Register Oriented Instruction Sets

James Brakefield

Outline and Goals:

The starting goal is to hit the sweet spot for FPGA soft core processors. The dominating resources are LUT or small block RAM for the register file and Block RAM for program and data memory. The LUT RAM is required to have two or more read ports and one write port. Program and data memory can be separate, easily accomplished by using a dual port RAM.

The secondary goal is good performance as indicated by say, CoreMarks per LUT. Herein, instead of CoreMarks or DMIPS, "instruction work" is used. The nominal value is 1.0 for a 32-bit word size running at one instruction per clock on a nominal instruction set. The nominal value is modified by word size[1], instruction set capabilities and instructions per clock. The performance metric is clock-speed in kilohertz times instruction-work divided by LUT count. Typical clock-speeds are 50 to 500MHz. Typical LUT[2] counts are 300 to 1000 (no floating-point).

The tertiary goal is an architecture spanning a variety of word sizes with little change to the instruction set. Prefix instruction(s) are used to lengthen immediate values (addresses, constants, offsets) so that any word length constant can be obtained. The register file word size sets the word size for the architecture instance.

It has turned out that it is convenient to have register zero of the register file always be zero. This is accomplished by initialization to zero and never writing to register zero. It is also convenient to have two representations of the condition code register: the compact form for save/restore and a non-compact form to save logic delay following the ALU. The non-compact form is the ALU result plus the carry and MSBs of the two operands. During conditional branches the non-compact form is evaluated where it is not in the critical path.

Instruction set encoding is not overly restricted by the above constraints. The simplest instruction set encoding is the "RISC" format with an op-code, two source designators and the result designator. This fits nicely into a 24-bit word giving a bias to a 12, 24 or 48 bit register word size. Another choice is a single register designator with the other two implied (12 or 16-bit instructions). This choice is most user-friendly if the register file is operated as one or more stacks[3]. Lastly one can have a stack machine where all registers are implied (6 to 9-bit instructions).

The overriding architectural characteristic is using a register file for computation operands and result; using load/store instructions to fetch/save data from "main" memory. Depending on address granularity, there are load/store instructions for each data size.

[1] The scaling factors as a function of word size: 8: 33%, 12: 40%, 16:67%, 24: 83%, 32: 100%, 48: 150%, 64: 200%
[2] There are four input LUTs and six input LUTs, here they are equivalent even though it takes 1.5 4LUTs to equal one 6LUT. ALUTs are about equivalent to 6LUTs.
[3] Typically there are a return address register pointer, one or two data pointers and a frame pointer. Instruction register numbers are offsets to these pointers (with one or two bits for pointer selection).