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

Design goal:
POWER EFFICIENCY

Metrics used:
- Energy consumption
- Energy-delay

Value Compression
Data size distribution for the Spec Integer 95 benchmarks
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

“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, 0xFFFFFFF00000000, R2 (i.e. R2?R1|0xFFFFFFF00000000)
- Mask operations
- Limited width fields
  - shift amounts

Value Range Specialization

Profiling based compile-time technique

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

Example of Value Range Propagation

Original C code:
```c
for (i=0; i<100; i++) {
    a[i]=i;
    a0 = @a
    a1 = 0
    a3 = a1*4
    a2=a0+a3
    mem[a2]= a1
    a1 = a1+1
    a1 < 100
    return
}
```

1. a0=<INTmin, INTmax>
2. a1=<0,0>
3. a3=<0,0>
4. a2=<INTmin, INTmax>
5. a1=<1,1>
6. tripcount=100
7. a1=<1,100>
8. a3 = <0, 396>
9. a1in = <0,99>

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 operands range, the input operands' ranges and the operation
Energy Savings

Energy-Delay\(^2\) Savings

Cooperative Hw-Sw
Advantages Hw
- ISA untouched
- Actual values

Advantages Sw
- Minimal hw complexity
- Wider program scope

Significance Compression
32-bit value

Size Compression
32-bit Value

Energy Savings
Summary

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