BROWN

Low Power VLSI System Design

Lectures 4 & 5: Logic-Level Power Optimization

Prof. R. Iris Bahar

EN2912

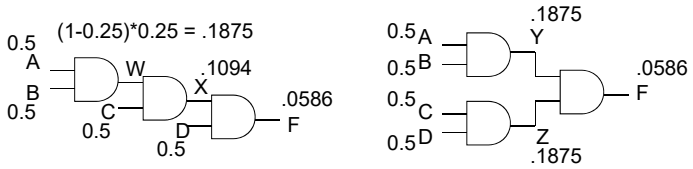
September 18 & 20, 2017

BROWN

Logic Restructuring Revisited

- Logic restructuring: Changing the topology of a logic network to reduce transitions

AND: $P_{0\to1} = P_0 \times P_1 = (1 - P_A P_B) \times P_A P_B$

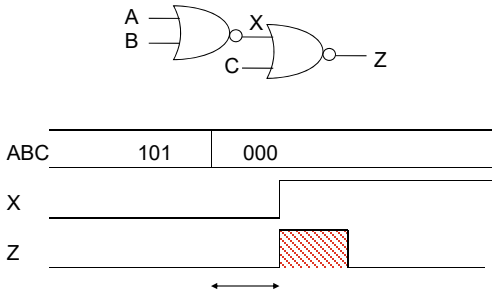


- Chain implementation has a lower overall switching activity than the tree implementation for random inputs
- Ignores glitching effects

BROWN

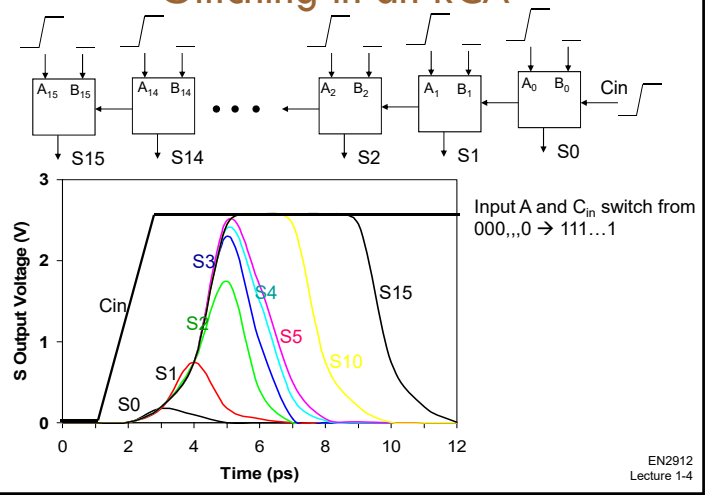
Glitching in Static CMOS Networks

- Gates have a nonzero propagation delay resulting in spurious transitions or glitches (dynamic hazards).



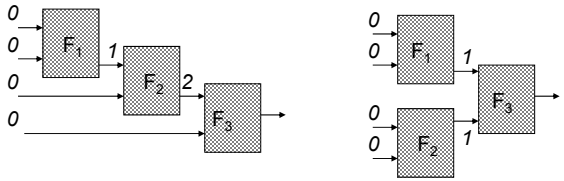
BROWN

Glitching in an RCA



Balanced Delay Paths to Reduce Glitching

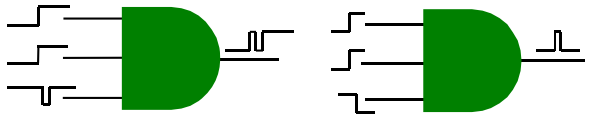
- Glitching is due to a mismatch in the path lengths in the logic network; if all input signals of a gate change simultaneously, no glitching occurs



- So equalize the lengths of timing paths through logic

Theorem 1

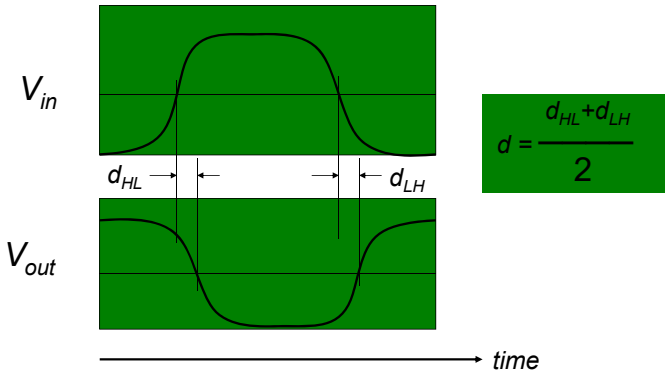
- For correct operation with minimum energy consumption, a Boolean gate must produce no more than *one* event per transition.



Output logic state changes
One transition is necessary

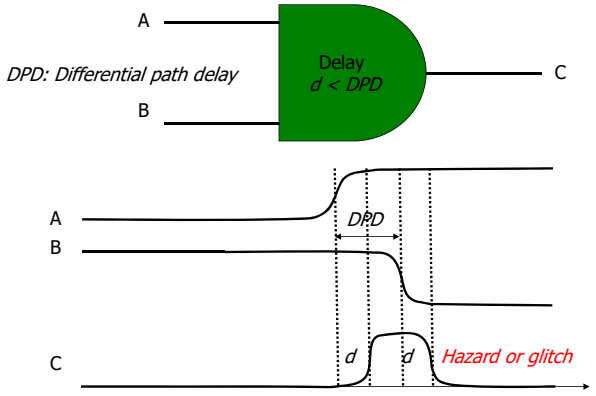
Output logic state unchanged
No transition is necessary

Inertial Delay of an Inverter

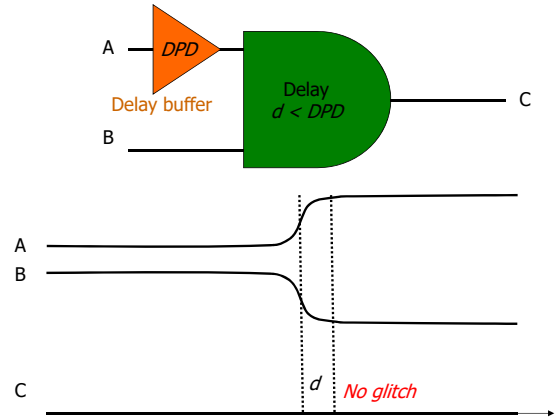


$$d = \frac{d_{HL}+d_{LH}}{2}$$

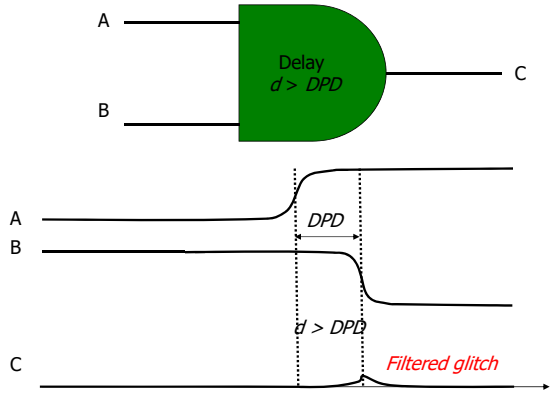
Multi-Input Gate



Balanced Path Delays



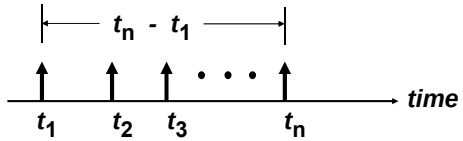
Glitch Filtering by Inertia



Theorem 2

- Given that events occur at the input of a gate with inertial delay d at times, $t_1 \leq \ldots \leq t_n$, the number of events at the gate output cannot exceed

$$\min\left(n,\ 1 + \left\lfloor \frac{t_n - t_1}{d} \right\rfloor\right)$$



Minimum Transient Design

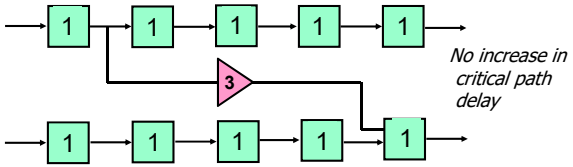
- Minimum transient energy condition for a Boolean gate:

$$|\, t_i - t_j \,| < d$$

Where
t_i and t_j are arrival times of input events and
d is the inertial delay of gate

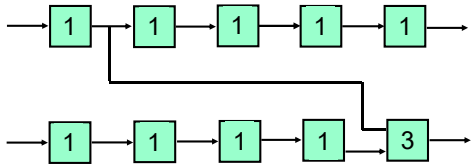
Balanced Delay Method

- All input events arrive simultaneously
- Overall circuit delay not increased
- Delay buffers may have to be inserted



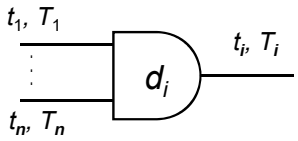
Hazard Filter Method

- Gate delay is made greater than maximum input path delay difference
- No delay buffers needed (least transient energy)
- Overall circuit delay may increase



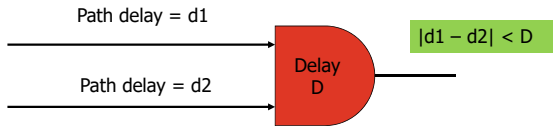
Define Timing Variables

- d_i Gate delay.
- Define two timing window variables per gate output:
  - t_i Earliest time of signal transition at gate i.
  - T_i Latest time of signal transition at gate i.
- Glitch suppression constraint: $T_i - t_i < d_i$



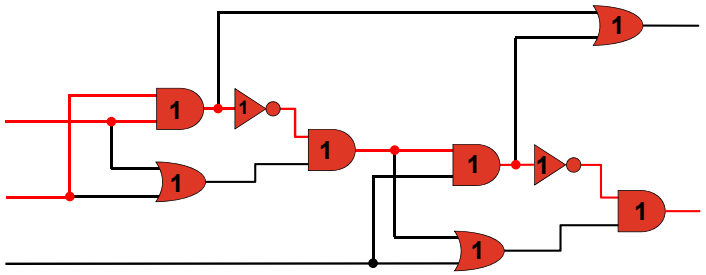
Designing a Glitch-Free Circuit

- Maintain specified critical path delay.
- Glitch suppressed at all gates by
  - Path delay balancing
  - Increasing inertial delay of gates
  - Inserting delay buffers when necessary.
- A linear program optimally combines all objectives.



BROWN

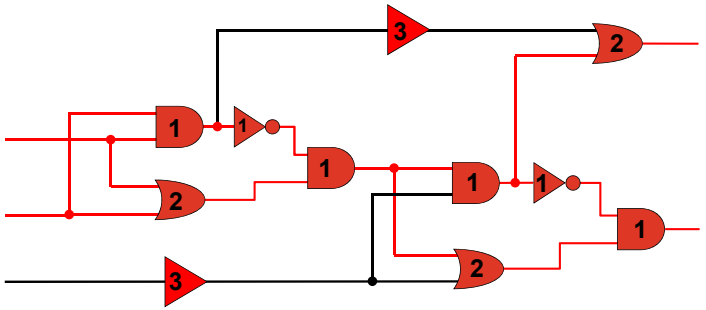
An Example: Full Adder



Critical path delay = 6

BROWN

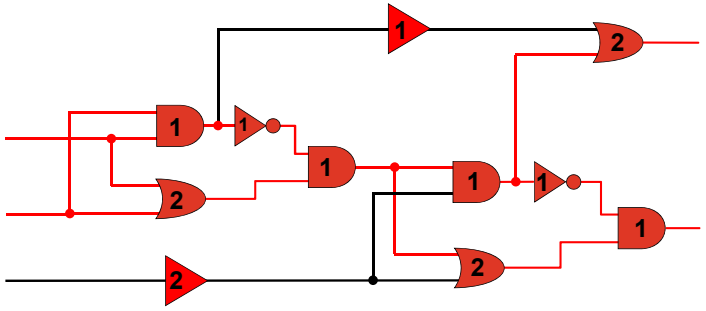
Solution: maxdelay = 6



Critical path delay = 6

BROWN

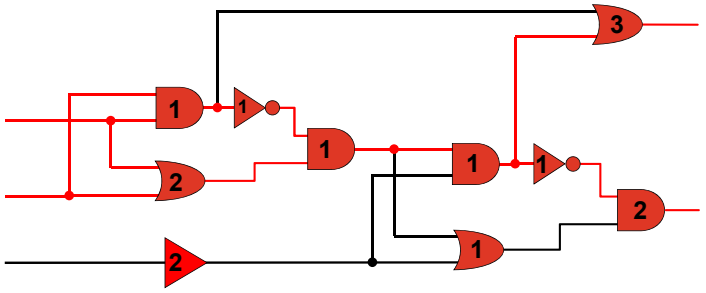
Solution: maxdelay = 6



Critical path delay = 6

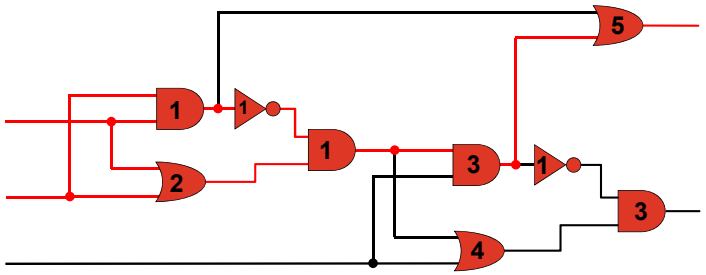
BROWN

Solution: maxdelay = 7



Critical path delay = 7

Solution: maxdelay ≥ 11



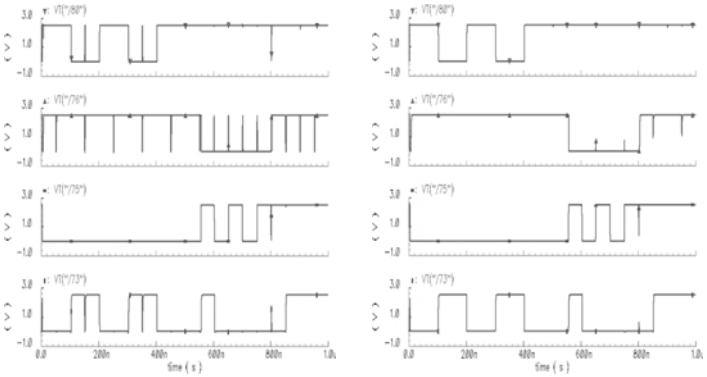
Critical path delay = 11

Four-Bit ALU

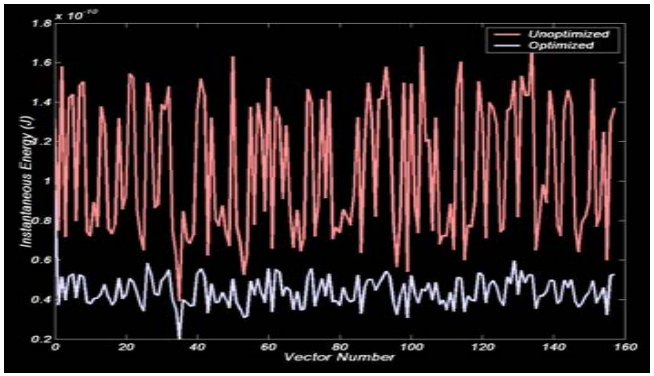
maxdelay	Buffers inserted
7	5
10	2
12	1
15	0

Maximum Power Savings (zero-buffer design):

ALU4: Original and Low-Power

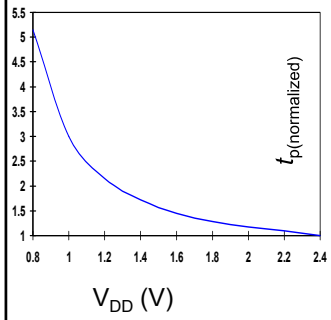


C7552 Circuit: Spice Simulation



Power Saving: Average 58%, Peak 68%

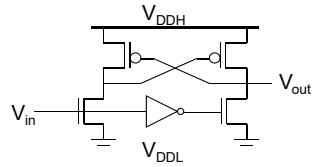
Power as a Function of V_{DD}



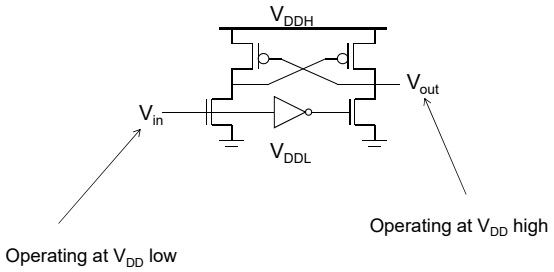
- Decreasing the V_{DD} decreases dynamic energy consumption (quadratically)
- But, increases gate delay (decreases performance)
- Determine the critical path(s) at design time and use high V_{DD} for the transistors on those paths for speed. Use a lower V_{DD} on the other gates, especially those that drive large capacitances (since this yields the largest energy benefits).

Multiple V_{DD} Considerations

- How many V_{DD}? → Two is common
  - Many chips already have two supplies (one for core and one for I/O)
- When combining multiple supplies, level converters are required whenever a module at the lower supply drives a gate at the higher supply (step-up)
  - If a gate supplied with VDDL drives a gate at VDDH, the PMOS never turns off
    - cross-coupled PMOS transistors do the level conversion
    - NMOS transistors operate on a reduced supply
  - Level converters are not needed for a step-down change in voltage
  - Overhead of level converters can be mitigated by doing conversions at register boundaries and embedding the level conversion inside the flip-flop

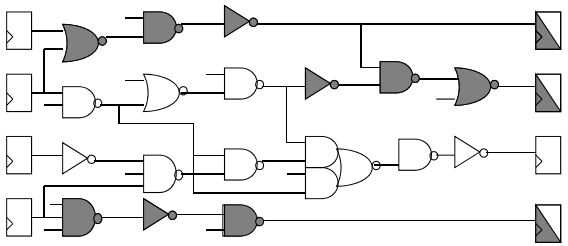


Multiple V_{DD} Considerations

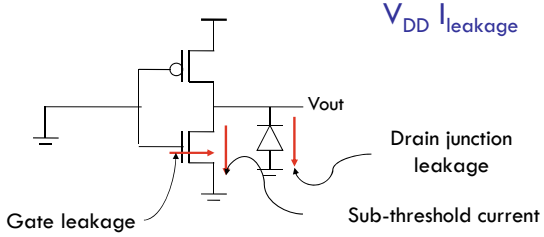


Dual-Supply Inside a Logic Block

- Minimum energy consumption is achieved if all logic paths are critical (have the same delay)
- Clustered voltage-scaling
  - Each path starts with V_{DDH} and switches to V_{DDL} (gray logic gates) when delay slack is available
  - Level conversion is done in the flipflops at the end of the paths



Leakage (static) power dissipation



Sub-threshold current is the dominant factor.

All increase exponentially with temperature!

Leakage (static) power dissipation

- 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.

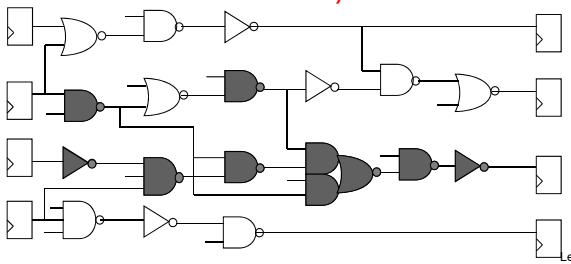
Contolling Leakage Current

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

- 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

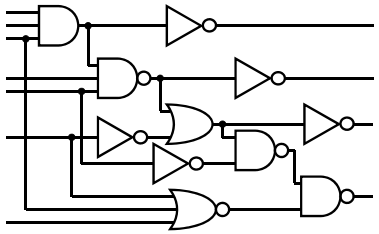
- Minimum energy consumption is achieved if all logic paths are critical (have the same delay)
- Use lower threshold on timing-critical paths
- *Only applied to off-critical paths*
- *Doesn't take into account circuit activity*



BROWN

Problem: Leakage Reduction

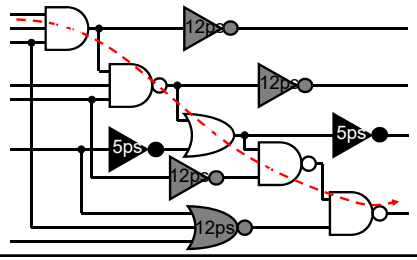
This circuit is designed in 65nm CMOS technology using low threshold transistors. Each gate has a delay of 5ps and a leakage current of 10nA. Given that a gate with high threshold transistors has a delay of 12ps and leakage of 1nA, optimally design the circuit with dual-threshold gates to minimize the leakage current without increasing the critical path delay. What is the percentage reduction in leakage power? What will the leakage power reduction be if a 30% increase in the critical path delay is allowed?



BROWN

Solution 1: No Delay Increase

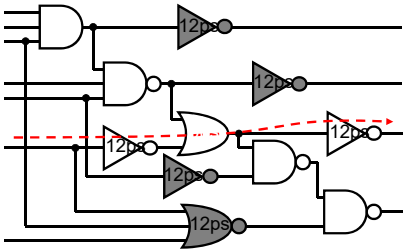
The critical path(s) are shown by the dashed line (delay = 25ps). None of these gates on the critical path can be assigned a high V_T. Also, the 2 inverters that are on four-gate long paths cannot be assigned high threshold because then the delay of those paths will become 27ps. The remaining 3 inverters and the NOR gate can be assigned high V_T (shaded grey). Reduction in leakage power = $1 - (4\times1+7\times10)/(11\times10)$ = **32.73%**
Critical path delay = 25ps



BROWN

Solution 2: 30% Delay Increase

Several solutions are possible. Notice that any 3-gate path can have 2 high threshold gates. Four and five gate paths can have only one high threshold gate. One solution is shown in the figure below where six high threshold gates are shown with shading and the critical path is shown by a dashed red line arrow.
The reduction in leakage power = $1 - (6\times1+5\times10)/(11\times10)$ = **49.09%**
Critical path delay = 29ps



BROWN

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)

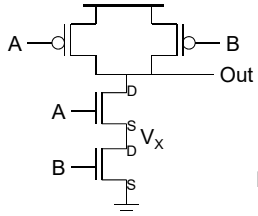
Stack Effect

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

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

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

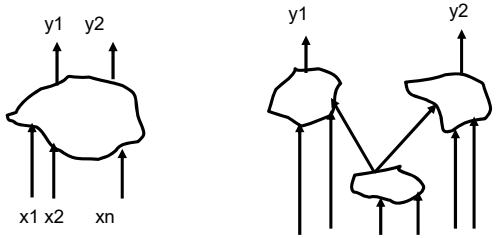
$V_x \approx V_{th} \ln(1+n)$



A	B	V_X	$I_{SUB} \propto f(I_S, V_{GS}, V_{BS})$
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***

Common Subexpression Extraction



Example

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 $P=1/2\ \alpha CV^2f_{clk}$
- Signal probabilities and transition densities of all nodes in a 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)

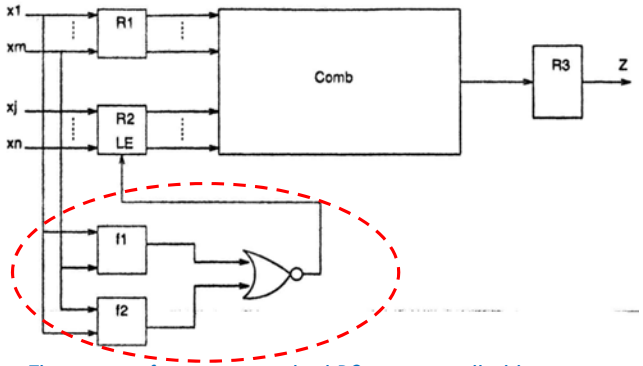
FSM and Combinational Logic Synthesis

- 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