Fourth LACCEI International Latin American and Caribbean Conference for Engineering and Technology (LACCET'2006)
"Breaking Frontiers and Barriers in Engineering: Education, Research and Practice"
21-23 June 2006, Mayagüez, Puerto Rico

Functional Microcontroller Design and Implementation

Vivekananda Jayaram, Ph.D. Candidate
Research Associate, Florida International University, Miami, Florida, U.S.A., vjaya002@fiu.edu

Subbarao Wunnava, Ph.D., P.E.
Professor, Florida International University, Miami, Florida, U.S.A., subbarao@fiu.edu

Abstract

In many situations, hardware description languages (HDL) such as VHDL, Verilog or SystemC is used to develop the functionality of the digital system, while the timing and control signal generation is either neglected or ignored. The authors have used a methodology wherein a hardware structure was conceptually laid out of the digital system under consideration. The system development started with top-down planning approach and the blocks were designed using bottom-up implementation. The programs were written, simulated and synthesized using Electronic Data Automation (EDA) tools such as ModelSim and Leonardo Spectrum. Instruction set such as transfer, arithmetic, logic, input, output and control instructions were implemented. This approach guaranteed the integrity of the system realization with proper timing and data flow, without the invisible ghost states. In this article, the authors have presented the design methodology of such a multi purpose microcontroller, and provided the functional HDL code, simulation and synthesis results. Also, the authors have presented the sequence in which this microcontroller can be made a general purpose controller unit which can be used by other system designs.

Keywords

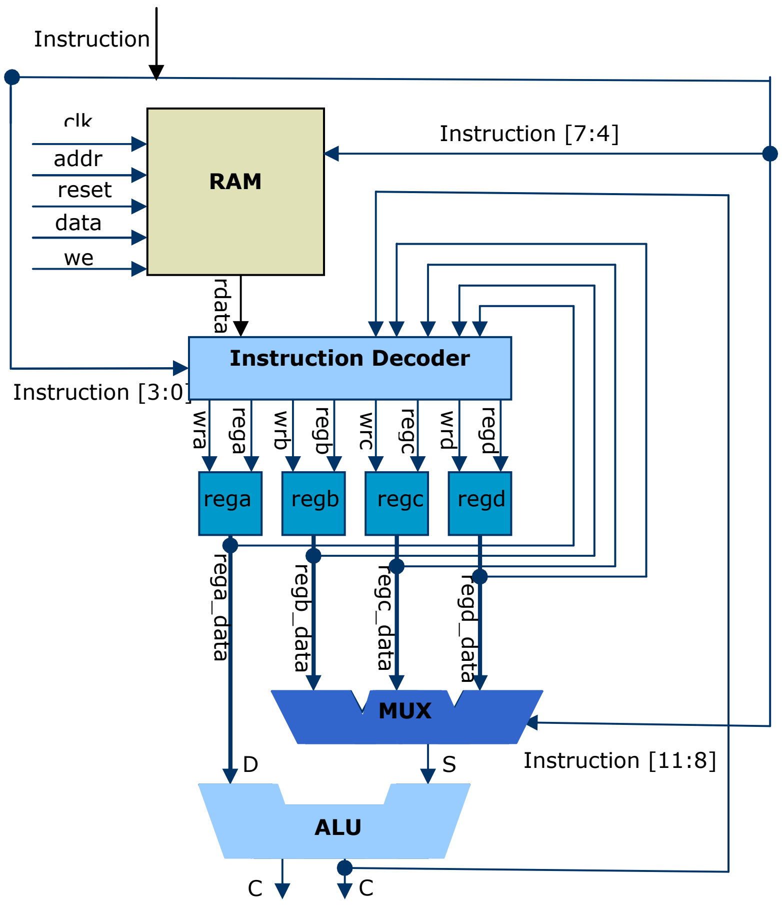
VLSI, VHDL, Microcontroller, Simulation, Synthesis

1. Introduction

In any control and controller system applications, microcontroller is an important module, which provides the control, timing and status signals, in any complex digital system realizations. (Bartbel, 1997) A microprocessor is usually defined as "a single chip that contains control logic and data processing logic, so that it can execute instructions listed in a program to operate on some data". Microcontrollers are nothing but microprocessors with on-chip memory. Whether using ASIC, FPGA or CPLD based realizations, it is essential to incorporate the microcontroller module, as an integral part of the system. Functional microcontroller has been developed using VHDL coding using structural design of logic blocks which generates control and timing signals used for the data processing operation.

2. Methodology

While designing a microcontroller module, it is important to determine the number data bits, which can be processed by the microcontroller in one cycle. In the process of designing, the instruction set, instruction width, internal register set, type of control unit, flags, amount of memory to be used has to be laid out. (Gloria, 1999) VLSI implementation point of view, the most relevant factor that intervenes in the decision of implementing a certain instruction lies on the number of operands needed to specify the instruction. This is not an absolute factor, it depends on the number of registers, and therefore on the number of bits required for coding a register reference. Present day VLSI technology has lead to the design and development of millions of gates on a chip. Hardware designers create several VLSI modules for their research and development purposes. It is often important to re-use these modules to reduce product development time, thereby minimizing the time to market. Therefore, it is important to design hardware in a modular fashion, so that these modules can be included in the development of a complex system. The design and development of such a modular design microcontroller helps other designers to incorporate



this module with minimal or no modifications to the hardware module. The block diagram of such a 4-bit microcontroller is as shown in fig.1.

Figure 1: 4-bit Slice Microcontroller Block Diagram

3. Module Development

In this study, 4-bit microcontroller was designed using a modular approach. Using top-down approach, the elements of the microcontroller were identified as basic registers, instruction decoders, ALU, RAM, Control and timing Unit. (Zabawa and Wunnava, 2004) propose the design of a microcontroller unit with the following building blocks:

- A 16-word by four bit port RAM.
- Four registers (RegA, RegB, RegC, RegD)

- An ALU selector which select two inputs. The D (Destination Input) will always be rega_data. The S (Source Input) will be either regb_data, regc_data, regd_data based on the selected output from the MUX.
- The MUX module used instruction [11:8] as the select input. If select is ‘0010’ then regb_data is set to signal S (Source Input). If select is ‘0100’ then regc_data is set to signal S. If Select is ‘1000’ then regd_data is set to signal S.
- A 4-bit ALU capable of doing arithmetic, logical, and bitwise functions on the selected source and destination words.
- An instruction decoder is used to decide whether to load the ALU output into RegA or whether to load the read data from the RAM. It also load RegB, RegC, and RegD from memory if the MOVB, MOVC, or MOVD operations occurs.

Table 1: Input and Output Signals

Port	Type	Bit Width	Description
Reset	In	1	Reset signal
Clk	In	1	Clock signal
Instruction	In	12	Instruction word
Addr	In	4	Address input to RAM
Data	In	4	Data input to RAM
We	In	1	Write/Read Enable for RAM
C	Out	4	Data output from chip
C4	Out	1	Carry output from ALU

Table 2: Instruction set

Field position	Description
instruction[3:0]	Opcode Select: ‘0000’: ADD ‘0001’: SUB ‘0010’: CMP ‘0011’: SHL ‘0100’: SHR ‘0101’: ROL ‘0110’: ROR ‘0111’: AND ‘1000’: OR ‘1001’: NOT ‘1010’: MOVA ‘1011’: MOVB ‘1100’: MOVC ‘1101’: MOVD
instruction[7:4]	Address
instruction[11:8]	Control Destination Selector for Mux Module to select input source (S) to ALU ‘0001’: Write to Register A ‘0010’: Write to Register B ‘0100’: Write to Register C ‘1000’: Write to Register D

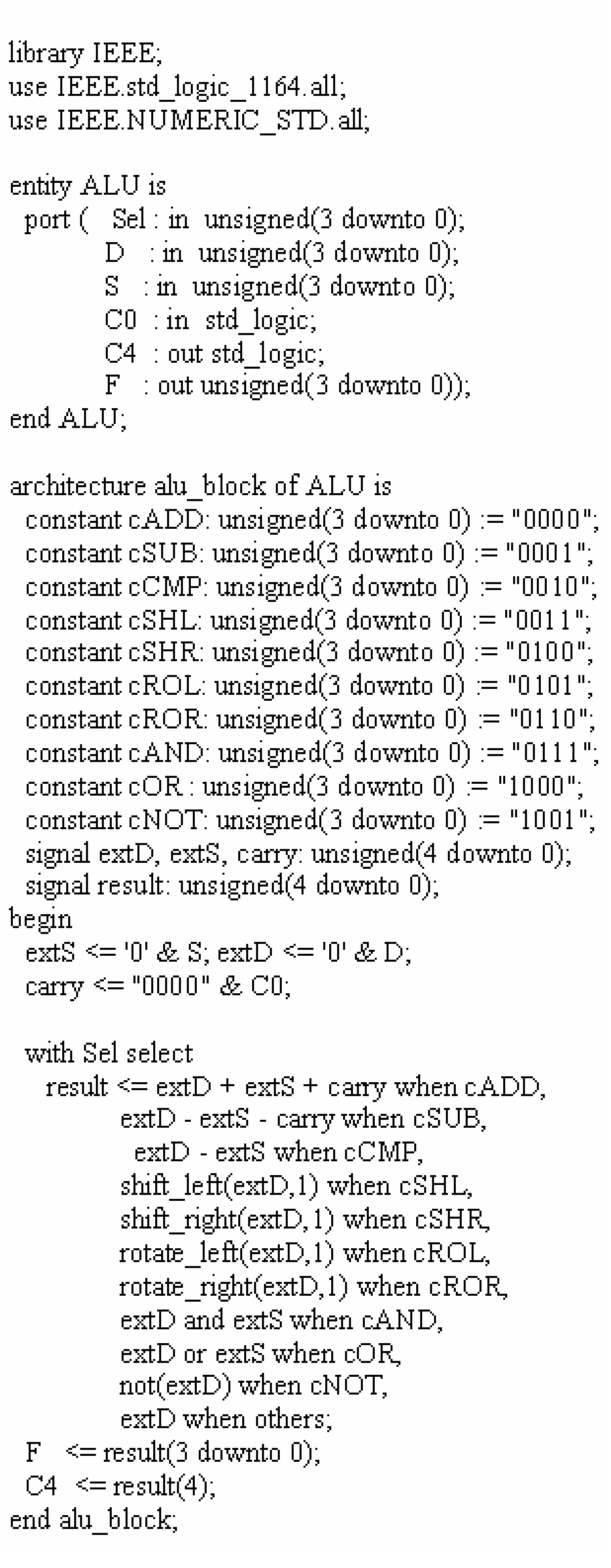
The implementation was done using a bottom-up approach. The basic hardware blocks like adders, flip flops, shift registers, counters, and comparators were designed. Later, these blocks were used to form ALU, RAM, and instruction decoder. Finally, in the top level module, these blocks were connected to form a functional microcontroller. The top level module has the I/O pins as shown in Table 1. The instruction set has a 12-bit instruction width, which has three 3-bit fields whose functions are as shown in Table 2. The ALU performs arithmetic, logic and shift operations as defined in Table 2. In all, 14 instructions can be performed by this ALU.

Table 2: Functions of ALU

Instruction Type	**Bit Field instruction[3:0]**	**ALU Function A, B, C, D are the register**
ADD	0000	A ← A + B
SUB	0001	A ← A - B
CMP	0010	A-B (set flag - Do not update A)
SHL	0011	A ← A shift left 1
SHR	0100	A ← A shift right 1
ROL	0101	A ← A rotate left 1
ROR	0110	A ← A rotate right 1
AND	0111	A ← A and B
OR	1000	A ← A or B
NOT	1001	A ← not A
MOVA	1010	A ← mem[addr]
MOVB	1011	B ← mem[addr]
MOVC	1100	C ← mem[addr]
MOVD	1101	D ← mem[addr]

4. VHDL Programming, Simulation and Synthesis

Programs for each building block of this system were written in VHDL. Mentor Graphics simulation tool ModelSim was used for writing the code, simulating the programs and to test its behavior. VHDL can be very difficult language to learn by reading the VHDL language Reference Manual (also called the LRM-Language Reference Manual) but should be learnt enough to start coding basic hardware blocks. The hardware programming was done in a structural fashion so that the modules can be easily used as the library for complex hardware development. The VHDL program for ALU is shown in fig 2. (Sosnowski, 2006) Developing test procedures for micro architectural level blocks is a cumbersome process. Appropriate test stimuli need sophisticated sequences of instructions. The programs were tested using test benches written using TEXTIO format, which enables us to write multiple test vectors for extensive test scenarios. The generated test benches were able to read in any test vectors from the test file and will act as stimuli to the designed hardware. The outputs thus obtained from these test benches were compared and verified for functional accuracy. (Lloyd et al, 1999) After the design has been developed and validated it becomes possible, through the use of VHDL descriptions, to employ field-programmable gate array logic that will allow a custom device to be rapidly prototyped and tested. Leonardo Spectrum Synthesis tool was used to synthesize the microcontroller module by choosing A1240XLPG132 from Actel family as the FPGA target device.



```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.NUMERIC_STD.all;

entity ALU is
  port (   Sel : in  unsigned(3 downto 0);
           D   : in  unsigned(3 downto 0);
           S   : in  unsigned(3 downto 0);
           C0  : in  std_logic;
           C4  : out std_logic;
           F   : out unsigned(3 downto 0));
end ALU;

architecture alu_block of ALU is
  constant cADD: unsigned(3 downto 0) := "0000";
  constant cSUB: unsigned(3 downto 0) := "0001";
  constant cCMP: unsigned(3 downto 0) := "0010";
  constant cSHL: unsigned(3 downto 0) := "0011";
  constant cSHR: unsigned(3 downto 0) := "0100";
  constant cROL: unsigned(3 downto 0) := "0101";
  constant cROR: unsigned(3 downto 0) := "0110";
  constant cAND: unsigned(3 downto 0) := "0111";
  constant cOR : unsigned(3 downto 0) := "1000";
  constant cNOT: unsigned(3 downto 0) := "1001";
  signal extD, extS, carry: unsigned(4 downto 0);
  signal result: unsigned(4 downto 0);
begin
  extS <= '0' & S; extD <= '0' & D;
  carry <= "0000" & C0;

  with Sel select
     result <= extD + extS + carry when cADD,
               extD - extS - carry when cSUB,
                 extD - extS when cCMP,
               shift_left(extD,1) when cSHL,
               shift_right(extD,1) when cSHR,
               rotate_left(extD,1) when cROL,
               rotate_right(extD,1) when cROR,
               extD and extS when cAND,
               extD or extS when cOR,
               not(extD) when cNOT,
               extD when others;
  F   <= result(3 downto 0);
  C4  <= result(4);
end alu_block;
```

Figure 2: VHDL implementation of ALU

After synthesizing the module the Register Transfer Level (RTL) schematic, Technology schematics and Critical Path schematic can be seen and compared to the original design. The synthesized RTL schematic of the top level of microcontroller module and the instruction decoder is as shown in fig 3 and fig 4 respectively. (Flynn, 1999) The most successful microprocessor implementations depend not simply on the use of the current state of the art in hardware algorithms, but more importantly in bringing together the knowledge of these algorithms together with projected advances in the technology and user state of the art. Therefore, the designer should not only depend on the technology to develop faster hardware modules but also need to find out ways to improve speed by minimizing redundant blocks in the design.

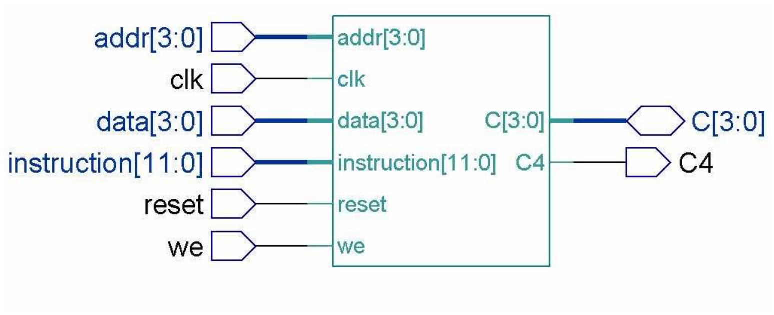


Figure 3: Top level entity of Microcontroller

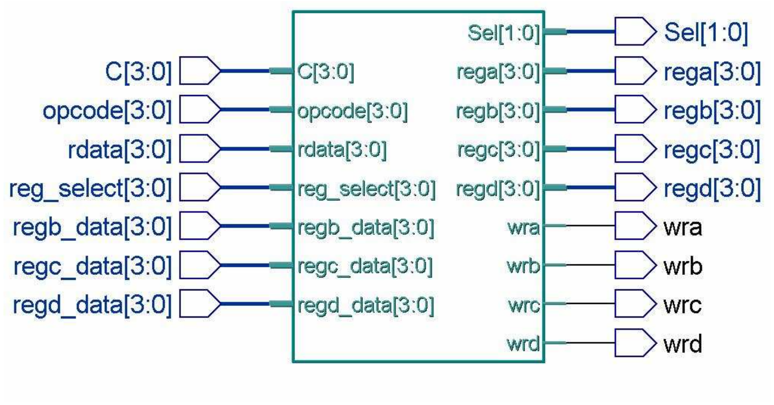


Figure 4: Top level entity of Instruction decoder

5. Results

The VHDL program was simulated in ModelSim using TEXTIO test benches with extensive test vectors. The simulation of instruction phase OR and NOT are shown in the fig 5. The device utilization of the target FPGA device A1240XLPG132, from Actel family is as shown in Table 3. The synthesized microcontroller module has 29 pins for address, data, I/O and for other signals. The Leonardo spectrum synthesis report shows 443 accumulated instances with 24 sequential and 280 combinational modules with the speed of operation of 14.47 MHz.

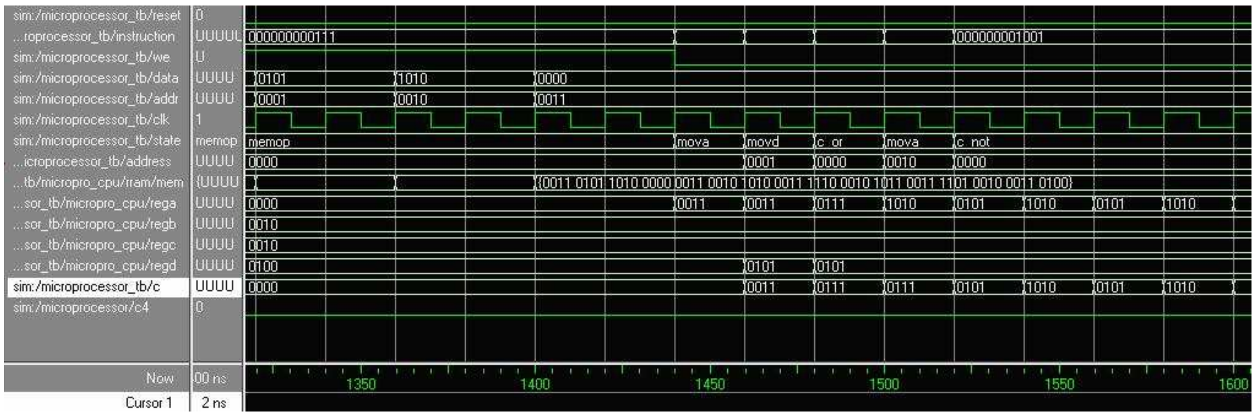

sim:/microprocessor_tb/reset
...roprocessor_tb/instruction
sim:/microprocessor_tb/we
sim:/microprocessor_tb/data
sim:/microprocessor_tb/addr
sim:/microprocessor_tb/clk
sim:/microprocessor_tb/state
...icroprocessor_tb/address
...tb/micropro_cpu/rram/mem
...sor_tb/micropro_cpu/rega
...sor_tb/micropro_cpu/regb
...sor_tb/micropro_cpu/regc
...sor_tb/micropro_cpu/regd
sim:/microprocessor_tb/c
sim:/microprocessor/c4
Now
Cursor 1
1350
1400
1450
1500
1550
1600

Figure 5: Simulation results for microcontroller instruction phase OR, NOT

Table 3: Device Utilization for Actel-FPGA: A1240XLPG132

Resource	Used	Available	% Utilization
I/O's	29	104	27.88
Modules	300	684	43.86
Sequential modules	24	348	6.9
Combinational modules	280	336	83.33

6. Conclusion

In this paper, the design and the development of a basic 4-bit microcontroller has been discussed. Functional verification shows the accurate results for instruction set Add, Subtract, compare, Shift-left, Shift-Right, Rotate-left, Rotate-right, AND, OR, NOT operations. The developed microcontroller module functions with simple control signals. The developed module is functionally built using structural approach in VHDL. Therefore, this microcontroller can be used as a module in complex board level designs. Also, because of the modular design and bottom-up implementation of this microcontroller, the VHDL code can easily be expanded to develop higher order microcontroller without making extensive changes. This VHDL program will be compatible with any other VHDL system that is compliant with either IEEE Standard 1076-1987 or 1076-1993. The speed of the design is limited to the target FPGA technology and can only be improved by using ASIC design methodology. Also ASIC design involves more resources and expensive development costs, and hence there needs to be a trade off of speed and economy.

7. References:

Actel FPGA (2006) http://www.actel.com (04/15/2006)

Bartbel, D (1997). “Architecture for microprocessors and DSPs”. *Microelectronic Engineering*, pp 255-262.

Flynn, M.J (1999). “Basic issues in microprocessor architecture”. *Journal of Systems Architecture*, pp 939-948

Gloria, A.D (1999). “Microprocessor design for embedded system”. *Journal of Systems Architecture*, pp 1139-1149.

IEEE VHDL Language Reference Manual. (2006). http://standards.ieee.org/reading/ieee/std_public/description/dasc/1076-1993_desc.html (04/15/2006)

Lloyd, L., Heron, k., Koelmans, A.M., and Yakovlev, A.V. (1999), “Asynchronous microprocessors: From high level model to FPGA Implementation”. *Journal of Systems Architecture*, pp 975-1000

Mentor Graphics, ModelSim, Leonardo Spectrum (2006). http://www.mentor.com (04/15/2006)

Sosnowski, J (2006). “Software-based self-testing of microprocessors”. *Journal of Systems Architecture*, pp 257-271

Zabawa, M. and Wunnava, S. (2004). “Efficient Digital System Design Methodology with SystemC Register Transfer Level Modeling”. *IEEE Southeastcon*. pp 395-399

Authorization and Disclaimer