AVR core in Verilog

Martin Vejnár

November 24, 2010

AVR architecture overview

- 32 8-bit registers
- 8-bit ALU, few 16-bit instructions (ADIW)
- jump, call, branch and skip instructions
- 16-bit wide program memory space
- 8-bit wide data memory space
  - 0x00–0x20 registers
  - 0x20–0x5f I/O memory
  - 0x60–0xffff SRAM
- gcc backend is available

Interface

- Clock and reset
- prog_data, prog_addr
- ram_data_in, ram_data_out, ram_addr, ram_rd, ram_wr
- For all intents and purposes, I/O and SRAM behave the same, some I/O registers are actually memory-mapped.
- The RAM interface includes file registers and I/O space. ram_rd and ram_wr are generated for file registers, but are satisfied internally
- Transactions cannot be stalled.
- No interrupt, WDT and SLEEP support.

Access timing

- Program data are assumed to be available on the clock cycle after the address was set, i.e. program instructions are prefetched.
- Data are expected to be available at the same clock cycle the address was exposed.
- This is necessary to satisfy a repeated sequence of `in r16, addr; out addr, r16`.
- This means that the SRAM block is clocked on the negative edge.

Instruction processing

- Instructions are decoded with a single long casez statement and executed immediately.
- Memory access is handled asynchronously (`in` instruction should completed in a single cycle)
- Sometimes the instruction takes more than one cycle
  - jumps (the instruction buffer to be flushed)
  - two-word instructions
  - program memory loads

Summary

- Worked with a program that uses UART and LCD.
- More peripherals can be added easily by mapping them into the data memory.
- The unfinished version can be clocked at 48MHz maximum (AVR chips support 20MHz only).
- Once finished, any program compiled with gcc targeting atmega48 or compatible core should be supported.

Related work

- OpenCores lists several implementations
  - avr_core, VHDL, classic core (no multiplier), precise timings
  - pAVR, VHDL, nice docs, claims to be faster than AVR
  - Navré, Verilog, longer cycles, similar to my implementation