Fundamentals of Computer Science, 2nd part

Lecture 04
Microcomputing
Recap

• Logical Level which is the nearest to hardware consists of gates
• Gates: small and fast switches
• Example of starting from a functionality and constructing the circuit
• Boolean Algebra to specify components
• Examples Arithmetic actions: *an adder*
• Chips: components made of gates ranging from SSI to VLSI
Multilevel machines (recap)

- Human friendly
- Digital logic
- Microarchitecture
- ISA
- Operating System
- Assembly
- Problem oriented lang.

Translation (assembler)
Translation (compiler)
Interpretation
Interpretation

Registers, ALU, data path simple program. Operation Controlled by Microprograms
This lecture

• Memory circuit
• Clocks and clock cycle
• Control Logic
• Microinstruction (execution)
• mapping clock phases to control signals
• implementing a multiplication algorithm
Memory circuit

- Remember NAND
- Write the table of truth?
Memory circuit

- Active Low SR-Latch.
- Latch? S: for set and R: for reset
Memory circuit

- Initial state: S=1 and R=1
- We end up with Q equal to 1 and its complement 0. Check!
Memory circuit

- What happens if \( S \) is set to 0 (while \( R \) is not changed)?
- The output remains unchanged. Check it!
Memory circuit

- Start from initial condition $S=R=1$ and reset, i.e. Change R to 0. [Change G2 first]
- Exercise: reset R to 1
Memory circuit

- Why do I need both Q and its complement?
- Start from initial state and change both S and R at the same time!
Memory circuit

- Put S=0 and R=0.
- Door open and closed Simultaneously!
- Inconsistent State!
Memory circuit

idea: remember previous value
Solution: the output independent of the input
Example: Active High SR latch
Clocks (provide sync & correct order of events)

- a circuit that emits a series of pulses with precise width for pulses and precise pulse width

- Clock Cycle Time: Time interval between the corresponding edges of two consecutive pulse phases
Clocks and control logic

Computer: various components connected via BUS
➢ So, need to Control the transfer to the Bus…
This is done by Control Logic.
Control Logic uses clock

In a circular form
Example: Microprogram Memory

Place a microinstruction address in MPC register

The contents available in MIR register

Phase 5 only

All examples are from chap 4.4 of Goldschlager & Lister
Microinstructions

Data is kept in **main memory**. How about instructions? Instructions (**microinstruction**) placed in **micromemory**. Example: the following 22 bits micromemory used in adder (next slide)
Adder

INPUT via bus 1, 2  OUTPUT via bus 3

Example:
if signal 1, 6 and 14 turned on, contents of A and MDR added and result placed in MAR

If 9 is on as well then result placed also in register A

Shift/subtract similar
Clock phase & Control Signals

During each phase only certain control signals are allowed to turn on, specified in the table.

<table>
<thead>
<tr>
<th>Clock phase</th>
<th>Control signals which can be turned on</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-8</td>
</tr>
<tr>
<td>2</td>
<td>9-14</td>
</tr>
<tr>
<td>3</td>
<td>15-16</td>
</tr>
<tr>
<td>4</td>
<td>17-22</td>
</tr>
<tr>
<td>5</td>
<td>the control signals marked ‘PHASES’</td>
</tr>
</tbody>
</table>
Multiplication Microprogram

- **Multiply**
  - contents of register C by contents of A
  - place result in main memory cell 1
- **The algorithm**
  set MDR to 0
  repeat A times
    add C to MDR
  move 1 to MAR and set control signal to write
Sketch of microprogram for multiplication

Micro Memory address: Microinstruction          Explanation

0: 0+0 -> MDR; MPC+1 -> MPC                   Initialise MDR
1: MPC + TESTZERO -> MPC                     Check if A zero
2: 5 -> MPC                                  Yes: exit from loop
3: C+MDR -> MDR; MPC+1 -> MPC                No: add C to MDR
4: A-1 -> A; 0+1 -> MPC                     Decrease A
5: 0+1 -> MAR; write; MPC+1 -> MPC           Goto top of loop at 1

Write product into cell 1

MDR:= Memory data register        MPC:=Microprogram Counter
Analysis of the Algorithm

Notice the loop executes $A$ times (worst case: the register for $A$ is $n$ bits, then $2^n - 1$ execution)

Is there any faster way? Yes, use shift left

(For the Algorithm see Page 158 of the book)

\[
\begin{array}{c}
\begin{array}{cccccc}
0 & 0 & 0 & 1 & 1 & 1 \\
\times & 0 & 0 & 0 & 1 & 1 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 1 & 0 & 1 & 0
\end{array}
\end{array}
\]

$=C$

$=A$

$=C$ shifted once

$=C$ shifted twice

$=\text{product}$
Summary

• How to remember previous values (Memory circuit)
• Clocks achieving synchronisation and …
• Control Logic
• How to execute in Microprograms Microinstruction
• implementing & analysing multiplication as a Microprogram