

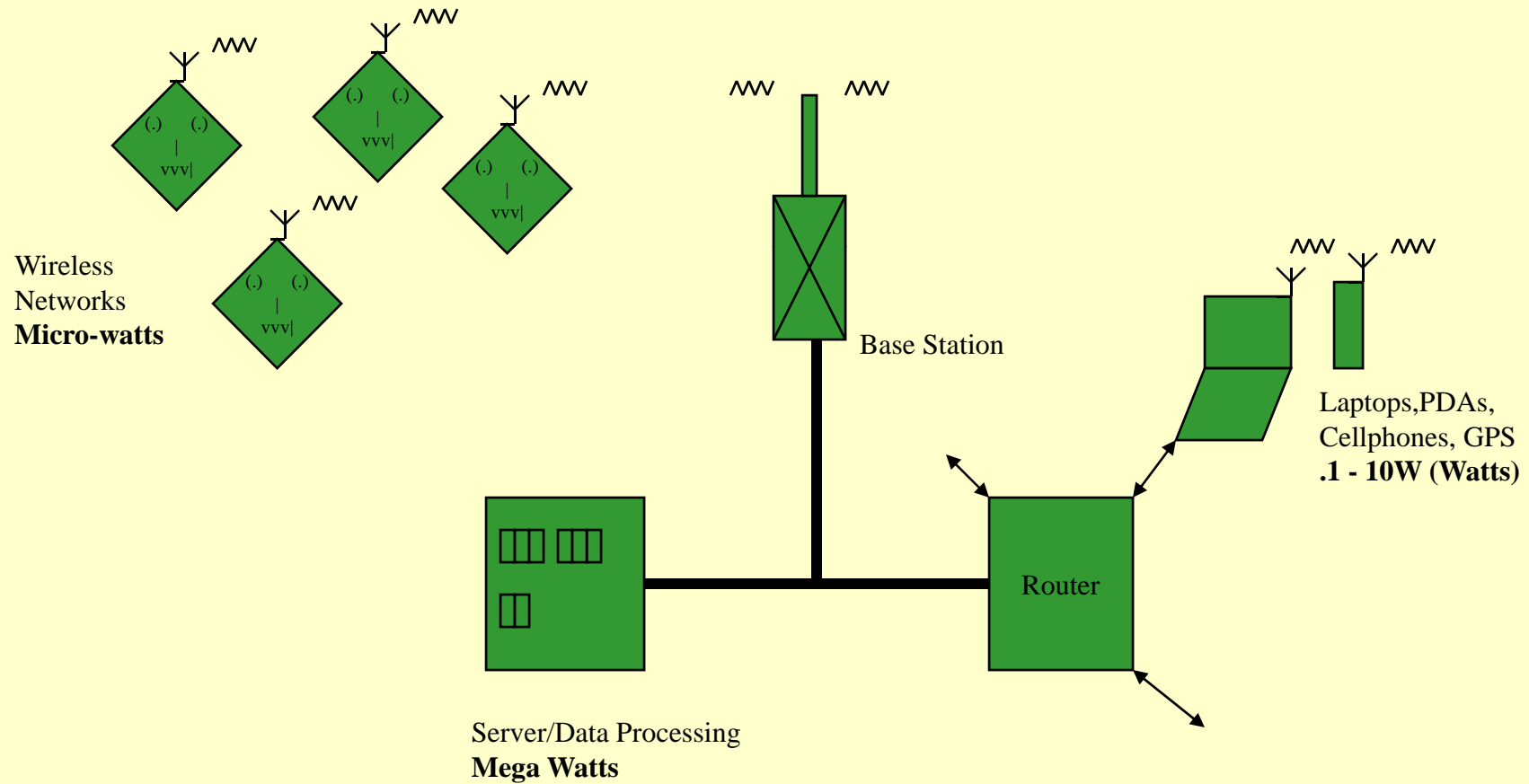
# Power Conscious Design of Embedded Systems

Rabi Mahapatra  
Computer Science

## Plan for today

- Some Power Models
- Familiar with technique to reduce power consumption
- Reading assignment: paper by Bill Moyer on “Low-Power Design for Embedded Processors” Proceedings of IEEE Nov. 2001

# Next Generation Computing: Watts metrics?



# Power Aware

- Increase in prominence of portable devices
- SoC complexity: heat generation
- Traditionally, **speed (performance), & area (cost),**
- **Now, add *power* as the new axis**

# Physics Revisited

- Energy is in Joules
- Power: Rate of energy consumption (joules/sec),  
in Watt
- $V_{dd} * I_d$ : instantaneous power

## Impact on embedded system

- **Energy consumed per activity reduces battery life**
  - Decreases battery capacity fast
- **IR drops in a battery due to flow of current**
  - Requires more Vdd & GND pins to reduce R, also, thick & wide wiring is necessary
- **Inductive Power-supply voltage bounce due to current switching**
  - Requires more & shorter pins to reduce inductance
  - Require on chip decoupling capacitance to help bypass pins
- **Power dissipation produces heat and high temperature reduces speed and reliability**

# Opportunities for Low-Power

<b>Algorithms</b>	Minimize Operation
<b>Source Code</b>	Optimized code
<b>Compiler</b>	Energy miser
<b>Operating System</b>	Scheduling
<b>ISA</b>	Energy Exposed
<b>Microarchitecture</b>	Clocked Gating
<b>Circuit Design</b>	Low voltage swing
<b>Manufacturing</b>	Low-k dielectric

# Some Power Models

- Macro level
  - Arithmetic
  - Software
  - Memory
- Activity Based
  - Empirical
  - Information-theoretic
  - Signal modeling-based

# Empirical

- Based on chip estimation system [Glaser ICCAD91]:

$$P = \alpha G(E_r + C_L * V_{dd}^2)f$$

G = number of equivalent gates

$E_r$  = energy consumed by an equivalent gate

$C_L$  = average loading per gate including fanout

$\alpha$  = activity factor

- Demerit: lacks consideration on different logic styles

# Information Theoretic

- Reference [Najm95]
- Based on activity estimation

$$P = k (C_L)(\alpha) = k(A)(h)$$

A = area, h = entropy factor (a function of entropy H)

- Limited accuracy, does not include possibility of encoding

# Signal Model Based

- Reference [Landman TCAD96]
  - Properties of 2's complement encoded data stream
  - Arithmetic blocks are regular
- Analytical Method: [Ramprasad TCAD97]
  - Word-level statistics
  - Auto-regressive Moving Average signal generation model
  - 2's complement & sign magnitude signal encoding

# Software Power

- Power consumed by a processor (P): Ref [TiwariTVLSI94]

$$P = V_{dd} * I$$

- Energy (E):  $E = P * T_p$ , program execution time
- Program Execution Time( $T_p$ )  $T_p = N * T_{clk}$

$$E = P * T_p = V_{dd} * I * N * T_{clk}$$

If  $V_{dd}$  and  $T_{clk}$  are assumed to be constant, Energy is measured by measuring current I.

- Low-power software: small value of N or fast execution time
- When  $V_{dd}$  and  $T_{clk}$  are varying? Current measurements?

# Instruction Level Power Modeling

- Reference: [Tiwari TVLSI97]
- Current consumption of a program with no loops but M instruction

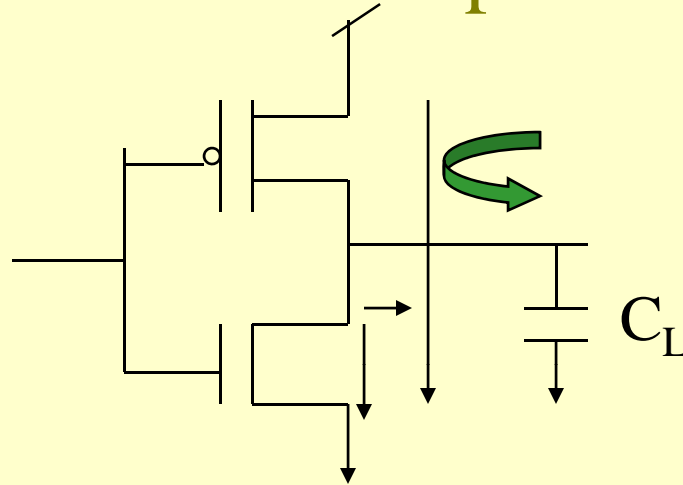
$$I = \sum_{i=0} B_k * N_k + O_{i,i+1 \bmod M} / \sum_{i=0} N_k$$

$B_k$  = Base current of kth instruction in the program

$N_k$  = Number of clocks required to complete kth instruction

$O_{i,j}$  = overhead of executing successive instruction

# Power Dissipation in CMOS



- Three sources:

$P_{\text{switching}}$ : Switching power (capacitive): dominant today

$P_{\text{leakage}}$ : Leakage Power, will dominant in 0.13 micron and below.

$P_{\text{shortcircuit}}$ : Short circuit component

# Switching Power Dissipation

- Occurs when device changes state or switching of charge in and out of  $C_L$  , capacitance
- Flow of current across the transistor's impedance
- $P_{\text{switching}} = t * C_L * V_{\text{dd}}^2 * f$ 
  - $t$  = average number of transition per cycle
  - $f$  = clock frequency
  - $C_L$  = effective capacitance
- Increases with clock frequency
- Decreases quadratically with supply voltage
- 85-90% of active power consumption

# Low-Power Techniques

- Low-power techniques reduces one or more of  $t$ ,  $C_L$ ,  $V_{dd}$ , and  $f$ 
  - $t$ : encoding
  - $C_L$  : fast algorithm, design layout
  - $V_{dd}$  : voltage scaling, variable voltage processor
  - $f$ : low-frequency and clock gating
- All of these are useful for embedded system

# Short Circuit Power Dissipation

- Occurs due to the overlapped conductance of both PMOS and NMOS transistors forming a CMOS logic gate as the input signal transitions
- $P_{\text{shortcircuit}} = I_{\text{mean}} * V_{\text{dd}}$
- 10-20% contribution to dynamic power
- Not important if all signals are guaranteed to have steep slopes

# Leakage Power Dissipation

- Occurs regardless of state change
- Due to leakage currents from reversed biased PN junction (OFF switches are not really off)
- Proportional to device area and temperature
- Increases exponentially with reduction in  $V_t$ , voltage scaling
- Significant when system is idle (Embedded Systems?)

# Static Power

- Not a factor in pure CMOS designs
- Sense amplifier, voltage references and constant current sources contribute to the static power
- Regardless of device state change

Total Power:  $P_{\text{switching}} + P_{\text{shortcircuit}} + P_{\text{static}} + P_{\text{leakage}}$

## Power – Delay Leverage

- Power & Delay trade off
- Speed is proportional to  $C_L * V_{dd} / (V_{dd} - V_t)^{1.5}$
- Trends: Reduce  $V_{dd}$  &  $V_t$  to improve speed
- Energy-delay product is minimized when  $V_{dd} = 2 * V_t$
- Reducing  $V_{dd}$  from  $3 * V_t$  to  $2 * V_t$  results in an approximately 50% decrease in performance while using only 44% of the power.

## Algorithmic Technique PR

- Focus on minimizing number of operation weighted by their cost: First order goal.
  - Underlying implementation: arithmetic or logical
- Recomputation of intermediate results may be cheaper than memory use
- Loop unrolling: reduces loop overhead
- Number representation:
  - fixed point or floating point
  - Sign-magnitude versus 2's complement is preferred in certain DSP when input samples are uncorrelated and dynamic range is minimized
  - Bit length (of course trade off accuracy)
  - Adaptive bit truncation in portable video encoder reduces 70% of the power over full bit width

## Architectural Technique PR

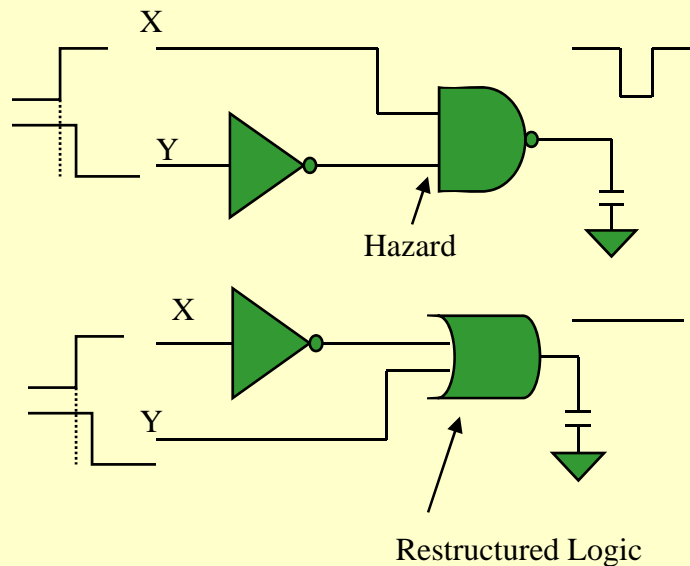
- Instruction set design and exploiting parallelism & pipelining are important
- Architecture driven voltage scaling method  
[Chandrakasan, IEEE J. Solid state Circuits 92]
  - Lower voltage for power but apply parallelism/pipeline to speedup
  - Possible if application has parallelism, trade-off with latency due to pipeline & data dependencies, and area
  - Speculative logic allowed if low overhead else detrimental
- Meeting required performance without over-designing a solution is fundamental to optimization
  - Extra logic power is not controllable and they still present even if parallelism is absent.

## Logic and Circuit Level PR

- Focus on reducing switched capacitance or/and signal swing
- Signal probabilities may favor either static or dynamic CMOS logic
  - Example: Two-input NAND gate with uniform distribution at inputs, probability of output being 0 ( $p_0$ ) is 0.25,  $p_1 = 0.75$
  - For static gate, probability of a power consuming transition from 0 to 1 is  $p_0 * p_1 = 0.1875$
  - For dynamic gate with the output is pre-charged to logic 1, power is consumed whenever the output was previously 0. Thus it has higher (by 0.25) transition at output than static.
  - However, dynamic circuit has lower input capacitance by a factor of 2 to 3.

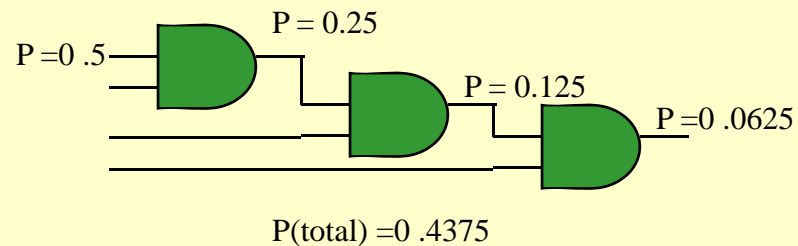
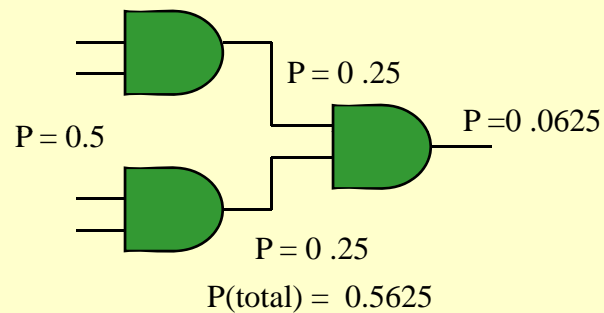
## Logic circuit PR

- For wider input static gate, say four input NAND,  $p_0 = 0.0625$  and  $p_0 > 1$  is  $0.0586$
- For dynamic version as above,  $p_0 = p_0 > 1 = 0.0625$
- Static logic suffers from *glitches*: needs restructuring and that adds up power more than 20%



# Logic circuit PR

- Mapping logic function to gates is tricky too
  - Example: transition probability



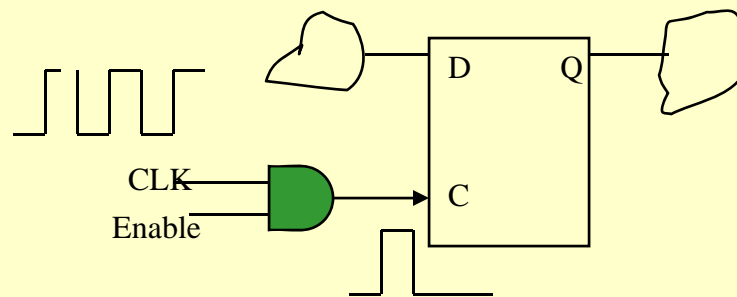
**(Four-input AND function)**

# Logic Circuit Reordering

- Reordering of equivalent inputs of gates and reordering of transistors in complex gates
  - Example of  $(A+B)*C$  in Fig.3
  - Affects amounts of switched internal capacitance, gate speed and static power
  - Signals with high probability of being off are placed nearest the output node of the gate, subject to timing constraint, and signals with high probability of being on are placed nearest the supply node
  - Signals with high probability of switching are placed nearest the output
  - 10% saving between best to worst case
  - Ref. Rules in [29] Shen et al ASP-DAC'95

# Logic circuit Clocking

- Clock generation & distribution can consume 30-40% of total system power: *minimize unnecessary transitions on clock signals*
  - Low-power STG of state machines and assignment of adjacent binary encoding
  - Clock gating at function unit level by inhibiting input updates
  - Fig. 4

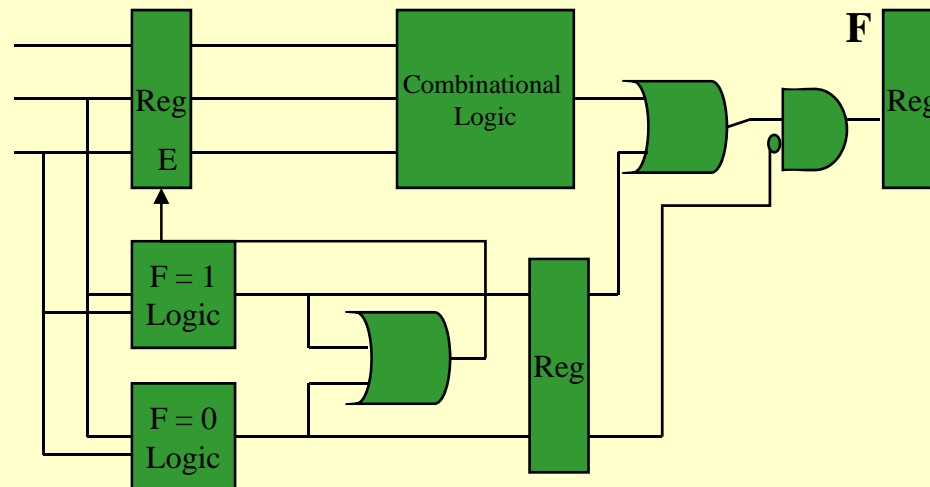


# Logic Circuit Clocking

- Overhead due to enable signal and limitation on granularity at clock gating can be applied
- Self-gating storage elements use local clocking enable
  - Stroll et al, Symposium Low-power Electronics & Design 2000
- Reduced swing clock drivers saves 63% [33] at the cost of twice the delay
- Differential clock signaling: ( $0.2 \cdot V_{dd}$ ), in static power, saves 60%; affects duty cycle and receiver skew

# Logic Circuit Pre-computation

- Selectively pre-computing output values
- Saves 11 – 66%. Ref. Monterio.. TCAD'98 [37]



# Device Technology

- Threshold voltage is key at device level for leakage
- Silicon on Insulator lowers the parasitic capacitance and reduces body effect
- Dual device threshold technology reduces standby power
  - High threshold for non-critical and low threshold for speed-critical paths
  - Alternative: raise threshold of all during standby

## Case Study: MCORE ISA

- 8-bit CPUs in watches, calculators (microwatts), 16-bit CPUs in handheld devices (milliwatts), 32-bit CPUs in notebooks (watts)
- ISA specification have significant effect on power-performance. (RISC, CISC, VLIW)
  - CISC is best suited for low-cost ES due to optimized code density
  - RISC & VLIW trade code density for simplified fetch/decode
  - Power spent on fundamental computing of an algorithm is important. However, Fetch/Decode/Sequencing overheads must consume less power

## ISA & Power

- RISC has reduced sequencing and control overhead but has increased instruction fetch bandwidth (32-bit words).
  - Compare 22-24 bits for CISC
- MCORE is not a pure RISC: 16-bit ISA with 16 GPR with load/store operation
  - Limited by long execution path length (due to reduced fields) & 2-operand instruction format
  - Using compiler-driven instructions, the limitations are minimized
  - For ES, optimized codes resides on-chip memory and reduced memory traffic due to careful selection of semantics also reduces power by 40%

## ISA Power

- Doubling I-cache reduces cache misses up to 50%
- For on-chip cache or memory, 32-bit data path is provided
  - Doubles the fetch bandwidth
- Using ISS, capture the frequency of execution of all instructions and order the *opcodes* accordingly (control unit design): could save 15% of power
- System-level power saving
  - Wait: disable only CPU
  - Doze: CPU plus some peripheral & clocks
  - Stop: deep power down state with stopping all clocks, reduced to off most supply voltages

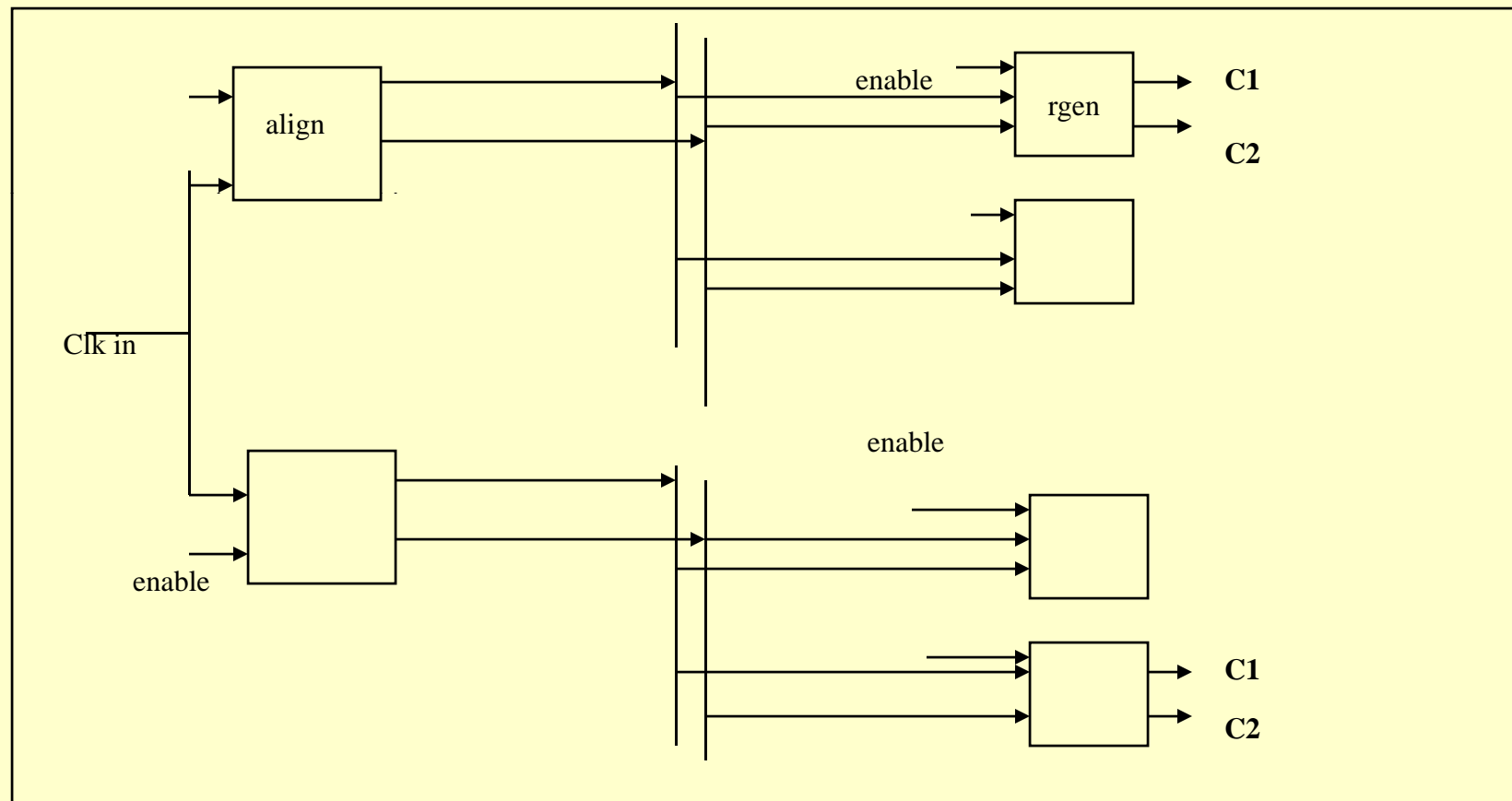
## Microarchitecture PR

- High-end processor microarchitectures exploit all possible ILP and correspondingly high level power inefficiency
- DSP & multimedia algorithms in embedded class have this opportunities: Special power aware architectures
- Mid range controllers use optimal pipeline architecture
- No free clocks in the datapath: gated and delayed clocks
- Floor planning of datapath element is evaluated executing a set of embedded benchmarks
  - Infrequently utilized functional units placed farther from reg-file & decoupled from bus appropriately

# Clock distribution

- Efficient distribution and gating mechanisms are vital
- Two-level approach:
  - Align clock edges in the first level for circuits and generate two non-overlapped phases
  - These two phases are further distributed to a set of regenerating cells with clock gating control inputs
  - Gating of clocks can be used at both levels per granularity
- Clock tree generation technique is used to allow both balanced and unbalance clock tree structures
  - For balanced clock tree, additional routing capacitance is used at all nodes
  - For unbalanced clock tree, resizing at intermediate aligner/regenerators (more effective for low-power)

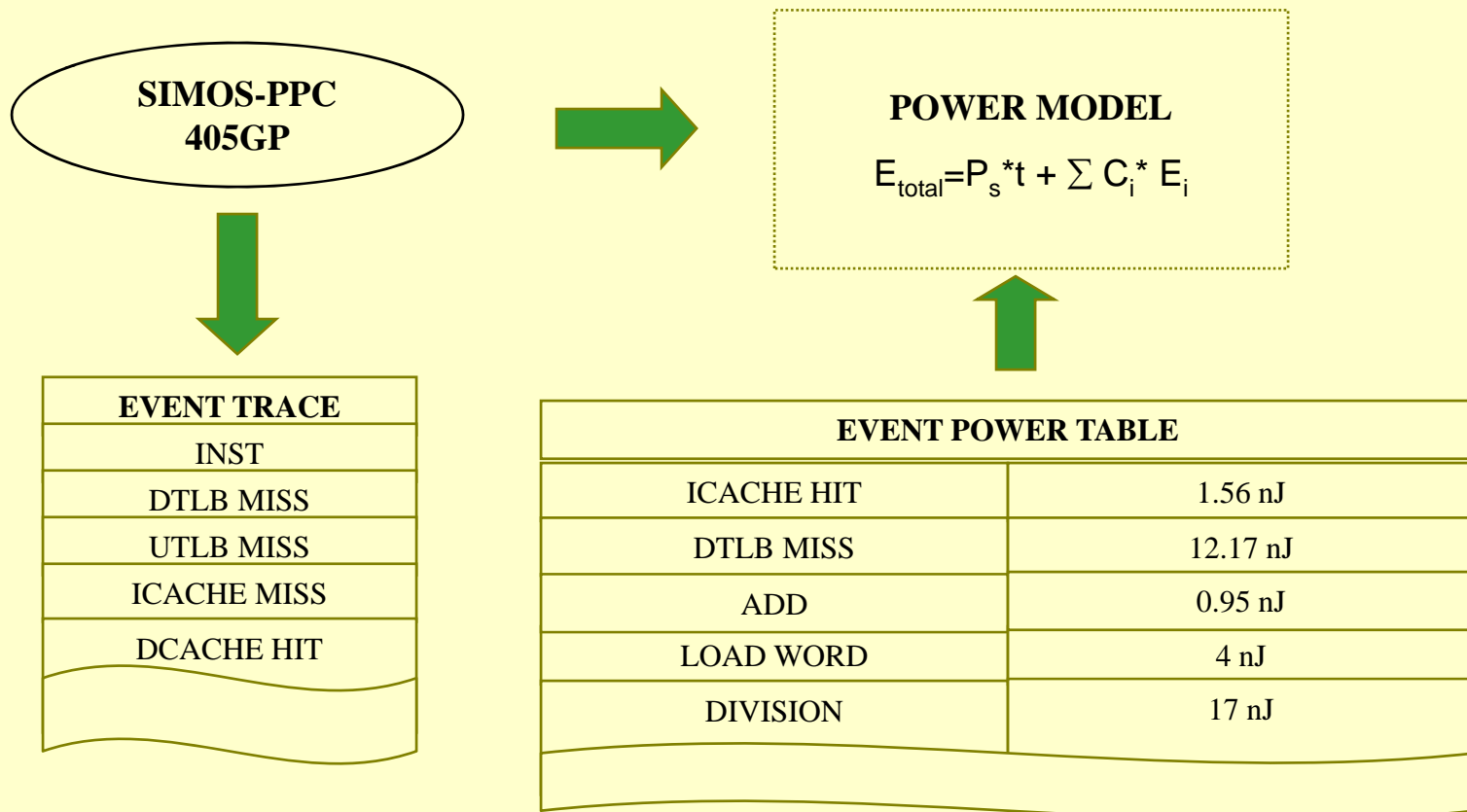
# Two-level clock distribution



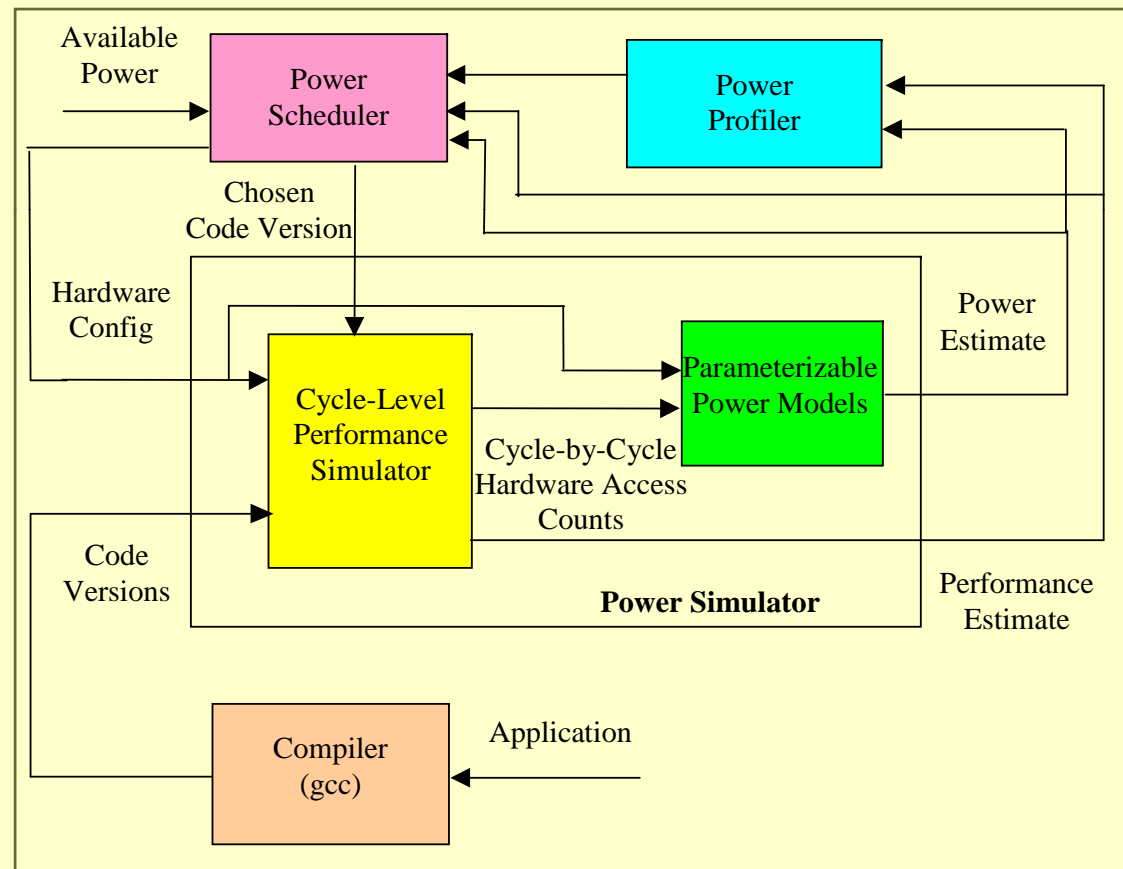
# Adapting power consumption profile to application and system needs

- IBM's power aware simulator
  - Use event driven complete simulator and power profiler
- UCI's COPPER project
  - Integrated view of compiler strategy for power managements

# IBM's power simulator



# UCI's Copper Project framework for power management



# Summary

- Power aware embedded system architectures
- Techniques to low power at circuit and device level
- Some recent efforts in use of tools for power efficient design

# Revisit Processor Architecture

- Execution Time:  
 $(\text{No. of Instructions}) * (\text{cycles/instructions}) * (\text{seconds/cycle})$
- Performance:  
 $(\text{Instructions/Cycle}) * (\text{Cycles/second})$
- It is mistake to treat these values as independent

# Architecture Revisited

- Clock frequency =  $f$  (device technology, circuit design,  $V_{dd}$ , architecture)
- How to improve clock rate?
  - Reduce complexity & use pipeline
- How do you feel about faster clock on power?
- Should we reduce clock?
- How about reduce  $V_{dd}$  at same clock? Will reduce power?

# Architecture Revisited

- Instructions/cycle
  - So much we can perform during a cycle
- How to improve more instruction issued during a cycle?
  - Exploit level of parallelism
  - Reduce/eliminate delays (memory & branch)
- ILP
  - VLIW
  - Superscalar
  - EPIC (explicitly parallel instruction computing)

## ILP & Power

- Increase instructions/cycle -> fewer cycles -> longer clock for same performance -> lower voltage->lower power
- Increase instructions/cycle ->more switching elements -> more transitions -> more power
- Also, more functional units means proportionately less performance but more power.
- It is tricky!

# Delay Elimination & Power

- Memory and Branch actions cause delay.
- Branch prediction helps reducing this delay.
  - **Guess** the next path most likely
  - If not, restore quickly with proper back up plans
  - Net gain when saved time is significant
- How to guess?
  - History based tables
  - Compiler assisted
  - Example: PCs save >95% time
  - but may not be feasible for low-end ES

# Prediction & Speculation

- Prediction
  - Use conditional statements in places of branches
  - Good for short if-then-else statements
  - Improves branch prediction performance and uses simple hardware
- Speculation
  - Move instructions from one path of branch to above branch

Add r4, r5, r6	Add r4, r5, r6
Beq elsewhere, r4, #6	Sub r8, r9, r10
Sub r8, r9, r10	Beq elsewhere, r4, #6

  - Improves ILP
  - If another path is taken, need to take care of it.

## More on speculation

- What to do if another path taken?
  - Speculate only instructions that can not cause problem
  - Preserve data, but allow exceptions
  - Provide HW to undo bad instructions
- Exceptions during speculation;
  - Only speculate non-excepting instructions
  - Ignore exceptions in speculated instructions
  - Delay exceptions until path is known

## Data speculation

- Move loads above stores that may not cause conflicts

Add r7, r8, r9

St r10, r7

Ld r2, r3

Add r1, r2, r3

Ld r2, r3

Add r7, r8, r9

St r10, r7

Add r1, r2, r3

# Power Issues

- Are we able to reduce cycle counts? Not offset by the extra power dissipation due to additional HW!
- For low-power
  - Compiler driven branch prediction
  - Prediction
  - Low-hardware instruction speculation
- High Power
  - Table based branch prediction
  - Hardware data and instruction speculation