

嵌入式微处理器系统

崔光佐

普适计算与应用实验室

北京大学现代教育技术中心

www.uclab.org



第一篇

第三讲 微处理器低功耗技术

2004.2.14

内容摘要

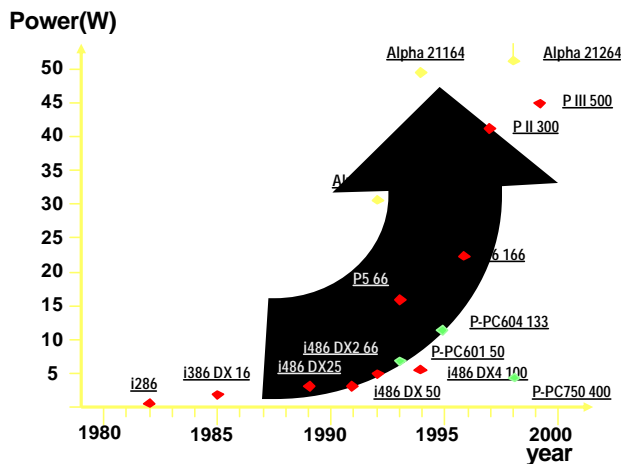
- 处理器中的功耗
- 算法级低功耗设计
- 系统级低功耗设计
- 结构级低功耗设计
- 逻辑级低功耗设计
- 电路级低功耗设计
- 微处理器设计举例

处理器中的功耗

"Power consumption has become an obsession for microprocessor designers in recent years and continues to be a hot topic as we chase Moore's Law. After all, increasing speeds and transistor counts have meant that chips require more energy"

Bijan Davari, IBM fellow and vice president of technology and emerging products

处理器中的功耗

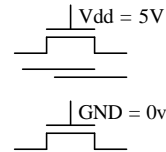


基本工艺：CMOS

- CMOS: Complementary Metal Oxide Semiconductor
 - NMOS (N-Type Metal Oxide Semiconductor) transistors
 - PMOS (P-Type Metal Oxide Semiconductor) transistors

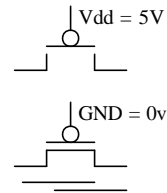
- NMOS Transistor

- 加高电平是导通
- 低电平截止



- PMOS Transistor

- 高电平截止
- 低电平导通

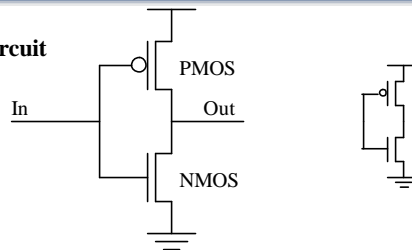


反向器

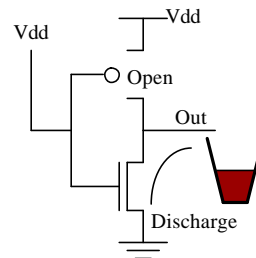
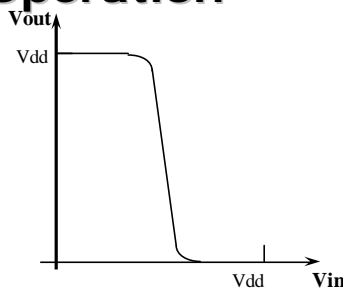
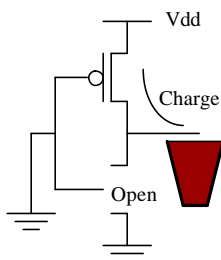
Symbol



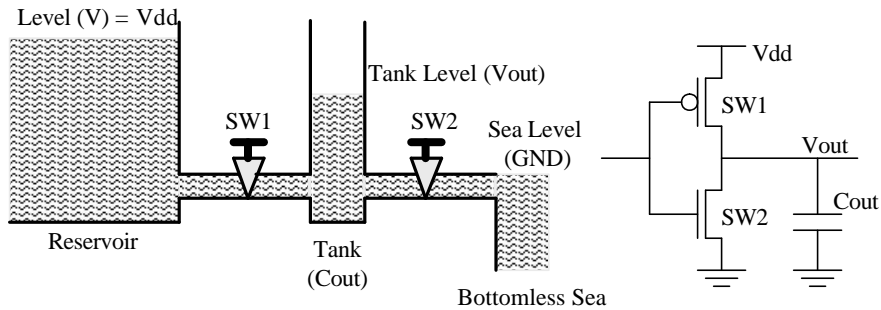
Circuit



- Inverter Operation

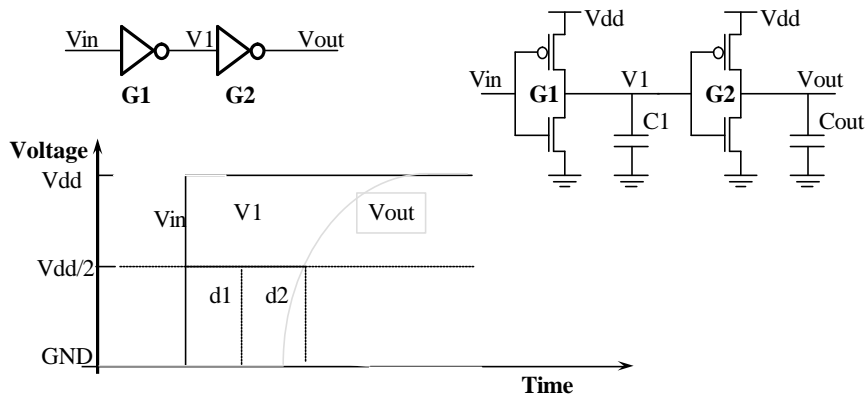


CMOS中的流体说明



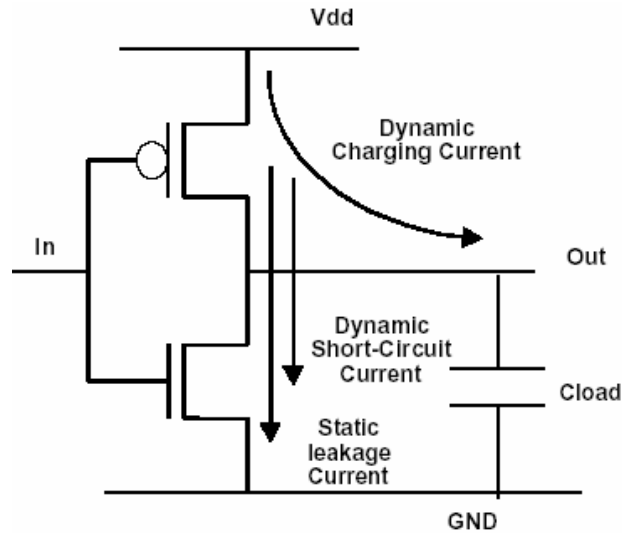
- Water $\hat{=}$ Electrical Charge Tank Capacity $\hat{=}$ Capacitance (C)
- Water Level $\hat{=}$ Voltage Water Flow $\hat{=}$ Charge Flowing (Current)
- Size of Pipes $\hat{=}$ Strength of Transistors (G)
- Time to fill up the tank proportional to C / G

串行延时



- Total Propagation Delay = Sum of individual delays = $d1 + d2$
- Capacitance $C1$ has two components:
 - Capacitance of the **wire** connecting the two gates
 - Input capacitance of the second inverter

处理器中的功耗



处理器中的功耗

动态功耗：

F:有效的频率；

C:电容；

V:电压值。

$$P = CV^2F$$

降低功耗的方法：

减小电容，降低供电电压，减小电压的变化频率。

静态功耗：

I:漏极电流

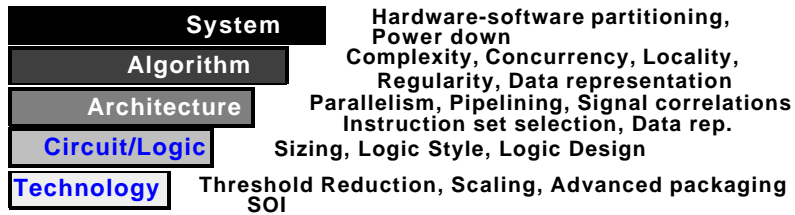
低功耗方法：

减小漏极电流；

减小供电电压。

$$P = I_{\text{leakage}} * V$$

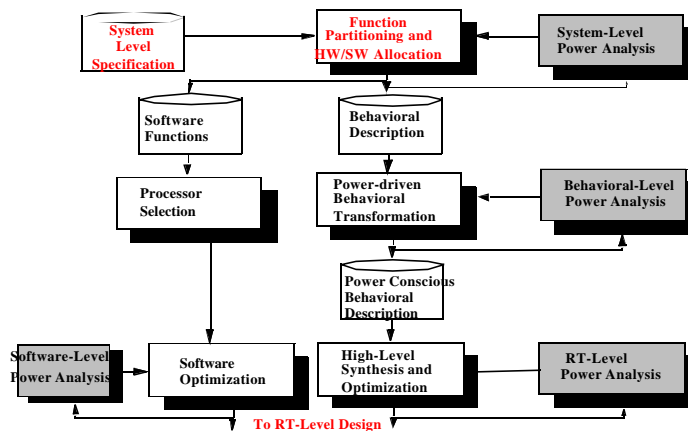
低功耗的层次



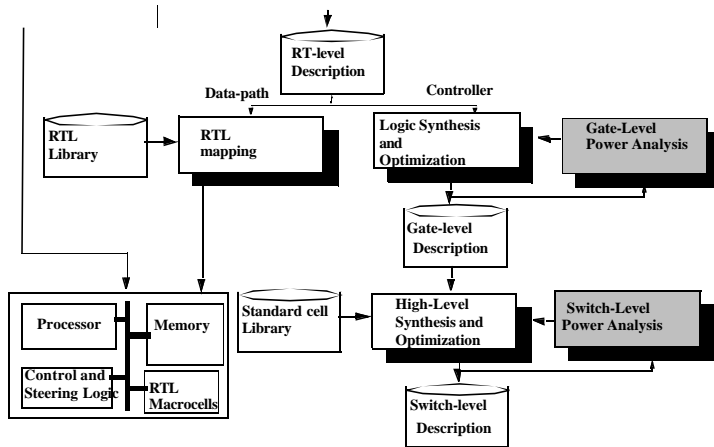
● Possible Power Savings at Different Design Levels

Level of Abstraction	Expected Saving
Algorithm	10 - 100 times
Architecture	10 - 90%
Logic Level	20 - 40%
Layout Level	10 - 30%
Device Level	10 - 30%

低功耗的设计



低功耗的设计



算法级低功耗设计

- ？ 电压作为设计参数
- 根据性能要求匹配电压与频率
- ？ 降低浪费（减小开关电容）
- 匹配计算与系统结构
- 保留算法内在的局部性
- 利用信号统计数据
- 按要求提供性能（能量）
- 更容易开发的定制设计（比可编程器件）

降低供电电压

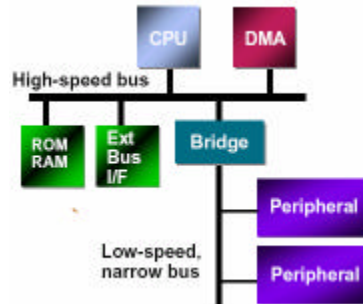
- 在满足系统需求的情况下，以最小的速度工作在最低供电电压
- 利用系统结构来优化并弥补部件的慢速操作；
 - 如，并发与流水

系统设计

- 算法选择，指令集,系统划分，数据表示等. 这些对系统的功耗起主要作用
- 处理器的设计者应该从全局的角度优化这些因素
- 操作系统也正在成为功耗的影响因素，也影响处理器的设计
- 系统不围绕低功耗设计，只是一个地功耗处理器很难起作用。

系统划分

- 分离高带宽和低带宽设备可以降低功耗
- 处理器的总线接口应该方便不同速率的回应：
 - split transactions
 - fire and forget
 - wait for critical response



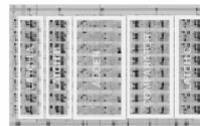
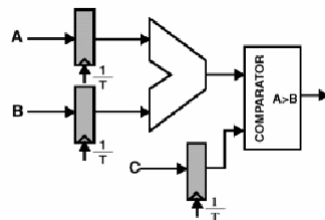
指令集设计

- 指令集对系统的功耗具有较大的影响
- 完成功能所需要的访存量越少越好
- 代码密度越高越好
- CACHE中的代码可减少外部存储访问
- RISC对于降低功耗不一定好，而CISC也不一定不好！

两种类型的处理

- 固定速率处理(如，多媒体通信中的信号处理)
 - 基于流的计算
 - 除了实时约束外无法得到更高的吞吐率
- 可变速率或突发模式计算 (如，通用计算)
 - 除了突发计算外，大多数空闲
 - 越快越好

固定速率处理中的折衷

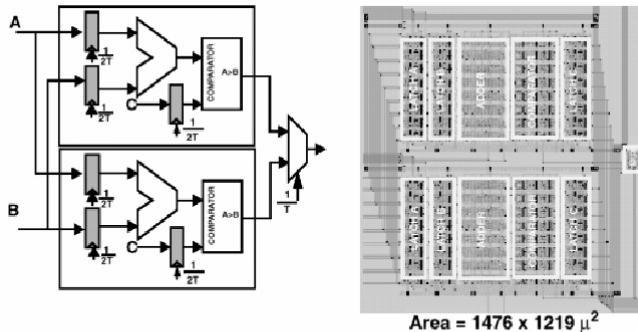


Area = 636 x 833 μ^2

- Critical path delay $\Rightarrow T_{\text{adder}} + T_{\text{comparator}} (= 25\text{ns})$
 $\Rightarrow f_{\text{ref}} = 40\text{Mhz}$
- Total capacitance being switched = C_{ref}
- $V_{\text{dd}} = V_{\text{ref}} = 5\text{V}$
- Power for reference datapath = $P_{\text{ref}} = C_{\text{ref}} V_{\text{ref}}^2 f_{\text{ref}}$

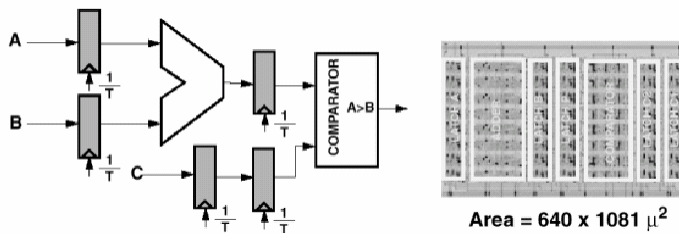
from [Chandrakasan92] (IEEE JSSC)

并行数据通路



- The clock rate can be reduced by half with the same throughput $\Rightarrow f_{\text{par}} = f_{\text{ref}} / 2$
- $V_{\text{par}} = V_{\text{ref}} / 1.7$, $C_{\text{par}} = 2.15C_{\text{ref}}$
- $P_{\text{par}} = (2.15C_{\text{ref}}) (V_{\text{ref}}/1.7)^2 (f_{\text{ref}}/2) \approx 0.36 P_{\text{ref}}$

流水线数据通路

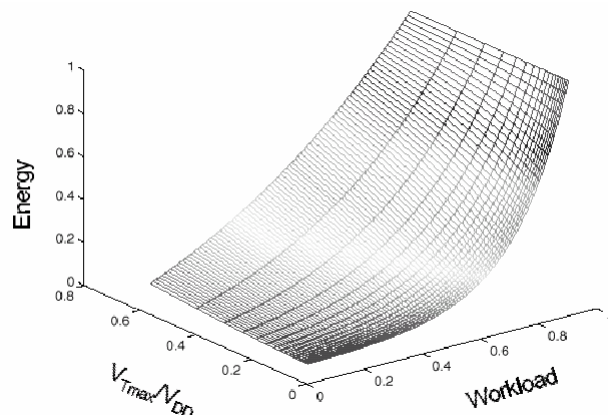


- Critical path delay is less $\Rightarrow \max [T_{\text{adder}}, T_{\text{comparator}}]$
- Keeping clock rate constant: $f_{\text{pipe}} = f_{\text{ref}}$
Voltage can be dropped $\Rightarrow V_{\text{pipe}} = V_{\text{ref}} / 1.7$
- Capacitance slightly higher: $C_{\text{pipe}} = 1.15C_{\text{ref}}$
- $P_{\text{pipe}} = (1.15C_{\text{ref}}) (V_{\text{ref}}/1.7)^2 f_{\text{ref}} \approx 0.39 P_{\text{ref}}$

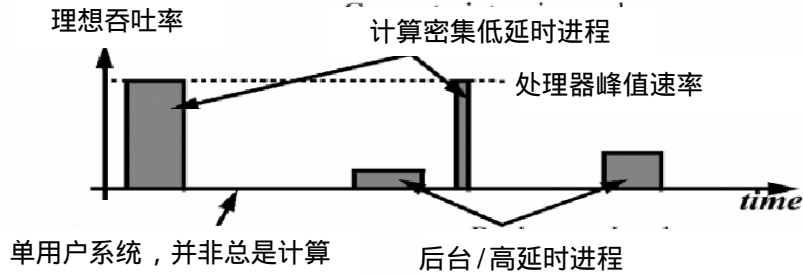
数据通路总结

Architecture type	Voltage	Area	Power
Simple datapath (no pipelining or parallelism)	5V	1	1
Pipelined datapath	2.9V	1.3	0.39
Parallel datapath	2.9V	3.4	0.36
Pipeline-Parallel	2.0V	3.7	0.2

可变速率处理

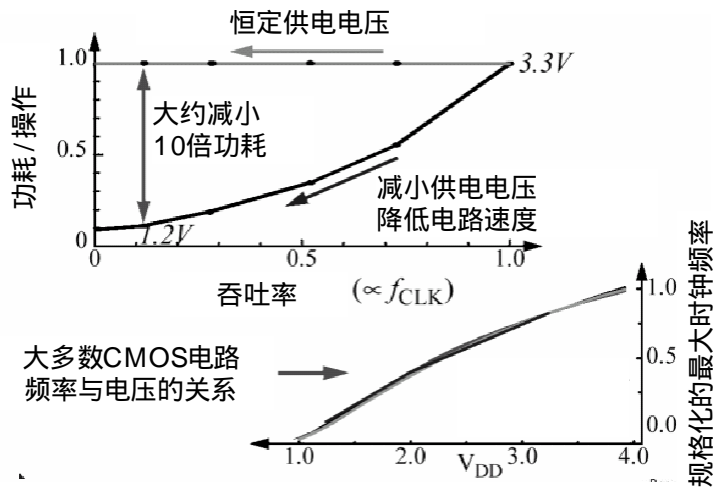


处理器使用模型

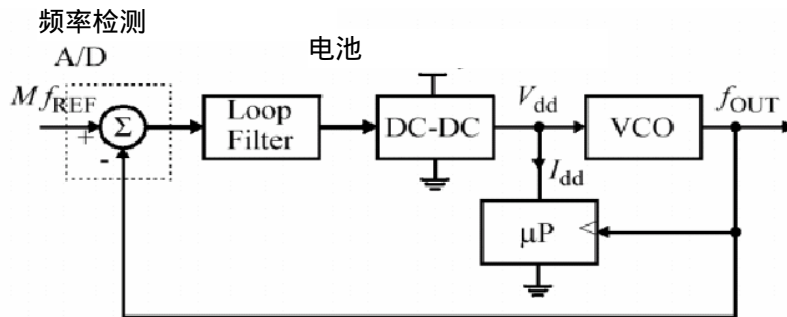


系统优化：
最大峰值吞吐率
每个操作的最小功耗
每个电池的最大计算量

时钟频率的伸缩



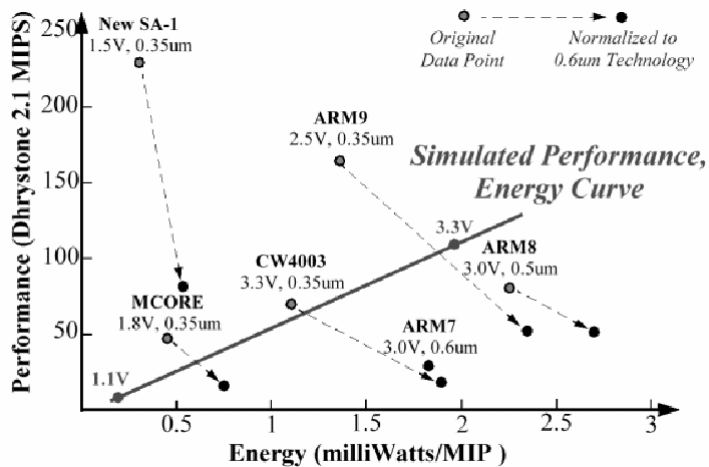
实现



VCO: 循环振荡器，与 μP 相匹配；

DC-DC: 执行数模转换，将电池转换成规则电压；

不同处理器功耗比较



功耗测量

- 没有一个好的测量标准，与性能一样
- MIPS/瓦较为常见
- 但不说明任何性能
- 摩托罗拉发明Mcore‘Powerstone’：15个嵌入式应用集

功耗的测量

- MIPS/Watt对正在运行的应用可能毫无疑义
- 确定处理器功耗最好的方法是运行应用
- Cache模型，总线行为,数据使用都会影响系统的功耗
- 系统的设计者应该站在全局的角度，建立高层模型

综合考虑

- 功能部件的增强意味着处理器完成更高的性能
- 大小,重量,封装,电池寿命意味着部件必须充分发挥功耗的效果
- 电池技术提高不大
- 制作工艺提高较大但带来新的问题
- 静态的漏极功耗与动态功耗一样,已经成为重要因素
- 在设计上如何考虑呢?

系统设计

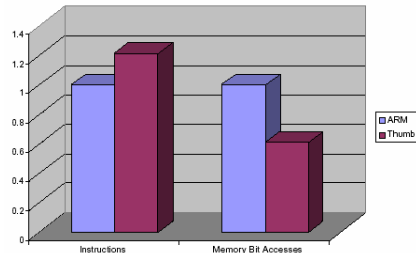
- 抽象的层次越高,降低功耗的余地就越大。

系统功耗与处理器的能力

- 16位Thumb指令系统是32位ARM指令系统的子集

Dhrystone 2.1应用程序测试：

- 取指增加21%
- 存储访问减少39%
- 系统功耗降低超过处理器的功耗增加



专用硬件与可编程处理器

- 专用硬件比非常灵活的处理器更有效
- 硬件乘法比软件例程更有效：减少取指和读寄存器
 - 专用DSP功能比处理器更有效
- 可配置处理器可提供更低的功耗
 - 该结构允许系统设计者根据硬件的复杂度情况增加指令和功能

系统级功耗控制

- 复杂的系统要求功耗觉察操作系统来降低系统的运行功耗
- 处理器需要提供显式控制给操作系统和应用系统
- 功耗控制包括静态的睡眠模式以及动态电压伸缩控制
- 嵌入式系统中的睡眠模式控制
 - On/off 电源控制
 - 时钟停止
 - 状态保存然后关掉电源

动态电压伸缩

- 电压伸缩技术应用于移动x86市场
 - • Intel - SpeedStep
 - • AMD - PowerNow
 - • Transmeta - Longrun
- 通过优化电压和频率，来降低给定工作负载的功耗
- 带来一个新的挑战 – 处理器设计者应保证电路的电压应在一定范围内线性变化

系统级设计总结

- 处理器设计者在考虑外部性能的同时
 - 功耗也十分重要
- 需要采用从上倒下的方法考虑低功耗，以得到低功耗的最终产品系统
- 软件和硬件的交互作用在设计中变得十分关键
- 如何设计电路来实现电压的伸缩变化

结构级设计

- 流水
- 并行
- 推测

微结构对于功耗的有效性

- 复杂 = 功耗
- 流水深度
 - 较多流水段增加D-触发器的数量和时钟负载
- 并行性
 - 并行执行单元增加总电容
- 推测
 - 推测错误浪费功耗
- 然而，性能要求也在提高
 - 以结构复杂度换取功耗

流水深度

- 深度流水意味着更多的寄存器
- 更多的寄存器-更多的负载-更多的功耗
 - 基于D触发器的低功耗设计
- P4具有20级流水线，42M晶体管，66w
- 驱动时钟占据功耗的重要部分
- Alpha 21264中时钟功耗占30-40%

推测功耗高，简单为好

- 转移预测经常用来改进处理器的CPI
- 包括历史纪录在内的复杂调度功耗较高
- 实际上在系统中存在另一个CACHE
- 简单的静态调度性能改进低，但功耗也低
- 两种情况下的错误猜测都产生功耗
- 任何推测都要求保存机器的原始状态

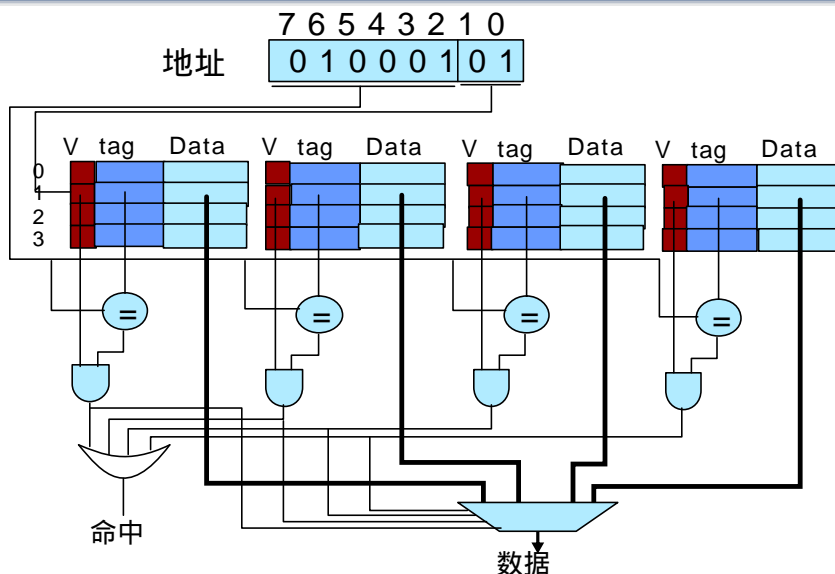
可扩展性

- 超标量和超长指令字处理器增加复杂度来增加指令的吞吐率
- 并行执行单元
- 资源冲突的硬件解决
- 增加的复杂度会严重消耗功耗
- Alpha 21264乱序发射逻辑消耗18%的芯片功耗
- VLIW核代码密度很差
- 导致存储系统的高功耗

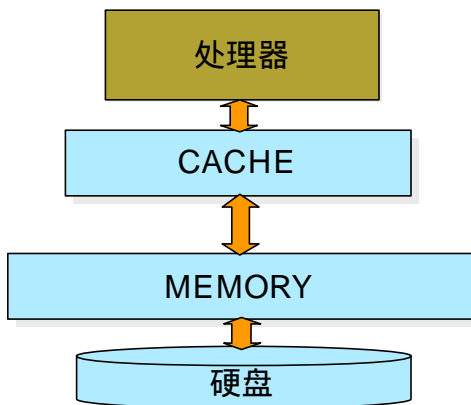
处理器是否越简单越好？

- Yes and no
- 如能剪裁操作的电压，则
- 在面积一定的约束下制作最高性能的处理器，但
 - 有效控制时钟频率
 - 确保电路能处理低电压
 - 并能在可能的最低电压下工作
- V^2 的节省可以大于C和F的增加
- 然而，开销以及系统的原因经常不允许改变操作电压：此时，应保持尽量简单

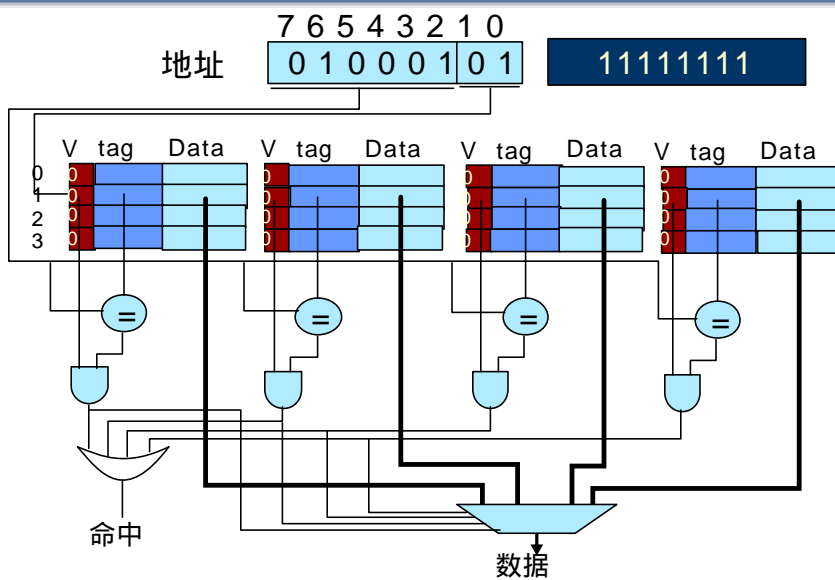
CACHE的功耗



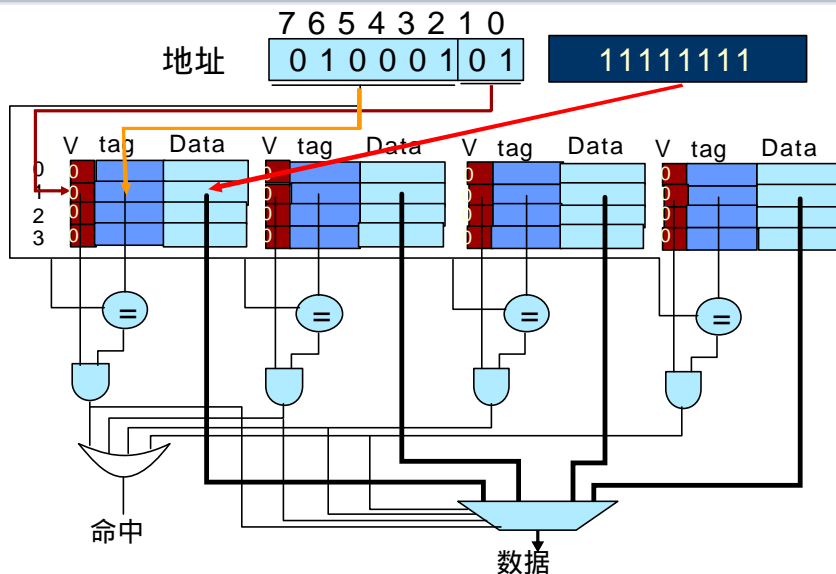
CACHE的功耗



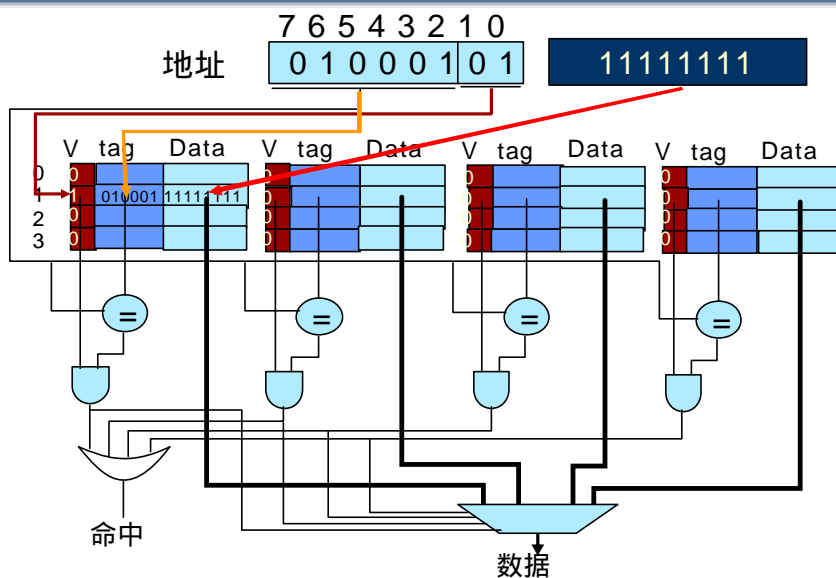
CACHE的功耗：写CACHE



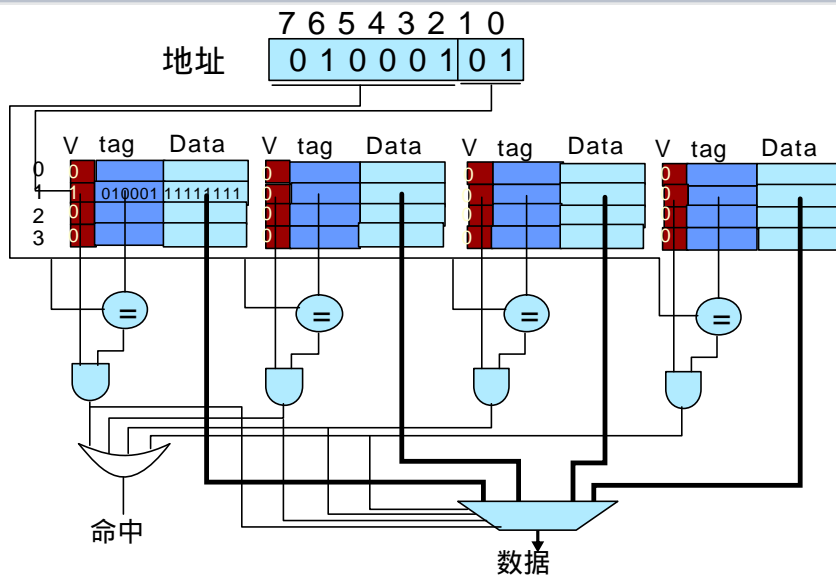
CACHE的功耗：写CACHE



CACHE的功耗：写CACHE



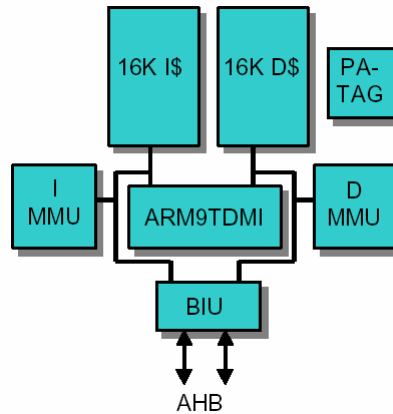
CACHE的功耗:读CACHE



- 究竟什么地方消耗能量？
 - ARM920T分析

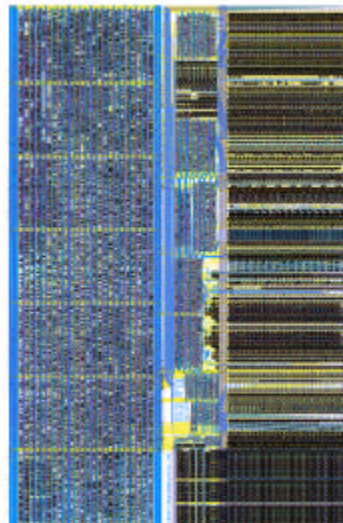
ARM9TDMI

- Harvard结构processor
ARM9TDMI核
2x16K caches
MMU support for VM
- 支持写返回和写穿透cache
 - 写穿透保持一致性
 - 写返回有利于低功耗
- 2.5 Million transistors
- TSMC 0.18mm:
- 1mW/MHz (1.8V)
- 200MHz (1.65V)
- 11.8mm²



ARM9TDMI

- 5 stage pipeline
- Harvard architecture
- ARM v4T compliant
 - ARM and Thumb instruction decoders
- 110,000 transistors
- TSMC 0.18μm:
 - 0.3mW/MHz (1.8V)
 - 220MHz (1.65V)
 - 1mm²



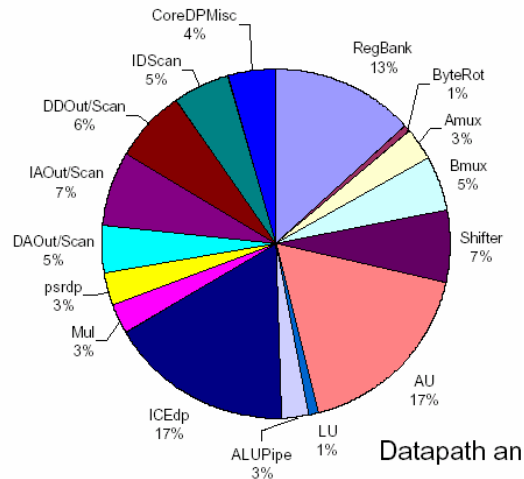
ARM9TDMI 的功耗分析

Datapath: 43%

- RegBank+AU+
Debug = 47%

Control logic: 53%

Clock driver: 4%

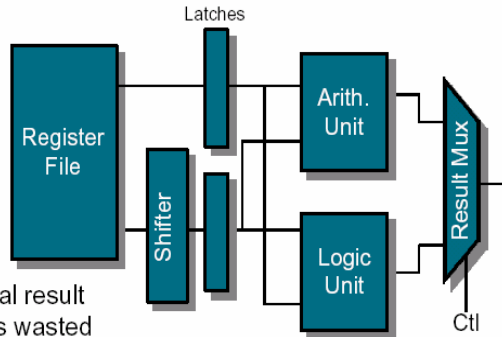


控制子模块分析

- 低功耗设计的规则:
 - 不要激活或时钟触发非活动模块的输入
 - 尽量分离功能模块
 - 尽量早地导出指令的控制信号以支持模块分离
 - 尽量降低产生控制信号的功耗

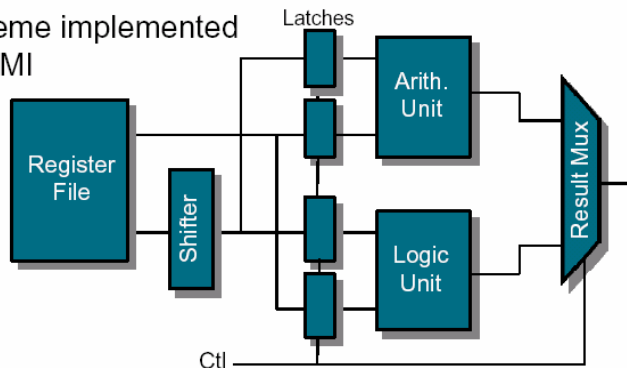
ARM7核功耗分析

- Analysis shows
 - AU used 61% of cycles
 - LU used 39% of cycles
- When LU in use the AU is unnecessarily driven
 - when calculating a logical result 40% of the total power is wasted by the adder
- Therefore isolating the AU can save significant power

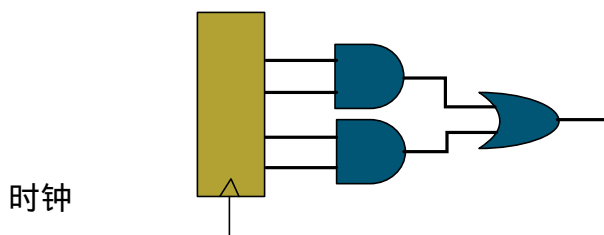


改进的设计

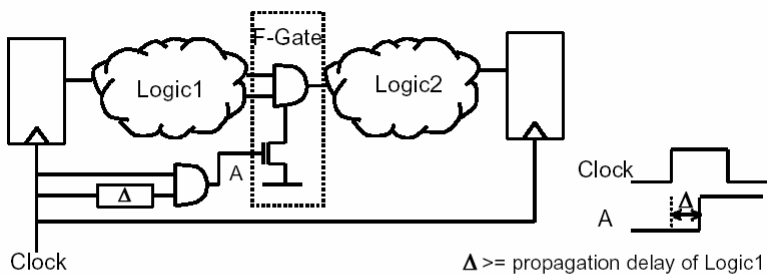
- Potential for 20% saving in power in the ALU
 - no cost in performance
 - extra 64 latches
- Similar scheme implemented in ARM9TDMI



冒险带来功耗

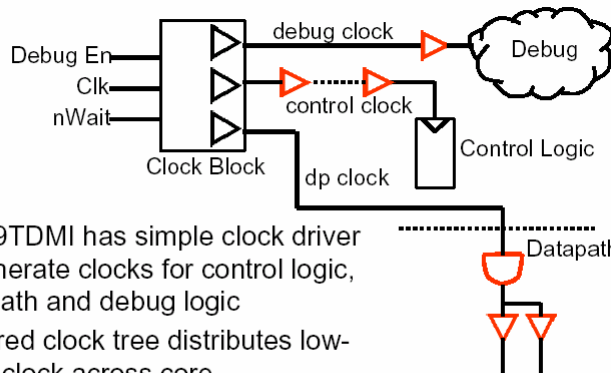


冻结门



- Logic2's inputs frozen while Logic1 evaluates
- Cells grouped together under common delayed clock signal
- Minimal impact on speed and area

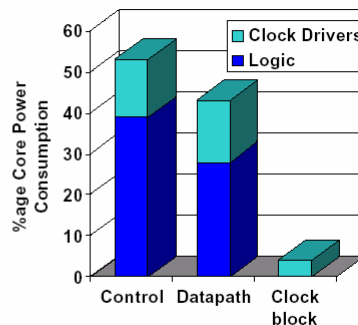
ARM7TDMI的时钟域



- ARM9TDMI has simple clock driver to generate clocks for control logic, datapath and debug logic
- Buffered clock tree distributes low-skew clock across core
- Global wait signal factored into all clocks

时钟功耗

- Clock driver power only 4% of total core consumption
- Power for entire clock tree totals 32% of core
 - excludes clk logic within latches
- Early gating of debug clock allows all debug logic to be frozen when not in use
 - saves ~10% of total core power
- Global wait signal factored into all clocks



异步电路

- Processors built using asynchronous design styles offer the potential for low power implementations
 - No clock, so no clock-tree power
 - side benefit is low electro-magnetic radiation
- Functional units automatically power up depending on which instruction is in execution
 - much finer resolution and no explicit idle-state decode logic
- Manchester University leading with AMULET3i⁽¹³⁾
 - Latest in family with performance similar to ARM9TDMI
 - First commercial use in DRACO wireless DECT chip

异步电路的前景

- Mainly still a research activity
- Few implementations to show that the technology can be employed in a commercially interesting way
- Designers brought up on synchronous techniques
- No commercial design tools to help
- But, great potential
 - low power, low emissions, no clock to distribute
- Using asynchronous interfaces between synchronous processing elements on a large SoC could provide a home for this technology

漏极电流

- On low voltage (low V_t) processes leakage currents can become significant to the overall power consumption
 - typically 10-20pA/transistor when $V_t \sim 0.7V$
 - increases to 10-20nA/transistor when $V_t \sim 0.2-0.3V$
- Combination of microarchitecture and circuit techniques required to control leakage currents

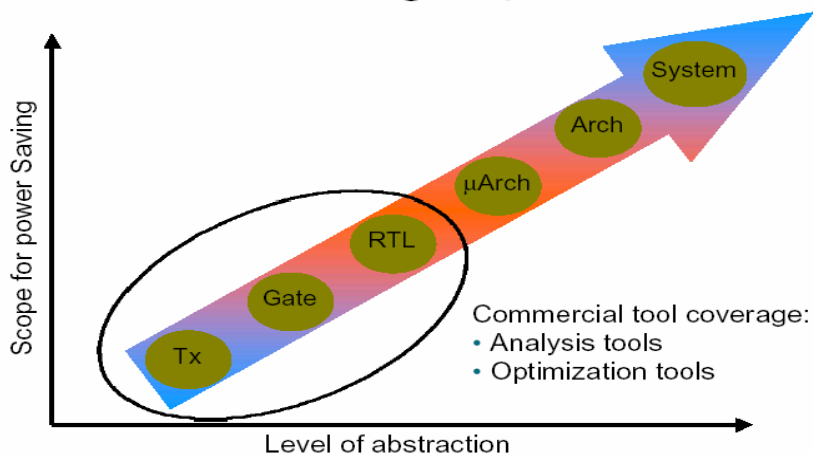
漏极电流的控制策略

- Processor sleep modes remove power to leaky sections of the design when not in use
 - eg a multiplier or a cache
- State often needs to be saved to memory
 - system trade off of power cost to write out to memory and then reload vs leakage power saved
 - need OS to determine how long the sleep period is likely to be
- At circuit level substrate biasing and MTCMOS are main approaches in addition to careful transistor sizing

总结

- Understand where the power goes in your CPU from historical data and then optimise your design
- Caches, ALUs and register files are big power burners
- Clocks contribute significantly to the power so choose your d-types well and clock gate early
 - or design out the clock all together!
- Ensure that infrequently used logic is clock-frozen, isolated from input activity and even remove Vdd
- Leakage is now a significant issue which affects your microarchitecture and your circuits

EDA工具：设计空间



晶体管级

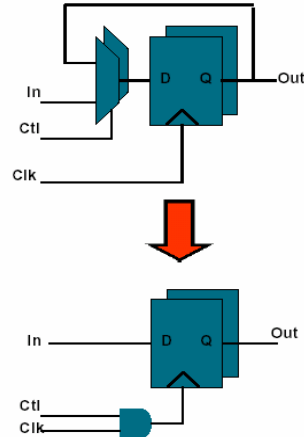
- Post-layout analysis tools predict what your power consumption will be, but too late in the project to allow large scale changes
 - Synopsys-EPIC PowerMill⁽¹⁵⁾
 - Mentor Mach PA⁽¹⁶⁾
- Claim close correlation with Spice and real silicon
 - better than 5% accurate
- AMPS⁽¹⁷⁾ offers transistor size optimization for power and performance
- Avant! Mars-Rail⁽¹⁸⁾ for power driven Place and Route

门级和RTL级分析

- Main tool offerings from Synopsys and Sequence
 - PrimePower⁽¹⁹⁾
 - WattWatcher⁽²⁰⁾
- Gate level simulations provide a graphical display of switching activity across a circuit to allow design optimizations for power
- Rely on using a power characterized cell library
- WattWatcher can also perform analysis on RTL
 - logic complexity estimated by analyzing RTL

门级与RTL级优化

- Synopsys' PowerCompiler⁽²¹⁾ insert gated clocks on registers
 - can also optimize logic gate selection based on library power characterization data
- Sequence's WattSmith⁽²²⁾ analyses the RTL and its WattBots then propose RTL level power optimizations



高层抽象

- Current design tools only cover low end of the design space today
- No tools exist today for microarchitecture and architecture analysis and optimization
 - the EDA industry needs to keep pushing up the abstraction graph
- The system designer needs power models of CPUs to make high level trade-offs
 - model the dependency on instruction and data patterns
 - allow the effects of code and data representation changes to be determined

结论

- Power consumption is as important a design criteria as performance
 - even if your application is plugged into the wall
- Low power design starts at the system level
 - a top down approach will yield greatest results
- Power management is a system issue that requires circuit, microarchitecture and software interaction
- Performance requirements are increasing and so microarchitectural complexity must go up without sacrificing power

结论（续）

- Set your power budget at the start of the design and measure it as you go
 - EDA tools are becoming available to help
 - But the EDA industry needs to be encouraged to do more
- Understand where the power goes in your designs today and use the data to improve future products
- Low power design presents new challenges
 - still an immature technology
 - reducing mW is much more interesting than increasing MHz!