL description must take into account the specific synthesizer being used.

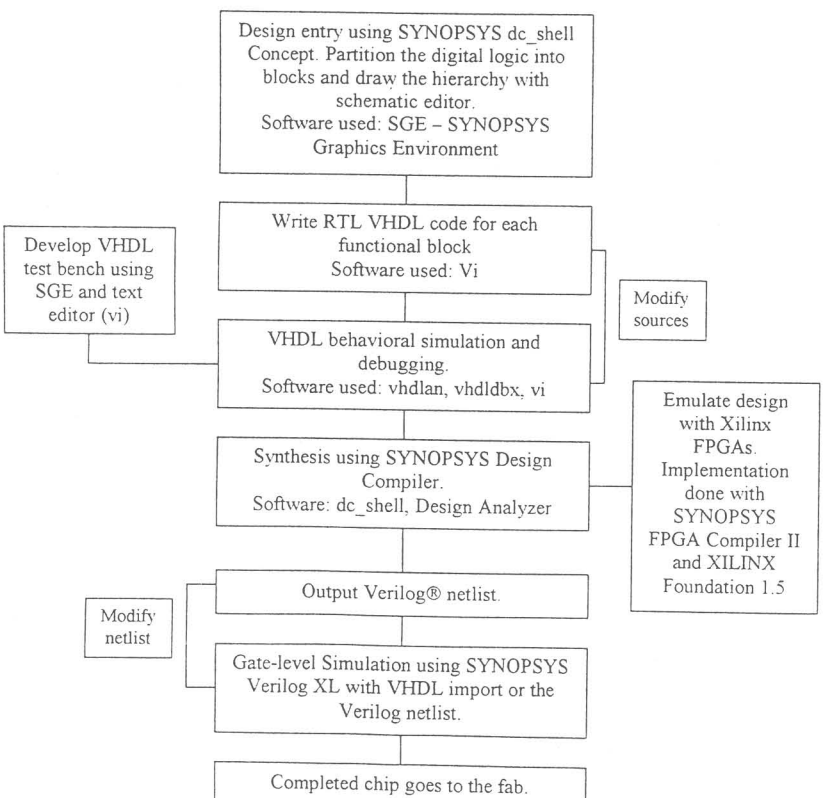

Figure 1.

In ECAD Laboratory Synopsys 1999.10 and XILINX foundation 1.5 are used for digital design. The flow for ASIC and FPGA design is presented at Figure 1.

XZ8 is a simple 8-bit processor and its VHDL representation is readable and understandable for students that makes their first steps in the field of digital design. All VHDL files are absolutely free available for educational goals at ftp://ecad.vmei.acad.bg/pub/ZX8. Its block diagram, representing the files relations is shown at Figure 2.

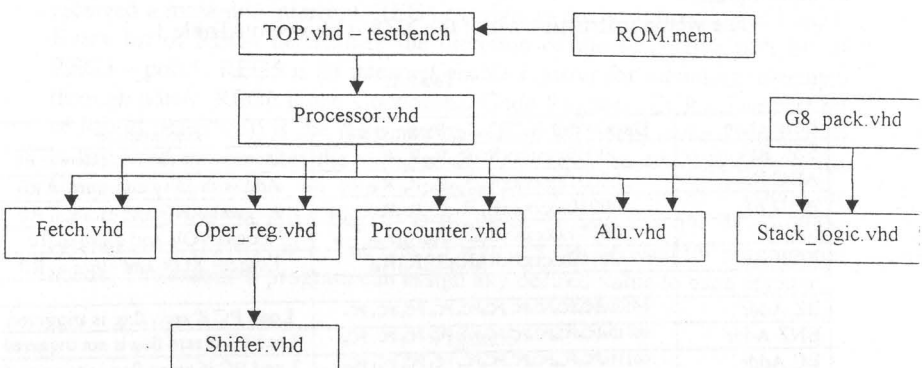

Figure 2.

The XZ8 is partitioned into five major blocks. The first block FETCH is responsible for the four phases that control the performance. One processor cycle consists of four clock pulses, i.e. the external clock generator must have four times greater frequency than the processor. Each of the rest four blocks: OPER_REG, PROCOUNTER, ALU and STACK_LOGIC is responsible for a specific type of instruction, i.e. in the OPER_REG is performed the instructions concerning the register block, PROCOUNTER – instructions concerning the 10-bit program counter, ALU – instructions concerning the arithmetic and logic instructions, STACK_LOGIC – instructions concerning the stack. The OPER_REG functional block has a sub-block included – a shift register.

The XZ8 has one input/output 8-bit port, which is connected directly to the register 3 (REG3) of the register block. The XZ8 works with 16-bits data bus, where the OP Code is stationed in the five most significant bits, excluding the two MOVE instructions whose OP Codes are two bits and the MSB is always 1. The address bus is 10-bits and it allows addressing of

1024 lines external ROM program. All jump instructions are performed in one processor cycle. There is a four level 10-bit stack for program counter storing during interrupts, in STACK_LOGIC block. This allows the program to handle deep up to 4 interrupts. There are three interrupts. The first and the major one is the external non-maskable interrupt. The second interrupt is also hardware and it is received via the port after some conditions are met (conditions such as port direction configuration, the mask flag of the CCR, and the triggered interrupt enable bits). This interrupt is maskable. The last interrupt is software interrupt and can be interpreted as calling a subprogram.

The whole instruction set of the XZ8 is shown on Table 1.

Table 1.

Instruction	Data Word (OP Code, Parameters)	Explanation
SBC Ri	00000xxxxxx$R_{i4}R_{i3}R_{i2}R_{i1}R_{i0}$	Subtract with carry and store in R0
ADC Ri	00001xxxxxx$R_{i4}R_{i3}R_{i2}R_{i1}R_{i0}$	Add with carry and store in R0
AND Ri	00010xxxxxx$R_{i4}R_{i3}R_{i2}R_{i1}R_{i0}$	Logical AND and store in R0
OR Ri	00011xxxxxx$R_{i4}R_{i3}R_{i2}R_{i1}R_{i0}$	Logical OR and store in R0
XOR Ri	00100xxxxxx$R_{i4}R_{i3}R_{i2}R_{i1}R_{i0}$	Logical XOR and store in R0
BZ Addr	00101x$PC_9PC_8PC_7PC_6PC_5PC_4$ PC_3 PC_2 PC_1 PC_0	Load PC if zero flag is triggered
BNZ Addr	00110x$PC_9PC_8PC_7PC_6PC_5PC_4$ PC_3 PC_2 PC_1 PC_0	Load PC if zero flag is not triggered
BC Addr	00111x$PC_9PC_8PC_7PC_6PC_5PC_4$ PC_3 PC_2 PC_1 PC_0	Load PC if carry flag is triggered
BNC Addr	01000x$PC_9PC_8PC_7PC_6PC_5PC_4$ PC_3 PC_2 PC_1 PC_0	Load PC if carry flag is not triggered
BN Addr	01111x$PC_9PC_8PC_7PC_6PC_5PC_4$ PC_3 PC_2 PC_1 PC_0	Load PC if negative flag is triggered
BNN Addr	01001x$PC_9PC_8PC_7PC_6PC_5PC_4$ PC_3 PC_2 PC_1 PC_0	Load PC if negative flag is not triggered
SHL Ri	01010xxxxxx$R_{i4}R_{i3}R_{i2}R_{i1}R_{i0}$	Shift one left through carry bit
SHR Ri	01011xxxxxx$R_{i4}R_{i3}R_{i2}R_{i1}R_{i0}$	Shift one right through carry bit
RTS	01100xxxxxxxxxx	Load PC with last value in the stack
CALL Addr	01101x$PC_9PC_8PC_7PC_6PC_5PC_4$ PC_3 PC_2 PC_1 PC_0	Load directly PC
STOP	01110xxxxxxxxxx	STOP all process until interrupt
MOV Ri, Const	11$R_{i4}R_{i3}R_{i2}R_{i1}R_{i0}xC_7C_6C_5C_4C_3C_2C_1C_0$	Load a Constant in a register
MOV Ri, Rj	11$R_{i4}R_{i3}R_{i2}R_{i1}R_{i0}$xxxx$R_{j4}R_{j3}R_{j2}R_{j1}R_{j0}$	Copy a register into register

X – Don't care
R – Register data
PC – Program counter data
C – Constant

The register block is organized in 32 8-bit registers. All of them are accessible for the programmer, although the first seven registers are system and the rest 25 are general-purpose registers. Register 0 (REG0) is the accumulator. It is always one of the parameters for the ALU instructions and the result from them is stored in it. REG1 is a pointer address register. It stores in its 5 least significant bits the address of the register, to which the pointer points. REG2 is the "peek" value of REG1 (*REG1), i.e. REG2 has always the same value as the register whose address is stored in REG1. REG3 is the input/output port. In some circumstances through REG3 can be received a maskable interrupt. REG4 configures the port direction of REG3. Every bit of REG4 determines the direction of the respective port bit of REG3 – portA. REG5 is an interrupt enable register for incoming interrupts through portA. REG6 is the Conditional Code Register (CCR). The first bit of REG6 (REG6(0)) is the mask interrupt flag, REG6(1) is the Zero flag (it is triggered in case that the value of REG is equal to 0), REG6(2) is the Carry bit. The Carry bit can be triggered by ALU or shift operation. REG6(3) is the Negative flag. It is triggered in case that the MSB of the Accumulator (REG1) is 1. The rest 25 registers can be used for programmer needs. The executed program can assign any desired value to each register.

Figure 3.

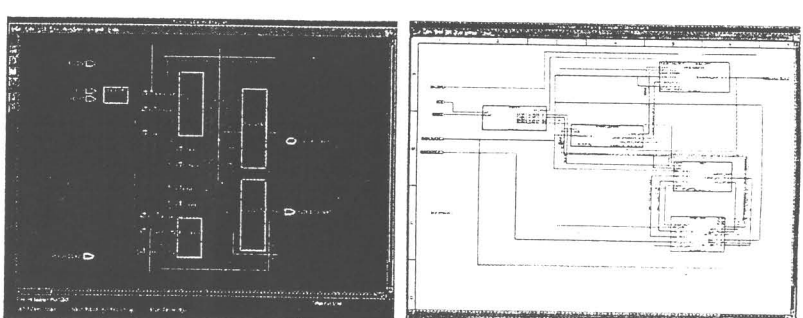

Figure 3 shows the Design Analyzer window with the synthesized processor and the SGE schematic block-diagram. These snapshots demonstrate the end and the beginning of the design flow.

At the stage of Design Analyzer, when the XZ8 is already synthesized, the project is saved in XNF gate-level format and is imported in XILINX Foundation 1.5. From the XNF is generated a schematic and the

project is programmed in XILINX FPGA within the XILINX Foundation 1.5 environment.

There are 18 instructions; every instruction is performed for one processor cycle, i.e. for 4 external clocks; this means that there are maximum 72 (18x4=72) internal states.

This processor will help our students to understand the basics of digital design, processor architectures and VHDL source writing. It is just a core. Additional units such as ADC, SPI, RS232, etc., can be added as VHDL sources. Upgrades are always welcomed and will be appreciated on our behalf. The key for its success, we believe, is in its non-commercial open source: ftp://ecad.vmei.acad.bg/pub/ZX8.

Literature:

1. Charles. H. Roth, Jr., Digital System Design using VHDL, 1998.
2. Douglas L. Perry, VHDL - third edition, 1998.
3. SOLD 1999.10, Synopsys Documentation.
4. XILINX Foundation Series On-line Help System.