Part IV MEng Electronic Engineering

ELEC6044 Advanced Design 2008/09

Tom J Kazmierski

tjk@ecs.soton.ac.uk, Zepler 4223, x 23520
Course aim and assessment

- **Aim:**
  to provide hands-on exposure to modern, commercial CAD tools

- **Assessment:**
  100% coursework based on electronic hand-ins

- **3 assignments:**
  - FPGA Synthesis of a complex system 60% (due Wk9)
  - Analogue simulations with HSPICE 30% (due Wk12)
  - Mixed-signal system description (VHDL-AMS) 10% (due Wk 14)
Reference material

- SystemVerilog and FPGA synthesis:
  - www.systemverilog.org – technical papers, tutorials
  - www.asic-world.com – SV examples
  - Andrew Rushton “VHDL for logic synthesis”, Wiley

- HSPICE (far more advanced than PSPICE):
  - most electronics textbooks have a SPICE primer
  - Andrei Vladimirescu, “The SPICE book”, J. Wiley,
  - ELEC6044 web notes have links to:
  - More SPICE manuals in pdf:
    - http://www.ecs.soton.ac.uk/~cad/spice/

- VHDL-AMS (VHDL with Analogue and Mixed-Signal extensions):
  - http://www.syssim.ecs.soton.ac.uk
Assignment 1

- Synthesise AVR processor core
- Synthesise Program Memory with a selection of small programs
  - Fixed-instruction execution
  - Copy Port B to Port c
  - Calculate n! (n factorial)
- Demonstrate a working system in software using Modelsim and Synplify or Altera Quartus
Embedded processor design flow

- System specification
- HDL entry
- Synthesis
- Place & Route
- Final layout
- RTL simulation
- Post Synthesis Verification
- Error?
Components of the AVR processor core to synthesise

- ALU, Status Register
- General-Purpose 32-Register File
- Program Memory, Program Counter, Instruction Register
- Two I/O ports
- A small SRAM, Stack Pointer Register
- Instruction Decoder
What is the AVR?

- RISC single chip controller developed by ATMEL in late 90s
- Uses flash memory, not an EPROM
- 1 instruction per clock cycle (except program flow control instructions)
- Dual-bus architecture with 32 general purpose registers connected to ALU
- Separate SRAM
- I/O ports similar to those in PIC
- Other circuitry: A/D, timer, USI, watchdog, etc.
AVR block diagram
Pipelined execution similar to that of the PIC
First try to develop and test simple registers
Then develop and test the Program Counter

```verilog
always_ff @ (posedge clk or negedge nreset)
    if (~nreset) begin
        PC_out = 10'b00000000000;
        PC_stop = 0;
    end
    else if (PC_incr) begin
        PC_out = PC_out + 1;
        PC_stop = 0;
    end
    else if (PC_Imme) begin
        PC_out = PC_out + Imme + 1;
        PC_stop = 0;
    end
    else if (PC_branch) begin
        PC_out = branch_add;
        PC_stop = 0;
    end
    else
        PC_stop = 1;
end
```

clk, nreset, Imme, branch_add, PC_incr, PC_Imme, PC_branch

PC_out, PC_stop
When integrating the design, first develop a cut down version with limited ALU functionality
Use the following module declarations for top level module and program memory module

// CPU – top level module
module CPU (input logic clk, reset; input logic [7:0] portAin, portBin; output logic [7:0] portAout, portBout);

// Program memory module
module program_memory #(parameter progno=0)(input logic [10:0] addr; output logic [16:0] data);
Required instructions

- **ALU instructions**
  - ADD, ADC, SUB, SBC, SUBI, LSL, LSR, AND, OR, EOR, ANDI

- **Data transfer instructions**
  - LDI, IN, OUT, MOV, POP, PUSH, LD

- **Program flow control instructions**
  - JMP, BREQ, CALL, RET

- **No operation**
  - NOP
### ALU instructions

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Opcode</th>
<th>Operands</th>
<th>Description</th>
<th>Operation</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD</strong></td>
<td>000011</td>
<td>Rd, Rr</td>
<td>Add without Carry</td>
<td>Rd &lt; Rd + Rr</td>
<td>Z,N,V,C</td>
</tr>
<tr>
<td><strong>LSL</strong></td>
<td>000011</td>
<td>Rd</td>
<td>Logical left shift</td>
<td>Rd(n+1)&lt;Rd(n), Rd(0)&lt; 0, C&lt; Rd(7)</td>
<td>Z,N,V,C</td>
</tr>
<tr>
<td><strong>ADC</strong></td>
<td>000111</td>
<td>Rd, Rr</td>
<td>Add with Carry</td>
<td>Rd &lt; Rd + Rr +C</td>
<td>Z,N,V,C</td>
</tr>
<tr>
<td><strong>SUB</strong></td>
<td>000110</td>
<td>Rd, Rr</td>
<td>Subtract without Carry</td>
<td>Rd &lt; Rd – Rr</td>
<td>Z,N,V,C</td>
</tr>
<tr>
<td><strong>SBC</strong></td>
<td>000010</td>
<td>Rd, Rr</td>
<td>Subtract with Carry</td>
<td>Rd &lt; Rd – Rr – C</td>
<td>Z,N,V,C</td>
</tr>
<tr>
<td><strong>OR</strong></td>
<td>001010</td>
<td>Rd, Rr</td>
<td>Logical OR</td>
<td>Rd &lt; Rd</td>
<td>Rr</td>
</tr>
<tr>
<td><strong>EOR</strong></td>
<td>001001</td>
<td>Rd, Rr</td>
<td>Exclusive OR</td>
<td>Rd &lt; Rd ^ Rr</td>
<td>Z,N,V,C</td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td>001000</td>
<td>Rd, Rr</td>
<td>Logical AND</td>
<td>Rd &lt; Rd &amp; Rr</td>
<td>Z,N,V</td>
</tr>
</tbody>
</table>

**Arithmetic and Logic Instructions (4-bit opcode)**

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Opcode</th>
<th>Operands</th>
<th>Description</th>
<th>Operation</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ANDI</strong></td>
<td>0111</td>
<td>Rd, K</td>
<td>Logical AND with Immediate</td>
<td>Rd &lt; Rd &amp; K</td>
<td>Z,N,V</td>
</tr>
<tr>
<td><strong>SUBI</strong></td>
<td>0101</td>
<td>Rd, K</td>
<td>Subtract Immediate</td>
<td>Rd &lt; Rd – K</td>
<td>Z,N,V,C</td>
</tr>
</tbody>
</table>
## Data transfer instructions

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Opcode</th>
<th>Operands</th>
<th>Description</th>
<th>Operation</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Transfer Instructions (4-bit opcode)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDI</td>
<td>1110</td>
<td>Rd, K</td>
<td>Load Immediate</td>
<td>Rd &lt; K</td>
<td>None</td>
</tr>
<tr>
<td><strong>Data Transfer Instructions (5-bit opcode)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IN</td>
<td>10110</td>
<td>Rd, A</td>
<td>In From I/O location</td>
<td>Rd &lt; I/O (A)</td>
<td>None</td>
</tr>
<tr>
<td>OUT</td>
<td>10111</td>
<td>A, Rr</td>
<td>Out to I/O Location</td>
<td>I/O (A) &lt; Rr</td>
<td>None</td>
</tr>
<tr>
<td><strong>Data Transfer Instructions (6-bit opcode)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>001011</td>
<td>Rd, Rr</td>
<td>Copy Register</td>
<td>Rd &lt; Rr</td>
<td>None</td>
</tr>
<tr>
<td><strong>Data Transfer Instructions (11-bit opcode)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POP</td>
<td>10010001111</td>
<td>Rd</td>
<td>Pop Register from Stack</td>
<td>Rd &lt; STACK</td>
<td>None</td>
</tr>
<tr>
<td>PUSH</td>
<td>1001001111</td>
<td>Rr</td>
<td>Push Register on Stack</td>
<td>STACK &lt; Rr</td>
<td>None</td>
</tr>
</tbody>
</table>
# Program flow control instructions

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Opcode</th>
<th>Operands</th>
<th>Description</th>
<th>Operation</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Branch Instructions (9-bit opcode)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BREQ</td>
<td>111100001</td>
<td>k</td>
<td>Branch if equal</td>
<td>If($Z=1$) then PC &lt; PC + k + 1</td>
<td>None</td>
</tr>
<tr>
<td><strong>Branch Instructions (10-bit opcode)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMP</td>
<td>1001010110</td>
<td>k</td>
<td>Jump</td>
<td>PC &lt; k</td>
<td>None</td>
</tr>
<tr>
<td>CALL</td>
<td>1001010111</td>
<td>k</td>
<td>Call Subroutine</td>
<td>PC &lt; k</td>
<td>None</td>
</tr>
<tr>
<td><strong>Branch Instructions (16-bit opcode)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RET</td>
<td>1001010100 001000</td>
<td>-</td>
<td>Subroutine return</td>
<td>PC &lt; STACK</td>
<td>None</td>
</tr>
</tbody>
</table>
Logical shift right and NOP

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Opcode</th>
<th>Operands</th>
<th>Description</th>
<th>Operation</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSR</td>
<td>0010100110</td>
<td>Rd</td>
<td>Logical Shift Right</td>
<td>Rd(n) &lt; Rd(n+1), Rd(7) &lt; 0, C &lt; Rd(0)</td>
<td>Z, C, V</td>
</tr>
</tbody>
</table>

Bit and Bit-test Instructions (11-bit opcode)

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Opcode</th>
<th>Operands</th>
<th>Description</th>
<th>Operation</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP</td>
<td>0000000000 000000</td>
<td>-</td>
<td>No operation</td>
<td>-</td>
<td>None</td>
</tr>
</tbody>
</table>

MCU Control Instructions (16-bit opcode)
Other desirable instructions

- RJMP, RCALL – absolute relative jump and call
- INC, DEC – increment, decrement
- BRNE (and other conditional branches if needed)
- SBI, SBO – set and clear I/O bit

Interrupt handling:
- BRIE, BRID, SEI, CLI, RETI

LD with and without increment/decrement
- LD rd, X
- LD rd, X+
- LD rd, -X