All Paths Pipelining:
An Algorithm for Scheduling Loops with IFs

Mark G. Stoodley and Corinna G. Lee

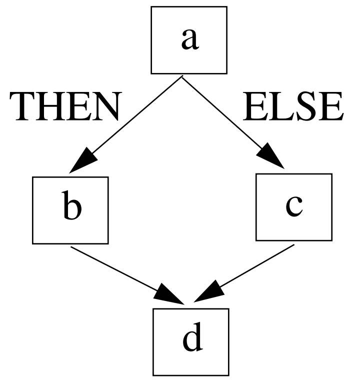
Department of Electrical and Computer Engineering

University of Toronto

Toronto, Ontario, Canada

Challenges of Scheduling Loops with IFs

- a loop that iterates N times has 2^N possible execution paths



one iteration path, T=abd
another iteration path, E=acd

possible execution paths:

T E T E T E T E T E T E T E . . . . .
T T T T T E E E E T T T T E . . . . .
E E E T T T T T T T E E E T . . . . .

- optimal execution time for runs of T's can be different from optimal execution time for runs of E's

Scheduling Loops with IFs

- ideal goal: find near-optimal schedule for all possible execution paths
- previous attempts:
  - Enhanced Percolation Scheduling [Ebcioglu, 1987]
  - Hierarchical Reduction [Lam, 1989]
  - Predicated Operations [Park & Schlansker, 1991]
  - Enhanced Modulo Scheduling [Warter, Bockhaus, *et al.*, 1992]

  all execution paths are sub-optimal or
  one execution path is near-optimal and all others are sub-optimal

Using All Paths Pipelining to Schedule Loops with IFs

- observation: many execution paths have runs of T’s and E’s
  ➥make long runs of T’s or E’s execute quickly
- execution paths with long runs of T’s or E’s are near-optimal and all others are possibly sub-optimal
- rather than paying a penalty for executing the “wrong” path, instead possibly pay a penalty for switching iteration paths
- algorithm extends to loops with multiple IFs

✔ no special hardware requirements

✔ does not rely on profiling information

Overview of All Paths Pipelining Algorithm

STEP 1: Create a trace for each iteration path in the loop. Add dependence edges to prevent global and speculative code motion.

STEP 2: Software pipeline each trace in isolation, possibly achieving different initiation intervals for each trace.

STEP 3: Merge the traces into one schedule by adding introductory and intermediate instructions that allow any execution path to be executed.

Example Loop with a Conditional Statement

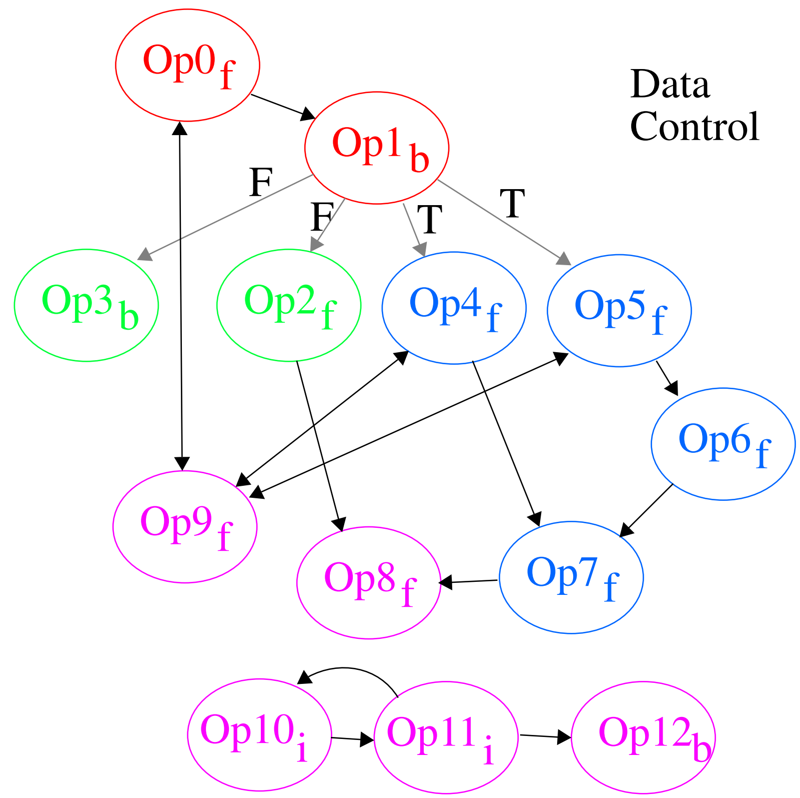
Example C Loop

```
for (i=0; i<N; i++) {
  if (x < threshold)
    z = threshold;
  else
    z = (x+1.0)*(x/2+3.0);
  sum += z;
  x = x*x;
}
```

	Assembly Code	Op Name
TEST	L7: fslt.s x,threshold,f1	Op0
	beqz.s f1,L5	Op1
THEN	fmov.s threshold,f10	Op2
	jmp L6	Op3
ELSE	L5: fadd.s x,f9,f2	Op4
	fdiv.s x,f8,f3	Op5
	fadd.s f3,f7,f11	Op6
	fmult.s f2,f11,f10	Op7
	L6: fadd.s sum,f10,sum	Op8
	fmult.s x,x,x	Op9
LOOP CONTROL	add.d i,d4,i	Op10
	slt.d i,N,d1	Op11
	bnez.d d1,L7	Op12

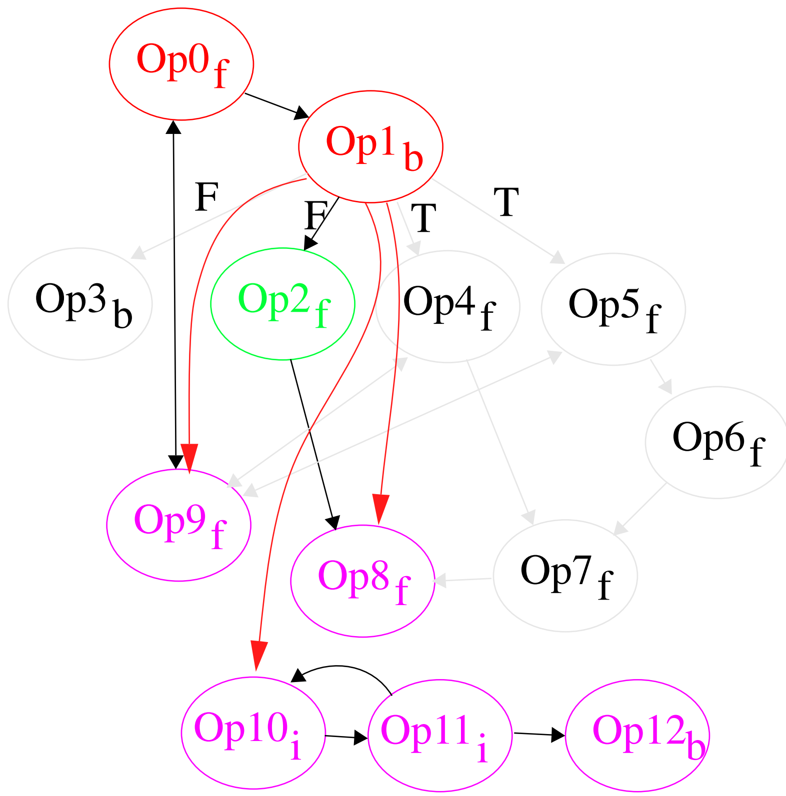
Program Dependence Graph

	Assembly Code	Op Name
BB0	L7: fslt.s x,threshold,f1	Op0
	beqz.s f1,L5	Op1
BB1	fmov.s threshold,f10	Op2
	jmp L6	Op3
BB2	L5: fadd.s x,f9,f2	Op4
	fdiv.s x,f8,f3	Op5
	fadd.s f3,f7,f11	Op6
	fmult.s f2,f11,f10	Op7
BB3	L6: fadd.s sum,f10,sum	Op8
	fmult.s x,x,x	Op9
	add.d i,d4,i	Op10
	slt.d i,N,d1	Op11
	bnez.d d1,L7	Op12



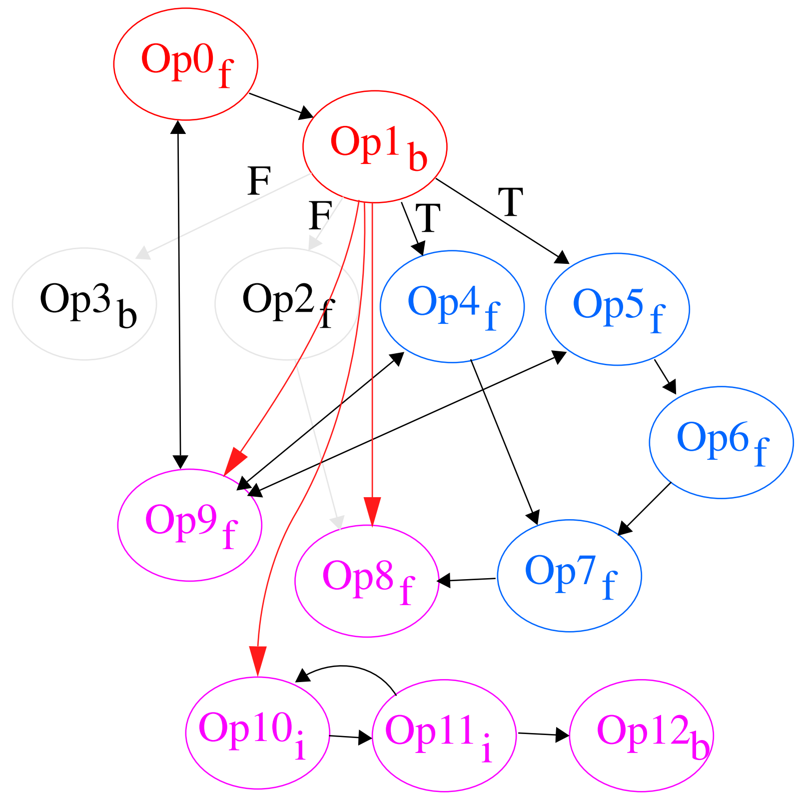
STEP 1: Extract Trace PDGs

THEN (False) Trace



minimum II >= 4 cycles

ELSE (True) Trace



minimum II >= 7 cycles

STEP 2: Software Pipeline Each Trace

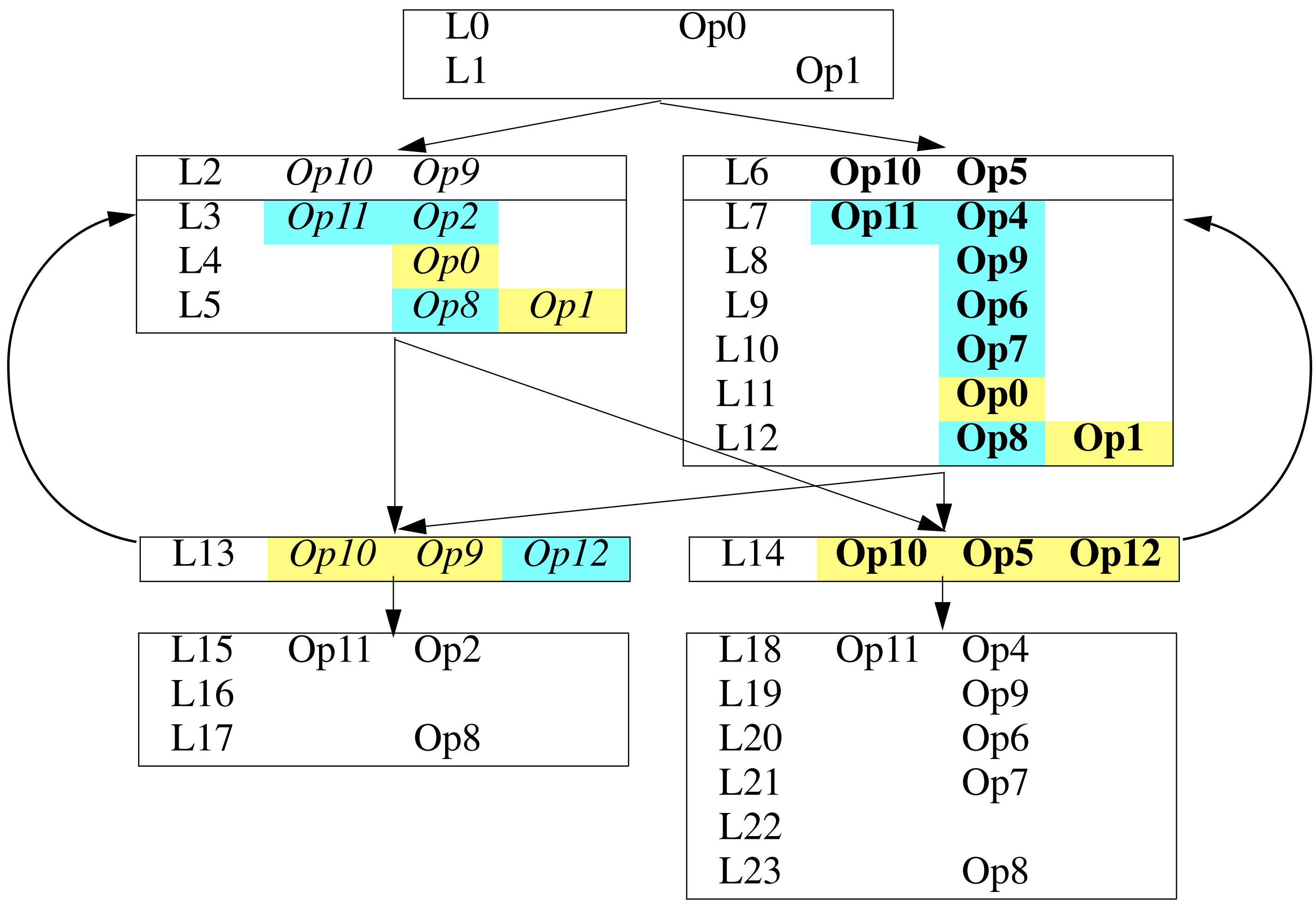
label	int	FP	branch
L3	*Op11*	*Op2*	
L4		*Op0*	
L5		*Op8*	*Op1*

L13	*Op10*	*Op9*	*Op12*

label	int	FP	branch
L7	**Op11**	**Op4**	
L8		**Op9**	
L9		**Op6**	
L10		**Op7**	
L11		**Op0**	
L12		**Op8**	**Op1**

L14	**Op10**	**Op5**	**Op12**

STEP 3: Combine Traces



Execution Time of APP-Scheduled Loop

- execution time depends on p_{THEN} = probability THEN path is executed
- number of cycles to execute 1 iteration:

$$T_I = 4p_{THEN} + (1 - p_{THEN})7 = 7 - 3p_{THEN} \text{ cycles}$$

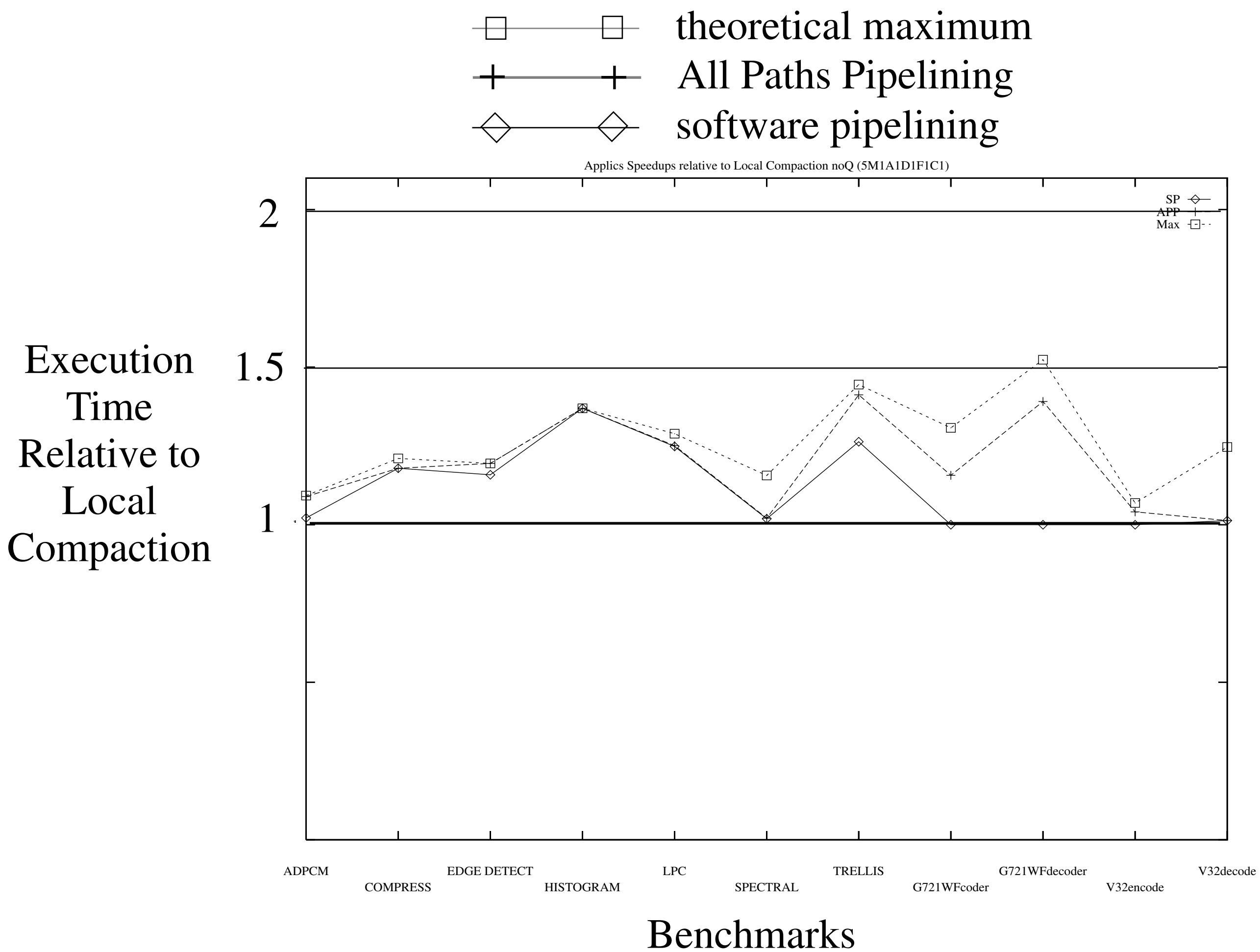
NOTE: no additional cycles needed to switch iteration paths!

- range of execution times:

$$4 \text{ cycles} \le T_I \le 7 \text{ cycles}$$

- ideal goal achieved for this loop
  BUT not possible in general

Performance Results for 5 Functional Units



Performance Results for 9 Functional Units