


Software-Controlled Operand-Gating


Ramon Canal, Antonio González and James E. Smith

Universitat Politècnica de Catalunya - Barcelona
University of Wisconsin - Madison




Outline

1. Introduction
2. Value Range Propagation
3. Value Range Specialization
4. Cooperative Hw-Sw
5. Summary & Conclusions



Introduction

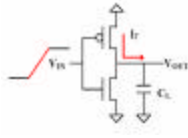

- Increasing miniaturization**
 - Portable handhelds (PDAs, notebooks, cell phones, etc.)
- Increasing computation requirements**
 - Multimedia processing (audio, video)
 - Speech-recognition
 - Audio and video encoding/decoding
 - Wireless communication
- Increasing demand for**
 - Longer battery life
 - Smaller heat dissipation



Motivation

Design for low power

- Becoming a critical design constraint
- Types of energy consumption
 - Dynamic (predominant source for current technology)
 - Charging/Discharging capacitors


Introduction

Design goal:

POWER EFFICIENCY

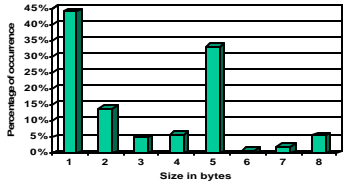
Metrics used:

- Energy consumption
- Energy-delay²




Value Compression

Data size distribution for the Spec Integer 95 benchmarks



Size in bytes	Percentage of occurrence
1	42%
2	15%
3	5%
4	5%
5	32%
6	2%
7	2%
8	5%




Software Techniques

Value Range Propagation

- Propagate at compile-time operand's range

Value Range Specialization

- Using value profiling information, specialize regions of code to certain value ranges



Value Range Propagation

Find initial value ranges


- Inst. declared with narrow operands (e.g. add_short)
- Assignments (e.g. Variable=constant)
- "if" condition statements

Forward propagation

- Set the range of the output operands
 - Depending on the input operands' ranges and operation

Backward propagation

- Set the range of the input operands
 - Depending on the output operand's range, the input operands' ranges and the operation




Value Range Propagation

"Useful" Range Propagation

Constrain the range of the values due to their operations

- Logical operations
 - AND R1, 0xFF, R2 (i.e. R2?R1&0xFF)
 - OR R1, 0xFFFFFFFF00000000, R2 (i.e. R2?R1|0xFFFFFFFF00000000)
- Mask operations
- Limited width fields
 - shift amounts



Value Range Propagation

Example of Value Range Propagation

- a0=<NTmin,INTmax>
- a1=<0,0>
- a3=<0,0>
- a2=<NTmin,INTmax>
- a1=<1,1>
- tripcount=100
- a1=<1,100>

```

a0 = 0a
a1 = 0
a3 = a1*4
a2 = a0+a3
mem[a2] = a1
a1 = a1-1
a1 < 100
return
    
```


a3 = <0, 396> 9.

a1in = <0,99> 8.

Original C code:

```

for (i=0; i<100; i++) {
    a[i]=1;
}
    
```




Value Range Specialization

Profiling based compile-time technique

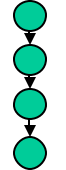
Three step process:

- Identification of instructions (candidates) where specialization may be profitable using basic block profiles (i.e. basic block counts)
- Computation of the ranges of values for the candidates with the support of profiling data
- Specialization of the candidates that are deemed profitable




Value Range Specialization

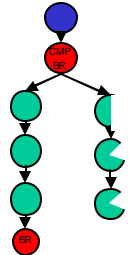
Candidates




Savings

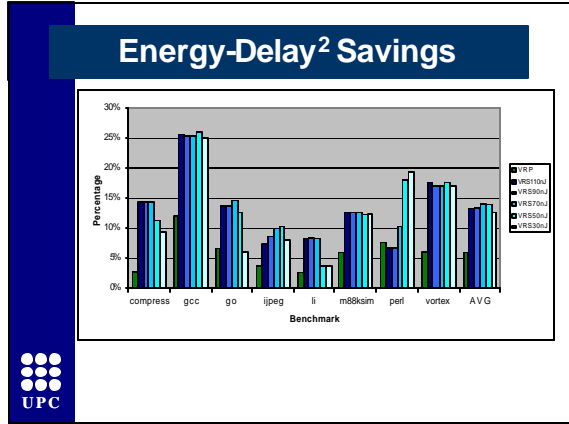
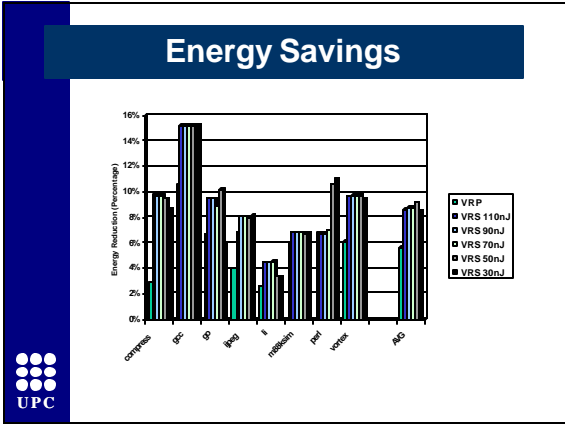


Specialization



3. Specialize candidates whose ranges and other characteristics are profitable according to savings





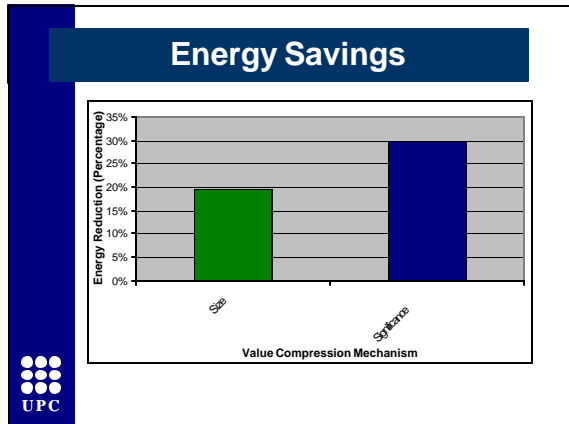
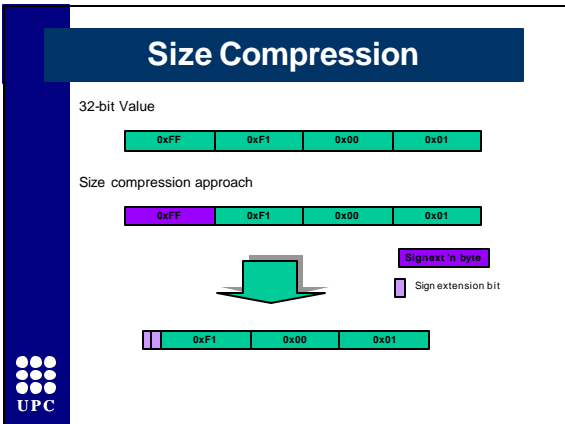
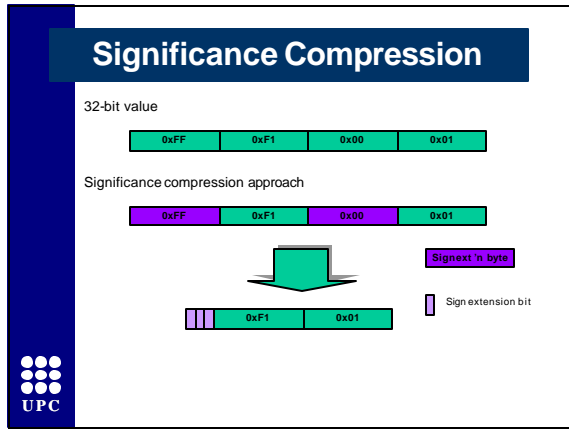
Cooperative Hw-Sw

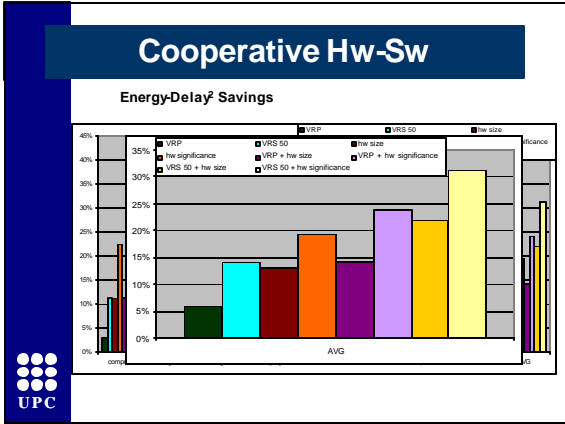
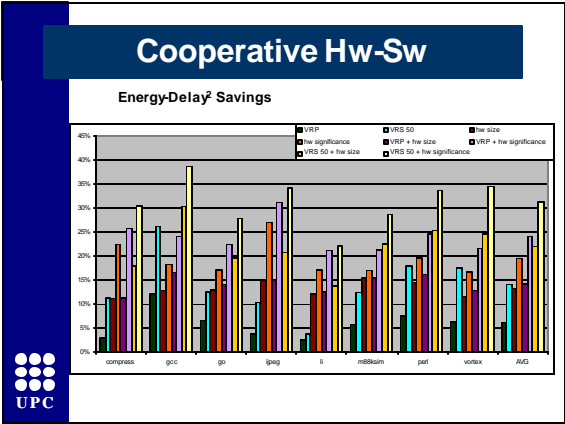
Advantages Hw

- ISA untouched
- Actual values

Advantages Sw

- Minimal hw complexity
- Wider program scope





- ## Summary
- Value compression is an effective way of reducing power requirements
 - Software less effective but less complex
 - Cooperative hw-sw for more effectiveness
- UPC