Building a two-bit processor

Wan Ahmad Tajuddin Wan Abdullah
Department of Physics, Universiti Malaya, 50603 Kuala Lumpur, Malaysia

Abstract: University electronics curricula teach digital electronics using logic gates, and microprocessors and their functional structure, but normally fail to substantially bridge the area between them. In this paper, the pedagogic exercise of building a two-bit processor using transistor-transistor logic (TTL) gates is described, wherein the main ideas in microprocessor architecture are encountered. These include central processing unit (CPU) design, bus management, and timing requirements.
Abstract: University electronics curricula teach digital electronics using logic gates, and microprocessors and their functional structure, but normally fail to substantially bridge the area between them. In this paper, the pedagogic exercise of building a two-bit processor using transistor-transistor logic (TTL) gates is described, wherein the main ideas in microprocessor architecture are encountered. These include central processing unit (CPU) design, bus management, and timing requirements.

I. INTRODUCTION

The typical university electronics curriculum teaches digital electronics using logic gates on standard TTL chips on one end, and microprocessors and their functional structure on the other end. The digital electronics course usually ends after building flip-flops and shift registers, whereas the microprocessor course can be rather abstract and sometimes somewhat descriptive, failing to connect directly where digital electronics left off (see for example the text by Wiatrowski and House¹). Laboratory exercises (see e.g. Lynch²) on microprocessors usually concentrate on how to use them rather then on their construction.
What seems to be missing is a bridge from the simple logic gates to the complex functional architecture of the microprocessor.

In this paper we describe the building of a two-bit processor from basic logic gates and devices, exhibiting the main functional components involved, which is useful for pedagogy. Note that with two bits, the architecture is the simplest one which contains non-trivial functionalities. In the following section, we discuss the architecture and functioning of the 2-bit processor and how it can be implemented using TTL gates. Then we look at timing requirements and other electronic concerns. Finally we conclude in the last section.

II. ARCHITECTURE

The basic components of a computer are its memory and the central processing unit (CPU), plus an underlying clock to drive the system. We shall not be concerned here with input/output components because of the restricted abilities of a 2-bit computer. The architecture of the 2-bit computer is shown schematically in Fig. 1. The data bus is bi-directional, while the CPU only writes onto the address bus and the memory reads. Only two control lines are involved in this simple system: the READ and the WRITE instructions to the memory, respectively for reading and writing data to the bus. The clock line provides a sequence of digital pulses to the CPU.

For a 2-bit computer, the data bus as well as the address bus are each 2 bits wide. This means that the memory can only consist of 4 locations of 2-bit data. Also there can only be 4 types of instructions, and to be representative, we have chosen the four as given in Table I. Below we describe how we can build the clock, the memory, and the central processing unit with this instruction set.
A. Clock

The clock provides a sequence of digital pulses to the system. For pedagogic reasons, we would like to be able to control the timings of these pulses manually. Thus a toggle switch is used. The toggle connects earth to either the PRESET or the CLEAR lines of a D-type flip-flop (available on the standard 7474 chip). The output line from the flip-flop provides for the clock output; the flip-flop is used to prevent effects of mechanical bounces in the switch.

B. Memory

The 4-location 2-bit memory can be composed of four 2-bit registers built from eight D-type flip-flops. The data lines and the output lines of the two flip-flops of each register connect to the 2-bit data bus through multiplexing switches to control the reading or writing of data from/to the bus.

To write to the bus, the outputs from the four registers are multiplexed using the 4-to-1 multiplexer on the chip 74LS153. The two select lines determine which of the four inputs are to be chosen as the output, and are connected to the address bus. The ENABLE lines are earthed so that the multiplexer is always enabled; writing to the data bus is controlled by 3-state buffers (on the chip 7094) between the multiplexer output and the bus. Besides the states 0 and 1 passed on through it, a 3-state buffer also provides a high impedance state which is needed when the multiplexer is not writing to the bus. The control lines of the 3-state buffers receive the WRITE control signal from the CPU.

To read from the data bus, the bus lines are demultiplexed using the 2-bit decoder on the 74LS139 chip (whose output are active low!). The address bus lines provide the input to the decoder to yield the output signal in one of the four output lines corresponding to the four registers. By ANDing (2-input AND gates on 7408) these respective lines with
the READ control signal line from the CPU and connecting them to the respective clock input lines of the flip-flops constituting the registers, and by connecting the respective data bus lines to the appropriate data input lines of the flip-flops, reading from the data bus is achieved.

C. Central Processing Unit

The main parts of a conventional processor are: the accumulator (ACC) which is the main data register, the program counter (PC) storing the address of the current instruction to be carried out, the arithmetic and logic unit (ALU), which carries out arithmetic and logic operations on data, and the control unit which oversees the execution of microinstructions. Their interdependence is shown in Fig. 2.

1. Accumulator

The accumulator is just a 3-bit data register, and can be built from two D-type flip-flops. It writes to the data bus through 3-state buffers at an ACC WRITE internal signal from the control unit which enables the buffers. It reads from the data bus at an ACC READ signal to the clock lines of the flip-flops.

2. Program control

The program counter is a 2-bit register storing the address in memory of the current instruction. A memory address register (MAR) is also needed to help the PC when the address needs to be incremented. The MAR is also a 2-bit register accepting input to its data lines output from the PC directly, and writes its output again to the PC at the PC INCR signal after incrementing it by 1. The increment is carried out by a 2-bit adder circuit: the
lowest bit goes through a NOT gate (available on 7404) and then through a 3-state buffer enabled by PC INCR. The simplicity of two bits only requires an XOR gate form the 2 bits of the MAR output into a similar 3-state buffer to the high bit of the PC. The clock lines in the PC receives a derivative of the PC INCR signal to instruct it to read when the output from the buffers are stable. A MAR READ signal into the clock lines of the MAR is used to instruct the MAR to read from the PC. These make up the first two microinstructions in the instruction cycle:

1. \( \text{MAR} \leftarrow \text{PC} \)  \hspace{1cm} \text{[MAR READ]} \\
2. \( \text{PC} \leftarrow \text{MAR} + 1 \) \hspace{1cm} \text{[PC INCR]} \\

The MAR writes to the address bus for instruction fetch. This can be done through AND gates with a MAR WRITE signal line providing the complementary inputs. To deal with jump instructions, the PC can read from the data bus through 3-state buffers enable by A PC READ signal. A derivative of the PC READ signal is also ORed (OR gates on 7432) with the PC INCR signal going to the PC clock lines.

3. **ALU**

The ALU carries out arithmetic and logic calculations on data. With the limited instruction set in Table I, we only need to handle the AND function between data. A 2-bit register ALM which reads from the data bus (at the ALU READ signal to the clock lines of the register) stores one of the arguments while the other argument to the AND function can be read directly from the data bus when instructed. So the output from ALM and connections to the data bus go through respective AND gates to carry out the AND function, with the outputs input into another 2-bit register ALK used to store the result. The output of ALK writes back to the data bus through 3-state buffers enabled by an ALU WRITE control signal while the clock lines of this register receive a derivative of an ALU EXEC control signal.
4. Control Unit

The workings of the central processor are overseen by the control unit. This unit sends out appropriate control signals depending on the microinstructions in an instruction cycle. The instructions are read from the data bus onto the instruction register (IR), which is just a 2-bit register. The clock lines receive an IR READ control signal. The output of the IR is decoded by the instruction decoder (using the 2-bit decoder on 7LS139 as before), yielding 4 lines which go into the control matrix.

We break up the instruction cycle into 8 periods to enable eight microinstructions to be carried out sequentially in a cycle (most of the instructions actually only require 7, the BRA instruction even less – see Table II). Table II shows the microinstruction sequences for the various instructions (note that the microinstructions in period 2 can be and are carried out simultaneously). The 8 periods are obtained by connecting the system clock to an 8-bit ring counter, made by joining the data and output data lines of 8 D-type flip-flops in a ring. The flip-flops are clocked by the system clock. The 8 output lines then define which period the system is in, and are input as the other dimension to the control matrix. The initial pattern in the ring counter need to be initialized beforehand. We choose the output lines to be active low, so only one of the 8 lines are low at one time. The output of the instruction decoder are also active low, so we can deploy NOR gates (7400) in the control matrix.

The control matrix is the matrix of 4 lines corresponding to instruction identity times the 8 lines for the microinstruction period which outputs appropriate control signals. The configuration of the control matrix can be inferred from Table II. For example the microinstruction period line 4 is NORed with the LDA instruction line to give PC READ, MAR WRITE and Memory WRITE signals corresponding to the microinstruction PC ←
(MAR) (i.e. transfer the contents of memory addressed by MAR to PC). The construction of
the control matrix employs NOR, OR and NOT gates.

III. TIMING AND OTHER CONSIDERATIONS

The description in the above section should be adequate to enable the construction
of a 2-bit processor or computer. However there are several technicalities which require
caution.

To ensure correct running of the CPU, we have to ensure that the sequence of
operations are carried out according to correct timing. Especially when simultaneous
writing and reading are done, the timings need to be arranged such that the start edge of the
READ signal occurs when the data to be read is already stable, i.e. it is slightly delayed as
compared to the WRITE signal (but still in the same microinstruction period(!)). Since each
microinstruction period consists of a high and a low system clock state (the ring counter
changes state at the positive-going edge), we use the later half (the low state) to activate the
READ signal. Thus the PC INC, PC READ, and ALU EXEC signals undergo respective
ANDing with the NOT of the system clock.

Other considerations in the circuit include concerns with fan-outs. Enough current
available to drive gates has to be ascertained. For display purposes, LEDs with appropriate
series resistors can be connected at various points in the circuit.

IV. CONCLUDING REMARKS

We have described how a 2-bit processor (and associated memory unit) can be
built for pedagogic purposes. This exercise can be carried out by the student having
background in practical digital electronics in about 20 hours.\textsuperscript{3} With more enthusiasm and time, a 4-bit extension of the architecture can be designed and constructed.\textsuperscript{4} These exercises are useful to the student and allows the practical appreciation of the functional components that are the basis for the modern microprocessor.

\textsuperscript{a}URL: http://fizik.um.edu.my/cgi-bin/hitkat?wat


\textsuperscript{3}Foo Ee Sin, “Pemproses 2-bit,” undergraduate project report \textbf{E/90-20} (Department of Physics, Universiti Malaya, Kuala Lumpur, 1990); Yeap Bee Khee, “2-bit processor,” undergraduate project report \textbf{E/90-21} (Department of Physics, Universiti Malaya, Kuala Lumpur, 1990).

\textsuperscript{4}Lee Kim Fook, “Microprocessor – 4 bit”, undergraduate project report \textbf{E/90-23} (Department of Physics, Universiti Malaya, Kuala Lumpur, 1991); Lim Lee San, “Mikropemproses 4 bit,” undergraduate project report \textbf{E/90-24} (Department of Physics, Universiti Malaya, Kuala Lumpur, 1991).

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Code</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA addr</td>
<td>00</td>
<td>Load contents of \textit{addr} to Accumulator</td>
</tr>
<tr>
<td>STA addr</td>
<td>01</td>
<td>Store contents of Accumulator to \textit{addr}</td>
</tr>
<tr>
<td>AND arg</td>
<td>10</td>
<td>Carry out logical AND between the contents of the Accumulator and \textit{arg}</td>
</tr>
<tr>
<td>BRA addr</td>
<td>11</td>
<td>Jump to instruction at \textit{addr}</td>
</tr>
</tbody>
</table>

Table I. Instruction set of the 2-bit processor
<table>
<thead>
<tr>
<th>Instruction Period</th>
<th>LDA</th>
<th>STA</th>
<th>AND</th>
<th>BRA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAR ← PC</td>
<td></td>
<td></td>
<td>[MAR READ]</td>
</tr>
<tr>
<td>2</td>
<td>PC ← MAR + 1</td>
<td>IR ← (MAR)</td>
<td>[PC INCR]</td>
<td>[IR READ, MAR WRITE, Memory WRITE]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[MAR READ]</td>
</tr>
<tr>
<td>3</td>
<td>MAR ← PC</td>
<td></td>
<td></td>
<td>[MAR READ]</td>
</tr>
<tr>
<td>4</td>
<td>PC ← (MAR)</td>
<td>ALM ← ACC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[PC READ, MAR WRITE, Memory WRITE]</td>
<td>[ALU READ, ACC WRITE]</td>
<td>[PC READ, MAR WRITE, Memory WRITE]</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MAR ← PC</td>
<td>ALK ← ALM(MAR)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[MAR READ]</td>
<td>[Memory WRITE, MAR WRITE, ALU EXEC]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ACC ←(MAR)</td>
<td>(MAR) ← ACC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[ACC READ, MAR WRITE, Memory READ]</td>
<td>[ACC WRITE, MAR WRITE, Memory READ]</td>
<td>[ACC READ, ALU WRITE]</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PC ← MAR + 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[PC INCR]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table II. Microinstruction sequences in an instruction cycle for the various instructions
Fig. 1. Schematic architecture of the 2-bit computer.

Fig. 2. The internal structure of the CPU.