LC-3B Microprocessor Design

Group:
Sagun Pai 12D070008
Saiprasad Koturwar 12D070025
Vivek Sangwan 12D070033

Problem Statement:
To design a LC-3B microprocessor with given design specifications(mentioned in ISA). Design an optimal datapath (optimal number of registers, functional units, and point-to-point communication based steering logic), and finite state machine based controller. Use VHDL or Verilog as hardware description language to specify your design. The design must be implemented on FPGA. Write a synthesizable Verilog/VHDL code so that it can be synthesized for FPGA.

Introduction:
The LC-3B specifies a word size of 16 bits for its registers and uses a 16-bit addressable memory with a 216-location address space. The register file contains eight registers, referred to by number as R0 through R7. All of the registers are general-purpose in that they may be freely used by any of the instructions that can write to the register file, but in some contexts some of the registers are used for special purposes. Instructions are 16 bits wide and have 4-bit opcodes. Our design implements instructions for fourteen of the sixteen possible opcodes, though some instructions have more than one mode of operation. Individual instructions' execution is regulated by a state machine implemented with a control unit and dataflow unit. The LC-3B instructions which can be broadly classified as following:-

ADD, AND, XOR, NOT, LSHF, RSHFA, RSHFL (Based on ALU)
BR, JMP, RET, JSR, JSRR, LEA (Based on PC manipulation)
LDB, LDW, STB, STW (Based on Memory handling)
More about these instructions can be found in the LC-3B ISA.

The LC-3B microprocessor could be split up into a number of major components

The overall task was broken into a number of weekly sub-tasks:
Design of ALU and Register file
Design and control of Memory (including Testbench)
Design of remaining components in Datapath
Design and testing of Control Path
System integration and Modelsim simulation
Final implementation on FPGA and verification of results using Signal Tap tool

Datapath design:
Based on the specifications, we have designed a basic datapath as follows:
Now we shall discuss each datapath elements

Memory elements:

Our memory consists of 256 8 bit registers. The way we access this data is through a 16 bit bus. This bus operates in two modes, byte mode and word mode. The memory is controlled by a signal mem_size. If mem_size is 1, it is in 16 bit mode where it accesses data from address register and address + 1 register. When mem_size is 0, it is in 8 bit mode, wherein the actual data is sent in the lower 8 bits of the data bus and higher 8 bits are set 0. The memory is thus controlled by mem_size and mem_r_w (used to specify whether to read = 1 or write = 0). Accordingly address is picked from mar and data is written/taken from mdr. Whenever there is a state change in input data/output data/address or mem_size, memory block starts. As mentioned earlier, MAR and MDR together control the memory. MAR is asynchronous and MDR is synchronous with the negedge of the clock.

Memory supporting elements:
MAR (Memory Address Register)
This is a temporary latch based register used to access memory. It stores the memory address where we have to read/write data. We have a latch signal ld_mar which sends address to memory if high. The size of data accessed is handled by mem_size signal (as discussed earlier)

MDR (Memory Data Register)
Just as MAR deals with memory address, MDR is another temporary latch register, which deals with actual memory data. It has the data to be written/read to memory. A signal ld_mdr controls MDR and sends/receives data only when high. Another signal mdr_sext handles whether we want data in normal (mdr_sext = 0) or signed extension mode (mdr_sext = 1). When data is in signed extension mode, 8 bit data is signed extended and sent/received from mdr.

ALU (Arithmetic Logic Unit)
ALU is a asynchronous unit used for basic computations, controlled by opsel signal. It implements functions such as AND(opsel=000), ADD(opsel=001), XOR(opsel=010), LSHF(opsel=011), RSHFA(opsel=101) and RSHFL(opsel=100). Another adjoining block is PSW, which tells whether the data is negative (N), positive (P) or zero (Z)

Reg file (Register file)
Controlled by rd (read by reg_r=1) or wr (write by reg_w=1), this is the block for registers R0 - R7. The index of register to be read or written is handled by index: rd_index1, rd_index2, wr_index.

PC handling blocks
PC is a posedge triggered, 8 bit latch based register which is controlled by ld_pc, while writing. Data is asynchronously read from PC to PC_out. PC is also connected to a +2 block which automatically increments PC. It also detects if the PC is odd and generates a odd_PC error signal
IR (Instruction Register)
Another negedge triggered temporary register, this block is responsible for decoding the instruction according to the opcode and ld_ir signal.

ADDER, LSHFT, SEXT blocks

These are basic asynchronous blocks for data manipulation. SEXT performs signed bit extension depending upon amount to be extended. ADDER is a 16 bit adder and LSHFT left shifts according to amount.