ICT Center of Excellence For Research, Innovation, Education, and life-long Learning

Politecnico di Milano

Power Optimization of Digital Systems

System-level power optimization

Lecturer:

Prof. William Fornaciari

Politecnico di Milano

fornacia@elet.polimi.it

www.elet.polimi.it/~fornacia

System-level optimization

cefriel
Politecnico di Milano

- System
  - Collection of components organized to provide a service
  - Each component has a power consumption profile
- Power optimization can be achieved by optimizing
  - Architecture and implementation of components
  - Communication among components
  - Use (exploitation) of components
- Typical actions for power optimization
  - Hw/Sw partitioning and allocation
  - Hw design
  - Sw design
  - Dynamic (es. runtime) power management

Hw/Sw Partitioning

cefriel
Politecnico di Milano

- Typical design toolchains support the designer in optimizing performance under area constraints
- Hw and Sw components are optimized later throughout the design flow, even if hw/sw allocation can impact of overall power consumption, dramatically
- Possible approaches, depending on the specification
  - Task-level
    - The system model is functional, where elements are tasks to be executed
  - Behavioral level
    - The model is structural, elements are Hw components performing specific operations

Task-level partitioning

cefriel
Politecnico di Milano

- Typical model is a task graph
  - Nodes are tasks
  - Edges are dependencies
  - Periodic execution is assumed (in general the bulk of consumption is originated by periodic activities)
- Solution space: set of processing elements (PEs)
  - GPPs, DSPs, Programmable Logic, custom processors (ASIPs, macrocells)
- Cost metrics: power/performance
  - Number of cycle (at given Vdd)
  - Energy per cycle
- Constraints
  - performance and cost
  - in some cases supply voltage can change

Task-level partitioning

cefriel
Politecnico di Milano

- Additional metrics
  - Cost of memory (local to the PEs and global) and communication power
  - Cost of multitasking
  - Cost of power management
- Solution
  - Solve a constraints scheduling and allocation problem
- Limitations
  - Pre-definition of tasks
  - Difficult exploitation of functional information is possible
  - Power characterization is required for any task on any PE

Behavioral-Level Hw/Sw partitioning

cefriel
Politecnico di Milano

- Model
  - Algorithmic specification in HL language with concurrent statements
- Solution space
  - System resources: macro architectural templates
- Cost metrics/constraints: same of Task-level
- Good power modeling accuracy thanks to templates
  - Processor core: instruction-level power model
  - ASIC cores: custom power models (behavioral/RTL)
  - Memory model and interconnect: parametric power models (word. Size, address size)
- Accuracy close to that of RTL models

Behavioral-Level Hw/Sw partitioning

cefriel
Politecnico di Milano

- Solution
  - Iterative improvement of an initial “schedule” according to some criteria
    - Refinement of initial schedule requires interaction with Instruction-level simulator
    - Code profiling is required to achieve data-dependent estimates of the criterion
- Examples
  - Criterion: Ur utilization of a resource (ratio of active cycles over total cycles in the initial schedule)
  - Refinement: if a resource has Ur on the cores w.r.t that of the processor, then realize it in Hw

Behavioral-Level Hw/Sw partitioning

cefriel
Politecnico di Milano

- Critical issues
  - Availability of accurate models is crucial
  - Problem is not even solved at the RTL
  - Assumption of architectural template may prevent the applicability to general purpose systems
  - The algorithmic specification maybe “non-neutral” and can bias the partitioning process

Hw power optimization

cefriel
Politecnico di Milano

- Choice of data representation
- Bus encoding
- Memory design
- Instruction encoding

Choice of data representation

cefriel
Politecnico di Milano

- This techniques aims at reducing the switching factor of C_{eff} by properly choosing the representation of data
- In most DSP chips (and micro) numbers are in 2's complement notation. The sign-bit extension is a problem when:
  - Number are switching around zero (+ or -) very frequently
  - Numbers do not utilize the entire bit-width (numbers are small…)
  - In the above case all the bits but one tend to toggle

Choice of data representation

cefriel
Politecnico di Milano

- Use of bit sign and magnitude
  - Produces less bit transitions
  - Is function of both dynamic range of represented numbers and signal correlation factor
- Significant advantages when
  - Dynamic range of represented numbers is small
  - Consecutive data words are highly negatively correlated
  - Sign/magnitude used for bus encoding
  - 2's complement useful for some arithmetic circuits (smaller so resulting in less power consumption)
- Best solution
  - Use of 2's complement inside arithmetic circuits while adding encoding/decoding logic at the interface with buses

Bus encoding

cefriel
Politecnico di Milano

- Load capacitance for the I/O are several orders of magnitude higher than the internal circuits nodes
- On average, power dissipation at I/O accounts for half of the overall switching power
- Power saving can be achieved decreasing switching activity of the I/O trough encoding, even by paying an increasing of the number of internal transitions
- Example of bit encoding
  - Active-high encoding: use of high-level voltage for logic “1” and low level for logic “0”
  - Transition-based encoding: voltage change for logic “1” and no change for “0”

Bus encoding

cefriel
Politecnico di Milano

- Word encoding
  - Assignment of patterns of 1's and 0's to each word of information to be transmitted
- Example of word encoding
  - Non-redundant code: n bits are used to encode 2^n possible patterns

Example: bus invert econding

cefriel
Politecnico di Milano

- Assumptions
  - N bit data bus with 2n patterns to be represented
  - Patterns are equiprobable and totally uncorrelated
  - Hence, the probability of = or 1 on each bus line is 0.5, on the bus there is and average of n/2 transitions per clock cycle
- Looking at two consecutive patterns
  - If Hamming distance between the current pattern and the next one is ≤ n/2 the pattern is transmitted as it is
  - If Hamming distance between the current pattern and the next one is > n/2 the pattern is first inverted before transmission

Example: bus invert econding

cefriel
Politecnico di Milano

- Advantage
  - The method guarantees that the max number f transition per clock cycle is n/2, half the original value
- Draw-back
  - One extra bus line is required toi indicate when the pattern is inverted
- Note
  - The average # of transition per clock cycle is also lowered by le less than 25% of the original value (because of the binomial distribution of the distance between consecutive patterns)

Transition level encoding

cefriel
Politecnico di Milano

- Transition based encoding instead of a level encoding schema is useful when lines are not equiprobable
- Let p(0) and p(1) be the input for the i-th line of a data bus, with p(0) > p(1)
- If active-high encoding is used, the average number of transition per clock cycle is:
  - p(0) p(1) + p(1) p(0) = 2p(1) (1-p(1))
- Example
  - If p(1)=0.25 and p(0) = 0.75, then the avg number of transition per colck cycle is 3/8
  - Using transition-based encoding, the abg transitions per clock cycle is simply p(1), since there is one transition for each 1 on the input line

Transition level encoding

cefriel
Politecnico di Milano

- Example
  - If p(1)=0.25 and p(0) = 0.75, then the avg number of transition per clock cycle is 33% less than the case of active-high encoding
  - The use of transition-based encoding is advantageous if the inputs are strongly equiprobable
- Solution
  - Encode input patterns using limited-weight codes, which guarantee a good control on the values of p(0) and p(1) of each bit line. Then apply bit encoding using transition-based scheme

Address Bus Encoding

cefriel
Politecnico di Milano

- The encoding schemes analyzed till now are suitable to transmit on the data bus
- Different techniques can be used to reduce switching on address bus lines
- Some popular approaches
  - Gray code (irreduntant)
  - T0 code (redundant)
  - Working Zone code (redundant)
  - Beach code (irredundant)

Gray code

cefriel
Politecnico di Milano

- Gray code
  - Two consecutive words of the code have Hamminh distance
- Observation
  - Due to instruction locality, most of the memory accesses are sequential by nature
- Consequence
  - A significant number of switches can be eliminated by using gray code addressing
  - In average binary addressing requires 2 switches per cycle for large address lines, while gray only 1 switch per cycle for any n.

Binary vs Gray

cefriel
Politecnico di Milano

- Savings about 37% transitions for some benchmarks
- For byte addressable CPUs, address elementary increments depends on word length
- Example
  - 32 bit CPU-> consecutive addresses have distance 4
  - 64 bit CPU-> consecutive addresses have a distance 8
- The gray code must be modified to keep the switching activity low
- Problem: address generation circuits are more complex than binary ones
- Literature (best solution depends on load capacitance and bus size)
  - Use of gray address generation converting binary address
  - Binary address generators and off-chip conversion circuitry

T0 code

cefriel
Politecnico di Milano

- Gray code is optimal for irredundant code, performing 1 transition per clock cycle
- By adding redundancy (as in the bus-invert code) performance can improve
- T0 ensures zero transition (asymptotic) for in-sequence addresses
- Basic idea
  - Add one bus line, INC, that is “1” when two consecutive addresses must be transmitted
  - Whenever INC=1, freeze value of address line
  - The receiver comutes the new address

T0 code

- Larger and more power consuming than Gray encoder and decoder
- Faster than Gray encoder and decoder
- When the probability of finding two consecutive addresses is >50%, T0 outperforms Gray code
- Example for MIPS processor, power saving w.r.t Gray
  - Instrcution bus: 39%
  - Data Address bus: negligible
  - Mupltiplexed bus 10%
  - Large saving w.r.t. binary addressing
- T0 Encoder/decoder implementation
  - Using Verilog RT-level and Synopsys design compiler onto 3.3V Motorola M5C library
  - Critical delay: 2.8 ns
  - For q>0.3, dissipation below 20 μW

Working Zone Code

cefriel
Politecnico di Milano

- When strict sequentiality is low, address locality may still exist
- Assumption
  - Application tend to privilege a few working zones of the address space
- Idea
  - Describe memory references by
    - An identifier of the working zone
    - An off-set
  - Send this encoding over the bus
- Off-set specification
  - W.r.t. the base address of the WZ
  - W.r.t. the previous reference to that zone

Working Zone Code

cefriel
Politecnico di Milano

- Hardware requirements
  - One register for each WZ at the transmitting end
  - One register for each working zone at the receiving end
  - Additional bus lines to signal to the receiver what WZ is active
- Implementation
  - With few WZs, no additional registers are necessary
  - Reduced number of additional bus lines
- Use existing bus lines to transmit the off-set encoded with a one-hot scheme
- When the reference does not correspond to any chosen working zone, the entire reference is transmitted (the case is signaled to the receiver)

Working Zone Code

cefriel
Politecnico di Milano

- Limitation
  - Application may have a large number of working zones
    - To reduce the additional registers, adopt a WZ caching mechanism -> smaller power savings
  - Adding bus lines maybe unacceptable
    - Use fewer existing lines for off-set transmission and the rest for the WZ identifier -> smaller power saving
- Results
  - Motion estimation algorithm
    - 47% saving w.r.t. binary
  - Quicksort
    - 67% saving w.r.t. binary

Beach code

cefriel
Politecnico di Milano

- There exist cases where addresses with low sequentiality and too many WZs
  - Gray code, T0 and WZ are not appropriate
- Basic idea
  - Discover coorelations (es. Block correlation between time-adjacent patterns) by inspecting typical address traces and determining a suitable encoding schema
- Applicability
  - Embedded systems where the dedicated processor repeatedly executes the same portion of embedded code

Beach Code calculation

cefriel
Politecnico di Milano

- Collect statistics identifying possible block correlation in the given address stream
- Group bus lines in clusters according to their correlations
- For each cluster automatically generate an encoding function
- Automatically synthesize encoding and decoding logic to be placed at the address bus terminals

Beach Code calculation

- Correlation measures (between pairs of bits)
  - Spatial correlation
    - Likelihood of correctly predicting the value of bit i of pattern k given the value j in pattern k
  - Spatio-temporal correlation
    - Likelihood of correctly predicting the value of bit i of pattern k given the value of bit i in pattern k-1 and of bit j in pattern k
  - Switching correlation
    - Likelihood of correctly predicting the occurrence of a transition on bit I given a transition on bit j in pattern k-1
- For each measure, a matrix C is built, whose entries c_{ij} represent the pairwise correlation between bit I and bit j
    - The matrix is considered as the adjacency matrix of a Graph onto which clustering is applied
    - From this clustering, the synthesis of encoding/decoding logic is automatizable
- Avg power saving over binary address encoding
  - 41.9%, with few minutes runtimes

Information-Theoretic Code

cefriel
Politecnico di Milano

- Use of the concept of correlator to simplify the encoding problem
- New problem formulation
  - Minimize word transition probabilities, that is, minimize the number of 1's being transmitted
  - Ensure decodability of the encoded message
- Rather tricky implementations, approximates solution are typically adopted (see literature)
- Power saving in the order of 40-50% over binary encoding

Memory design

cefriel
Politecnico di Milano

- Memory and caches of a digital systems usually account for a large fraction of the total system power
  - Es. L1 and L2 of a Alpha processor dissipate 25% of the total power
- Target
  - Minimization of the power due to memory accesses
  - Minimization of power due to data transfer

Memory design

cefriel
Politecnico di Milano

- Minimization of memory access power
  - Fixed memory access patterns
    - Optimize memory hierarchy
  - Fixed memory architecture
    - Optimize memory access patterns
  - Concurrent optimization of memory architecture and access patterns is still an open issues
- Minimization of data transfer power
  - Code density optimization

Memory access power optimization

cefriel
Politecnico di Milano

- Basic idea: Close vs far memory accesses
  - Close: faster, less power, small block size
  - Far: slower, more energy consuming, larger block size
- Strategy
  - Enforce locality in the cache and memory subsystems
- Solutions
  - Data Replication
  - Alternative to caches (e.g. scratch-pad buffers)
  - Cache/Memory partitioning

data replication

cefriel
Politecnico di Milano

- Implicit data replication
  - Use of a filter cache
    - Introduce an extra L0 cache (e.g. 256 byte)
    - Energy delay product is reduced
- Explicit data replication (some proposals)
  - Exploit buffers along I-cache and D-cache, No latency penalty
  - Use of buffers as victim cache
    - Accesses on main cache miss
    - Hit: datum is promoted to main cache, the replaced line in the cache is moved to the victim cache
    - Miss: L2 cache is accessed
      - The incoming datum is put in the main cache
      - The replaced line in the cache is moved to the victim cache

Cache memory partitioning

cefriel
Politecnico di Milano

- Multi-bank caches
  - Use independently-addressable bancks
  - Two dimensional partitioning: M modules with B bancks each
  - Power saving achieved through exploitation of reduced capacitance of smaller memories
  - Ad hoc, low-power bank selection circuitry is used
- Partitioned memories
  - Memory hierarchy with independently-addressable memory bancks
  - Exploit sleep-mode features to shut down individual banks
  - Design memory partition so as to maximize the sleep time
  - Typ traces are used to drive the partitioning process (available for embedded systems)
  - Other approaches map most frequent address onto small partitions close to the processor

Code density optimization

cefriel
Politecnico di Milano

- Basic idea
  - Minimize program memory occupation so as to reduce the bandwidth of processor-memory communication
- Approaches
  - Custom instruction sets
  - Object code compression

Custom instruction sets

cefriel
Politecnico di Milano

- Viable solution for General purpose processors
- Es. ARM Thumb code
  - Interleaving of regular (32 bit) and thumb (16 bit) instructions
  - Requires modifications to the basic processor architectures
  - Requires specific compilers and software development Kits

Object code compression

cefriel
Politecnico di Milano

- Viable solution for embedded processor
- Idea
  - Exploit the small subset of instructions used by firmware code running on embedded processor
- Approaches
  - Full compression
  - Selective compression

Compression

cefriel
Politecnico di Milano

- Full compression
  - Replace all instruction with binary patterns of minimum width. A table (IDT) is used do decomp. The code
  - Use of standard compilers, no hw modification
  - Limitation: frequently the # of distinct instructions used is not small: large size of IDT , poor compression ratio
- Selective compression
  - Often program traces are covered by a small subset of instructions, only this subset is used for compression
  - Candidate: Instructions ensuring best program coverage
  - Program is a mix of compressed and uncompressed instructions, IDT is fixed
  - Fetching/decompressing is simplified
  - The controller has to handle instruction fetching
  - Average saving around 45%

Instruction encoding

cefriel
Politecnico di Milano

- Target is the minimization of the power consumed by a processor running typical software apps
- Focus on the circuitry for fetching and decoding
  - It is a small fraction of the overall power, but it can minimized at no cost
- Strategies consider the cost of loading binary pattern in registers
  - Power saving is reduced by assigning binary codes to the instruction op-codes so to minimize such power
- Example
  - MIPS R4000 (64 bit)
  - Avg saving 15-40% in switching activity