PIC16C5x :Verilog Example

박 정 훈
High Speed Digital Circuit Lab.
Kyungpook National University

Contents

- PIC16C5x Spec.
- Refine Internal Architecture
  - Instruction Decoder
  - Program Counter
  - ALU
  - Registers
- Verification
- Synthesis
- Summary

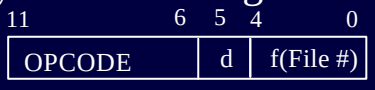
PIC16c5x Spec.

- Harvard Architecture
  - Program : 12bit X 2^{11} Data : 8bit X 72
- 33 instructions
  - 31 instructions available (except tri,clrwdt)
- Two level deep stack
- Direct, indirect and relative data/instruction addressing mode

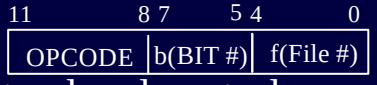

High Speed Digital Circuit Lab 2

Instruction Format

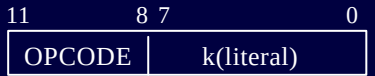
- Byte-oriented register file operations



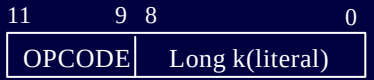
- Bit-oriented register file operations



- Literal and control operations (except GOTO)

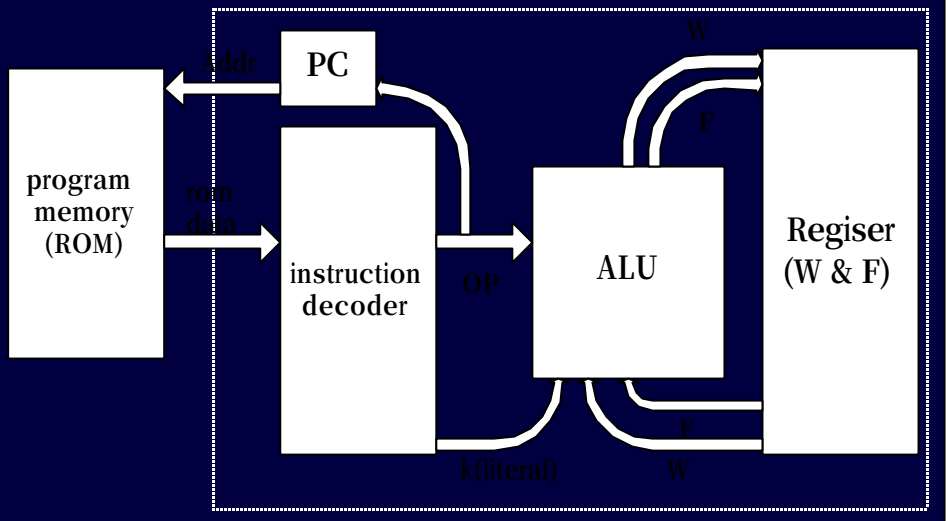


- Literal and control operations-GOTO instruction

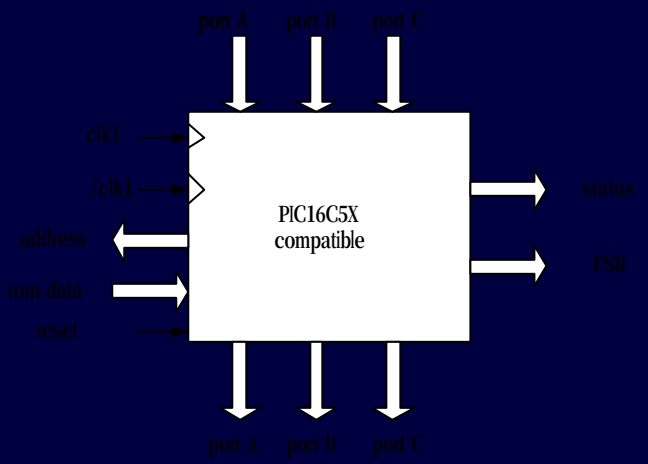



High Speed Digital Circuit Lab 3

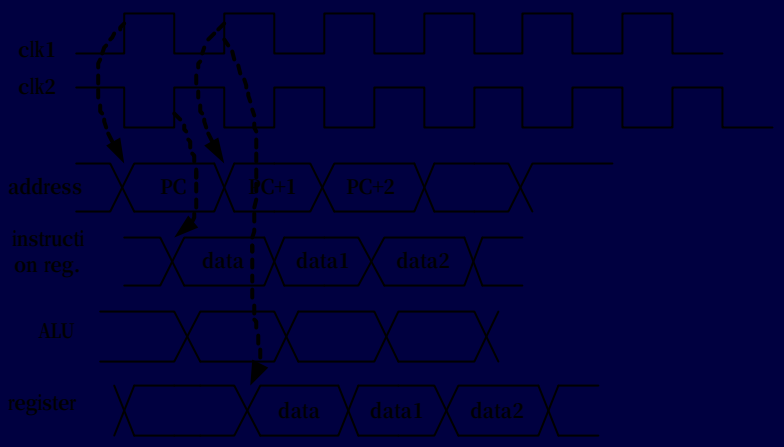
Internal Architecture



Top module



Machine Cycle



Instruction Decoder

- Instruction_decoder_top
  - instruction register
  - instruction decoder
  - mux carry, zero write
  - mux few
  - mux indirect addressing

Instruction register

```
always @(posedge clk or posedge reset) begin
    if( reset == 1'b1)
            inst_buf <= 12'b0000_0000_0000;
    else begin
            if(/*call_inst == 1'b1 || goto_inst == 1'b1 ||*/ skip == 1'b1)
                    inst_buf <= 12'b0000_0000_0000;
            else
                    inst_buf <= rom_data;
    end
 end
assign inst = inst_buf;
 assign k  = inst_buf[7:0];
 assign longk = inst_buf[8:0];
 assign fsel = (inst_buf[4:0] === 4'b0000) ? fsr_rs:inst_buf[4:0];
 assign d = inst_buf[5];
```


High Speed Digital Circuit Lab 8

Instruction decoder

```
assign   op_nop = (inst[11:0] == NOP) ? 1 : 0;
assign   op_movwf = (inst[11:5] == MOVWF) ? 1 : 0;
assign   op_clrw = (inst[11:0] == CLRW) ? 1:0;
assign   op_clrf = (inst[11:5] == CLRF) ? 1:0;
assign   op_subwf = (inst[11:6] == SUBWF) ? 1:0;
assign   op_decf = (inst[11:6] == DECF) ? 1:0;
                        .
                        .
                        .
assign   op_andlw = (inst[11:8] == ANDLW) ? 1:0;
assign   op_xorlw = (inst[11:8] == XORLW) ? 1:0;
```


High Speed Digital Circuit Lab 9

Mux_few, mux_cz_write

- assign few = (op_clrf == 1) ? 1 :
  (op_movwf == 1) ? 1 :
  (op_bcf == 1) ? 1:
  (op_bsf == 1) ? 1:
  (op_subwf == 1 && d == 1) ? 1:
  (op_decf == 1 && d == 1) ? 1 ……
- assign z_write = (op_clrw == 1 ||
  op_clrf == 1 ||
  op_subwf == 1 || ……
- assign c_write = (op_subwf == 1 ||
  op_addwf == 1 ||
  op_rrf == 1 || ….

Program Counter

- Normal Instruction: pc = pc + 1
- Branch instruction - CALL, GOTO, RETLW
  - Next literal address or poped address from stack1 or 2
- SKIP?? - DECFSZ, INCFSZ, BTFSC, BTFSS
  - Insert NOP instead of prefetched instruction.

Program Counter (cont.)

```
always @(posedge clk or posedge reset) begin
    …
  else begin
    if(op_goto == 1) begin
      pc[8:0] <= longk;  pc[10:9] <= status_pa;
    end
    else if(op_call == 1) begin
      ……
      pc[7:0] <= k;    pc[8] <= 0;    pc[10:9] <= status_pa;
    end
    else if(op_retlw == 1) begin
      ……
    end
    else if((fwe == 1) && (fsel == 5'b00010)) begin
      pc[7:0] <=fin;
    end   ……
```

ALU

- mux_alu_a
  - f or k or w. Default: w
- mux_alu_b
  - f or k or w or 1. default :f
- alu_op_gen
- alu_op

ALU_OP_GEN

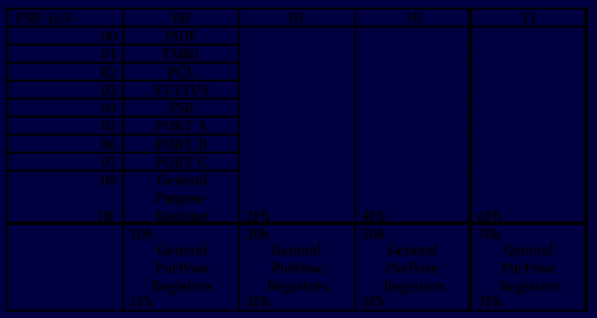
```
assign          alu_op =
                (op_addwf == 1 ||op_incf ==1 || op_incfsz == 1) ?  ALU_ADD:
                (op_subwf == 1|| op_decf == 1 || op_decfsz == 1) ? ALU_SUB  :
                (op_andlw  == 1|| op_andwf == 1) ? ALU_AND :
                (op_iorwf == 1|| op_iorlw  == 1|| op_movwf==1 || op_movf==1 ||
    op_movlw==1 || op_retlw==1) ? ALU_OR :
                (op_xorwf==1 || op_xorlw==1 || op_clrw==1 || op_clrf==1) ? ALU_XOR :
                (op_comf==1) ? ALU_COM :
                (op_rrf==1) ? ALU_RR:
                (op_rlf==1) ? ALU_RL :
                (op_swapf==1) ?  ALU_SWAP:
                (op_bcf==1) ? ALU_BC:
                (op_bsf==1) ? ALU_BS:
                (op_btfsc==1) ? ALU_BTC:
                (op_btfss == 1) ? ALU_BTS:
                4'b1111;
```

ALU_OP

```
always @(op or a or b or carry_in or bitdecode) begin
   case (op)
   ALU_ADD: out <= a+b;
   ALU_SUB: out <= a-b;
   ALU_AND: out <= {1'b0, a & b};
   ALU_OR: out <= {1'b0, (a | b)};
   ALU_XOR: out <=  {1'b0, (a ^ b)};
   ALU_COM: out <= {1'b0, ~a};
   ALU_RR: out <={a[0], carry_in, a[7:1]};
   ALU_RL: out <={a[7:0], carry_in};
   ALU_SWAP:out <= {1'b0, a[3:0], a[7:4]};
   ALU_BC: out <= {1'b0, a & (~bitdecode)};
   ALU_BS: out <= {1'b0, a | bitdecode};
   ALU_BTC:out <= {1'b0, a & bitdecode} ;
   ALU_BTS:out <= ((a & bitdecode) != 8'b0000_0000) ? 9'b0_0000_0000 : {carry_in, a};
   default : out <= {carry_in, a};
   endcase // case(op)
```

Registers

- Special Function Register
  - INDF, STATUS, FSR, PORT A,B,C
- Register File



Registers (cont.)

- Mux_reg_w, Mux_reg_f
  - Input of W register and register file
- Reg_w, reg_general, reg_status, reg_fsr reg_io

Register File

```
assign reg_addr = (fsel == 5'b00000) ? fsr[4:0]: fsel;
assign bank_addr = fsr[6:5];

 always @(posedge clk_q4) begin
   if(fwe == 1) begin
      if(reg_addr > 15) begin
        case (bank_addr)
          2'b00:gpr0[reg_addr] <= fin;
          2'b01:gpr1[reg_addr] <= fin;
          2'b10:gpr2[reg_addr] <= fin;
          2'b11:gpr3[reg_addr] <= fin;
        endcase
      end
      else if(reg_addr > 7) gpr[reg_addr] <= fin;
   end // if (fwe == 1)
 ……
```

Verification

- Instantiation UUT
- Stimuli and response
- Clock Generator

→ *Verify using test assembly programs.*

Verification(cont.)

- Stimuli

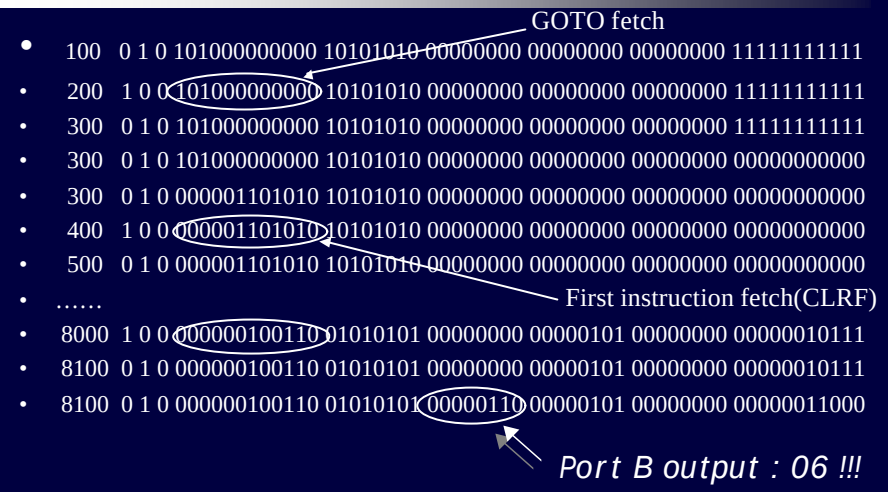
```
assign rom_data = rom[rom_addr];
initial begin
   reset =0;
   pa_in=8'haa;
   pb_in=8'hbb;
   pc_in=8'hcc;
   #2 reset =1;
   #2 reset =0;
   $readmemb("rom.dat", rom);
   #5020
   pa_in = 8'h55;
   #5030
   pa_in = 8'hAA;
 end
```

Verification (cont.)

- always begin
  #100 clk_q1=0; clk_q2=1;
  #100 clk_q1=1; clk_q2=0;
  end

- always @(posedge clk_q1 or posedge clk_q2) begin
  $display($time,,,"%d fsr:%b status:%b f_out:%h %b %h %b %d", clk_q1,
  fsr_out, status_out, f_out, pa_in, pb_out, rom_data, rom_addr);
  end

Verification (cont.)



Synthesis(XILINX spartan S40PQ240)

- Total accumulated area :
- Number of BUFG : 2
- Number of CLB Flip Flops : 680
- Number of CY4 : 12
- Number of FG Function Generators : 798
- Number of H Function Generators : 270
- Number of IBUF : 37
- Number of OBUF : 67
- Number of Packed CLBs : 411

- Number of ports : 106
- Number of nets : 2011
- Number of instances : 1881
- Number of references to this view : 0