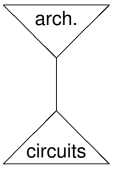
Logic Circuit Styles

Ken Stevens

Logic Circuit Styles

- Two main aspects determine quality of design
  - Architecture
  - Circuits



“Wine Goblet”

Logic Circuit Styles

- Topology is a major factor in the quality of circuits
- Major freedom and creativity of CMOS canvas
- How to use topology to meet design goals is critical
- For example
  - Sample action potentials in cat’s brain
  - Test and regulate temperature of exhaust from manifold
  - Sample and store optical wavelengths for video/image capture
  - . . .

Design Point

- Choose topology that best meets design targets
- Primary metrics may be:
  - Performance
  - Power
- Secondary metrics may be:
  - cost
  - robustness
  - testability
  - TTM (Time To Market/Money)
  - Area
  - Gain
  - . . .

Logic Choice

- Logic family should be based on:

1. Knowledge of circuit alternatives and characteristics
2. Matching of design requirements to circuit topology (both FET and interconnect)
3. Interfacing issues (timing, voltage, etc.)
4. CAD support: HLD, design, synthesis, ATPG, Formal Verification, . . .
5. Products requirements: performance, power, area, reliability, TTM, . . .

General Logic Families

- Two orthogonal “classes”
- Logic Style
  1. Static logic
  2. Dynamic logic
- Functional Style
  1. Combinational
  2. Sequential

	static	dynamic
combinational		
sequential		

Static Logic

- The standard logic
- Complementary *on-set* and *off-set* logic
- Boolean function representation
- Transitions not explicitly specified or optimized
  - Example: $o = \overline{a+b}$
    - on-set: $\{\overline{a}\overline{b}\}$
    - off-set: $\neg$on-set $= \{a, b\}$

Dynamic Logic

- *Transition* logic
- Specifies *set* and *reset* transitions
- Static states not explicitly specified or optimized
- Efficient datapath (logical effort) logic
  - Example: rendezvous or *c-element*
    - Set function: $\overline{a}\overline{b} \mapsto o\uparrow$
    - Reset function: $ab \mapsto o\downarrow$
    - What happens in states $\notin$ set $\cup$ reset ???

Dynamic Logic

- *Transition* logic
- Specifies *set* and *reset* transitions
- Static states not explicitly specified or optimized
- Efficient datapath (logical effort) logic
  - Example: rendezvous or *c-element*
    - Set function: $\overline{a}\overline{b} \mapsto o\uparrow$
    - Reset function: $ab \mapsto o\downarrow$
    - What happens in states $\notin$ set $\cup$ reset ???
      Output is FLOATING!

Combinational Logic

- Function is fully specified by inputs
- Our most common design style
  - Example: $f = a + b + c$

c \ ab	00	01	11	10
0	0	1	1	1
1	1	1	1	1

Sequential Logic

- Current state depends on previous state
  - Example: truth table for inverting c-element:

inputs	**output**
00	1
01	q
10	q
11	0

- Circuits typically contain **feedback**
- **Timing** may be critical to correct behavior
- Typical design mode for **asynchronous** circuits
  - why?

Sequential Logic

- Current state depends on previous state
  - Example: truth table for inverting c-element:

inputs	**output**
00	1
01	q
10	q
11	0

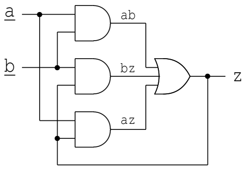
- Circuits typically contain **feedback**
- **Timing** may be critical to correct behavior
- Typical design mode for **asynchronous** circuits
  - why? Async implements sequential state machines

Sequential Logic

- How do you implement sequential logic with combinational gates?

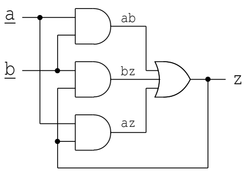
Sequential Logic

- How do you implement sequential logic with combinational gates?
  - Feedback



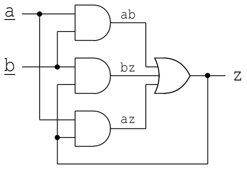
Sequential Logic

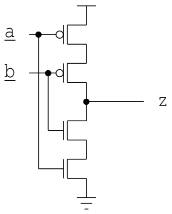
- How do you implement sequential logic with combinational gates?
  - Feedback
- How do you implement sequential logic with dynamic gates?



Sequential Logic

- How do you implement sequential logic with combinational gates?
  - Feedback
- How do you implement sequential logic with dynamic gates?
  - You can get it for free!
  - Dynamic gates are naturally sequential





Logic Circuit Styles

- The following will be discussed:

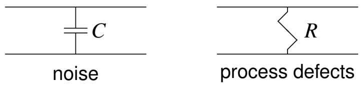
1. **Static combinational logic**
2. Pass transistor logic
3. Dynamic logic
4. Self-timed and asynchronous logic
5. Self-resetting (pulse) logic

Static Combinational Logic

- Used in **every** design!
- Main workhorse of our chips
- All commercial libraries have these cells
- Direct support based on CBD (cell based design), P&R, . . .

Key Circuit Effects

- Static Logic has good characteristics for key effects:
  - Reasonable power consumption (reactive, activity factors)
  - Simple & direct synthesis
  - High noise immunity
  - Timing validation (most common model)
  - Tolerant to process variation (time independent)
  - Tolerant to minor defects (high resistive bridging faults)
  - Scalable (shrinkable) to different process nodes
  - Testability (stuck at models, fault propagation, etc.)

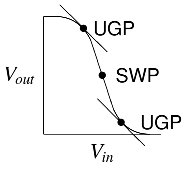


Additional Important Circuit Effects

- These are also important to consider for all logic families
  - Transfer function
  - Delay variability
  - Switching thresholds
  - Body effect and sensitivity to signal ordering
  - Miller effect
  - Glitching
  - Crossover current

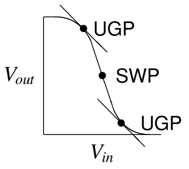
Transfer Function

- Useful for performance and noise characterization
  - Switch-point is location on transfer function where $V_{out} = V_{in}$
  - Unity Gain Points determine signal attenuation and amplification voltages
- What happens to performance and noise when we skew SWP and UGP locations?



Transfer Function

- Useful for performance and noise characterization
  - switch-point is location on transfer function where $V_{out} = V_{in}$
  - Unity Gain Points determine signal attenuation and amplification voltages
- What happens to performance and noise when we skew SWP and UGP? Response *design dependent* (can be worse!)

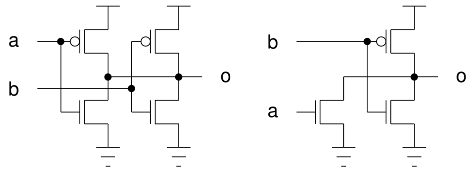


Performance and Noise Metrics

- Electrical gain
  - FO-4 delay
  - Logical effort: $f = gh \approx 4$
  - Too high – slow operation
  - Too low – wasted power and area
- *Beta ratio*, or *gate skew*
  - Balancing pullup/down currents:
    - equal rise/fall times
    - switch point at $V_{DD}/2$
    - improved noise immunity
  - Skewed sizing:
    - improved performance in one direction

Gate Design Quiz

Following are a couple of gates:



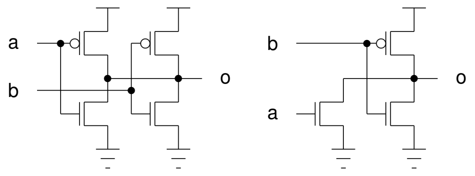
Are these static or dynamic implementations?

What is the logic function of the above gates?

Hint: The function is the same for both.

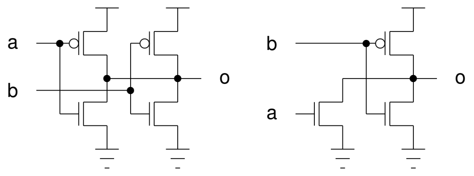
Ratioed NORs

- “Ratioed NOR” gates are much faster than static NORs
- These gates have relatively high crossover current in some states (which)?
- How do you size this device?
- What are the noise properties?



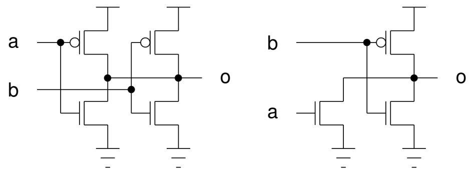
Ratioed NORs

- “Ratioed NOR” gates are much faster than static NORs
- These gates have relatively high crossover current in some states (which)? when inputs not equal (1,0 or 0,1)
- How do you size this device?
- What are the noise properties?



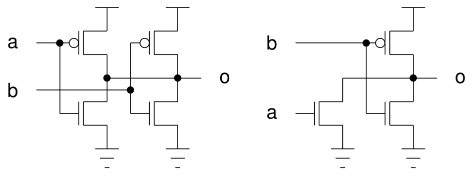
Ratioed NORs

- “Ratioed NOR” gates are much faster than static NORs
- These gates have relatively high crossover current in some states (which)? when inputs not equal (1,0 or 0,1)
- How do you size this device? weak p to n ratio
- What are the noise properties?



Ratioed NORs

- “Ratioed NOR” gates are much faster than static NORs
- These gates have relatively high crossover current in some states (which)? when inputs not equal (1,0 or 0,1)
- How do you size this device? weak p to n ratio
- What are the noise properties? output may not be gnd



Gate Delay Variability

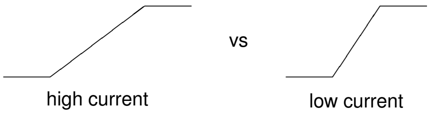
- Internal parasitic values
- Signal switching order
- Multiple input switching
- MCF (Maximum capacitive Coupling Factor)
- Fanout load on gates
  - With inversion channel, effective t_{ox} increases, decreasing cap from gate to substrate
    - This gives different gate cap based on the logic state of the gate, and results in a varying capacitance as a gate switches!!!
- Body effect

Glitching and Power

- **Activity factor** has direct effect on energy
  - Idle circuit only consumes leakage power $P_{\text{off}} = W_{\text{tot}} V_{\text{dd}} I_{\text{off}}$
- Combinational logic is *reactive*
- Transient glitches propagate
- Approximately 15% of the power is due to *dynamic hazards*

Crossover or Crowbar Current

- Current rushing between power and ground as p- and n-type devices are both on
- Dependent on slew rate of input signal
- Dependent on threshold of gate
- Consumes approximately 10% of the power in static gates with proper slopes



Classifications of Static Gates

- Two flavors:
  - Simple
    - logical: single minterm
    - structural: no internal branching
  - Complex
    - logical: SOP format with multiple minterms
    - structural: *combination of multiple simple gates*

Complex vs Simple Gate Design

- Classical design optimization problem:
  - Where is optimal benefit derived from jointly optimizing or partitioning logic functions or blocks?
  - What characteristics can be exploited or avoided?
  - What is local minima?
- Q: How does scaling effect our optimal choices?

Complex vs Simple Gate Design

- Classical design optimization problem:
  - Where is optimal benefit derived from jointly optimizing or partitioning logic functions or blocks?
  - What characteristics can be exploited or avoided?
  - What is local minima?
- Q: How does scaling effect our optimal choices?
  Reduces the number of series transistors to output

Simple Static Gate Design

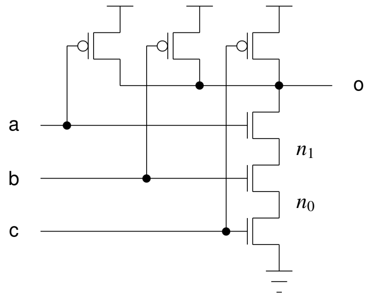
- SOP minterm: $\overline{f} = abc$
  - AND function
  - $\rightarrow$ n-type transistors in series
- POS minterm: $\overline{f} = a + b + c$
  - OR function
  - $\rightarrow$ n-type transistors in parallel
- Note gates are naturally *inverting*
- Apply negative function directly to n-type devices
- Apply logical or structural duality transformation for p-types

Simple Static Gate Design

- Example: $\overline{f} = abc$
- Structural Algorithm
  1. Map minterm to n-type device
  2. P-types are structural dual of n-type tree
     - Series devices are in parallel
     - Parallel devices are in series
- Logical Algorithm
  1. Map minterm to n-type device
  2. Apply DeMorgan's theorem to minterm
  3. Implement p-type tree from derived equation

Simple Static Gate Design

- $\overline{f} = abc$
- Show structural and logical derivations



Structural Complex Static Gate Design

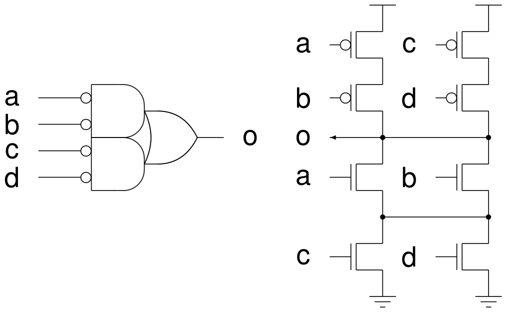
- Structural Derivation Algorithm 1
  1. Create p-type and n-type trees for all minterms
  2. Combine p-type and n-type minterms in series (SOP) or parallel (POS)
  3. Optimize network by sharing common transistors
- Structural Derivation Algorithm 2
  1. Create n-type (p-type) trees for all minterms
  2. Combine n-type (p-type) minterms in series (SOP) or parallel (POS)
  3. Optimize network by sharing common transistors
  4. Structurally take the dual of the network for the p-type (n-type) tree

Logical Complex Static Gate Design

- Logical Derivation Algorithm
  1. Create minimized n-type pulldown logical representation using associative and distributive laws
  2. Map logic directly to n-type structure
  3. Apply DeMorgan's theorem to function
  4. Create minimized p-type pullup logical representation using associative and distributive laws
  5. Map logic directly to p-type structure

Complex Static Gate Design

- Example: $\overline{f} = ac + ad + bc + bd$



Logic Circuit Styles

- The following will be discussed:
  1. Static combinational logic
  2. **Pass transistor logic**
  3. Dynamic logic
  4. Self-timed and asynchronous logic
  5. Self-resetting (pulse) logic

Pass Transistor Logic

- Used in *almost* every design
  - DRAM and SRAM cells
- Fine example of understanding capabilities and application needs
- Applied in other ways as well:
  - ALUs
  - Muxing
  - Interconnect
  - . . .

Pass Transistor Design

- Single-ended
  - Only pull-down function
- Differential
  - Pulldown function with dual
  - Both trees implemented in n-FET
  - *Sense amp* required for differential sensing
- Optimization point favors high complexity over simple functions

Pass Transistor Logic Characteristics

- Potential significant performance advantage
- Linear resistive delay
- Can admit *very* complex functions in single DCN
- Pull up and down through same source transistor network
- Best differential performance with reset voltage between power rails

Pass Transistor Logic Characteristics

- Can use *sense amp* (SA) for improved performance
  - Static power draw when active
    - Pulsed clocking
  - Limited active range
    - Normally skewed low
      - n-type logic
      - low voltage or ground as common reset reference

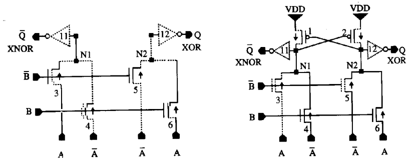
Pass Transistor Logic Characteristics

- High sensitivity to noise:
  - Highly sensitive to differential noise
  - **Common mode** noise rejection
- Interfacing issues if not using SA
  - High voltage level restoration
  - Limited fan-in and fanout loads
  - Body effect

Complementary Pass-Gate Logic

- Fast, common style for Function block pass logic
- Three varieties:

1. Standard (p-type in driver not shut off)
2. Keepers (dynamic logic inspired)
3. Cross coupled (SA integration)



(a) Standard CPL

(b) Cross-coupled CPL

Figure 7.5 Complementary pass-gate logic implementations of XOR: standard CPL (a); cross-coupled CPL (b) [6].

Complementary Pass-Gate Logic

- Clear difference in plots between standard and SA inspired
  - Full voltage generation
  - Reduced performance due to extra capacitance
- **Low threshold** transistor options for pass-gates
  - Noise and leakage ramifications

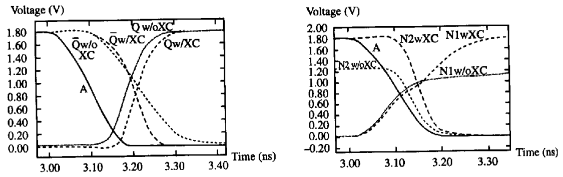


Figure 7.6 CPL operation of circuits shown in Fig. 7.5; external (a) and internal (b) node behavior [6].

11 → 01 transition on XOR

LEAP Logic

- ◆ Random logic library, bidirectional cells
  - ◆ Full functional block synthesis
  - ◆ Single-ended methodology, “full” swing outputs
  - ◆ Keeper to shut off p-FET in inverter
  - ◆ Multiplexor based synthesis idea

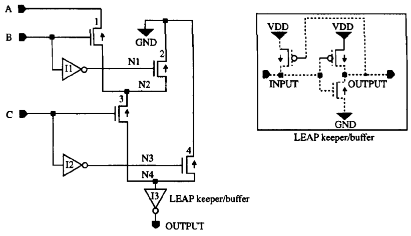


Figure 7.7 LEAP pass-gate logic of NAND function [6].

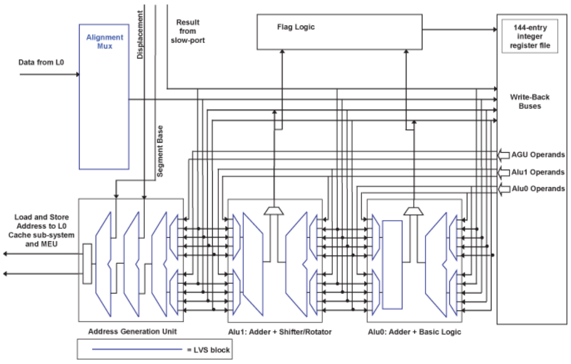
LVS Case Study: Double Pumped ALUs

Intel Pentium® 4 $2\times$ frequency low voltage swing integer core:[1]

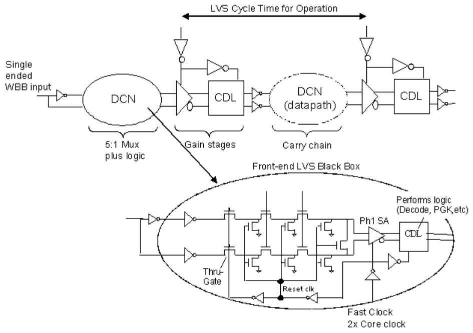
- Performance benefit from halving the latency of integer operations
  - ◆ Reduce wait and impact of instruction dependencies
  - ◆ Improved bypass operations
  - ◆ Store to load forwarding mechanism
- Greater complexity than entire Pentium Pro® design (6.8 million non-array transistors)
- Pulsed clocks

[1] US Patent 6,557,149: Algorithm for finding vectors to stimulate all paths and arcs through an LVS gate”, Morrise & Stevens.

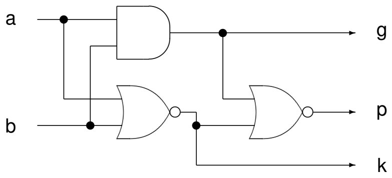
Low Voltage Swing Logic Bypass Architecture



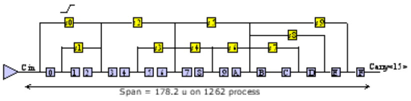
Low Voltage Swing Core

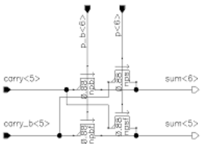


Manchester Carry Chain

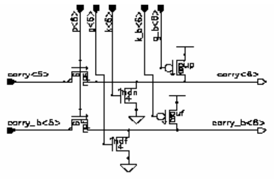


Low Voltage Swing Adder

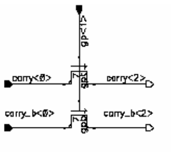




LVS XOR CELL

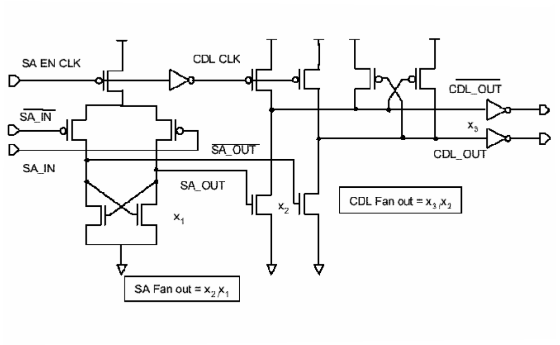


LVS PGK CELL



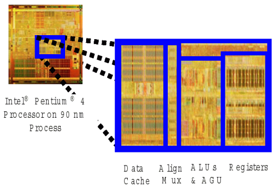
LVS SKIP CELL

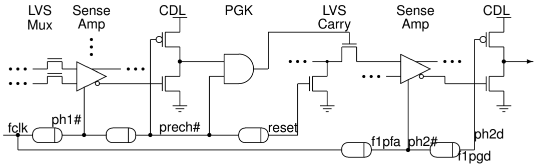
Sense Amp and Drive Gain



Low Voltage Swing Validation

1. 6.8 GHz in 3.4 GHz product
2. 6 transistor stages in 2 FO4 delays
3. Pulse clock
4. Complex path calculation
5. Common mode noise





Logic Circuit Styles

- The following will be discussed:

1. Static combinational logic
2. Pass transistor logic
3. **Dynamic logic**
4. Self-timed and asynchronous logic
5. Self-resetting (pulse) logic

Dynamic Logic

- *Transition* logic
- Specifies *set* and *reset* transitions
- Static states not explicitly specified or optimized
- Efficient datapath (logical effort) logic
  - Example: rendezvous or *c-element*
    - Set function: $\overline{a}\overline{b} \mapsto o\uparrow$
    - Reset function: $ab \mapsto o\downarrow$
    - What happens in states $\not\in$ set $\cup$ reset ???

Dynamic Logic

- *Transition* logic
- Specifies *set* and *reset* transitions
- Static states not explicitly specified or optimized
- Efficient datapath (logical effort) logic
  - Example: rendezvous or *c-element*
    - Set function: $\overline{a}\overline{b} \mapsto o\uparrow$
    - Reset function: $ab \mapsto o\downarrow$
    - What happens in states $\not\in$ set $\cup$ reset ???
      Output is tristated

Dynamic Logic Characteristics

- Monotonic functions
  - Due to dynamic state, glitching inputs can be fatal
- Cannot assert both set and reset functions
  - Automatically controlled with “footed” logic. (“footed” will be defined)

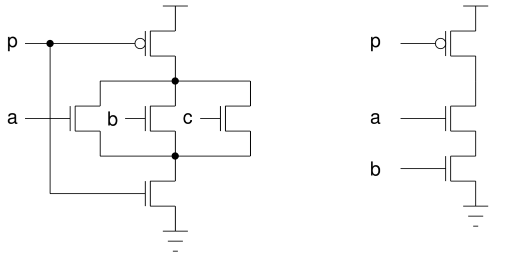
Dynamic Logic

- **Domino logic** is most common form of dynamic logic
- Used in two forms:
  1. Precharge logic
     a. clocked precharge logic
     b. self-resetting logic
  2. Set reset functions

Domino Logic

- Optimizes the pullup (p-FET) trees:
  - *Only a single p-type pullup!*
- Allows complex pulldown function.
- Two classes
  - Footed
    - The pullup signal is ANDed with pulldown function
  - Unfooted
    - The pullup signal does not exist in the pulldown function

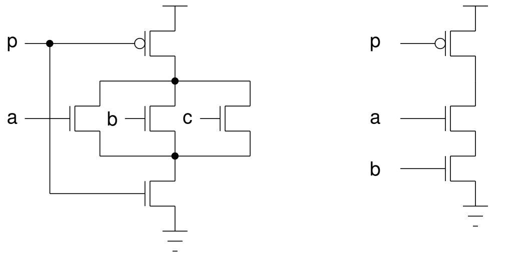
Domino Logic



footed, $\overline{f} = a + b + c$ unfooted, $\overline{f} = ab$

- Do we want footer at top or bottom of stack? Why?

Domino Logic



footed, $\overline{f} = a + b + c$ unfooted, $\overline{f} = ab$

- Do we want footer at top or bottom of stack? Why?
  Bottom, to minimize parasitics and body effect.

Domino Advantage

Can you have a gate with a logical effort less than one?

Domino Advantage

- Low data input load to output drive ratios (*gain*)

Gate	Static	Domino	Prech
2-NAND, unfooted	$4/3$	$2/3$	$2/3$
2-NAND, footed	$4/3$	$3/3$	$5/3$
3-NAND, unfooted	$5/3$	$3/3$	$2/3$
3-NAND, footed	$5/3$	$4/3$	$6/3$
2-NOR, unfooted	$5/7$	$1/3$	$2/3$
3-NOR, unfooted	$7/3$	$1/3$	$2/3$
3-NOR, footed	$7/3$	$2/3$	$4/3$

- What about logical effort of complex pulldown functions?

Domino Advantage

- Low data input load to output drive ratios (*gain*)

Gate	Static	Domino	Prech
2-NAND, unfooted	$4/3$	$2/3$	$2/3$
2-NAND, footed	$4/3$	$3/3$	$5/3$
3-NAND, unfooted	$5/3$	$3/3$	$2/3$
3-NAND, footed	$5/3$	$4/3$	$6/3$
2-NOR, unfooted	$5/7$	$1/3$	$2/3$
3-NOR, unfooted	$7/3$	$1/3$	$2/3$
3-NOR, footed	$7/3$	$2/3$	$4/3$

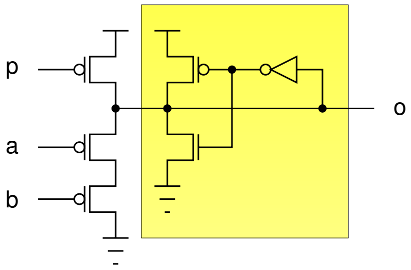
- What about logical effort of complex pulldown functions?
  Also MUCH improved!!!

Domino Disadvantages

- Only three main disadvantages:
  - Noise, noise, and noise
    - Due to: coupling, parasitic charge sharing, leakage
- Other argued disadvantages...
  - Energy consumption
    - Precharge activity factor (clocked modes)
  - Additional timing requirements

Dynamic Gate Noise

- **Keepers** can be used to mitigate this somewhat
  - Size of keeper limited by relation to set/reset stack strength
  - Nominally used to combat leakage current
  - Increases crowbar current of gate as it fights almost the whole transition.



Keepers

- Jam-latch keepers are W ratioed to approximately 5–20% of reset tree.
- Should feedback inverter be in the output path?

Keepers

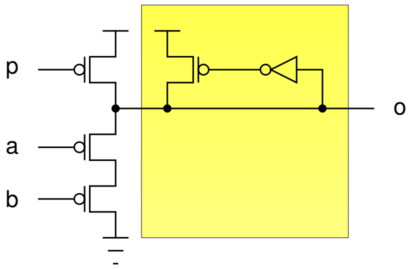
- Jam-latch keepers are W ratioed to approximately 5–20% of reset tree.
- Should feedback inverter be in the output path?
  NO, due to:
  - Coupling control
  - Latency
- Do we need to use both a p- and n-type keeper FET?

Keepers

- Jam-latch keepers are W ratioed to approximately 5–20% of reset tree.
- Should feedback inverter be in the output path?
  NO, due to:
  - Coupling control
  - Latency
- Do we need to use both a p- and n-type keeper FET?
  Depends
  - Usage of gate (precharge or set/reset)
  - N-type removed for many precharge logics.

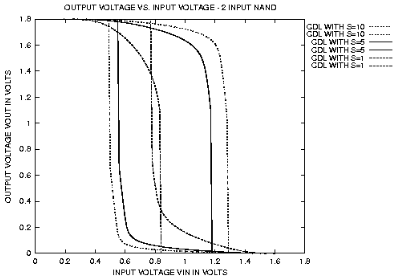
Dynamic Gate Noise

- **Keepers** can be used to mitigate this somewhat
  - Size of keeper limited by relation to set/reset stack strength
  - Nominally used to combat leakage current
  - Increases crowbar current of gate as it fights almost the whole transition.



Domino Advantage

Go figure - there actually *can* be a noise advantage with domino.



From research by Santhanam & Stevens on *Complementary Dynamic Logic.* (Related to Noise Tolerant Precharge NTP)

Clocked (Precharge) Domino Implications

- Clock connected to precharge
- Each domino input is low at start of evaluate phase
- Clock allows synchronization and latching
- Disadvantages
  - *Power:* precharge occurs independent of data activity factor
  - *Performance:* footed implementation adds series device
  - *Performance:* unbalanced evaluate and precharge phases
    - Evaluate dependent on logic depth
    - Concurrent precharge in constant time (two gate delays)
  - *CAD:* more difficulty to synthesize, place & route, etc.

Clocked (Precharge) Domino Configuration

- Alternate domino and static gates
- Optimize evaluation path
  - Always evaluate data pulldown in domino
  - Always evaluate data pullup in static
  - Therefore, skew logic:
    - Up to a factor of 6/1 sizing skew
    - Skew both static and domino
    - . . . but this decreases noise margins!
  - Therefore, very simple static gates

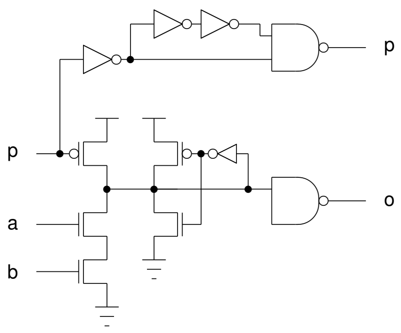
Timing for Precharge Domino

1. Precharge pulse sufficient to charge outputs
2. Falling input transitions require positive setup before precharge ends
3. Rising edge setup sufficient to discharge & latch output
4. Rising inputs require hold time to fully drive output

Clocked Unfooted Domino

- First stage normally footed
- Internal stages can be unfooted
  - Crowbar current serious reliability, power, noise issue
  - Avoid by delayed clock
    - Precharge clock delay $\geq$ logic precharge
    - Logic precharge skewed slow?
    - Symmetric delay effects evaluation delay

Clocked Unfooted Domino



unfooted, $\overline{f} = ab$

Dynamic Noise Sources

- Leakage current can be substantial in wide OR structures
- Charge sharing
  - precharge internal nodes
- Interconnect coupling
- Ground bounce

Dynamic Noise Sources

- **Dynamic Carrier Noise Injection**
  - Nodes driven below ground and above V_{DD}
  - Forward biases diodes
  - Injects minority carriers into substrate and well
- Collected by nearby diffusion
- Particularly onerous near pads an clock drivers
  - Pads always use “guard ring” for this purpose
- Effect reduces with voltage scaling due to diode forward bias threshold

Area & Leakage

- Dynamic logic uses less area than static:
  - PMOS tree replaced with single FET
  - NMOS tree can combine multiple gates
- Translates to substantially lower W_{tot} (leakage)

Dynamic Noise Sources

- **Miller and Backgate Coupling**

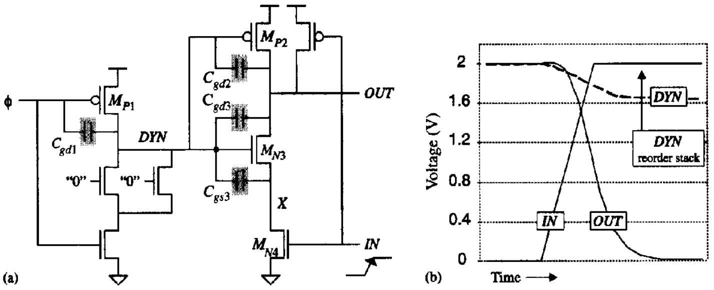
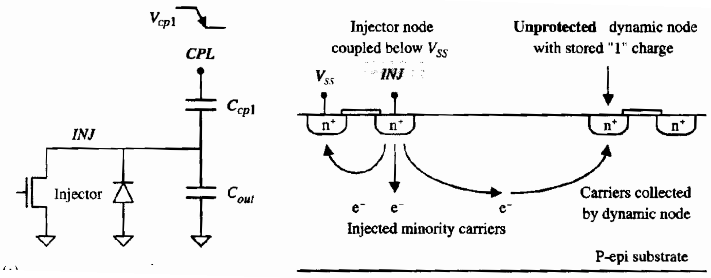
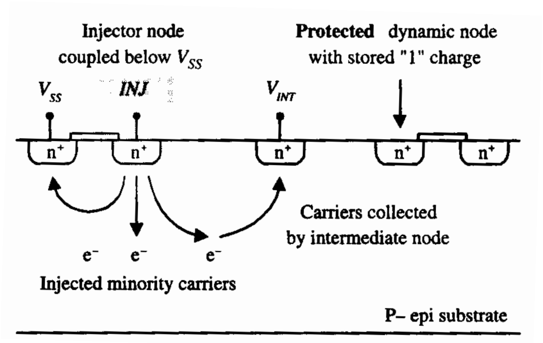


Figure 8.13 Back-gate coupling: (a) circuit diagram; (b) signal waveforms.

Dynamic Carrier Noise Injection

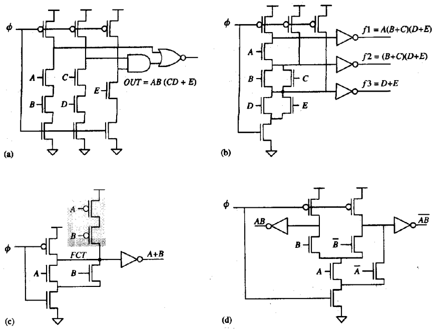


Dynamic Carrier Protection



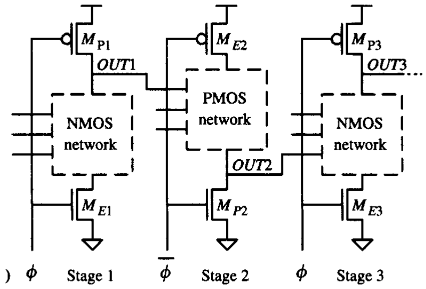
Other Domino Forms

- Compound Domino, Multiple-output domino, noise-tolerant precharge (NTP), dual-rail domino



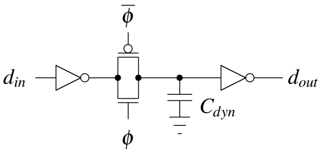
Other Dynamic Structures

- NORA Logic

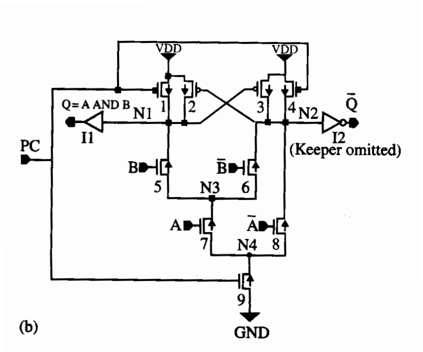


Other Dynamic Logic

- DRAM
- Dynamic Latches:



Dual Rail Domino



DCVS

- A combination of pass-gate logic and Domino
- True dual-rail domino
- Sense-Amp inspired

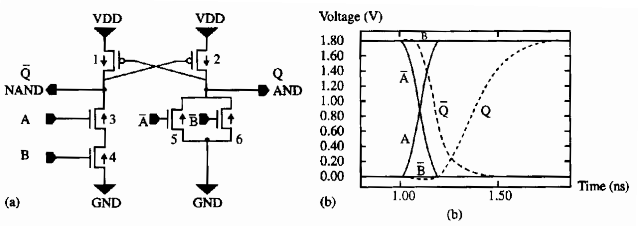


Figure 7.4 Basic DCVSL *AND* gate with PMOS cross-coupled loads [6].

Logic Circuit Styles

- The following will be discussed:

1. Static combinational logic
2. Pass transistor logic
3. Dynamic logic
4. **Self-timed and asynchronous logic**
5. Self-resetting (pulse) logic

Set Reset Domino

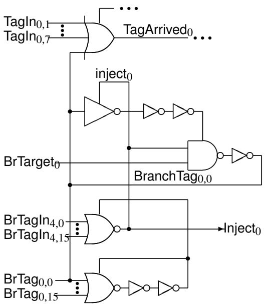
- Normally applied in Asynchronous Logic
  - Maps very well to sequential protocols
- Resolves clocked dynamic problems:
  - Not precharged
    - No phases - dependent on set/reset transition delays
    - Data activity factor ∝ active power
  - Sequential asynchronous synthesis quite mature

Asynchronous Domino Example

- Sequential circuit design natural for domino gates
- Ultra low power and high gain sequential design
  - Both active and leakage energy
  - Activity factor dictates active energy
  - No glitching
- Not required to have static gate between dominos
  - Based on transitions and logic levels

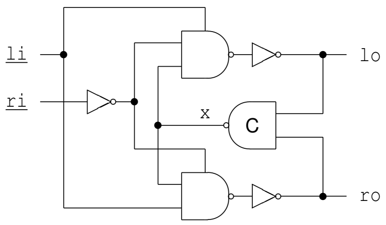
Asynchronous Domino Example

Asynchronous Pentium instruction length decoder branch control circuitry

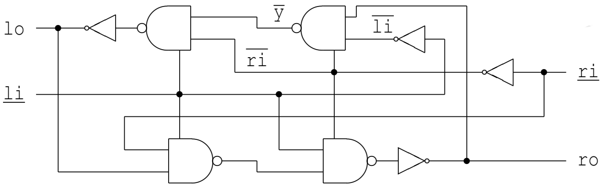


Asynchronous Domino Example

High performance pipeline controller

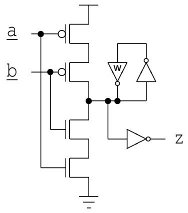


Domino FIFO controller



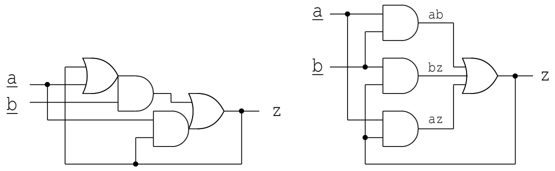
Non-Domino Precharge

- C-element
- Generalized C-element
  - Complicated multiple PMOS structure
  - Slower than domino
  - Hazard removal, reduced timing constraints



Static Asynchronous Logic

- Can implement with static gates
- Generally worse than with dynamic gates
  - slower
  - higher power
  - larger area



(a): SIC

(b): SC

Logic Circuit Styles

- The following will be discussed:

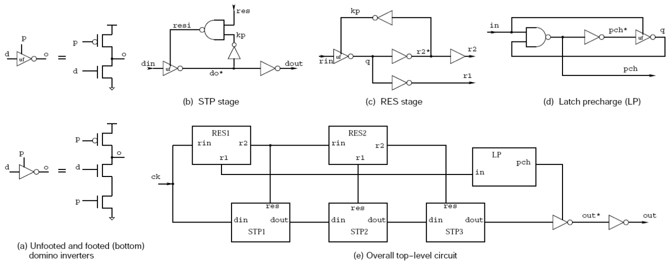
1. Static combinational logic
2. Pass transistor logic
3. Dynamic logic
4. Self-timed and asynchronous logic
5. **Self-resetting (pulse) logic**

Self-Resetting Domino

- Pulse-logic
- Combination between precharge and set/reset design
  - Self-precharged after each evaluation
  - Only precharges if data is set
    - allows energy ∝ activity factors
- Challenging design issues:
  - Pulse-width
  - Pulse filtering
  - Pulse ANDing

Intel “Self-Terminating Precharge”

Double Pumped ALU



(b) STP stage

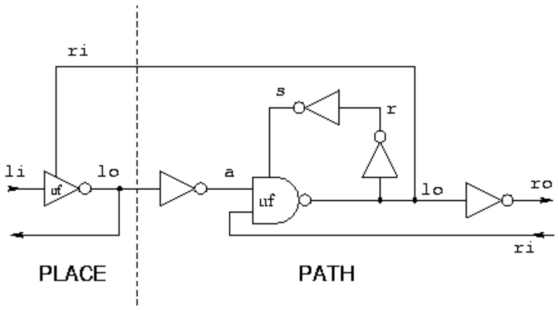
(c) RES stage

(d) Latch precharge (LP)

(a) Unfooted and footed (bottom) domino inverters

(e) Overall top–level circuit

Asynchronous GasP



Logic Circuit Styles Review

- key metrics for choosing logic design style
- difference between static and dynamic logic
- combinational vs sequential logic
- static logic characteristics
- circuit skewing
- ratioed gates
- sources of gate variability
- MCF

Logic Circuit Styles Review

- signal switching order
- fanout loads
- activity factor's effect on circuit power
- glitching
- crowbar current
- simple and complex gate transistor optimization
- static logic optimization point
- common pass transistor logic applications

Logic Circuit Styles Review

- single vs differential pass gate design
- pass gate optimization point
- sense amp
- pass gate noise: common mode
- threshold effect on pass gate logic
- monotonic nature of dynamic logic
- characteristics of domino logic
- footed vs unfooted domino

Logic Circuit Styles Review

- primary domino advantage and disadvantage
- keepers, ratio, p vs n-FET
- precharge domino configuration, disadvantages, skews, . . .
- dynamic logic noise source
- dynamic carrier noise injection
- async domino characteristics
- async domino advantages
- self-reset domino advantages and challenges

Review

- Compare logic styles in terms of:
  - gate capacitance (logical effort)
  - complementary p-type trees and their effect on power & performance
  - power (due to reactive nature of logic (activity factors))
  - glitching, crowbar power, voltage level restoration, etc.