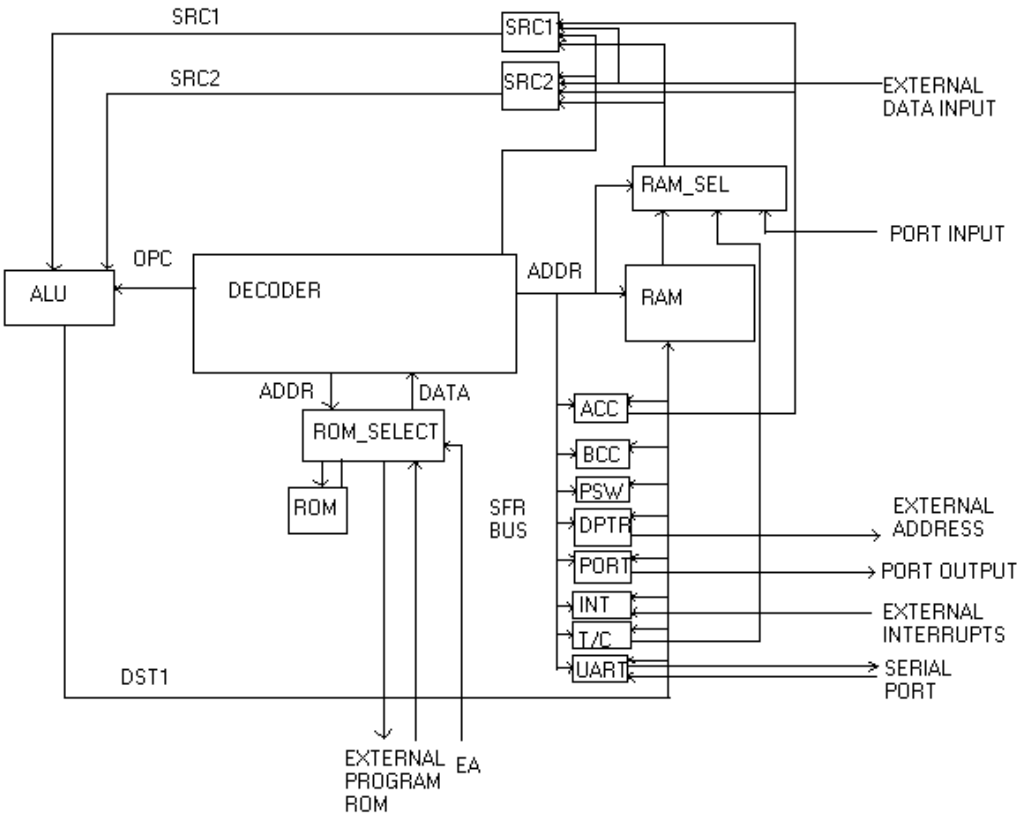
Introduction

This is the plan for 8051 core design, in my opinion the most effective and easy implementation. It also includes uart, which we (me and Jaka) don't intend to implement (at least not in first version).

If anyone is interested for this project, the current progress can be seen on project homepage (www.opencores.org/cores/8051/), just pick free module and let me know.

Simon Teran
simont@opencores.org



alu (aritmetic logic unit. includes divide.v, multiply.v)

- rst (input):(active hi) reset. alu outputs don't care values during reset
- op_code [3:0] (input): defines alu operation (see Defines.v)
- src1 [7:0] (input): source operand 1
- src2 [7:0] (input): source operand 2
- src3 [7:0] (input): source operand 3 (used for signed operation on pc)
- srcCy (input): carry in bit (7th bit)
- srcAc (input): carry in bit (4th bit)

- des1 [7:0] (output): destination 1
- des2 [7:0] (output): destination 2 (used for multiplication and division)
- desCy (output): carry out (7th bit)
- desAc (output): carry out (4th bit)
- desOv (output): overflow

decoder (main unit, decodes instruction and generate control signals)

- rst (input)
- clk (input): clock
- rom_addr [15:0] (output): rom address (actually PC)
- op_in [7:0] (input): input from rom (for operation code and operands)
- ram_addr [7:0] (output): address for internal ram
- int (input): interrupt signal
- int_src [2:0] (input): interrupt source (see Defines.v)
- src1 [1:0] (output): choose witch data signal will be on alu source 1
- src2 [1:0] (output): choose witch data signal will be on alu source 2
- data_out [7:0] (output): data output (used for immediate operands)

rom (internal program memory 4k)

- addr [11:0] (input): address from rom select
- data [7:0] (output): data output
- clk (input): clock signal

rom_select (specify from which (internal, external) program memory we will read)

- ea (input) (pin): (external access) if ea=0 we work with internal program memory, if ea=1 we work with external memory
- addr_in [15:0] (input): requested address (pc)
- data_out [7:0] (output): data from internal/external rom to decoder (instruction)
- addr_in_rom [11:0] (output): address of internal rom (used if ea=0)
- addr_ex_rom [15:0] (output) (pin): address of external rom (used if ea=1)
- data_in_rom [7:0] (input): data from internal rom
- data_ex_rom [7:0] (input) (pin): data from external rom

ram (data memory 256 b)

- clk (input): clock
- addr [7:0] (input): memory address
- data_in [7:0] (input): data input
- data_out [7:0] (output): data output
- r_w (input): read active low, write active high

acc (accumulator register; address:E0)

- clk (input): clock
- sfr_addr [7:0] (input): address bus (see Defines.v)
- r_w (input): read (active high), write (active high)

- data_in [7:0] (input): data input
- data_out [7:0] (output): data output
- p (output): parity (connected to psw.0)

psw (program status word; address: D0)

- clk (input): clock
- sfr_addr [7:0] (input): address bus (see Defines.v)
- r_w (input): read (active high), write (active high)
- data_in [7:0] (input): data input
- data_out [7:0] (output): data output
- cy_out (output): carry output (7^{th} bit) – connected to alu.srcCy
- ac_out (output): auxiliary carry output (6^{th} bit) – connected to alu.srcAc
- rs [1:0] (output): register select (bits 4 and 3) – connected to decoder ??
- cy_in (input): carry input – connected to alu.desCy
- ac_in (input): auxiliary carry input – connected to alu.desAc
- ov_in (input): overflow flag input – connected to alu.desOv
- set (input) (active high): set signal if 1 values from cy_in, ac_in and ov_in are written to register.
- p_in (input): parity flag – connected to acc.p

dptr (data pointer; address: low bits 82, high bits 83)

- clk (input): clock
- sfr_addr [7:0] (input): address bus (see Defines.v)
- r_w (input): read (active high), write (active high)
- data_in [7:0] (input): data input
- data_out [15:0] (output) (pin): address for external data memory

port_out (output from ports 0-3; address: P0= 80, P1=90, P2=A0, P3=B0)

- clk (input): clock
- sfr_addr [7:0] (input): address bus (see Defines.v)
- r_w (input): read (active high), write (active high)
- data_in [7:0] (input): data input
- data_out_p0 [7:0] (output) (pin): port 0 output;
- data_out_p1 [7:0] (output) (pin): port 1 output;
- data_out_p2 [7:0] (output) (pin): port 2 output;
- data_out_p3 [7:0] (output) (pin): port 3 output;

interrupt_control (module witch will handle with interrupts. address: IP=B8, IE=A8)

- clk (input): clock
- sfr_addr [7:0] (input): address bus (see Defines.v)
- r_w (input): read (active high), write (active high)
- data_in [7:0] (input): data input
- t1 (input): interrupt from timer 1

- t0 (input): interrupt from timer 0
- x1 (input) (pin): external interrupt 1
- x0 (input) (pin): external interrupt 0
- s (input): interrupt from serial port
- int (output): interrupt signal – connected to decoder.int
- int_src [2:0] (output): interrupt source (see Defines.v) connected to decoder.int_src
- ie1 (output): external interrupt – connected to timer_counmter.ie1
- ie0 (output): external interrupt – connected to timer_counmter.ie0
- t1_run (output): timer 1 run
- t0_run (output): timer 0 run

timer_counter (timer/counter 0 and 1. addresses: 89, 88, 8C, 8A, 8D, 8B)

- clk (input): clock
- sfr_addr [7:0] (input): address bus (see Defines.v)
- r_w (input): read (active high), write (active high)
- data_in [7:0] (input): data input
- int_t0 (output): interrupt from timer/counter 0 – connected to interrupt_control.t0
- int_t1 (output): interrupt from timer/counter 1 – connected to interrupt_control.t1
- data_out [7:0] (output): data output
- ie1 (input): external interrupt 1
- ie0 (input): external interrupt 0
- t1_run (input): timer 1 run
- t0_run (input): timer 0 run
- t1 (input) (pin): timer/counter external input 1
- t0 (input) (pin): timer/counter external input 0

uart (serial port. addresses: SCON 98, SBUF: 99)

- clk (input): clock
- sfr_addr [7:0] (input): address bus (see Defines.v)
- r_w (input): read (active high), write (active high)
- data_in [7:0] (input): data input
- int (output): serial port interrupt – connected to interrupt_control.s
- s_in (input) (pin): serial input
- s_out (output) (pin): serial output

ram_sel (ram select, selects from witch source we will read. This is used in case we access to port inpus, or timer/counters witch could be different from those in ram)

- addr [7:0] (input): address by comparison with we choose witch source will be on destination
- ram_data [7:0] (input): data from ram
- p0 [7:0] (input) (pin): input pin from port 0
- p1 [7:0] (input) (pin): input pin from port 1
- p2 [7:0] (input) (pin): input pin from port 2

- p3 [7:0] (input) (pin): input pin from port 3
- tim_con [7:0] (input): input from timer/counter
- des [7:0] (output): destination

alu_src1_sel, alu_src2_sel (selects witch data will be on alu source 1,2)

- sel [1:0] (input): select signal – connected to decored.src1 (decored.src2)
- immediate [7:0] (input): connected to decoder.data (immediate operands)
- acc [7:0] (input): connected to acc.data_out
- ram [7:0] (input): connected to ram_sel.des
- ext [7:0] (input): connected to external data ram input
- des [7:0] (output): destination - connected to alu.src1 (alu.src2)