Clocking of VLSI Systems

(a) Moore machine

(b) Mealy machine

Figure 15.3 Moore and Mealy state machines

- LARGE DIGITAL BLOCKS ARE TYPICALLY MADE OF COMBINATORIAL LOGIC BLOCKS FOLLOWED BY REGISTERS TO LATCH THE DATA
- REGISTERS ARE TYPICALLY MADE UP OF D-FIIP FLOPS (SMALLER AND FASTER THAN JK FLIP FLOPS, ETC.)

Figure 15.2 Timing in a DFF

- GATE DELAY OF FLIP-FLOP = Δt
- DATA FLOW IS SYNCHRONIZED TO 1 OR MORE CLOCKS
CLOCKS

Figure 15.1 Ideal clocking signal

- CLOCKS CAN BE 1 PHASE $\phi$, OR 2 PHASE, OR EVEN MORE!

Figure 15.6 Complementary clocks

- MOST COMMON VLSI CLOCKING SCHEME USES CLOCK & $\overline{CLOCK}$
- REAL CLOCKS HAVE RISE & FALL TIMES (NOT STEP FUNCTIONS)

Figure 15.15 Clocking waveforms with finite rise and fall times
**CLOCK GENERATION**

**Figure 15.36** Phase-locked loop (PLL) stabilization circuit

- PLLs are usually used to synchronize the chip's clock to an external board clock.
- On-chip clock may be >> higher frequency than external reference clock.
- PLL = "phase locked loop" = complex analog block

**Figure 15.10** Clock generation circuit

- Once a single clock is available, CLK & CLE can easily be created.
- But, simple inverter CKTS yield "clock skew"

**Figure 15.11** Clock skew
**Figure 15.36** Skew minimization circuit

- **TO MINIMIZE SKEW BETWEEN CLK & \( \bar{CLK} \), SCALE EACH INVERTER CHAIN TO EQUALIZE DELAYS \( \Rightarrow \) DESIGN SO THAT \( \alpha_1 = \alpha_2 \)

- **BUT, CAN STILL GET SKEW FROM DIFFERENCES IN THE DISTRIBUTION LINES** (MORE ON THIS LATER...)

**Figure 15.37** Inverter-based clock generation circuit
**Figure 15.19** A dual-clock finite-state machine design

- **Some VLSI digital blocks use 2, 3 or more clock phases (less common)**
- "Non-overlapping" clocks sometimes used to get "break before make" (only 1 clock on at a time)

**Figure 15.17** Dual non-overlapping clocks

"Non-overlap time" can be tuned by the number of inverter delays

**Figure 15.40** Circuit for producing non-overlapping clocks
**Clock Distribution**

![Clock Distribution Diagram](image)

**Figure 15.33** Clock distribution to on-chip modules

![Clock Skew Diagram](image)

**Figure 15.34** Example of clock skew

- **If clock lines are distributed to various blocks on a chip in a simple fashion,**
  - **SKREW!**

- **Skew occurs because of differences in capacitive loading and distribution line length for a metal line:**
  - (neglects fringe capacitance)
  - \[ R = R_s \left( \frac{1}{W} \right) \]
  - \[ C = W L C_{ox} \]

  \[ R C = \left[ R_s \left( \frac{1}{W} \right) \right] [W L C_{ox}] \]

  \[ \gamma = R C = (R_s C_{ox}) L^2 = B L^2 \]

  \[ \Delta t_i = B \left( L_B^2 - L_A^2 \right) \]

  - \( B \) is a constant for a given metal (M1, M2, etc.)
CLOCK DISTRIBUTION (CONT.)

- One way to improve skew is to use a single point clock driver.

* But, still get skew because of differences in line length from buffer to end of line.

A better way is to use a "clock tree" where the clock travels an equal distance to each point.

- Can also repeat the "H" pattern at different levels of hierarchy.

2 Problems:
1) Long lines give long RC time constants for slow clock edges (need to add buffers).
2) Can't always equalize line lengths, due to various size circuit blocks.

Figure 15.49 Single driver tree with multiple outputs

Figure 15.43 Geometrical analysis of the letter "H"

Figure 15.44 Macro-level H-type distribution tree

- The lengths and electrical characteristics must be the same for every clock path to produce the desired effect, and
- The load capacitance at every receiver point must be the same.
**CLOCK DISTRIBUTION (CONT.)**

(a) Driver tree  
(b) Application to H-tree

**Figure 15.45** Driver tree arrangement

- **ADD BUFFERS (INVERTERS) IN A "TREE" FASHION**  
  ➔ **KEEPS CLOCK EDGES SHARP, EVEN WITH LONG LINES AND LARGE CAPACITIVE LOADS**

- **CLOCK TREE DOESN'T HAVE TO BE AN "H" GEOMETRY, AS LONG AS ALL DELAYS ARE EQUAL.**

**Figure 15.47** Distribution scheme with equivalent driver segments

- **CAN SCALE EACH BUFFER ACCORDING TO THE LOAD IT DRIVES**  
  ➔ **ALL BUFFERS DON'T WANT TO BE IDENTICAL!**
CLOCK DISTRIBUTION (CONT.)

"Clock tree insertion" is typically done during place & route, after initial placement is decided (and thus routing distances and loads).

- To equalize the delays through a clock tree, must take into account the gate loading on each "branch", as well as the lines' RC delays.

- This can be done for a non-symmetrical clock tree, but tough!

Figure 15.46 Driver tree design with interconnect parasitics

Figure 15.48 Electrical circuit for a non-symmetrical distribution
Asynchronous Clocking

- Different parts of chip use different clock phases
  → Skew only matters over a smaller distance
  → Interface blocks required between clock regions

- Operates using "hand shaking" between blocks
  → Data "requests" & "acknowledgement" signals

![Figure 15.50 Asynchronous system clocking](image)

![Figure 15.51 Operation of a self-timed element](image)

- Seldom used on a single VLSI chip, mainly due to tool issues
- Often used at system level (chip-to-chip)
"Pipelining" of Data Paths

- Logic chains between input & output registers
  → Clock frequency is limited by the delay through the longest logic chain

- But, most circuits in a given logic chain are only active for a short period as the data "ripples" through the logic
  → Can increase the clock frequency by inserting more registers!

**Figure 15.28** Logic chains in a clocked system

**Figure 15.29** Circuit activity in a logic cascade

**Figure 15.30** Progression times in the logic cascade
PIEPLINING OF DATA PATHS (CONT.)

Figure 15.31 A 4-stage pipeline

- BREAK LOGIC CHAIN INTO SMALLER PARTS, SO THAT EACH LOGIC BLOCK STAYS ACTIVE MORE
  - HIGHER CLOCK FREQUENCY POSSIBLE
  - HIGHER "THROUGHPUT" OF DATA
  - BUT, LATENCY REQUIRED TO "FILL" & "FLUSH" THE PIPE

Figure 15.32 Pipeline with positive edge- and negative edge-triggering

- CAN ALSO LATCH DATA ON BOTH + AND - CLOCK EDGES
  - DOESN'T INCREASE DATA THROUGHPUT
  - DOES ALLOW A LOWER FREQUENCY CLOCK TO BE USED (EASIER DISTRIBUTION) BUT NOW DUTY CYCLE MATTERS!
    - % OF TIME CLOCK IS HIGH
**LOGIC TIMING**

**Figure 15.26** Basic pipelined stage for timing analysis

- **Each flip-flop has**
  1. A "clock-to-Q" delay = $\tau_{FF}$
  2. A "setup" time = time data must be present before clock edge
  3. A "hold" time = time data must be present after clock edge

**Figure 15.27** Waveform quantities for timing analysis

- **Clock period must be long enough to allow for flip-flop & logic delays, as well as set-up time and clock skew**
  
  \[ T > \tau_{FF} + \tau_D + \tau_{SU} + \tau_s \]  
  
  \[ \uparrow \quad \uparrow \quad \uparrow \quad \uparrow \]  
  
  Flop delay logic delay setup clock skew  
  
  **Limits how much logic can go between flops!**

- **If logic block has a short delay, look out for "race conditions" where the next flop's input data changes before the hold time =&gt; wrong data can be latched!**
Clocking of Dynamic Logic Cascades

Figure 15.9 A clocked cascade

- Data is passed from stage to stage on alternate clock phases

Figure 15.13 Shift register circuit

- Dynamic, so subject to leakage effects which limit minimum clock speed

- Can provide feedback around latches to hold charge, but then the circuit isn't dynamic anymore - what's the point?

Figure 15.16 Static shift register design
CLOCKING OF DOMINO LOGIC CASCADES

Figure 15.22 Operation of a domino logic stage

- "PRECHARGE" AND THE "EVALUATE" CLOCK PHASES
- EVALUATE PHASE MUST BE LONG ENOUGH FOR THE DATA TO RIPPLE THROUGH ALL STAGES

Figure 15.23 A dynamic logic cascade

Figure 15.24 Timing sequence in the domino cascade
**DUE TO THE HIGHLY REPEATED NATURE OF THE LOGIC, CAN LOOK AT ONE BIT "SLICE"**

**CAN BUILD ONE "SLICE", AND THEN BUILD A MULTI-BIT CIRCUIT BY STACKING THEM**

**CAN ALSO HAVE 2, 4, ETC. BIT SLICES**
**VLSI System Techniques (Cont.)**

(a) Basic system  
(b) Cache modification

**Figure 15.50** Adding cache memory

- ON-CHIP MEMORY: "CACHE" USED FOR BOTH DATA AND INSTRUCTIONS, TO SPEED OPERATION
- AVOIDS LONG OFF-CHIP DELAYS TO MAIN MEMORY

**Figure 15.57** Block diagram of a dual-issue superscalar machine

- MODERN CPU: PROCESS > 1 INSTRUCTION AT ONCE
- PARALLEL PROCESSING USED FOR ULTRA HIGH SPEEDS!

**Figure 15.58** Regular patterning in a parallel processing network