TriMedia CPU64

PHILIPS

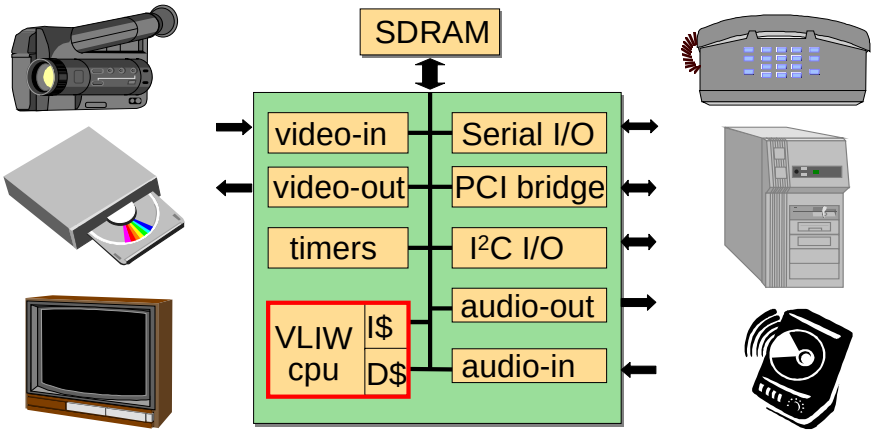
K.A. Vissers
Philips Research
Trimedia Technologies, Inc.
and
J.T.J. van Eijndhoven, G.J. Hekstra,
E.J.D. Pol, A.K. Riemens
Philips Research
Eindhoven, The Netherlands

Outline

- Introduction
- Application benchmark suite
- CPU64 architecture
- Development environment
- Design space exploration
- Conclusions

DAC System Level Design with Embedded Platforms Tutorial

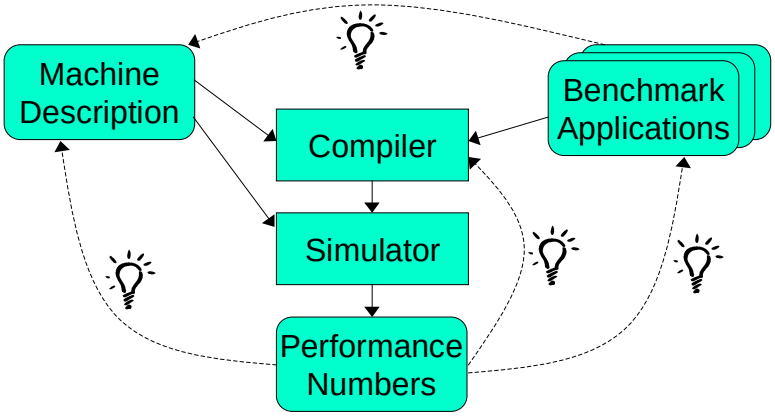
Design Problem



Initial Design Considerations

- Goal: 6-8 times TM1000 performance
- Standard ANSI-C, reuse of code
- Utilize instruction and data parallelism
- Limited complexity (embedded core)
- Compatibility through recompilation
- VLIW architecture

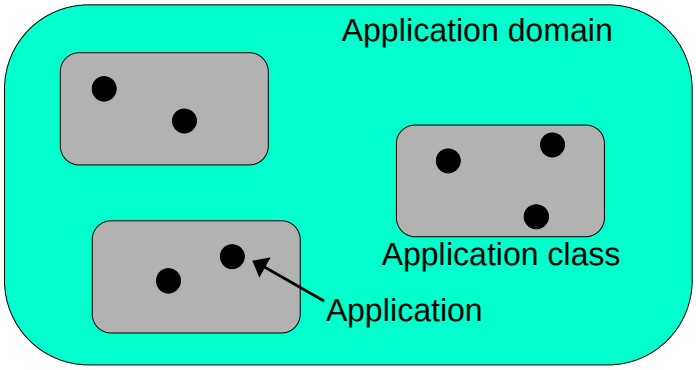
Design Approach: Y-chart



Outline

- Introduction
- Application benchmark suite
- CPU64 architecture
- Development environment
- Design space exploration
- Conclusions

Terminology



Benchmark suite: set of all applications

Benchmark Suite Characteristics

- Each application: typical for a class of applications within application domain
- The set covers a significant area of the application domain
- Each benchmark is sufficiently well tuned to the architecture to measure its performance

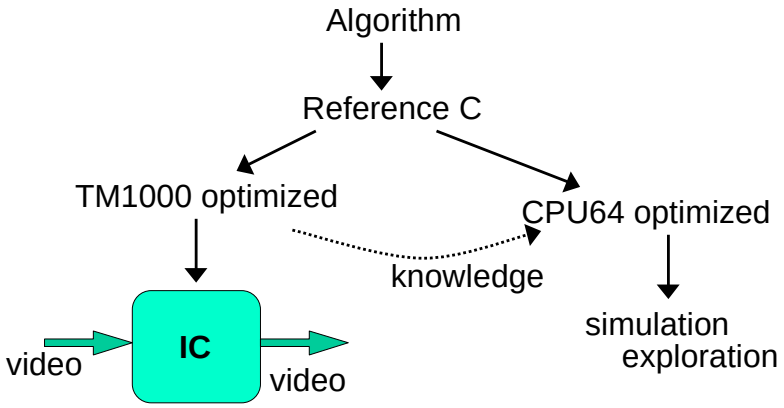
The Benchmark Suite

Data comm.	Viterbi decoding
Audio coding	AC3 decode
Video coding	MPEG2 encode DVC decode
Video proc.	Layered natural motion Dynamic noise reduction Peaking
Graphics	3D renderer library

Processing Characteristics

- Signal rates
  audio, video (at block and pixel rate)
- Basic data types
  byte (8), half-words (16), words (32), float (32)
- Data access patterns
  sample stream, bitstream, random access
- Data independent and dependent load
- Control processing and signal processing

Application Development



Initial Design Considerations

- Goal: 6-8 times TM1000 performance
- Standard ANSI-C, reuse of code
- Utilize instruction and data parallelism
- Limited complexity (embedded core)
- Compatibility through recompilation
- VLIW architecture

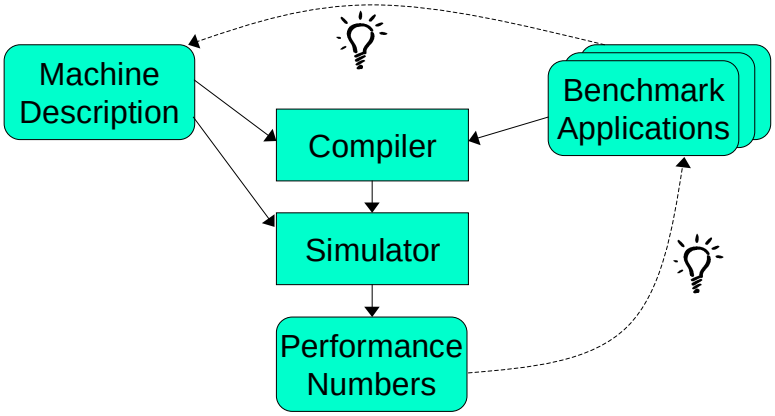
Initial Design Choices

- Double vector length: 32 → 64
- Enriched media instruction set

Instruction Set Considerations

- A machine operation must be sufficiently generic within the application domain
- Sufficiently powerful operations
- Limited number of operations
- Consistency and orthogonality

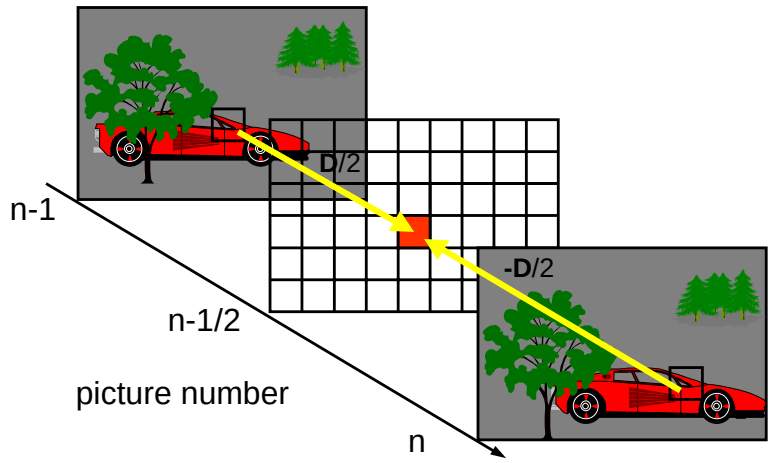
The Y-chart for applications



Application Code Example

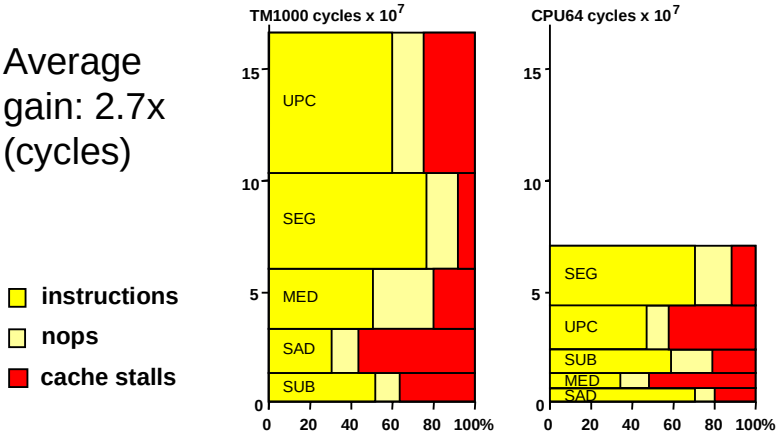
```
int calc_sad(vec64ub *prv, vec64ub *cur, int s)
{
    vec64ub left, right;
    int i; int sad = 0;
    for (i=0; i<8; i++)
    {   left = prv[i*s];   right = cur[i*s];
        sad += ub_me(left, right);
    }
    return(sad);
}
```

Natural Motion Application



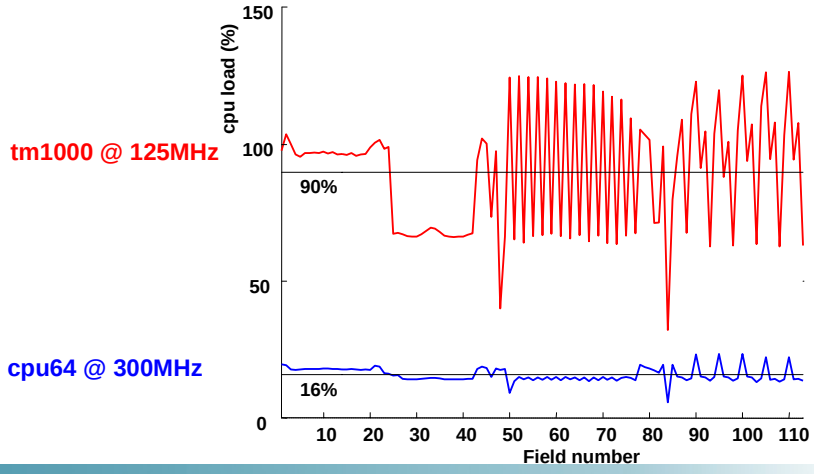
Natural Motion Results

Average gain: 2.7x (cycles)



- instructions
- nops
- cache stalls

Natural Motion Dynamic Load



Outline

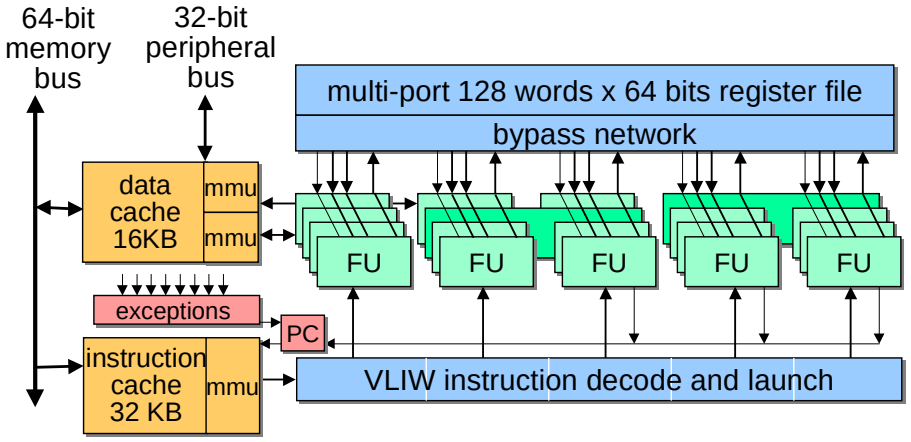
- Introduction
- Application benchmark suite
- CPU64 architecture
- Development environment
- Design space exploration
- Conclusions

DAC System Level Design with Embedded Platforms Tutorial

Architectural Speedup

- Extended SIMD: uniform 64-bit design supporting 1-, 2-, and 4-byte elements (data throughput per cycle)
- New, extensive, media instruction set (functionality per cycle)
- Improved cache control (prefetch, dismiss)

CPU64 Architecture



DAC System Level Design with Embedded Platforms Tutorial

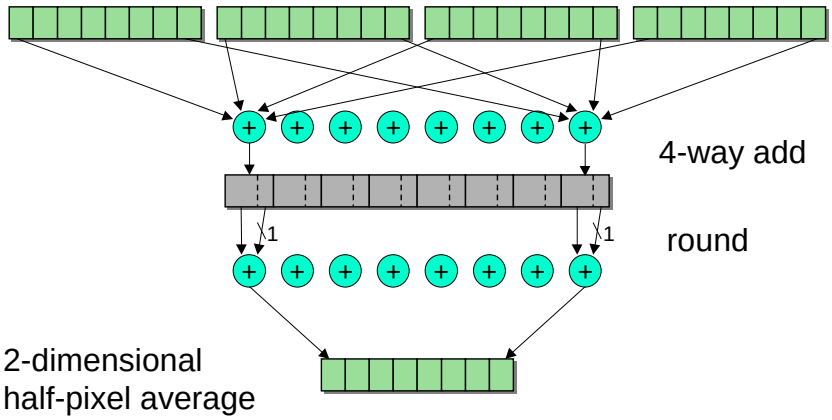
Architecture VLIW+SIMD

- Issues a 5-slot instruction every cycle
- Each slot supports a selection of FUs
- All FUs support vectorized (SIMD) data
- Double-slot FU allows powerful multi-argument, multi-result operations
- All FUs are pipelined, latency 1 to 4 (except floating point divide and sqrt)

Instruction Set

Category	Cnt	Comment
Load/store ops	39	vectorized, endian-correct
Byte shuffles	67	vector type convert
Bit shifts	48	round, fields
Multiplies	54	round, clip, wrap around
Integer arith	104	clip, wrap around
Floating point	59	scalar and vectorized
Lookup table	6	vectorized
Special ops	23	MMU, cache, special regs
Branch	10	(un)interruptible, trap

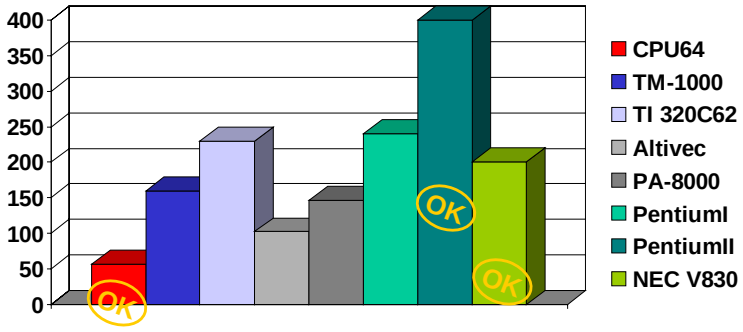
Example: 4-way average



2-dimensional
half-pixel average

2D-IDCT Instructions

Execution time in cycles, excluding data cache misses



Implementation

Now in construction at Philips Semiconductors, Sunnyvale

- 7M transistors (target)
- 300Mhz clock (target)
- 0.18µ technology
- 1.8 volt

Outline

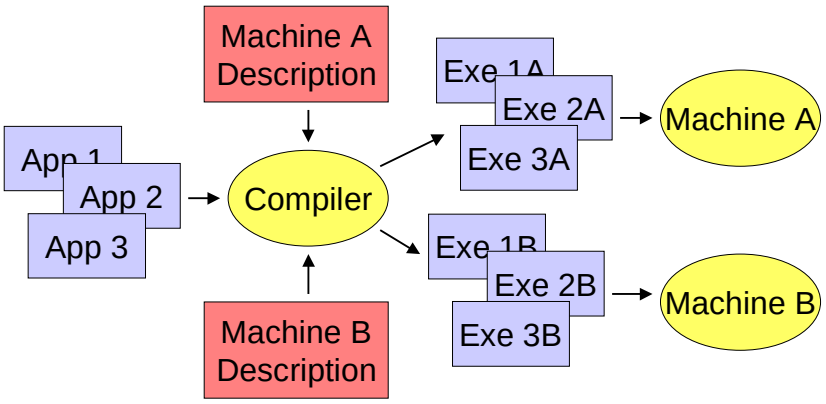
- Introduction
- Application benchmark suite
- CPU64 architecture
- Development environment
- Design space exploration
- Conclusions

Goal

Make application development
as simple as possible!

- Higher levels of parallelism
- More special-purpose hardware
- Range of CPUs will be available

Retargetable Compilation



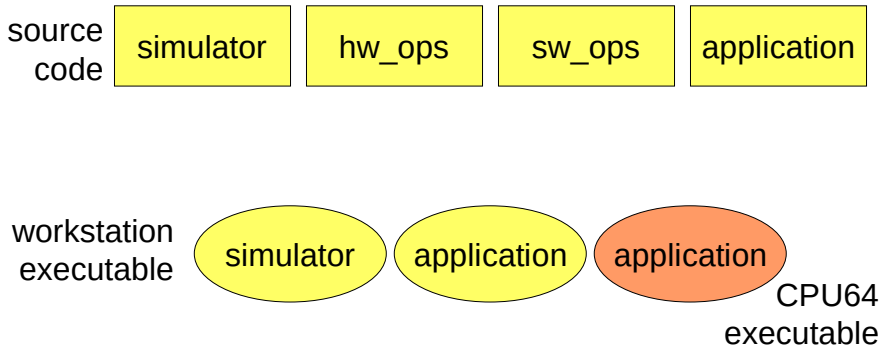
MD file contents

- MD describes instance from class of machines
- Contains information such as:
  - number & width of registers
  - number of issue slots
  - number & placement of function units
  - instruction latencies

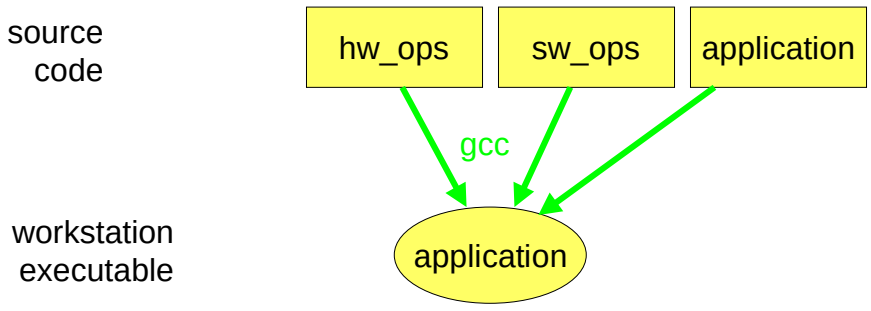
MD file fragment

```
CPU
   ISSUESLOTS 5
   FUNCTIONAL UNITS
         alu      SLOT 1 2 3 4 5    LATENCY 1
                  OPERATIONS
                           iadd(12), isub(13),
                           igtr(15), igeq(14),
         dspalu
                  SLOT 1 3          LATENCY 2
                  OPERATIONS
                           dspiadd(66), dspuadd(67)
   REGISTERS  r SIZE 32 NUMBER 128;
   READ BUSES REGISTERS r NUMBER 10;
OPERATIONS
   SIGNATURE (r:r,r->r) PURE iadd, isub,
   SIGNATURE (r:PAR,r->r) PARAMETER (0 to 127) PURE iaddi,
   SIGNATURE (r:r,r->r) LOADCLASS ld32x,
```

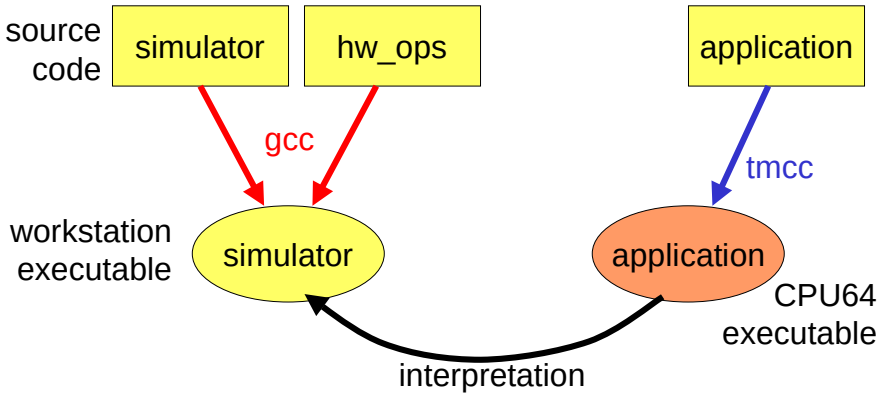
Library structure



Functional Development



Code Tuning



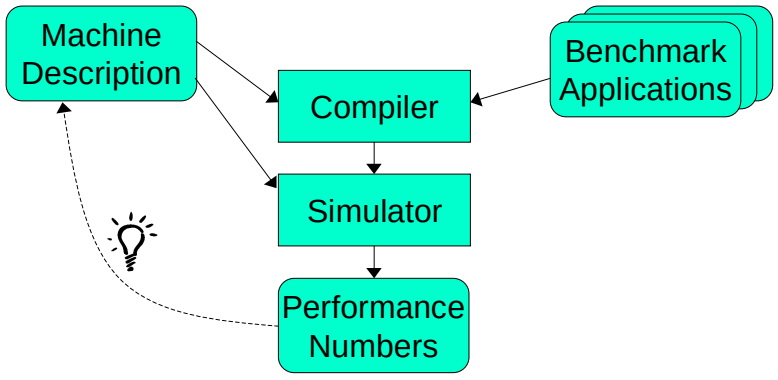
General optimization guidelines

- Design application architecture (data flow)
- Determine data formats (scalar/vector types)
- Arrange data locale (memory/cache/registers)
- Design control architecture (loop structure)
- Optimize computations (custom-ops)
- Fine tune cache behavior (prefetch/alloc)

Outline

- Introduction
- Application benchmark suite
- CPU64 architecture
- Development environment
- Design space exploration
- Conclusions

The Y-chart for DSE



DAC System Level Design with Embedded Platforms Tutorial

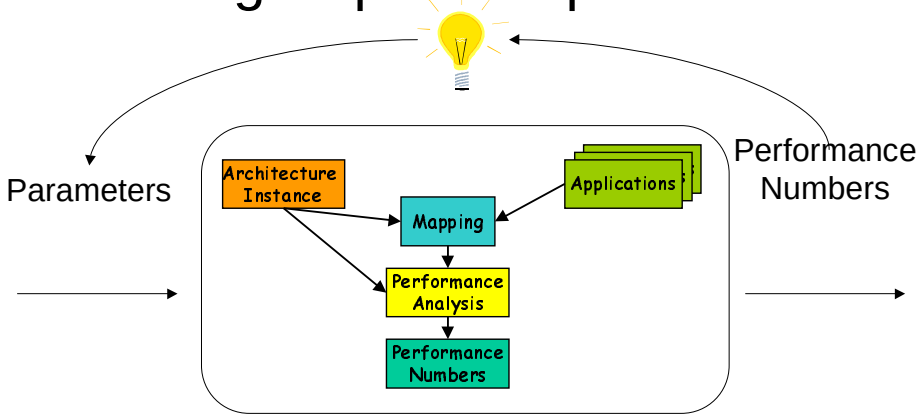
Challenge for design space exploration

- Simulation time for the benchmark for a single machine is 18 hours
- The number of design points for functional unit configuration alone: $\approx 10^{15}$
- Results in 2 000 000 000 000 years of computation time

Essential tooling

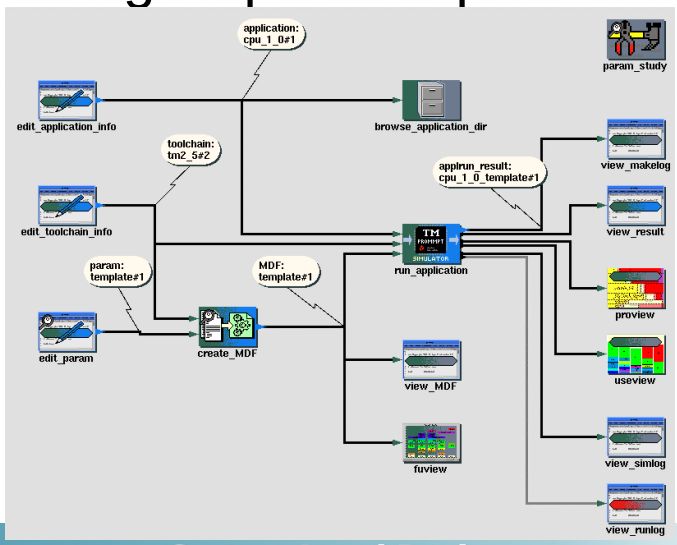
- Retargetable toolchain
  - Core compiler, scheduler, simulator
- Design and experiment management
- DSE support tools
  - Machine generation, pseudosim, analysis
- Glue software

Design Space Exploration

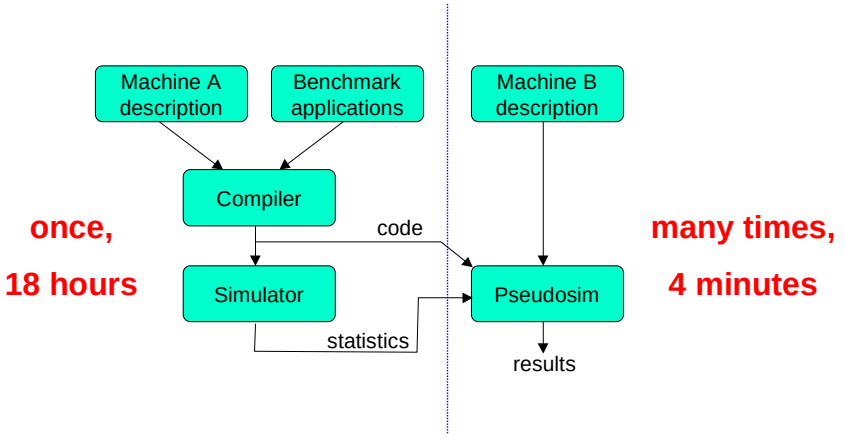


The Acquisition of Insight

Design Space Exploration



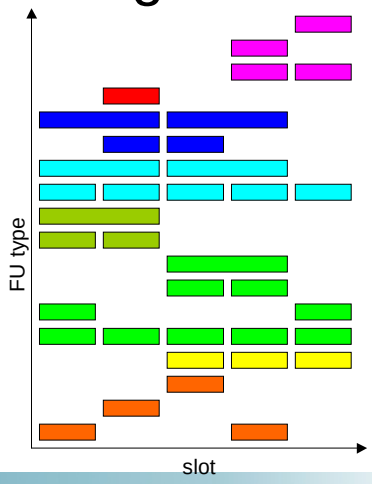
Pseudo-simulation



Function unit configuration

Problem:

- How many of each type of FU do I need?
- Where do I place them?
- How do I prevent simulating too many machine variations?

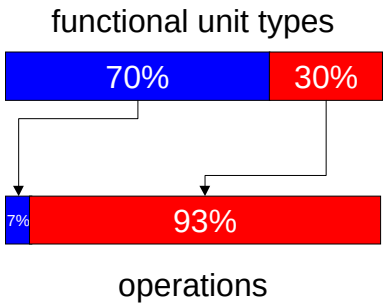


Systematic exploration

- It is not feasible to exhaustively compute all design points in the design space
- Therefore we probe, partition, and then explore the design space
- Careful set-up of experiments

Reduction of the design space

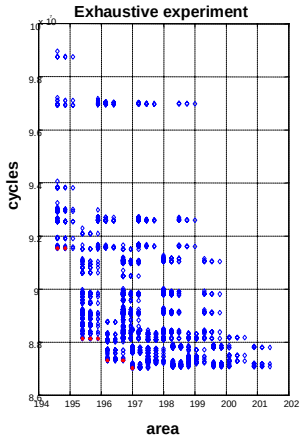
- Observation: over 93% of operations are done in 30% of FU types
- Action: partition space
  - Exhaustive exploration of important FU's
  - Greedy exploration of the remainder



DAC System Level Design with Embedded Platforms Tutorial

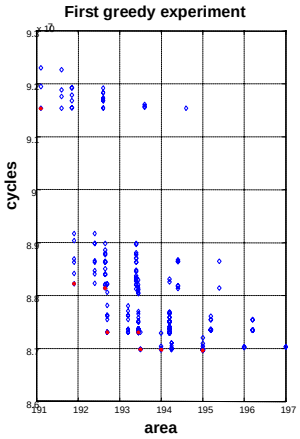
Exhaustive experiment

- Close to 3000 machine variations
- Only 4 machines survive
- Large performance variation due to placement
- Time taken = 200 hours



Greedy experiments

- Less machine variations per experiment
- More machines survive
- Less performance variation due to placement
- Close to another 3000 machine variations over all greedy experiments



Outline

- Introduction
- Application benchmark suite
- CPU64 architecture
- Development environment
- Design space exploration
- Conclusions

Conclusions

CPU64 :

- runs 6-8 times faster than TM1000
- is tuned by benchmark of media processing applications
- offers a rich set of custom operations
- supports vector processing
- hides endianness through vector load/store
- provides MMUs for robust multitasking

References

- A.K. Riemens, K.A. Vissers, R.J. Schutten, F.W. Sijstermans, G.J. Hekstra, G.D. La Hei. “Trimedia CPU64 Application Domain and Benchmark Suite”. In Proc. ICCD October 1999, Austin Texas.
- J.T.J. van Eijndhoven, F.W. Sijstermans, K.A. Vissers, E.J.D. Pol, M.J.A. Tromp, P. Struik, R.H.J. Bloks, P. van der Wolf, A.D. Pimentel, H.P.E. Vranken. “Trimedia CPU64 Architecture”. In Proc ICCD October 1999, Austin Texas.
- E. Pol, B. Aarts, J. van Eijndhoven, P. Struik, F. Sijstermans, M. Tromp, J. van de Waerdt, and P. van der Wolf, “TriMedia CPU64 Application Development Environment”, In Proc. ICCD October 1999, Austin Texas.
- G. Hekstra, G. La Hei, P. Bingley, and F. Sijstermans, “TriMedia CPU64 Design Space Exploration”, In Proc. ICCD October 1999, Austin Texas.
- Http://www.trimedia.philips.com
- Http://www.trimediatechnologies.com