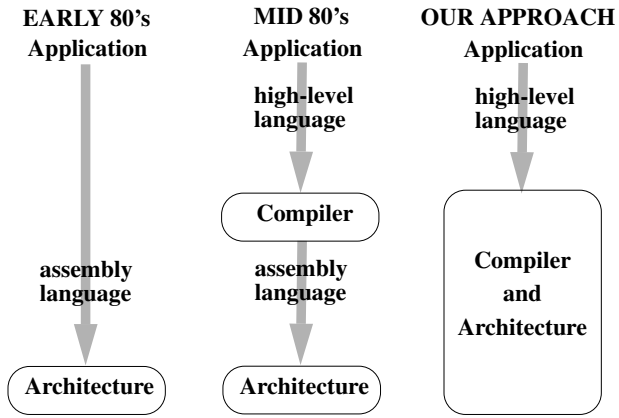
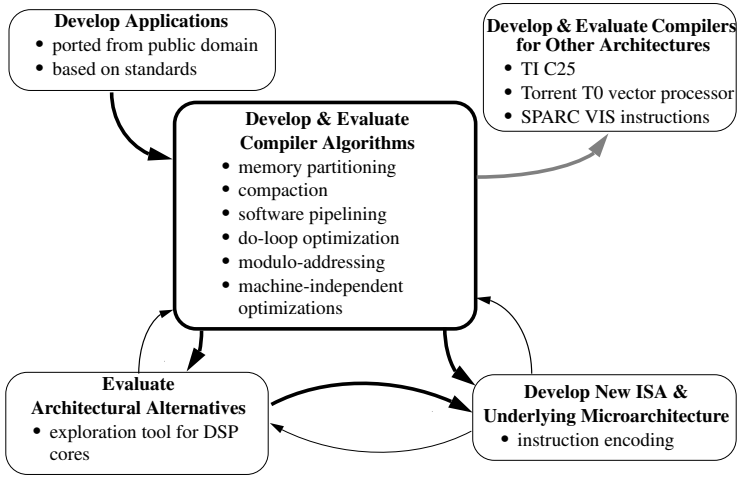
Compiler-Directed Architectural Design for Digital Signal Processors

Paul Chow and Corinna G. Lee
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada

Different Approaches for Designing DSP Architectures



Research Projects



Graduate Students: Past & Present

Student	Year	Project
Mike Takefman	1990	Improving the Performance of a DSP Microprocessor Architecture
Vijay Singh	1992	An Optimizing C Compiler for A General-Purpose DSP Architecture
Mazen Saghir	1993	Architectural and Compiler Support for DSP Applications
6 undergraduate students	sum 1994	validation of software tools; develop applications
Wen-Yen Lin	1995	An Optimizing Compiler for the TMS320C25 DSP Processor
Sanjay Pujare	1995	Machine-Independent Compiler Optimizations for the UofT DSP Architecture
Mark Stoodley	1995	Scheduling Loops with Conditional Statements
2 undergraduate students	sum 1996	finish porting SUIF compiler to tool set; develop applications
Derek Devries	1996	A Vectorizing SUIF Compiler
Mazen Saghir	1996	Exploiting Dual Data Memory Banks
Mark Stoodley	1997	Compiler Support for Multimedia Instructions
Mazen Saghir	1997	Synthesizable ISA and Microarchitecture for DSP Applications

Mike Takefman (1990):
Improving the Performance of a DSP Microprocessor Architecture

- investigated performance of deeper pipelines and a load/store architecture for DSP applications
- modified Motorola 56000 architecture
- used benchmark programs written in 56000 assembly language
- showed that proposed architecture improves performance by 1.25 to 2.11 on a per clock basis

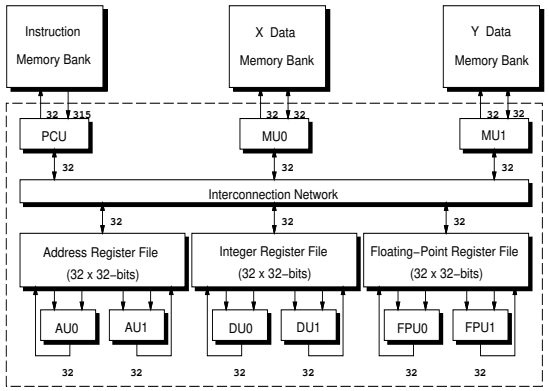
Vijay Singh (1992):
An Optimizing C Compiler for A General-Purpose DSP Architecture

- developed gcc-based compiler and VLIW architecture (support for integer arithmetic only)
- developed processor-specific optimizations: do-loop generation, memory partitioning, compaction
- wrote several kernel benchmark programs in C
- showed that compiler-generated code performs 10% slower than optimal hand-coded programs and almost 3 times faster than the best commercial system on a per clock basis

Mazen Saghir (1993):
Architectural and Compiler Support for DSP Applications

- developed instruction-set simulator for VLIW architecture
- modified compiler and architecture to include support for floating-point arithmetic
- improved processor-specific optimizations: new memory partitioning algorithm, modulo addressing
- wrote application benchmark programs in C
- showed that performance improvements differ significantly between kernels and applications

VLIW Architecture (starting point only)



load/store architecture

Harvard memory architecture:
- 1 instruction memory bank
- 2 data memory banks

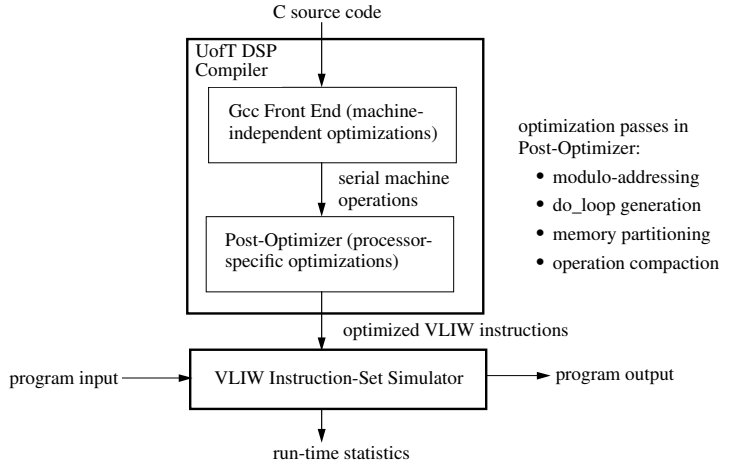
9 functional units

3 multi-ported register files

all operations execute in 1 CP

- VLIW architecture chosen for the flexibility it offers the compiler
- hardware trade-offs being considered currently

Tools & Evaluation Methodology



optimization passes in Post-Optimizer:
- modulo-addressing
- do_loop generation
- memory partitioning
- operation compaction

Benchmark Suite

kernel benchmarks: common DSP algorithms that form the core of most DSP applications
each algorithm executed with two different data sizes (small and large)

- radix-2, in-place, decimation-in-time FFT
- FIR filter
- IIR filter
- normalized lattice filter
- least-mean-squared adaptive FIR filter
- matrix multiplication

application benchmarks: from the areas of speech and image processing, telecommunications

- three versions of the ADPCM speech encoder
- LPC speech encoder
- spectral analysis using periodogram averaging
- image compression using the discrete cosine transform
- edge detection using Sobel operators
- image enhancement using histogram equalization
- V.32 modem encoder and decoder
- trellis block decoder

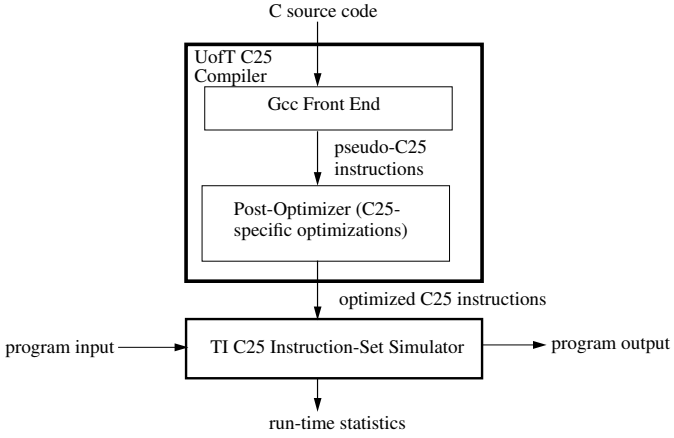
Performance of Post-Optimization Passes for Kernels and Applications

$$\text{speedup} = \frac{\text{execution time without optimization}}{\text{execution time with optimization}}$$

Optimizing Pass	Kernels	Applications
low-overhead looping	1.78	1.23
memory partitioning	1.29	1.05 → 1.08
operation compaction	3.48	2.11
Overall Speedup	4.86	2.83

applications achieve just less than 60% of the speedup obtained for kernels

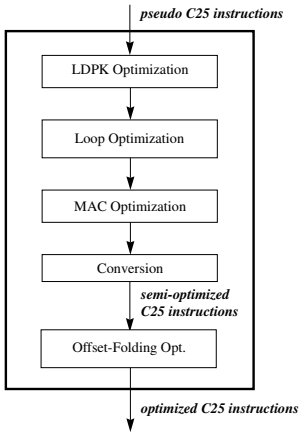
Wen-Yen Lin (1995)
An Optimizing Compiler for the TMS320C25 DSP Processor



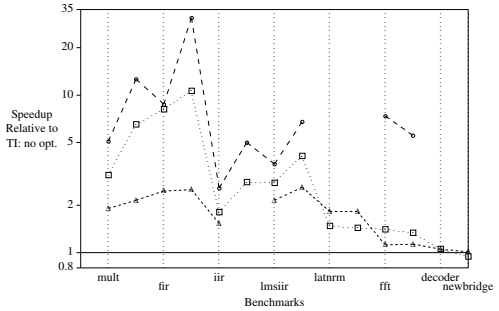
Format of Pseudo-C25 Instructions

Name	Format	Conversion Example	
		Pseudo C25 Instruction	Real C25 Instructions
Basic indirect addressing	*op* (ar*n*)	ADD (ar1)	LARP ar1 ADD *
Pre-inc/dec addressing	*op* ++(ar*n*) *op* --(ar*n*)	ADD ++(ar1)	LARP ar1 ADRK 1 ADD *
Post-inc/dec addressing	*op* (ar*n*)++ *op* (ar*n*)--	ADD (ar1)++	LARP ar1 ADD *+
Index addressing with offset	*op* (ar*n*) + *const* *op* (ar*n*) - *const*	ADD (ar1)+2	LARP ar1 ADRK 2 ADD * SBRK 2
Addition of ar0 to any auxiliary register	ADDR ar0, ar*n*	ADDR ar0, ar1	LARP ar1 MAR *0+
Arithmetic operation for auxiliary register	ADRK ar*n*, *const* SBRK ar*n*, *const*	ADRK ar1, 10	LARP ar1 ADRK 10

C25 Post-Optimizer



Performance of UofT C25-Generated Code

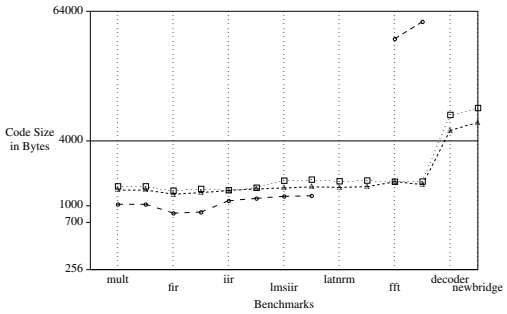


○ Hand-written C25 assembly code

□ UofT: LDPK+Loop+Offset+MAC

Δ TI: optimization level 2

Size of UofT C25-Generated Code

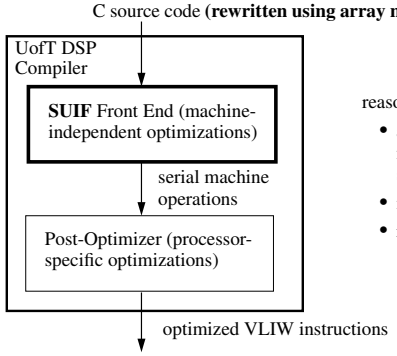


○ Handwritten C25 Code

□ UofT Compiler

Δ TI Compiler

Sanjay Pujare (1995)
Machine-Independent Compiler Optimizations



reasons for using SUIF:

- able to write C code in a more natural style without sacrificing performance
- more powerful alias analysis
- recently became available

Using DSP-Style Programming for Performance

C-style

```
for (i = 0; i < N; i++) {
    b[i] = a[i];
}
```

✗ cannot translate into hardware do-loop instruction because:
- loop-index variable `i` is used inside the loop

DSP-style

```
pa = a; pb = b;
for (i = 0; i < N; i++) {
    *pb++ = *pa++;
}
```

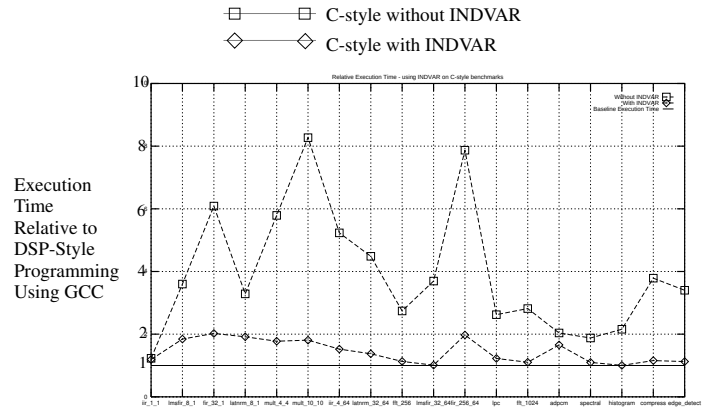
✓ can translate into hardware do-loop instruction because:
- no reference to loop-index variable by introducing extra variables
- use of pointers to access array elements

✗ less readable

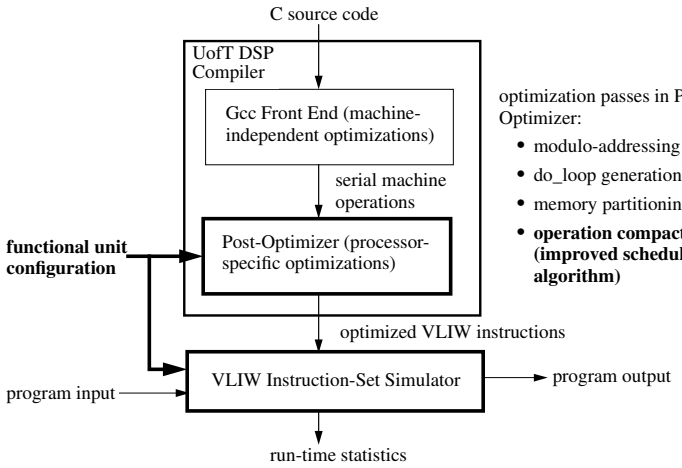
✗ low-level programming

How much performance could be lost when using C-style programming?

Performance of Machine-Independent Optimizations (INDVAR)



Mark Stoodley (1995): Scheduling Loops with Conditional Statements



optimization passes in Post-Optimizer:
- modulo-addressing
- do_loop generation
- memory partitioning
- **operation compaction (improved scheduling algorithm)**

All Paths Pipelining (APP) Scheduler

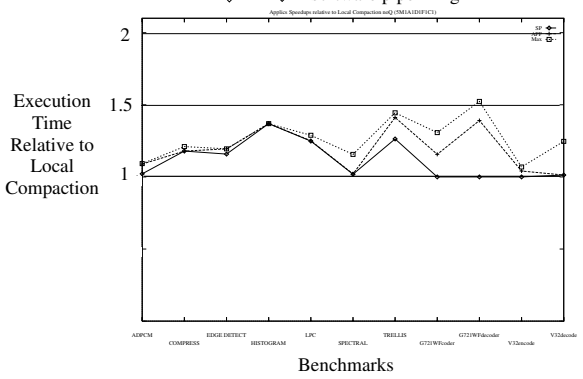
- motivation for APP scheduler:
  - many DSP algorithms are dominated by loops
  - standard technique for scheduling loops is to use software pipelining (SWP)
  - SWP is well-established for straight-line code but not for loops with conditionals
  - SWP originally implemented at source-code level
- APP scheduler generates near-optimal schedule for runs of iteration paths and efficient transition code between paths
- APP scheduler does not require any special hardware requirements, any profiling information, nor any program constructs specific to DSP applications

Evaluation of Architectural Alternatives

- scheduler and instruction-set simulator augmented to handle any number of integer, floating-point, and address units
- used to determine “best” configuration for given number of functional units

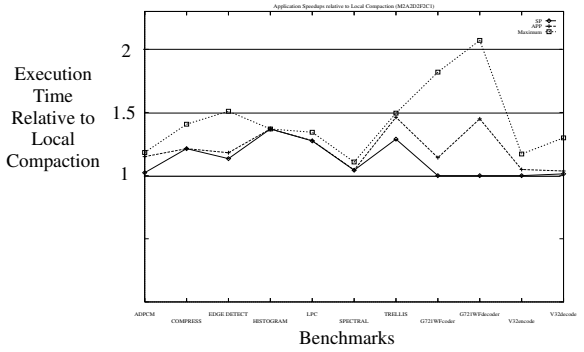
Performance of All Paths Pipelining for 5 Functional Units

theoretical maximum
All Paths Pipelining
software pipelining



Performance of All Paths Pipelining for 9 Functional Units

theoretical maximum
All Paths Pipelining
software pipelining



Paul Hellyar & Munish Sharma (Summer 1996)

- architectural analysis of MPEG-2 audio decoders
- complete integration of SUIF front-end into tool set
- include support for executing system calls in simulator

➥ make tool set robust enough to compile and simulate wider range of programs

Derek Devries (1996): A Vectorizing SUIF Compiler

- modify SUIF compiler to identify vectorizable loops
- motivation:
  - vector architecture is an inexpensive technique to exploit abundance of parallelism
  - corroborated by introduction of multimedia extension instructions in Sun UltraSPARC, HP Precision, and Intel Pentium Pro
- target architectures:
  - current target is Torrent T0, a vector extension of the MIPS architecture, developed at UC Berkeley for speech recognition
  - future target will be VIS instructions of the Sun UltraSPARC
- more details in later presentation