Lecture 19
Dynamic Logic - Adders
(that is – wrap-up)

Administrative Stuff

- Hw 6 due on Th
- No lab this week
- Midterm 2 next week
- Project 2 to be launched week 12
Class Material

- Last lecture
  - Adders
- Today’s lecture
  - Wrapping up dynamic logic and adders
  - Intro into multipliers

Dynamic Logic
**Dynamic Gate**

Two phase operation
- **Precharge** (CLK = 0)
- **Evaluate** (CLK = 1)

**Some undesirable effects**
- Leakage
- Charge Sharing
- Clock Feedthrough
- Backgate Coupling
**Cascading Dynamic Gates**

Only $0 \rightarrow 1$ transitions allowed at inputs!

**Domino Logic**

Only $0 \rightarrow 1$ transitions allowed at inputs!
**Why Domino?**

Like falling dominos!

**Properties of Domino Logic**

- Only non-inverting logic can be implemented
- Very high speed
  - static inverter can be skewed, only L-H transition
  - Input capacitance reduced – smaller logical effort
Designing with Domino Logic

The first gate in the chain needs a foot switch. Precharge is rippling – short-circuit current. A solution is to delay the clock for each stage. Can be eliminated!

Footless Domino

Inputs = 0 during precharge
**Differential (Dual Rail) Domino**

\[ \text{Out} = \overline{A}B \]

A
B
Clk
\[ \text{Out} = \overline{A}B \]

\[ \text{out} \]
\[ \text{on} \]

\[ \text{Solves the problem of non-inverting logic} \]

**np-CMOS**

\[ \text{In1} \]
\[ \text{In2} \]
\[ \text{In3} \]
\[ \text{Clk} \]
\[ \text{PDN} \]
\[ \text{Out1} \]

\[ \text{In4} \]
\[ \text{In5} \]
\[ \text{Clk} \]
\[ \text{PUN} \]
\[ \text{Out2} \]
(t to PDN)

\[ \text{Only 0} \rightarrow 1 \text{ transitions allowed at inputs of PDN} \]
\[ \text{Only 1} \rightarrow 0 \text{ transitions allowed at inputs of PUN} \]
NORA Logic

WARNING: Very sensitive to noise!

Adders
Adder Delays - Comparison

C_{o,k} = f(A_k, B_k, C_{o,k-1}) = G_k + P_k C_{o,k-1}

LookAhead - Basic Idea
**Look-Ahead: Topology**

Expanding Lookahead equations:

\[ C_{n-k} = G_k + P_k(G_{k-1} + P_{k-1}C_{n-k-2}) \]

All the way:

\[ C_{n-k} = G_k + P_k(G_{k-1} + P_{k-1}(... + P_1(G_0 + P_0C_{i0}))) \]

---

**Logarithmic Look-Ahead Adder**

Flow diagram showing the logarithmic look-ahead adder with:

- \( t_p \sim \log_2(N) \)
- \( t_p \sim N \)
Carry Lookahead Trees

\[
C_{0,0} = G_0 + P_0 C_{i,0}
\]

\[
C_{0,1} = G_1 + P_1 G_0 + P_1 P_0 C_{i,0}
\]

\[
C_{0,2} = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{i,0}
\]

\[
= (G_2 + P_2 G_1) + (P_2 P_1)(G_0 + P_0 C_{i,0}) = G_{2:1} + P_{2:1} C_{0,0}
\]

Can continue building the tree hierarchically.

Tree Adders

16-bit radix-2 Kogge-Stone tree
Tree Adders

16-bit radix-4 Kogge-Stone Tree

Sparse Trees

16-bit radix-2 sparse tree with sparseness of 2
Tree Adders

Brent-Kung Tree

Example: Domino Adder

Propagate

Generate
Example: Domino Adder

Example: Domino Sum
Next Lecture

- Multipliers
- Other datapath operators

Multipliers
The Binary Multiplication

\[ Z = \tilde{X} \cdot Y = \sum_{k=0}^{M \cdot N - 1} Z_k 2^k \]

\[ = \left( \sum_{i=0}^{M-1} X_i 2^i \right) \left( \sum_{j=0}^{N-1} Y_j 2^j \right) \]

\[ = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} X_i Y_j 2^{i+j} \]

with

\[ X = \sum_{i=0}^{M-1} X_i 2^i \]
\[ Y = \sum_{j=0}^{N-1} Y_j 2^j \]
The Array Multiplier

The $M \times N$ Array Multiplier — Critical Path

$$t_{mult} + (M-1)t_{carry} + (N-1)t_{sum} + (N-1)t_{and}$$
The Carry-Save Multiplier is a parallel multiplier designed to reduce the delay and area compared to a conventional multiplier. The design uses a combination of half adders (HAs) and full adders (FAs) to perform the multiplication operation efficiently. The formula for the delay of the Carry-Save Multiplier is given by:

\[ t_{\text{multi}} = (N-1)t_{\text{carry}} + (N-1)t_{\text{and}} + t_{\text{merge}} \]

The Multiplier Floorplan shows the physical layout of the multiplier, with each cell representing a specific functionality, such as a HA Multiplier Cell, FA Multiplier Cell, or Vector Merging Cell. The X and Y signals are broadcasted through the complete array to ensure parallel processing.
Wallace-Tree Multiplier

Multipliers — Summary

- Optimization Goals Different Vs Binary Adder
- Once Again: Identify Critical Path
- Other possible techniques
  - Logarithmic versus Linear (Wallace Tree Mult)
  - Data encoding (Booth)
  - Pipelining

FIRST GLIMPSE AT SYSTEM LEVEL OPTIMIZATION