RISC-V is becoming popular as an open-source and royalty-free instruction set architecture. The RISC-V ISA is defined as a basic instruction set and other extended instruction sets. RISC-V can accommodate necessary instruction sets according to the application required. The goals in defining RISC-V includes providing an open ISA that is simple and avoids “over-architecting”, which supports both 32-bit and 64-bit address variants, which has highly parallel multicore with IEEE 754 floating-point standard and which also has experimentation with user-level ISA extensions and specialized variants. RV32I is the basic 32-bit integer instruction set.

Methodology

RV32I was designed to be sufficient to form a compiler target and to support modern operating system environments. The ISA was also designed to reduce the hardware required in a minimal implementation. RV32I contains 47 unique instructions. RV32I can emulate almost any other ISA extension (except the A extension, which requires additional hardware support for atomically). Following shows user-visible integer registers. There are 31 general-purpose registers R1-R31 while R0 is constant. Each register is 32 bit wide. The Program Counter holds the address of current instruction.

Optimizations

According to the related work [2], the circuit speed is faster by using exclusive OR instead of multiplexer to select the operation result for the ALU optimization on FPGA. Therefore, in this design of RV32I, exclusive OR is used to select the 32-bit executed result of ALU. As mentioned in the related work [17], one-hot encoding is used instead of the usual binary encoding for the control signal generation to select the ALU calculation result. As only one bit of the bit vector is 1 and the other bits are 0, and the control decisions are determined by the corresponding flip-flop bit in parallel. Therefore, the proposal adopts a one-hot encoding for ALU.

RV32I has five load instructions which are load byte (LB) to load 8-bit signed data, load byte unsigned (LBU) to load 8-bit unsigned data, load halfword (LH), load halfword unsigned (LHU), and load word (LW). Therefore, align/extend unit has to align the loaded data by shifting 8, 16, or 24 bits right depends on the memory address and operation code of the load instructions. Finally, the unit selects a proper value using a large multiplexer depends on the operation code of the instruction. The module is optimized similar to optimizations in ALU module.

According to the implementation done in related work [2], the circuit speed is made faster by using exclusive OR instead of multiplexer to select the operation result for the ALU optimization on FPGA. Following figure

Module rv32ialu

Detailed RTL Component Info : 
---Adders : 
  3 Input 32 Bit Adders := 1 
  2 Input 32 Bit XORS := 1 
  11 Input 32 Bit Mixes := 1 
  5 Input 5 Bit Mixes := 1 

Module rv32ialu_optimized

Detailed RTL Component Info :
---Adders : 
  3 Input 32 Bit Adders := 1 
  2 Input 32 Bit Adders := 1 
  2 Input 32 Bit XORS := 1 
  2 Input 32 Bit XORS := 1 
  2 Input 32 Bit Mixes := 0 
  2 Input 1 Bit Mixes := 2

Above figure shows synthesis logs of unoptimized and optimized design. According to the work done in [3] the preliminary evaluation of the operating frequency of the ALU alone targeting Xilinx Artix-7 FPGA showed that the frequency of the typical ALU was 230MHz while the frequency of the optimized ALU was 240MHz. This optimization is expected to improve the operating frequency of ALU by about 10MHz.