State Holding and Sequential Circuit Design

Andreas G. Andreou
Pedro Julian

Electrical and Computer Engineering
Johns Hopkins University

http://andreoulab.net

*Lecture Notes adapted from Lecture 05 in MIT 6.004 Computation Structures
http://6004.csail.mit.edu/

and from Pedro Julian’s VLSI class notes*
Progress so far: all about levels of abstraction and representation
Something we can’t build (yet)

What if you were given the following design specification:

When the button is pushed:
1) Turn on the light if it is off
2) Turn off the light if it is on

The light should change state within a second of the button press

What makes this circuit so different from those we’ve discussed before?

1. “State” – i.e. the circuit has memory
2. The output was changed by a input “event” (pushing a button) rather than an input “value”
Digital machine model

Plan: Build a Sequential Circuit with stored digital STATE –

- Memory stores CURRENT state, produced at output
- Combinational Logic computes
  - NEXT state (from input, current state)
  - OUTPUT bit (from input, current state)
- State changes on LOAD control input
Combinational logic is *stateless*: valid outputs always reflect current inputs.

To build devices with state, we need components which *store* information (e.g., state) for subsequent access.

- **ROMs** (and other combinational logic) store information “wired in” to their truth table
- **Read/Write** memory elements are required to build devices capable of changing their contents.

**How can we store -- and subsequently access -- a bit?**

- Mechanics: holes in cards/tapes
- Optics: Film, CDs, DVDs, …
- Magnetic materials
- Delay lines; moonbounce
- Stored charge
Dynamic Storage: using capacitors

“store” a charge on a capacitor!

N-channel fet serves as access switch $V_{\text{REF}}$

To write:
Drive bit line, turn on access fet,
force storage cap to new voltage

To read:
precharge bit line, turn on access fet
detect (small) change in bit line vol
e-DRAM cell

Need cell with big capacitance!
Dynamic Storage: using capacitors

Pros:
- compact – low cost/bit (on BIG memories)

Cons:
- complex interface
- stable? (noise, …)
- it leaks! $\Rightarrow$ refresh

Suppose we refresh CONTINUOUSLY?
Static Storage: using feedback

**IDEA:** use *positive feedback* to maintain storage indefinitely. Our logic gates are built to restore marginal signal levels, so noise shouldn’t be a problem!

![Diagram of a bistable storage element]

Result: a bistable storage element

Three solutions:
- two end-points are **stable**
- middle point is unstable

Not affected by noise

VTC for inverter pair

Feedback constraint: \( V_{IN} = V_{OUT} \)

We’ll get back to this!
Settable storage element

It’s easy to build a settable storage element (called a latch) using a lenient MUX:

If $G=1$ \( Q \) follows \( D \)

If $G=0$ \( Q' = Q \) (stable)
New device: “transparent” D latch

\[ Q' \]
\[ D \]
\[ G \]

- **G=1:**
  - \( Q \) follows \( D \)
- **G=0:**
  - \( Q \) holds

- \( T_{PD} \)

- **Transparent mode**
  - \( t_{setup} \)
  - \( t_{hold} \)

\[ V_1 \]
\[ V_2 \]

Introduction to VLSI Systems
Latch: CMOS realization

- Switches implemented using transmission gates
- Switches are complementary
  - L1 passes input to output in transparent mode
  - L2 closes loop and keeps the state
CMOS latches: types

- Positive
  a) [Diagram of positive CMOS latch]
  b) [Detailed diagram of positive CMOS latch]

- Negative
  c) [Diagram of negative CMOS latch]
  d) [Detailed diagram of negative CMOS latch]
Let’s try it out!

Plan: Build a Sequential Circuit with one bit of STATE –

- Single latch holds CURRENT state
- Combinational Logic computes
  - NEXT state (from input, current state)
  - OUTPUT bit (from input, current state)
- State changes when $G = 1$ (briefly!)

What happens when $G=1$?
Combinational cycles

When $G=1$, latch is Transparent...

... provides a combinational path from D to Q.

Can’t work without tricky timing constraints on $G=1$ pulse:

- Must fit within contamination delay of logic
- Must accommodate latch setup, hold times

Want to signal an INSTANT, not an INTERVAL...
Edge-triggered Flip-Flop

Transitions mark instants, not intervals

G = 0

G = 1

D

in

D

hold

D

transparent

0 -> 1
Edge-triggered Flip-Flop

Observations:
- only one latch “transparent” at any time:
  - master closed when slave is open
  - slave closed when master is open
  $\Rightarrow$ no combinational path through flip flop
  (the feedback path in one of the master or slave latches is always active)
- Q only changes shortly after $O \rightarrow 1$
  transition of CLK, so flip flop appears to be “triggered” by rising edge of CLK

Introduction to VLSI Systems
Flip-Flop waveforms

Negative latch + positive latch

Rising edge FF

master closed
slave open

slave closed
master open
Edge triggered FF

- Positive (rising) edge FF
  - Master negative, slave positive
Edge triggered FF

- Falling (negative) edge FF
  - Master positive, slave negative
Single-clock synchronous circuits

We’ll use Flip Flops and Registers – groups of FFs sharing a clock input – in a highly constrained way to build digital systems:

**Single-clock Synchronous Discipline**

- No combinational cycles
- Single periodic clock signal shared among all clocked devices
- Only care about value of register data inputs just before rising edge of clock
- Period greater than every combinational delay + setup time
- Change saved state after noise-inducing logic transitions have stopped!
Model: Discrete Time

Active Clock Edges punctuate time ---

- Discrete Clock periods
- Discrete State Variables
- Discrete specifications (simple rules - eg tables - relating outputs to inputs, state variables)
- ABSTRACTION: Finite State Machines (next lecture!)
Summary

“Sequential” Circuits (with memory):

**Basic memory elements:**
- Feedback, detailed analysis => basic level-sensitive devices (e.g., latch)
- 2 Latches => Flop
- Dynamic Discipline: constraints on input timing

**Synchronous 1-clock logic:**
- Simple rules for sequential circuits
- Yields clocked circuit with $T_S$, $T_H$ constraints on input timing

**Finite State Machines**
Next Lecture Topic!
Computer Aided Design Tools

MICROWIND Tool Design Flow

Contact:
Sales: sales@microwind.net
Support: support@microwind.net

SPICE Simulator (3rd Party)

Schematic Modeling
Analog & digital Library models

Digital Simulation
Verilog Extraction
SPICE Extraction

Tape out to FAB.

MICROWIND 3

Verilog File

Verilog Compiler
nanoLambda
Layout Editor

Constraints
Technology rule files

Analysis
DRC, ERC
Delay Analyzer
Crosstalk Analyzer
2D Cross section
3D Analyzer

Place & Route

Layout Extraction

Layout Conversion
SPICE, CIF

LTSpice
WindSpice

http://www.microwind.net