Automatic Generation of Application Specific Processors

David Goodwin
Darin Petkov

Tensilica, Inc.

Copyright ©2003 by Tensilica, Inc.

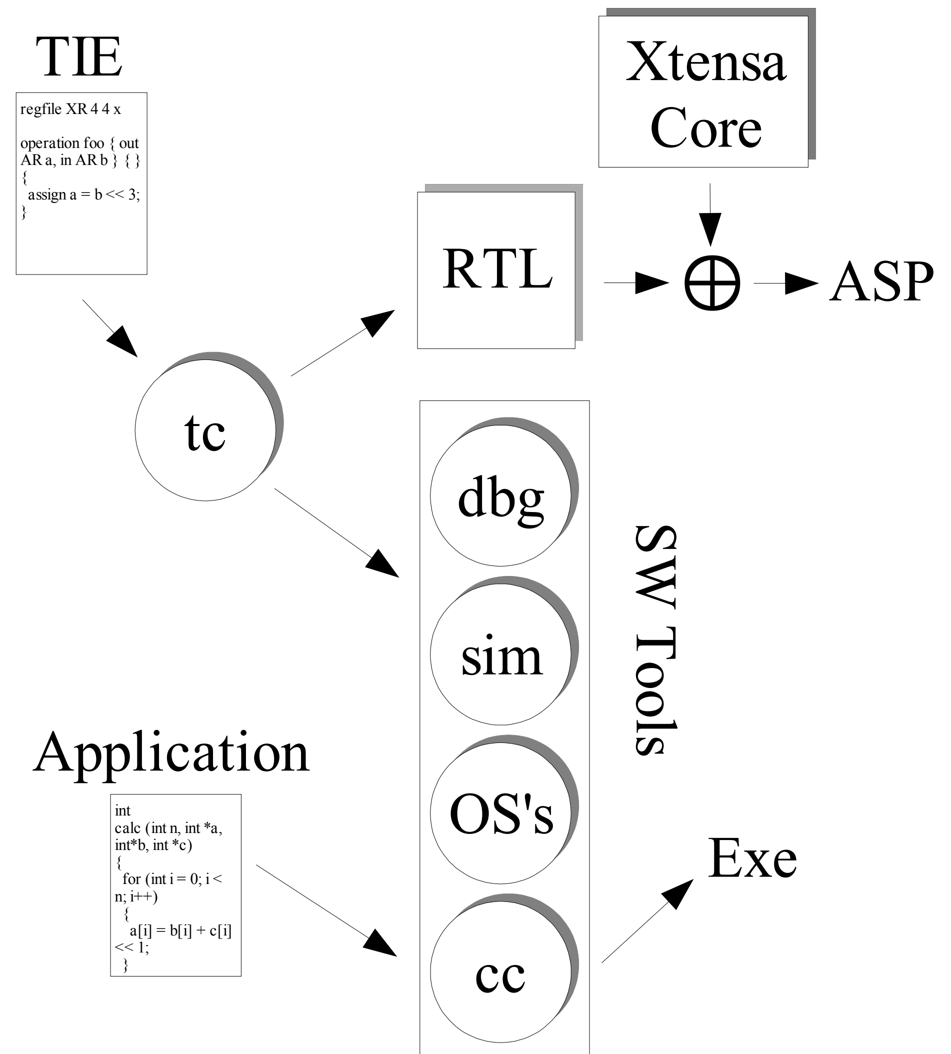
Application Specific Processors

- ASP ideal for many embedded applications
  - versus general purpose processor
    - lower power and size
    - higher performance
  - versus hard-wired logic
    - shorter time-to-market
    - programmability
- ... but users must design ASP that best trades-off various architectural techniques to balance cost and performance

Copyright ©2003 by Tensilica, Inc.

Current ASP Flow

- Tensilica Instruction Extensions (TIE) language
  - Describes VLIW formats, operations, regfiles, datatypes, etc.
  - Generates RTL for HW
  - Generates SW Tools



 Copyright ©2003 by Tensilica, Inc.

Performance Enhancing Techniques

- VLIW
  - + General (software-pipelining, scheduling)
  - − Replication of regfile ports, function units
- Vector (SIMD)
  - + High performance (automatic vectorization)
  - − Expensive vector regfiles, function units
- Fusion
  - + High performance for low cost
  - − Less general

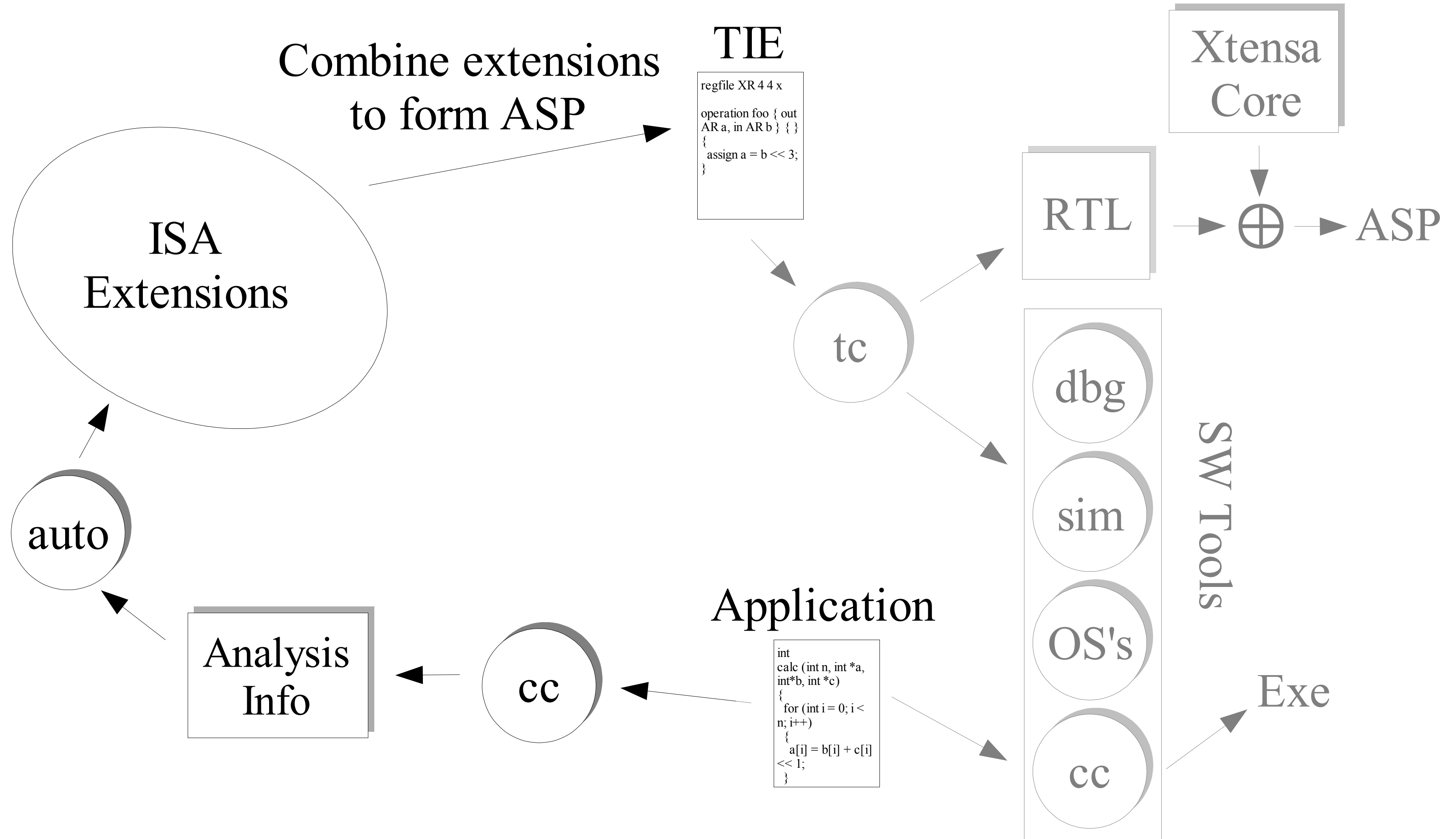
Copyright ©2003 by Tensilica, Inc.

AutoTIE

- Automatically extend base processor
  - VLIW instructions
  - Vector operations, register files
  - Fused operations
- Tradeoff performance and hardware cost
- Extensions automatically exploited by compiler

Copyright ©2003 by Tensilica, Inc.

AutoTIE Flow

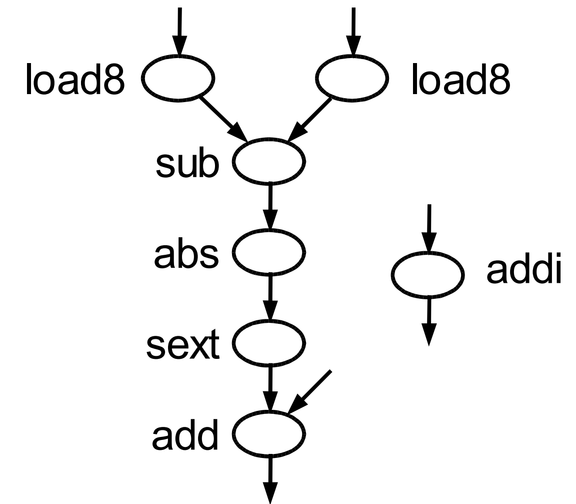


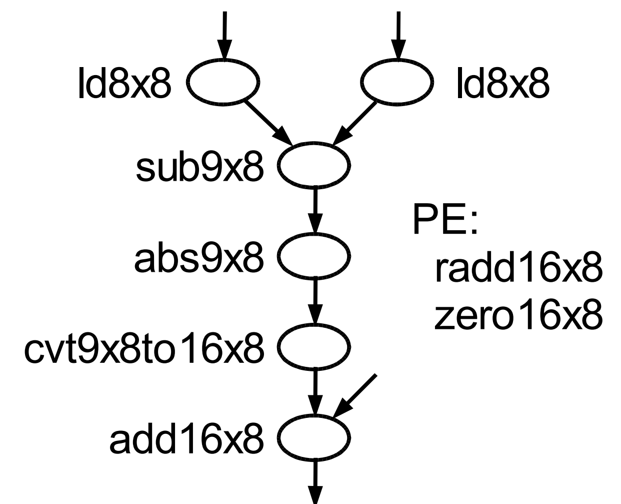
Copyright ©2003 by Tensilica, Inc.

Analysis

- For each loop...
  - Scalar DFG of loop body
  - Vector DFG of loop body, for each vector length
  - Loop prologue and epilogue operations
  - char and short guard bits

```
for (row=0; row<128; row++)
    for (col=0; col<128; col++)
        total += abs(im1[row][col] - im2[row][col]);
```





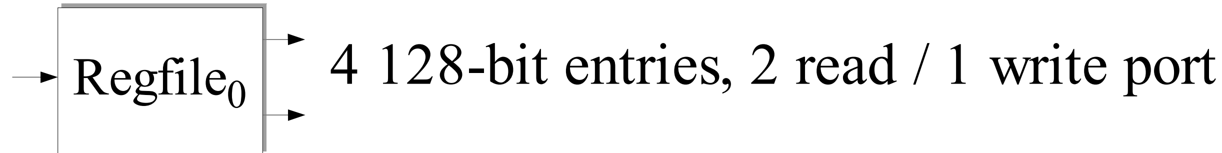
Copyright ©2003 by Tensilica, Inc.

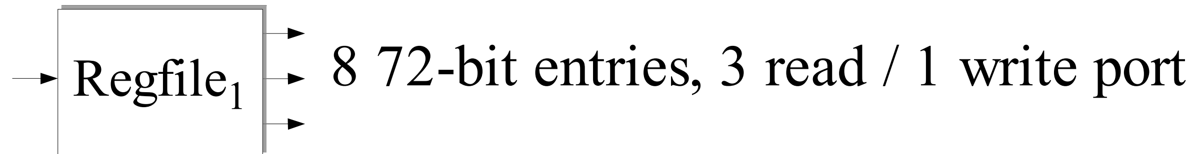
Configuration

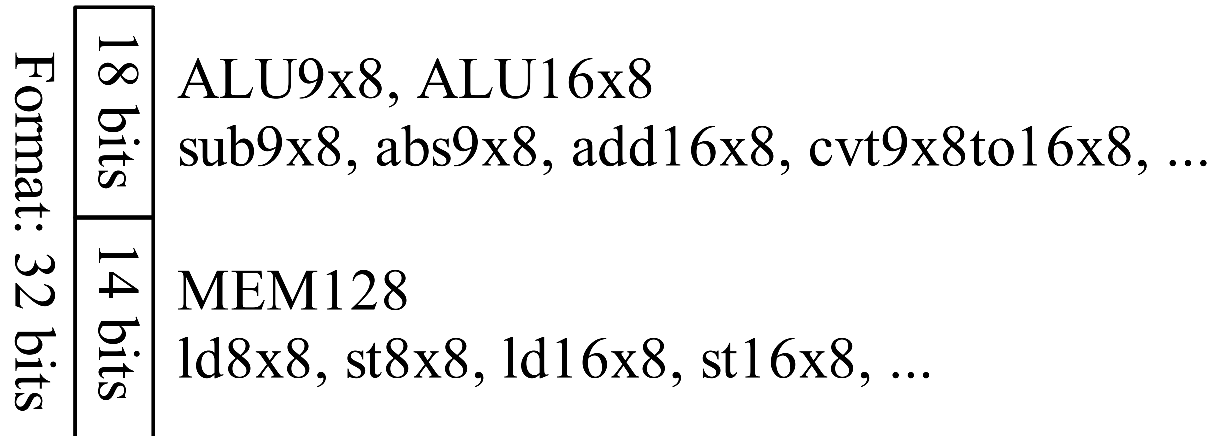
- Compact representation of set of ISA extensions
  - Vector regfiles: entries, bit-width, read/write ports
  - VLIW format: size, slots
  - Operations in each slot
  - Resources in each slot
    - Model function units
    - Characterized by *class*: ALU,MUL,SHIFT,LOGIC,MEM
    - Characterized by bit-size and vector length, e.g. ALU16x8

Copyright ©2003 by Tensilica, Inc.

Configuration Example







Copyright ©2003 by Tensilica, Inc.

Configuration Generation

- Generate configurations for each loop
  - *Seed* configurations represent minimum requirements
  - *Expand* seed configurations to create additional configurations
  - *Slot* configurations
    - Assign resources and operations to slots
    - Estimate cycle count
    - Estimate hardware cost

Copyright ©2003 by Tensilica, Inc.

Non-Fusion Seed Configuration

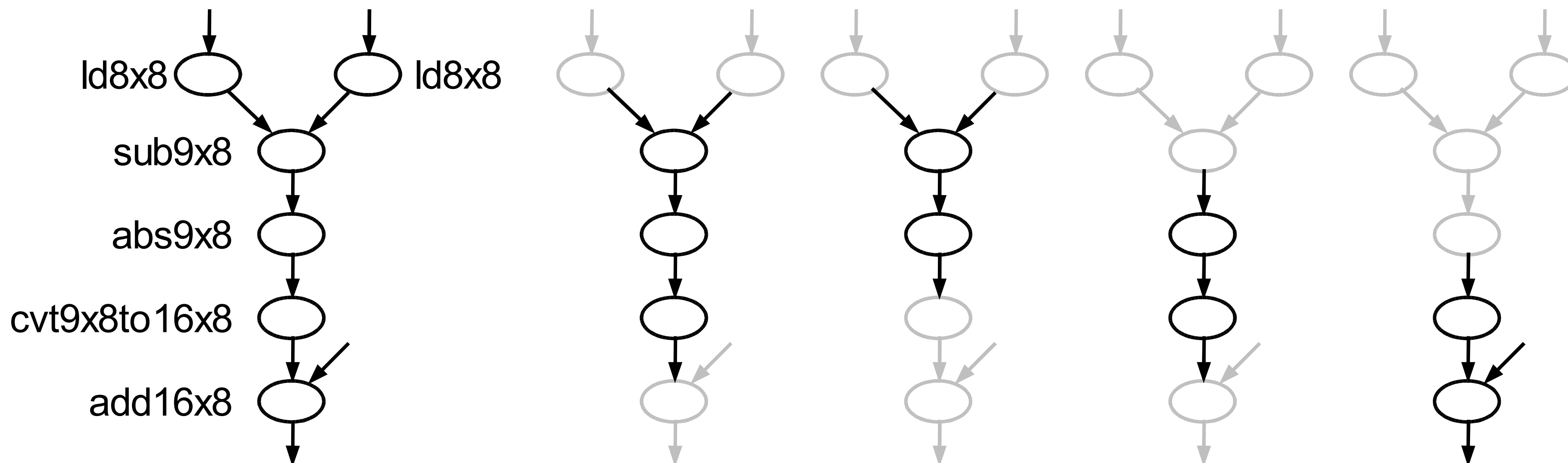
- Minimum scalar ISA required by loop

- Minimum vector ISA for each vector length

Copyright ©2003 by Tensilica, Inc.

Fusion Seed Configuration

- Collect subgraphs of loop DFG
  - Each subgraph represents possible fused operation
  - Input, output, and cycle limits restrict fused operations
- Estimate cycles after bit-wise logic optimization

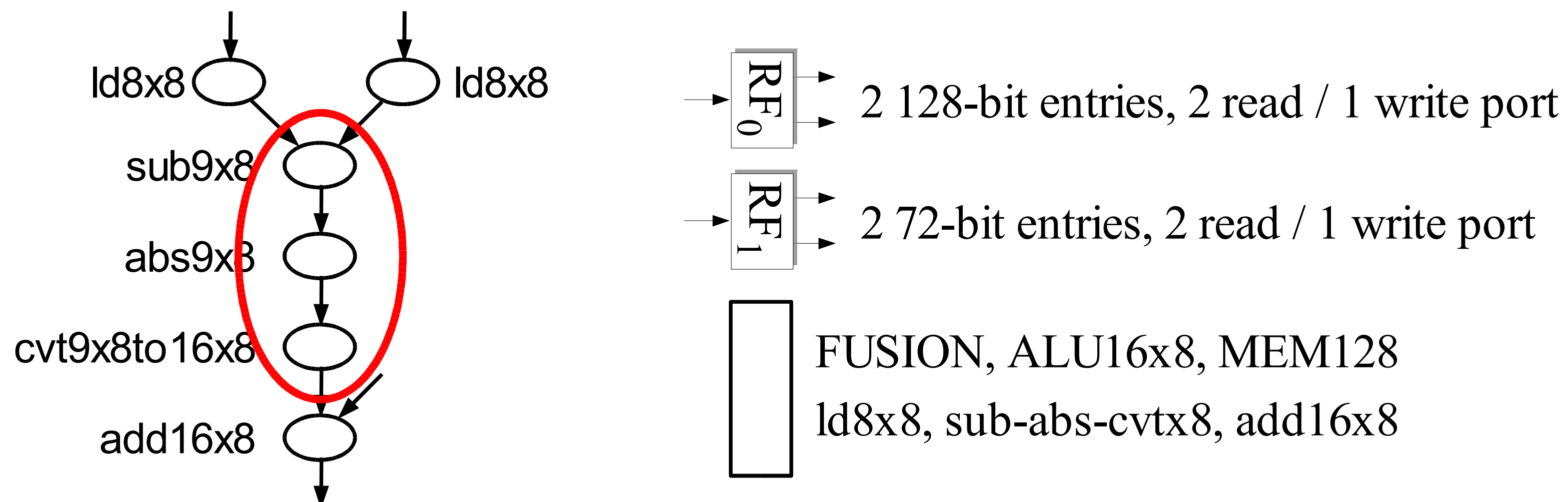


Limits: 2 inputs, 1 output, 1 cycle

Copyright ©2003 by Tensilica, Inc.

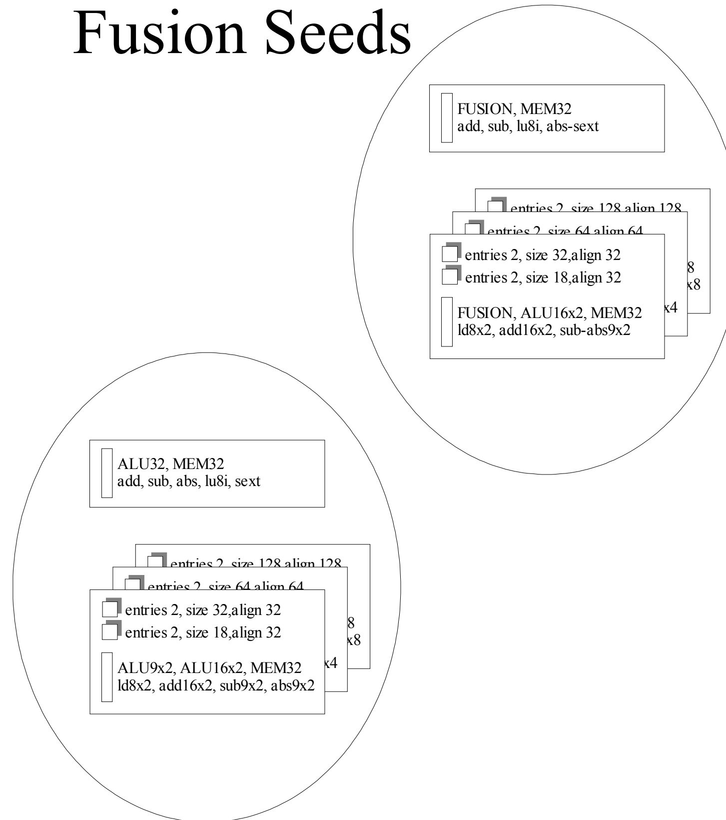
Fusion Seed Configuration (cont.)

- Select fused operations w/ greedy graph-covering
  - Select fused operation with greatest cycle saving
  - Continue selecting until no additional fusions
- Create fused seed configuration containing fused operations in place of original operations



Copyright ©2003 by Tensilica, Inc.

Generation Flow



 Copyright ©2003 by Tensilica, Inc.

Configuration Expansion

- Extra resources added to each seed configuration
  - Critical resource limits loop performance
  - Critical resource increased → new configuration
  - Expansion stops when critical resource bounded by user or processor limit (issue-width, memory ports)

$$MinII_R = \left\lceil \frac{Ops_R}{Num_R} \right\rceil$$

Copyright ©2003 by Tensilica, Inc.

Expansion Example

MEM128 = { ld8x8, ld8x8 } → Ops[MEM128] = 2
ALU9x8 = { sub9x8, abs9x8, cvt9x8to16x8 } → Ops[ALU9x8] = 3
ALU16x8 = { add16x8 } → Ops[ALU16x8] = 1

ALU9x8 (x1), ALU16x8 (x1), MEM128 (x1) — ← Seed Configuration

MinII[MEM128]	= 2 / 1	= 2 cycles
MinII[ALU9x8]	= 3 / 1	= 3 cycles
MinII[ALU16x8]	= 1 / 1	= 1 cycles
MinII[ISSUE]	= 6 / 1	= 6 cycles

ALU9x8 (x1), ALU16x8 (x1), MEM128 (x1) — → New Configuration

MinII[MEM128]	= 2 / 1	= 2 cycles
MinII[ALU9x8]	= 3 / 1	= 3 cycles
MinII[ISSUE]	= 6 / 2	= 3 cycles

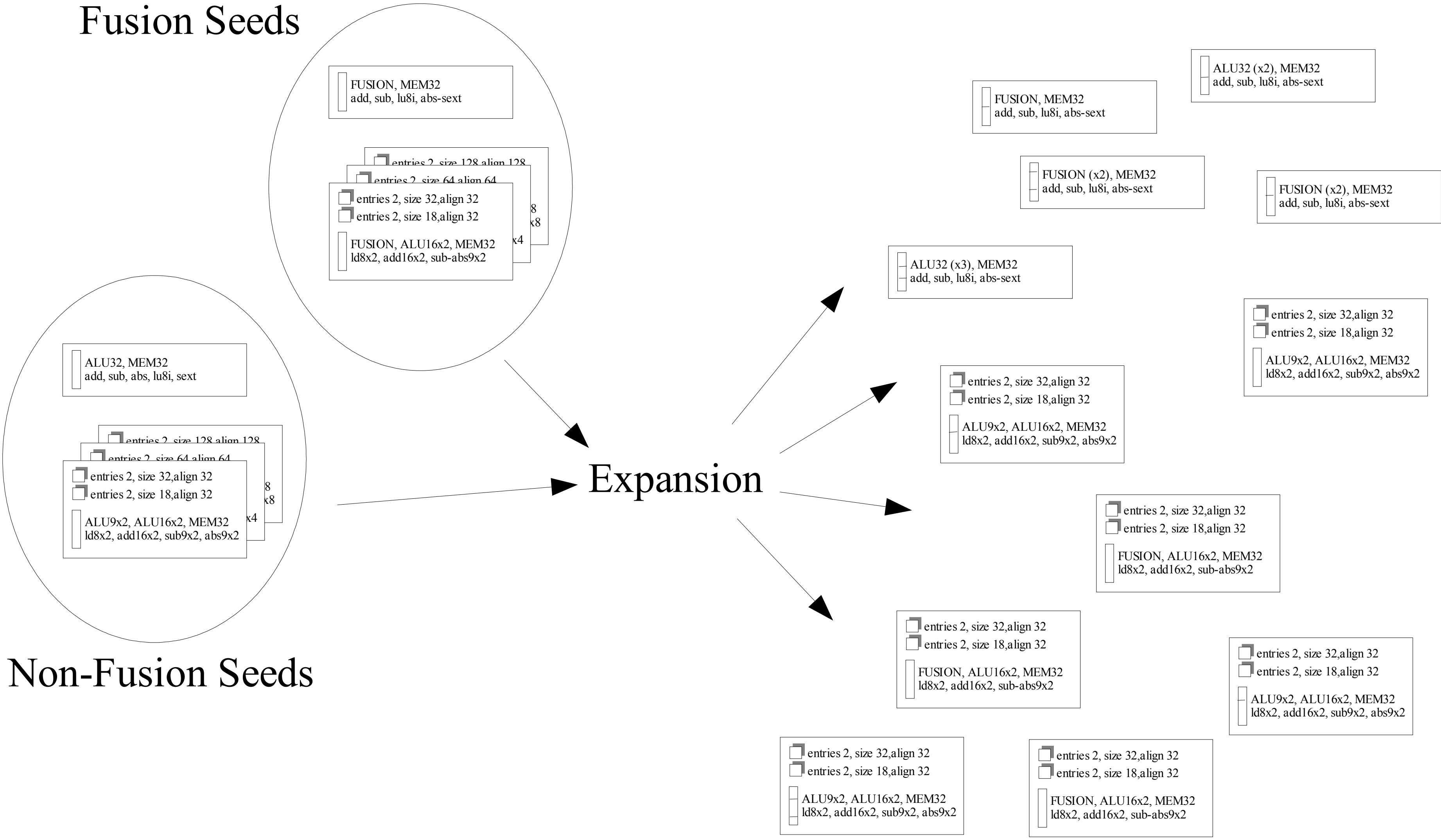
ALU9x8 (x2), ALU16x8 (x1), MEM128 (x1) — → New Configuration

MinII[MEM128]	= 2 / 1	= 2 cycles
MinII[ALU9x8]	= 3 / 2	= 2 cycles
MinII[ISSUE]	= 6 / 2	= 3 cycles

⋮

Copyright ©2003 by Tensilica, Inc.

Generation Flow



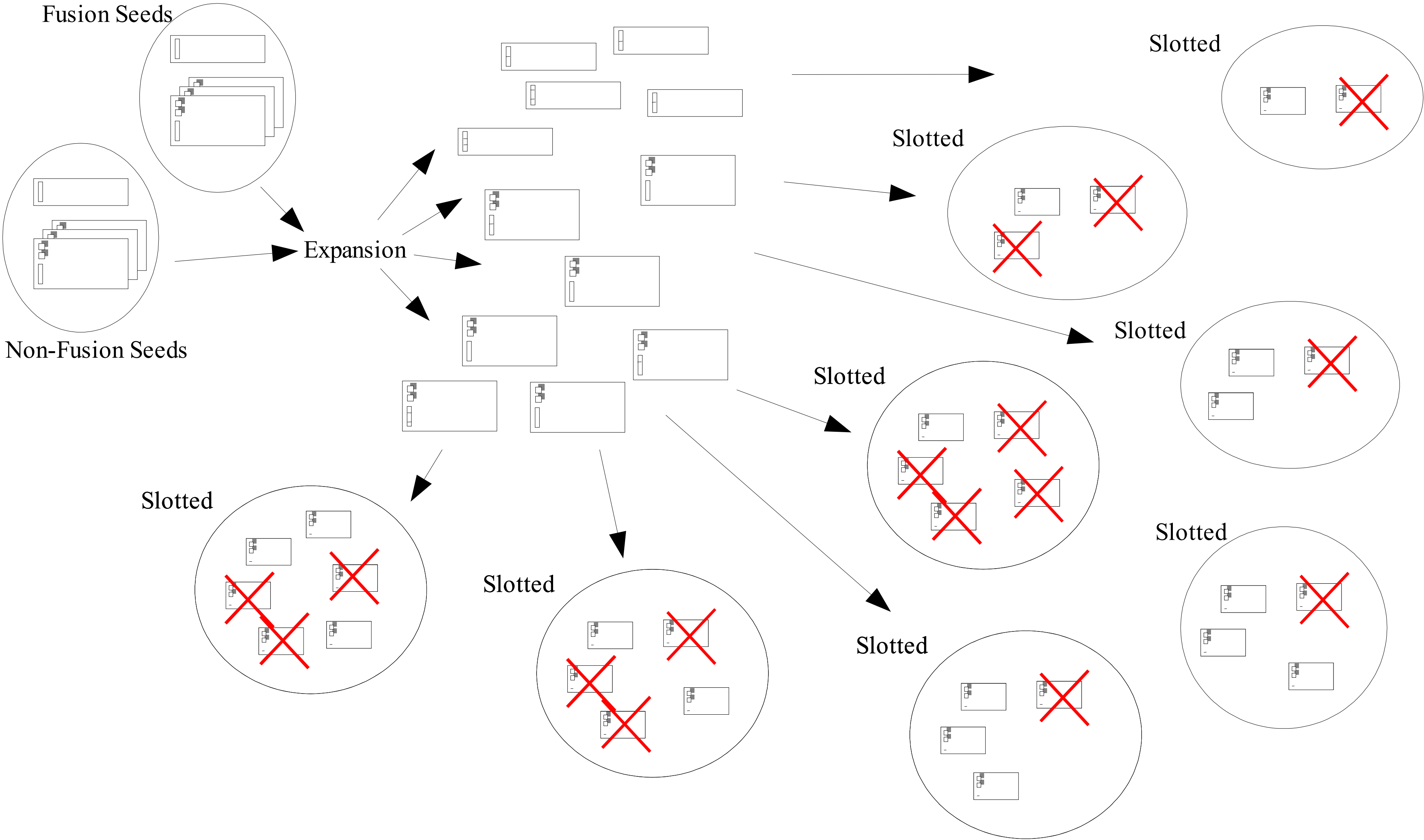
Copyright ©2003 by Tensilica, Inc.

Configuration Slotting

- Resources assigned to slots
  - Enumerate large number of possible assignments
- For each resource assignment...
  - Assign opcodes to slots containing required resource
  - Attempt to find assignment that minimizes cycles
- Create *slotted* configuration for each assignment that has minimum cycle count

Copyright ©2003 by Tensilica, Inc.

Generation Flow



Copyright ©2003 by Tensilica, Inc.

ASP Generation

- Slotted configuration represents set of ISA extensions
  - estimate performance benefit for each loop
  - estimate hardware cost
- Greedily merge configurations to form ASP
  - User sets ASP hardware cost limit
- Generate TIE describing ASP

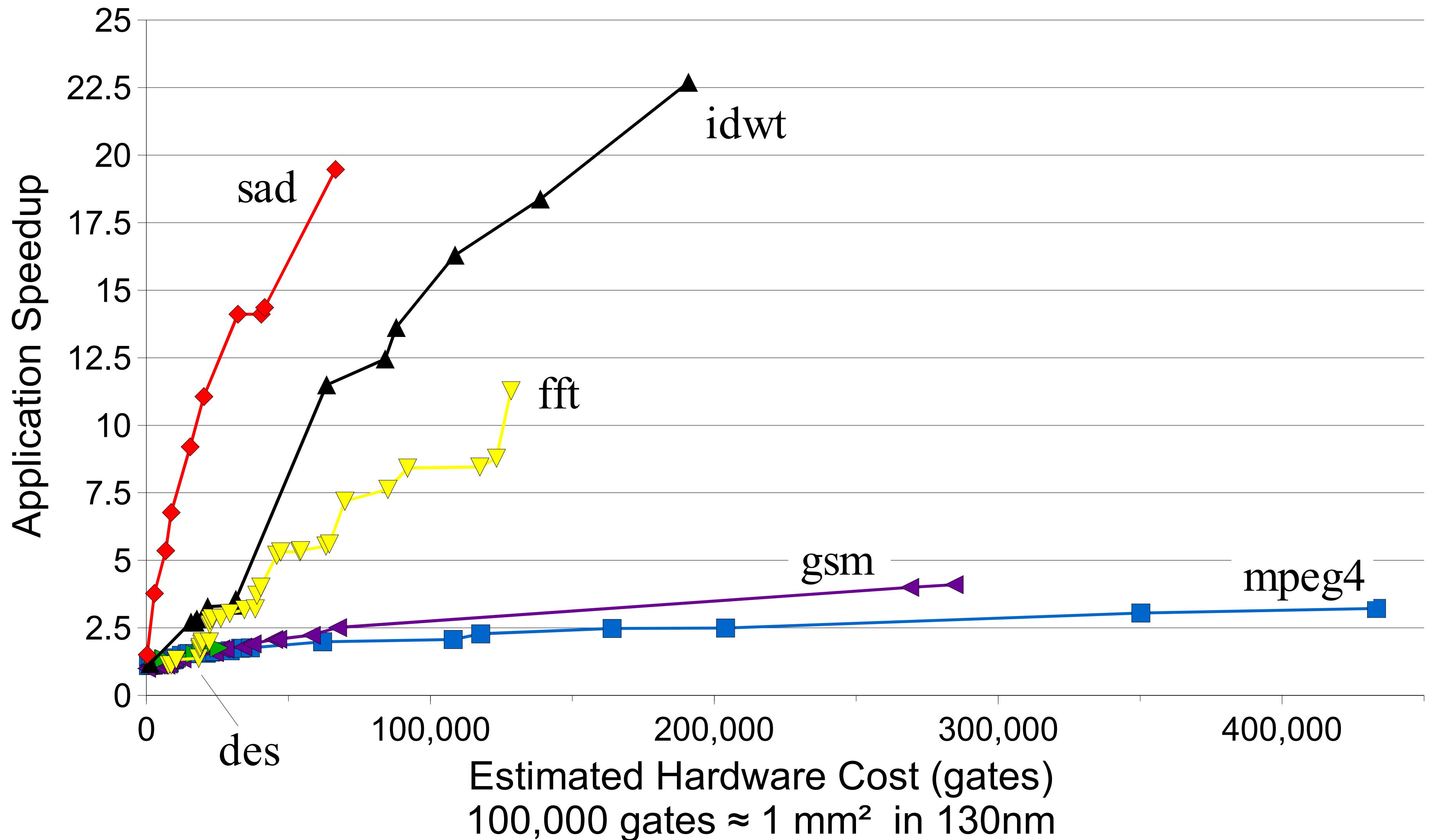
Copyright ©2003 by Tensilica, Inc.

Benchmarks

Benchmark	Description	Configurations Considered	Solution Time
sad	Sum of absolute differences	1.8K	10s
fft	Radix-4 fast Fourier transform	34K	55s
idwt	Inverse discrete wavelet transform	4.5K	21s
gsm	GSM audio encoder	489K	2h5m
mpeg4	MPEG4 encoder	2.1M	5h56m
des	DES encryption	766K	16m

Copyright ©2003 by Tensilica, Inc.

Performance / Cost Trade-off



Copyright ©2003 by Tensilica, Inc.

Conclusion

- AutoTIE effectively searches large design space
  - Enumerate large number of possible ISA extensions
  - Combine sets of ISA extensions to form ASP
  - Performance and cost estimates speed search
- Generated ASP automatically targeted
  - C/C++ compiler performs analysis and transformation

Copyright ©2003 by Tensilica, Inc.