BROWN

Low Power VLSI System Design

Lecture 6 & 7: Low Power Sequential Circuits & Multi-threshold CMOS (MTCMOS)

Prof. R. Iris Bahar
EN2912 E
September 25 & 27, 2017

BROWN

Seminar today

Abhishek Bhattacharjee
Department of Computer Science
Rutgers University

- **CIT 368**
- **12-1pm**

TITLE: Computer System for Neuroscience

Prof. Bhattacharjee will discuss the extreme energy needs of hardware used in brain implants, and the challenges posed by the computational and data requirements of large-scale brain modeling software.

BROWN

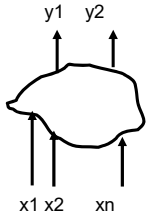
Homework #2

- You will need to use the CCV machines with Cadence tools for this assignment
- 3 tutorials are now posted on the course webpage:
  1. Login on to CCV machines Cadence setup
     - Note that some screen shots show info for 15nm technology → we are using 45nm technology
  2. Cadence CMOS transistor design (using Virtuoso)
  3. Cadence transient simulation (creating symbolic testbench)
- Please complete tutorial #1 by Wednesday, #2, #3 by Thursday
- Homework #2 will be posted on Wednesday
- Shanshan Dai will be helping out as a TA for this assignment
- TA hours Tuesday 3-5pm in B&H 191 (Computer Lab)

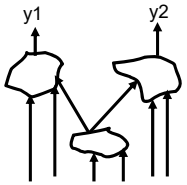
BROWN

Common Subexpression Extraction

y1 = ac + b'c + de
y2 = ae' + b'e' + e'f



t1 = (a + b')
y1 = t1c + de
y2 = t1e' + e'f



Low Power Logic Synthesis

- How can the sub-expression impact the power dissipation of the circuit?
- Remember $P=1/2\ \alpha CV^2 f_{clk}$
- Signal probabilities and transition densities of original nodes in circuit remain unchanged with factoring
- What changes?
  - Capacitances at output of driver gate of each node containing literals found in sub-expression
  - Cuts down switching activity of internal nodes in sub-expression term (term appears once instead of L times)

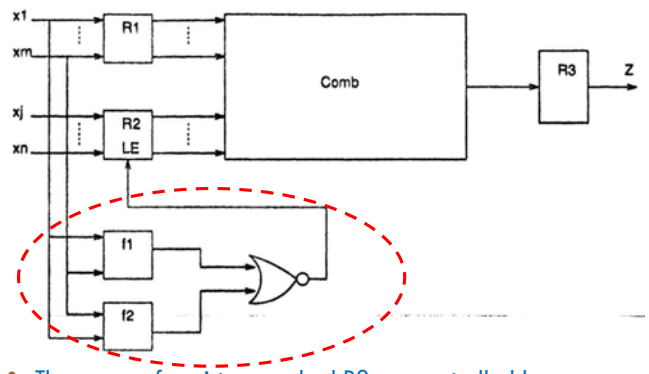
FSM and Combinational Logic Synthesis

- Combinational Logic:
  - Consider signal activity when selecting best common sub-expression to pull out during multi-level logic synthesis
    - Factor highest-activity common sub-expression out of all affected expressions
- Latches:
  - Consider conditions under which computation is needed
    - Pre-compute logic ahead of time and selectively prevent latch from updating
  - Consider likelihood of state transitions during state assignment
    - Minimize # signal transitions on present state inputs V

Precomputation-Based Optimization

- Basic idea:
  - Precompute circuit output logic values 1 cycle before they are needed
- Use precomputed information in next clock cycle to disable unneeded hardware,
  - Reduces switching activity
  - Disable via clock enable to state bit latches
- Must be careful: Precomputation hardware can add to area and lengthen clock period

Precomputation Architecture

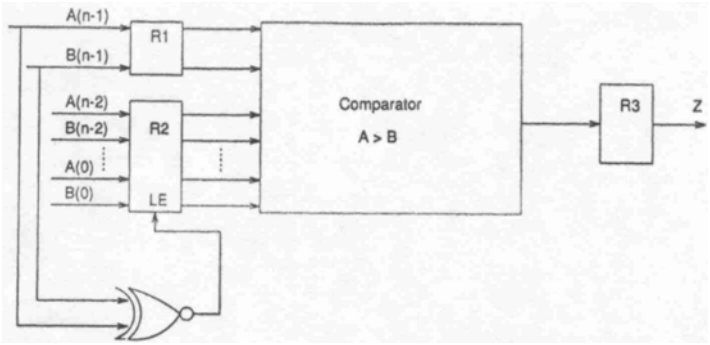


- The group of registers marked R2 are controlled by precomputation logic

Explanation

- $f_1 = 1 \rightarrow Z=1$, $f_2 = 1 \rightarrow Z=0$
  - When both functions are 0, indicates that no prediction of output value is possible
- When prediction happens (we know definitely that output is 1 or 0), we turn off R2
  - Reduces activity in combinational logic block
  - R1 still is active, so Comb block still computes correct output
  - More effective if P (f1 + f2) is large and inputs to R2 is large

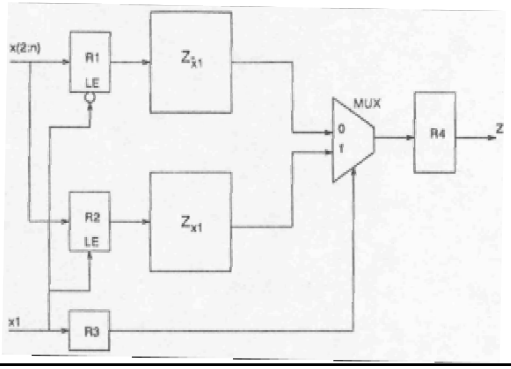
Example: *n*-bit Comparator



- If the MSBs of the comparator differ, this alone is enough to compute the result and all other inputs can be ignored

Precomputation with Shannon's Expansion Theorem

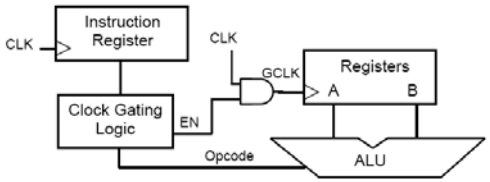
$$Z = x_j Z_{x_j} + x_j' Z_{x_j'}$$



Clock Gating

- Clock gating provides a way to selectively stop the clock
  - Force the circuit to make no transition
  - Assumes computation at next clock cycle is not needed
- Think of a reactive circuit that stays in an idle state until certain stimulus is received
- Changing inputs to the circuit can be ignored if they will not be used.

Example: Clock Gating in µPC



- Under what conditions do you need to enable the ALU?
- What determines the register values?
- What is the downside of this approach?

Probabilistic State Transition Graphs (STGs)

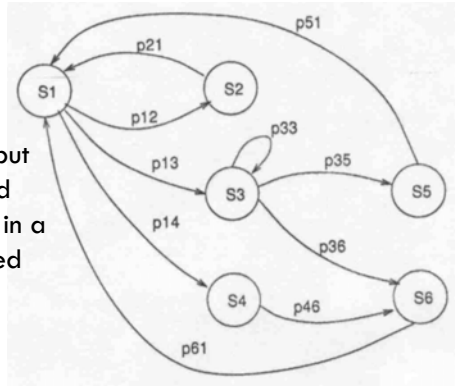
- Directed edge from state S_i to S_j in a state transition graph indicates:
  - input values causing transitions
  - resulting outputs
  - *conditional probability, p_{ij}, of transition*
- Given that machine is in state S_j

$$\sum_m p_{j,m} = 1$$

  - Directly related to signal probabilities at primary inputs
  - Introduce self-loops in STG for don't care situations to transform incompletely-specified machine into completely-specified machine

Example

- The sum of the output probabilities should always equal to 1 in a completely specified machine.



Relationship Between State Assignment and Power

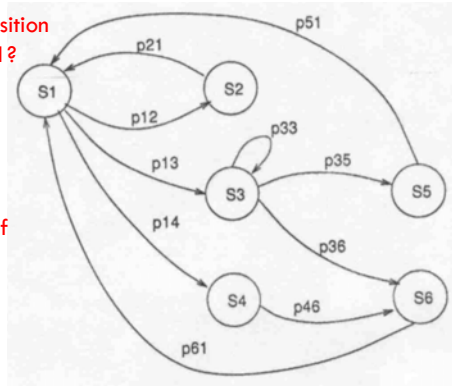
- Hamming distance between states S_i and S_j:
  - H (S_i, S_j) = # bits in which the assignments differ
- Average Power: $Power_{avg} = \frac{1}{2} V_{DD}^2 \sum C_i \cdot T(i)$
  - $T(i)$ = signal activity at node i (transition activity)
  - Approximate C_i with fanout factor at node i
- Average power proportional to:

$$\varphi = \sum_i \text{fanout}_i \cdot T(i)$$

BROWN

Example

- What is the state transition probability of S5→S1?
- What about S2→S1?
- What can we assume about the likelihood of state transitions S1←→S2?



BROWN

Handling Present State Inputs

- Find state transitions $H(S_i, S_j)$ of highest probability
- Minimize $H(S_i, S_j)$ by changing state assignment of S_i, S_j
- Requires system simulation of circuit over many clock periods, noting signal values and transitions
- If one-hot design is used, note that H = 2 for all states
  - May not be possible to obtain optimum power reduction
  - Uses too many flip-flops
- By minimizing $H(S_i, S_j)$ for high-transition edges, we assume:
  - We minimize transition activity of present state inputs
  - Combinational logic of FSM can be more efficiently synthesized for low power.

BROWN

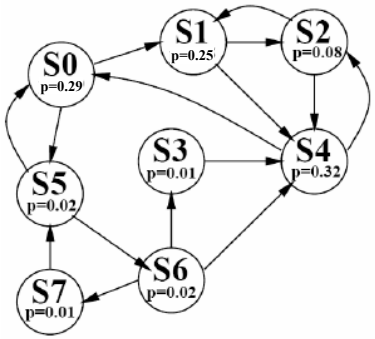
Computational Kernels

- Sequential circuits can contain many reachable states
- BUT… in normal operation, only a small subset are visited.
- Similarly, only a few distinct output patterns may be generated.
- IDEA:
  - Focus on making circuit fast & power-efficient only for typical input stimuli
  - Other inputs will produce correct behavior, but less efficiently

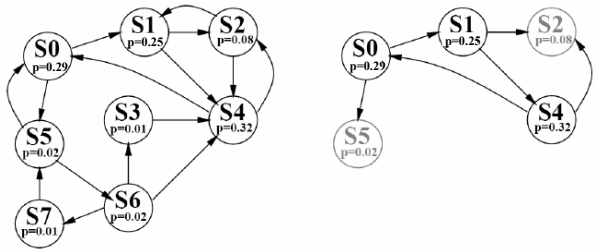
BROWN

p-order Computation Kernel

- Find the subset of states S_p whose steady-state occupation probabilities are greater than p
- Also include states directly reachable from this subset S_p
- *What is the p-order kernel when p=0.25?*

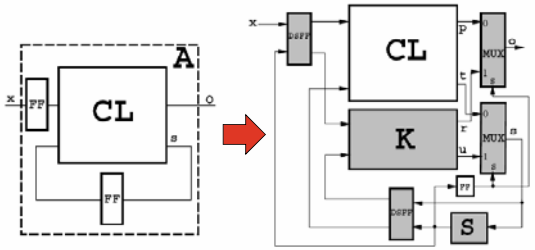


p-order Kernel (cont.)



- S_p ={S0,S1,S4}
- Optimize the next-state and output logic for these most common states.

Kernel-Based Optimization



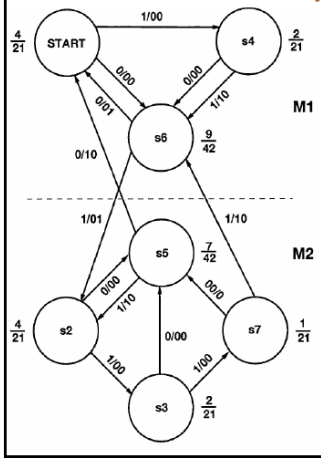
- Computation kernel K is a simplified/partial implementation of circuit A
  - Reduced complexity, reduced power
- Selector S determines which block to use in following cycle based on next state bits in current cycle

State Machine Decomposition

- Decompose the State Transition Graph of an FSM into two or more STGs
- Except for transitions between sub-FSMs, only one needs to be clocked.

- Searching for subsets:
  - Find set of states that transition among states with high probability
  - Goal: find small sub-FSM that is active most of the time.
- Larger FSM can be disable most of the time

Decomposition Example



- Given steady state probabilities, FSM can be partitioned into two sub-machines
- Minimize transition between states

Back to Leakage Current…

- Reducing V_{DD} reduces dynamic power dissipation
  - BUT… reduced V_{DD} also means reduced performance
- Improve performance by reducing V_{th}
  - BUT… reduced V_{th} means increased leakage current

$$I_{leakage} \approx I_0 e^{\frac{V_{GS}-V_{th}}{nVT}} \left(1 - e^{-\left(\frac{V_{DS}}{VT}\right)}\right) \approx I_0 e^{\frac{V_{GS}-V_{th}}{nVT}}$$

- I_0 is a function of gate oxide, mobility, and size of device
- VT is the thermal voltage (26mV at T=300K)
- Notice that V_{ds} is much larger than V_{gs} for a device that is off.

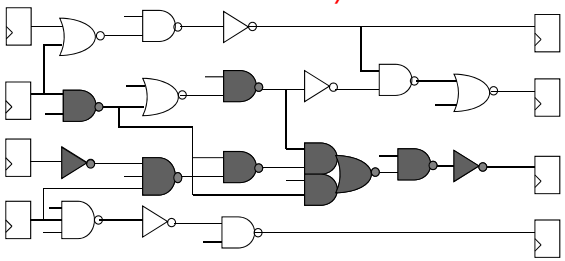
Contolling Leakage Current

$$I_{leakage} \approx I_0 e^{\frac{V_{GS}-V_{th}}{nVT}}$$

- What can we do to control leakage current?
- V_{GS}, V_{th}, VT….
- What about Dual V_{th} synthesis?
- What are the limitations of Dual V_{th}?
- Remember that leakage occurs whether devices are switching or not (i.e., active or not)

Static Dual-V_{th} Assignment

- Minimum energy consumption is achieved if all logic paths are critical (have the same delay)
- Use lower threshold on timing-critical paths
- *Only applied to off-critical paths*
- *Doesn't take into account circuit activity*



Standby/Active Modes

- We want computation to be as fast as possible when needed
- When computation is not needed, we don't care:
  - Clock gating: inhibits switching, reduces dynamic power
  - Standby mode: cuts off or reduces power supply, reduces static power
- When do we enter standby mode?
  - After some period of inactivity (reactive)
  - Analyzing resource needs of current computation

Reducing static power

$$I_{leakage} \approx I_0 e^{\frac{V_{GS}-V_{th}}{nVT}}$$

- V_{th} has an exponential effect on leakage current
- Increasing V_{th} will lower static power dissipation
  - Control directly (using high V_{th} devices)
  - *Control indirectly* (taking advantage of circuit topology and input values)

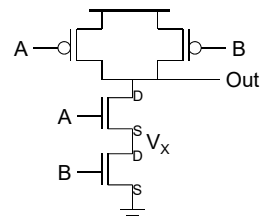
Stack Effect

$$V_{th} = V_{T0} + \gamma\left(\sqrt{|-2\varphi_F + V_{SB}|} - \sqrt{|-2\varphi_F|}\right)$$

- Consider a 2-input NAND gate
- Leakage is lowest when both inputs A=B=0V.
  - Intermediate node V_x settles to:

$$I_D \approx I_0 e^{\frac{V_{GS}-V_{th}}{nkT/q}}\left(1-e^{-\left(\frac{V_{DS}}{kT/q}\right)}\right)$$

$V_x \approx V_{th} \ln(1+n)$



A	B	V_X	$I_{SUB}=f(I_S,V_{GS},V_{BS})$
0	0	$V_{th}\ ln(1+n)$	$V_{GS}=V_{BS}= -V_X$
0	1	0	$V_{GS}=V_{BS}=0$
1	0	$V_{DD}-V_T$	$V_{GS}=V_{BS}=0$
1	1	0	$V_{SG}=V_{SB}=0$

Leakage reduction due to stacked transistors is called the ***stack effect***

Main Ideas of MTCMOS

- Lowering V_{DD} reduces power, but drastically hurts performance
- Lowering V_{th} helps to recover some performance, but will exponentially increase leakage power
- MTCMOS allows designs to use low V_{th} devices, but effectively only in *active mode*.
- In standby mode, virtual V_{DD} is lowered to the point where leakage is drastically reduced.
- Essentially takes advantage of stack effect

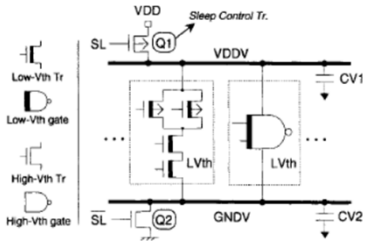
- Ideas presented in lecture are based on published paper:

S. Mutoh *et al. IEEE Journal on Solid-State Circuits*, vol. 30, no. 8, August 1995.

Access from the course webpage (lecture slides reading)

MTCMOS circuit scheme

- MTCMOS: Multithreshold -voltage CMOS
- NAND gate is made of LVT devices
- Not connected directly to VDD and GND
- Path to supply rails is through HVT devices controlled by sleep signal



BROWN

MTCMOS Operation

- Active Mode:
  - When SLEEP is off (active mode), Q1 and Q2 are on
  - VDDV and GNDV function as real power lines
  - NAND operates normally
- Standby (Sleep) Mode:
  - Q1 and Q2 are turned off
  - VDDV and GNDV are floating
  - Leakage current of gate is determined by leakage of HIGH Vth DEVICES.

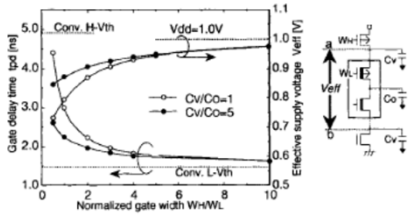
BROWN

Design Constraints

- When in active mode, the voltage drop across Q1, Q2 should be small:
  - Larger drop reduces drive capability from Vdd to (Vdd-Vx)
- Solution: reduce resistance of Q1, Q2
  - Make the transistor W large
  - BUT this takes extra area and power…
  - Several gates can share a single sleep transistor

BROWN

Gate width vs. delay

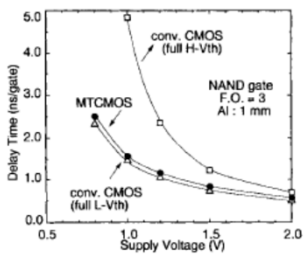


- Gate delay and effective supply voltage Veff depend on width of sleep transistors
- Large capacitance on virtual power lines also better for retaining voltage
  - voltage lines are not greatly affected by switching of internal logic gates.

BROWN

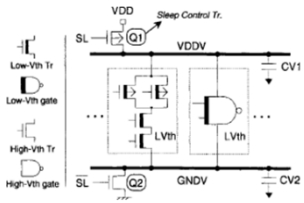
MTCMOS vs. Conventional CMOS

- Voltage dependence of MTCMOS gate is less than for an H-Vth gate
- MTCMOS gate delay is reduced by 70% compared to H-Vth gate.
- Note that MTCMOS is most advantageous at low Vdd

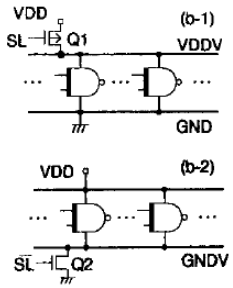


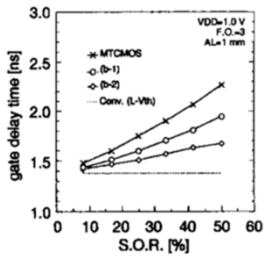
Standard cell design

- In standard cells, sleep transistors are typically shared by many gates
- What happens to the voltage when many gates switch at the same time?
- How does this affect delay?



Affect on switching rate





- As long as switch rates remain below 30% delay is not greatly affected.
- What about adjusting the width W of QL1 and QL2?

Clustering sleep transistors

- What happens if the switching-on-rate (SOR) is greater than 30%?
- A single pair of sleep transistors will cause too much delay (due to voltage drop)
- Instead, cluster gates with mutually exclusive switching outputs together
- More generally, group together as many as possible without violating max current constraint.
- Each cluster has low SOR rate
- No single cluster will see big drop in virtual supply rails

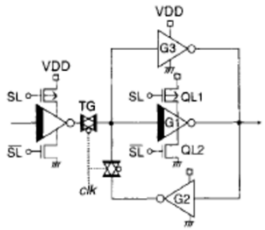
Data in sleep mode

- So far we have just seen how to use sleep transistors in simple static gates.
- What about flip-flops or other memory elements?
- What happens to the voltage levels when it is put into sleep mode?
- What happens when the circuit returns to active mode?

- We need a slightly different strategy

MTCMOS latch

- Note that G2 and G3 are always connected to the true power supplies (even in sleep mode)
- G1 and TG using L-Vth devices make fast switch times possible.
- QL1, QL2 cut the leakage current path.



- G2 and G3 are H-Vth gates
- Clock is fixed in sleep mode
- G1 and TG are L-Vth gates
- QL1, QL2 are H-Vth gates

Homework

- Simulate a circuit using an MTCMOS design
- Measure affect on delay and leakage reduction
- All work is done using Cadence Virtuoso (schematic capture)
- See assignment posted on webpage for more details.