Diverge-Merge Processor(DMP):
Dynamic Predicated Execution of Complex Control-Flow Graphs Based on Frequently Executed Paths

Yousun Ko

Contents

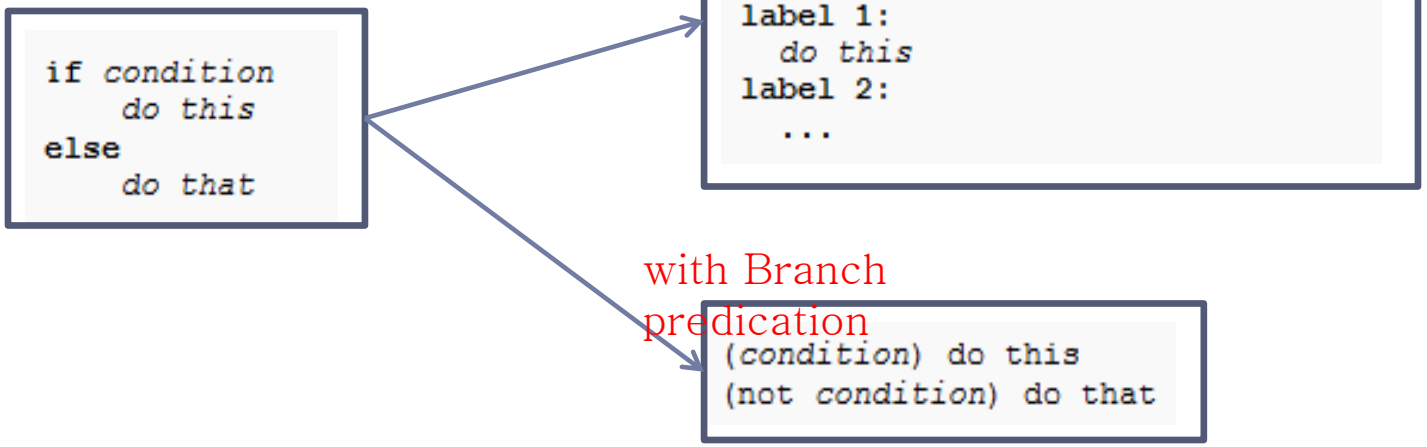
What Is This Paper About?

- Propose
  - new processor architecture for handling hard-to-predict branches,

    the diverge-merge processor (DMP)

- Goal
  - Eliminate branch mispredictions due to hard-to-predict dynamic branches by dynamically predicting them without requiring instruction set architecture(ISA) for predicate registers and predicated instructions.

Branch Predication



Example adapted from wikipedia

Branch Predication

- Problems/limitations
  - Requires significant support from ISA
  - Statically predicated code incurs the performance overhead
    - Have to fetch instructions from both paths of an if-converted branch
    - Dependent data have to be stall until the predicate value is resolved
  - Large subset of control-flow graphs are usually not converted to predicated code

Diverge-Merge Processor

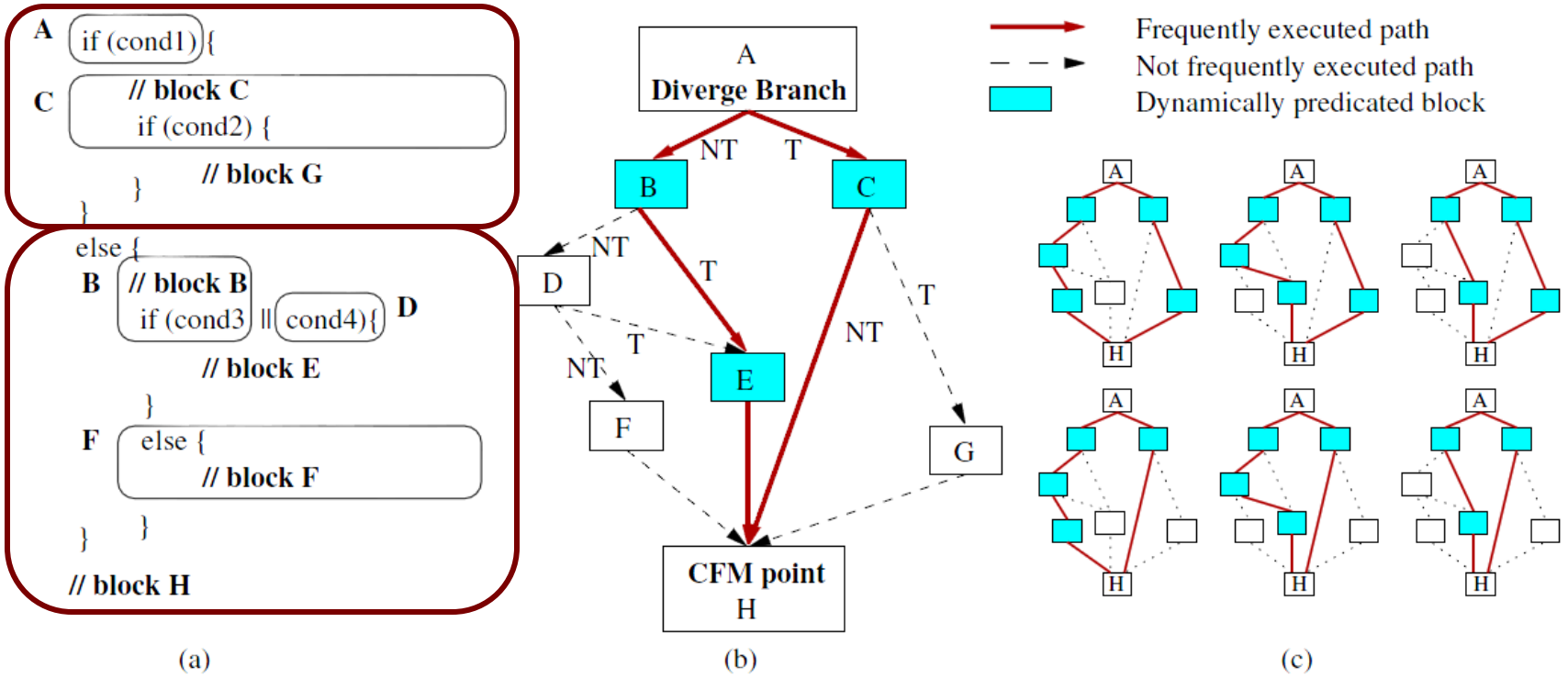
- Dynamically predicates simple and complex CFG
- Not require predicate registers and predicated instructions
  - Matter of course, no need of large hardware/energy cost
- Only predicates the frequently executed paths
  - Reduce overhead due to predication
  - Can predicate a large set of CFG
- Based on Dynamic-hammock-predication

Hammock Predication

- Hammock

Example adapted from Yard Envy

Hammock Predication



(a) source code (b) CFG (3) Possible paths

Basic Idea

- Processor fetches a diverge branch
- Estimates whether or not the branch is hard to predict
  - uses branch confidence estimator to estimate
  - This estimator prefers frequently executed basic blocks in CFG. Information is from compile-time profiline.
- Enters *Dynamic predication mode (dpred-mode)* if the diverge branch has low confidence

Basic Idea

Instruction Fetch Support

- Processor Fetches instructions from both directions of a diverge branch
- If reaches CFM, exits dpred-mode

Select-μops

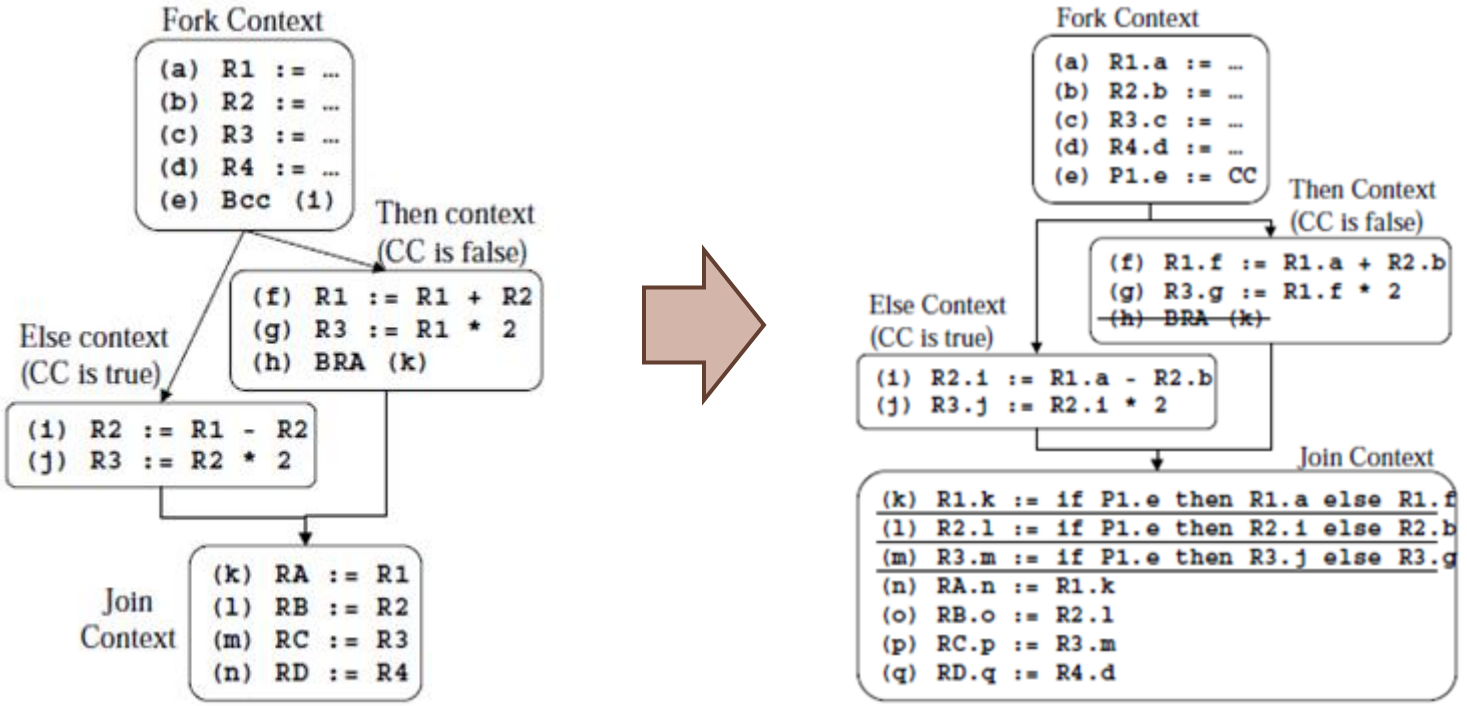
- Not waiting for resolution of right path to go, make data dependent
  - Similar to Φ-function SSA

Loop Branches

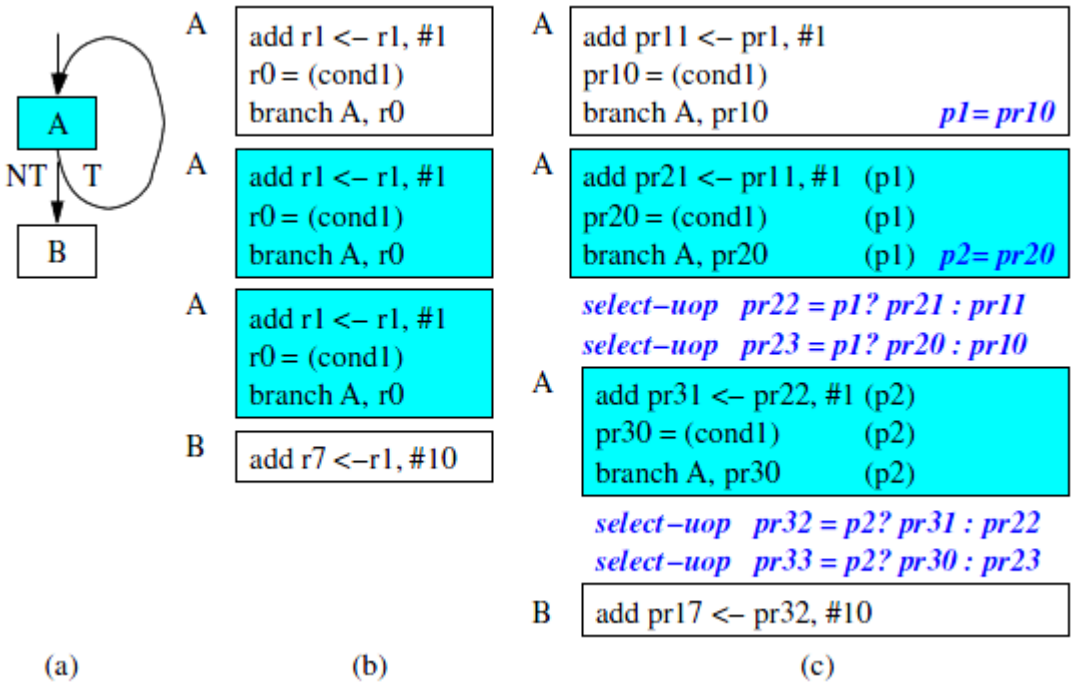
- Each iteration needs to be predicated separately.
- Use different predicate register for each iteration

Select-μops

- Φ-function SSA
  - Compilation technique, Every variable is assigned exactly once. *(very smart thinking to use! I can say…)*

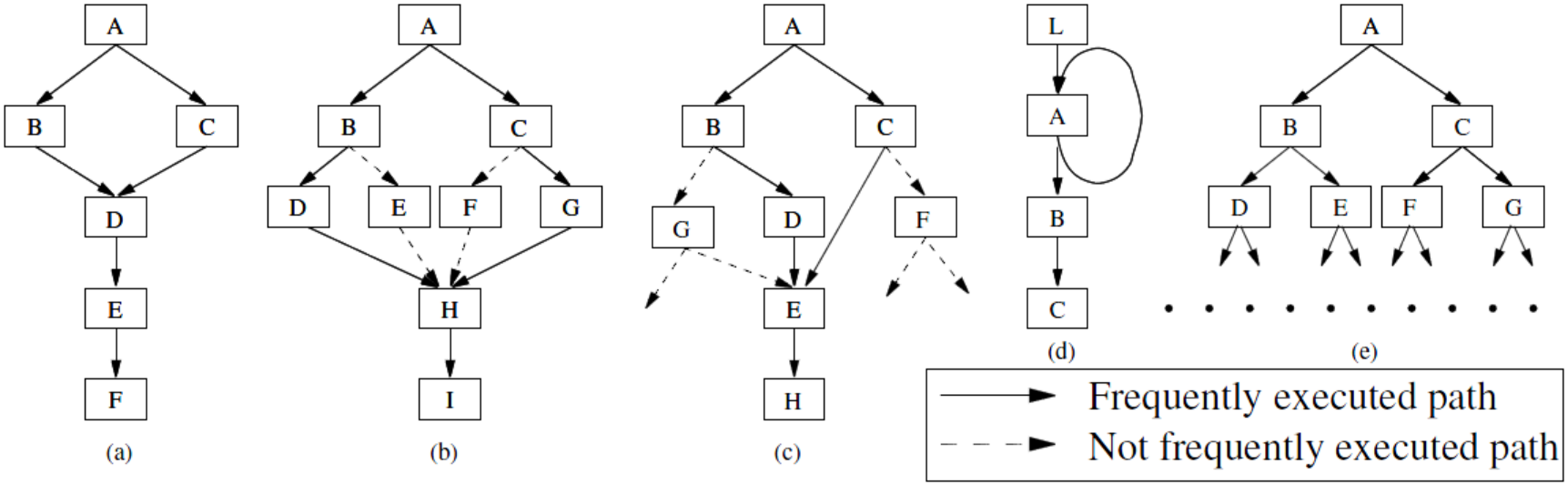


Loop Branches



- Negative Effects
  - Instruction that source the results of a previous loop iteration cannot be executed until the loop-type diverge branch is resolved.

Power of DMP



Processing model	simple hammock	nested hammock	frequently-hammock	loop	non-merging
DMP	B, C, D, E, F	B, C, D, G, H, I	B, C, D, E, H	A, A, B, C	can't predicate
Dynamic-hammock-predication	B, C, D, E, F	can't predicate	can't predicate	can't predicate	can't predicate
Software predication	B, C, D, E, F	B, C, D, E, F, G, H, I	usually don't/can't predicate	can't predicate	can't predicate
Wish branches	B, C, D, E, F	B, C, D, E, F, G, H, I	usually don't/can't predicate	A, A, B, C	can't predicate
Dual-path	path1: B, D, E, F path2: C, D, E, F	path1: B, D, H, I path2: C, G, H, I	path1: B, D, E, H path2: C, E, H	path1: A, A, B, C path2: B, C	path1: B ... path2: C ...

1. dpred-mode

- Front-end stores the address of the CFM point associated with the diverge branch into a buffer
- Front-end forks the RAS and the GHR

- If *1) CFG has more than one CFM point from same branch*, compiler provides multiple CFM and processor chooses the CFM point reached first on any path of the diverge branch and uses it to end dpred-mode.

1. dpred-mode

- What if *2) selected path does not have CFM point? 3) Or meet another branch before end of mode?* For this situation which makes dynamic predication less beneficial, DMP has policies to exit dpred-mode.
  - Counter policy(deals with *case 2*): set counter/threshold for transition steps of block(N) by heuristic!
  - Yield policy(deals with *case 3*): exits dpred-mode for the earlier diverge branch and enters dpred-mode for the later branch.
    - dpred-mode-causing diverge branch usually has a higher probability to be mispredicated.
    - dpred-mode-causing diverge is smaller so as overhead.

2. Select-µops

- When enters select-µops, processor forks 2 RATs for each path of diverge branch.
- Extend RAT with one extra bit per entry to indicate that the corresponding architectural register has been renamed in dpred-mode.
- When reaches the register renaming stage, the select-µop insertion logic compares two RATs.

3. Resolution of diverge branches

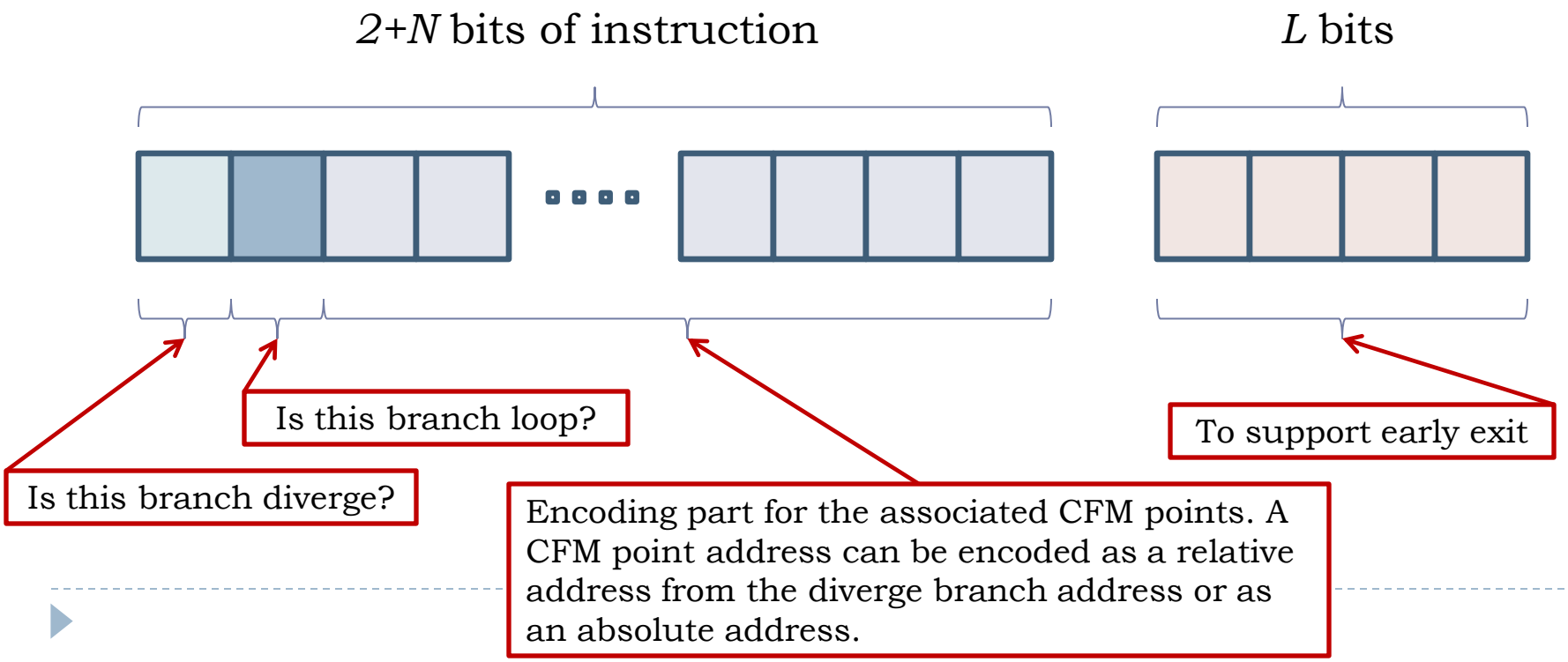
- If dpred-mode is resolved:
- Broadcasts the predicate register id of the diverge branch with the correct branch direction.
  - same predicate id & same direction
    = predicated-TRUE
  - same predicate id & different direction
    = predicated-FALSE
- If the processor is still in dpred-mode for the predicate register id, it simply exits dpred-mode and continues fetching only from the correct path as determined by the resolved branch.

4. And so on…

- Instruction Execution and Retirement
  - proceeded just like normal instructions
- Load and Store Instructions
  - Dynamically predicated load inst. is executed like normal load inst.
  - Dynamically predicated store inst. Is sent to the store buffer with their predicate register id.
- Interrupts and Exceptions
  - There is no need of saving and restoring predicate register, unlike software predications.
- Hardware Complexity Analysis
  - Increases h/w complexity, decreases energy cost

5. ISA Support for Diverge Branches

- How the compiler can transfer diverge branch and CFM point information to the hardware through simple modifications in the ISA?
  - ISA branch instruction format



Hardware Support

Hardware	DMP	Dynamic-hammock	Dual-path/Multipath	Software predication	Wish branches
Fetch support	CFM registers, +1 PC round-robin fetch	fetch both paths in simple hammock	+1/m PC round-robin fetch	-	selection between branch/predicated code
Hardware-generated predicate/path IDs	required	required	required (path IDs)	-	-
Branch pred. support	+1 GHR, +1 RAS	-	+1/m GHR, +1/m RAS	-	-
BTB support	mark diverge br./CFM	mark hammock br.	-	-	mark wish branches
Confidence estimator	required	optional (performance)	required	-	required
Decode support	CFM point info	-	-	predicated instructions	predicated instructions
Rename support	+1 RAT	+1 RAT	+1/m RAT	-	-
Predicate registers	required	required	-	required	required
Select-μop generation	required	required	-	optional (performance)	optional (performance)
LD-ST forwarding	check predicate	check predicate	check path IDs	check predicate	check predicate
Branch resolution	check flush/no flush predicate id broadcast	check flush/no flush	check flush/no flush	-	check flush/no flush
Retirement	check predicate	check predicate	selective flush	check predicate	check predicate

Experimental Result

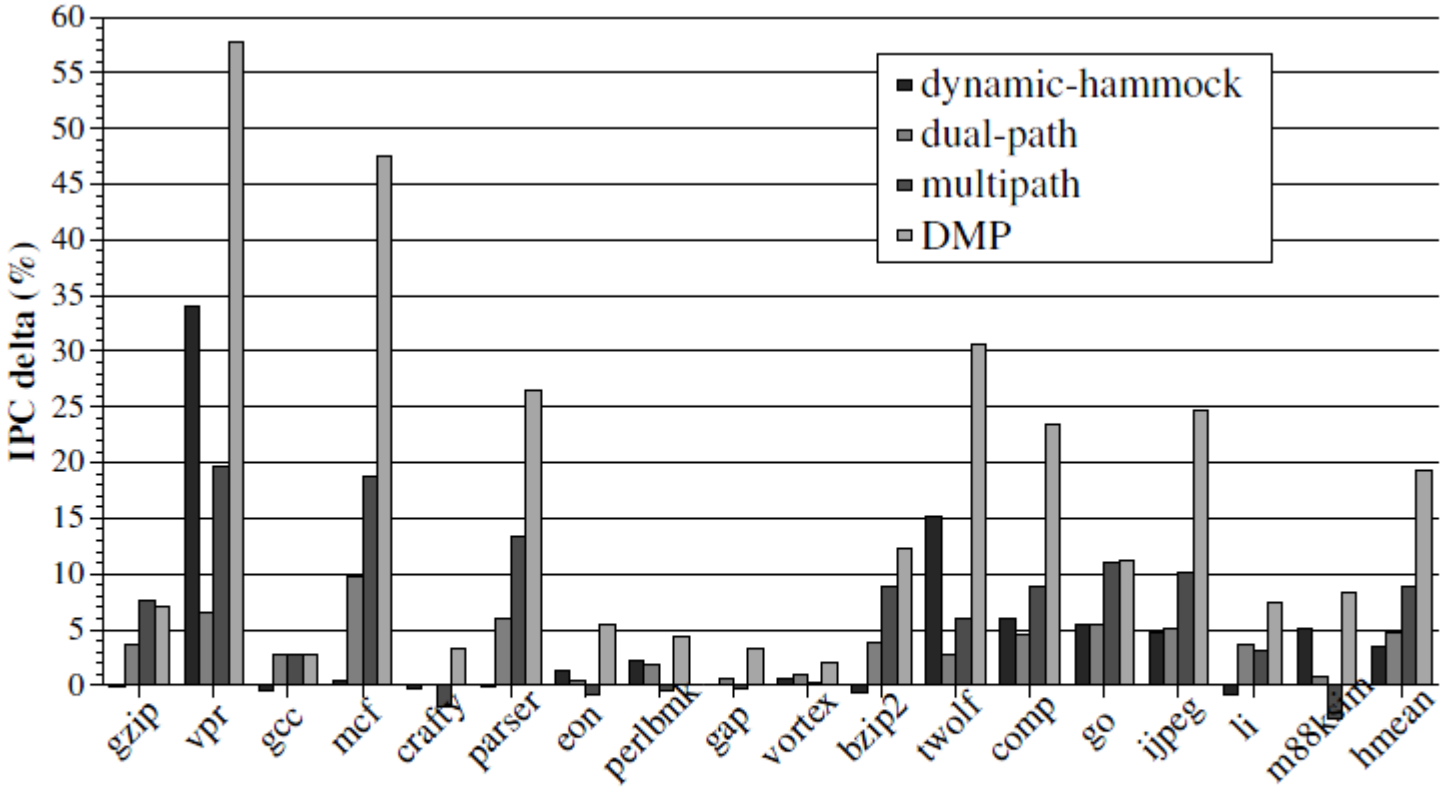


Figure 6. Performance improvement provided by DMP vs. dynamic-hammock-predication, dual-path, and multipath execution

Experimental Result

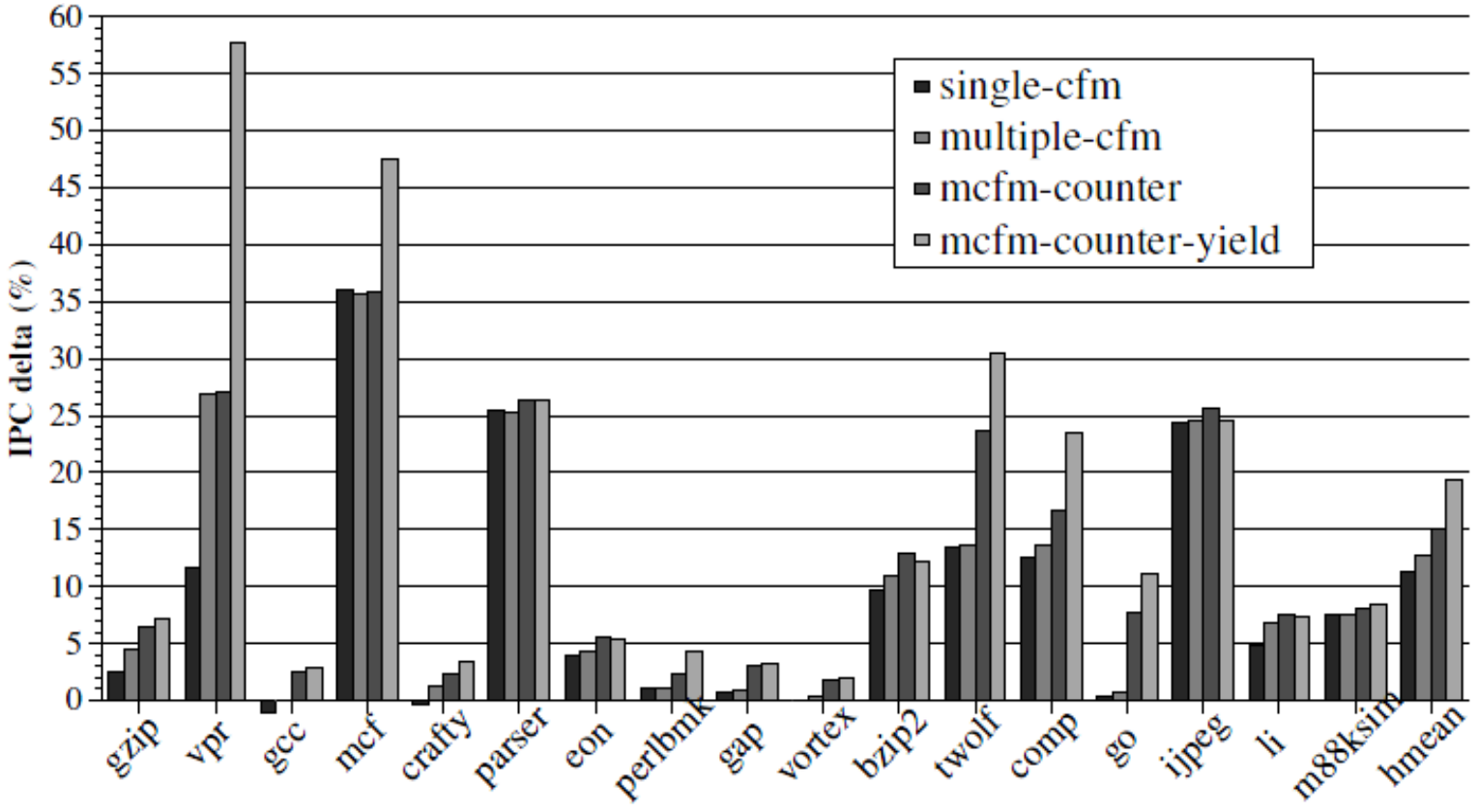


Figure 11. Performance impact of enhanced DMP mechanisms

Experimental Result

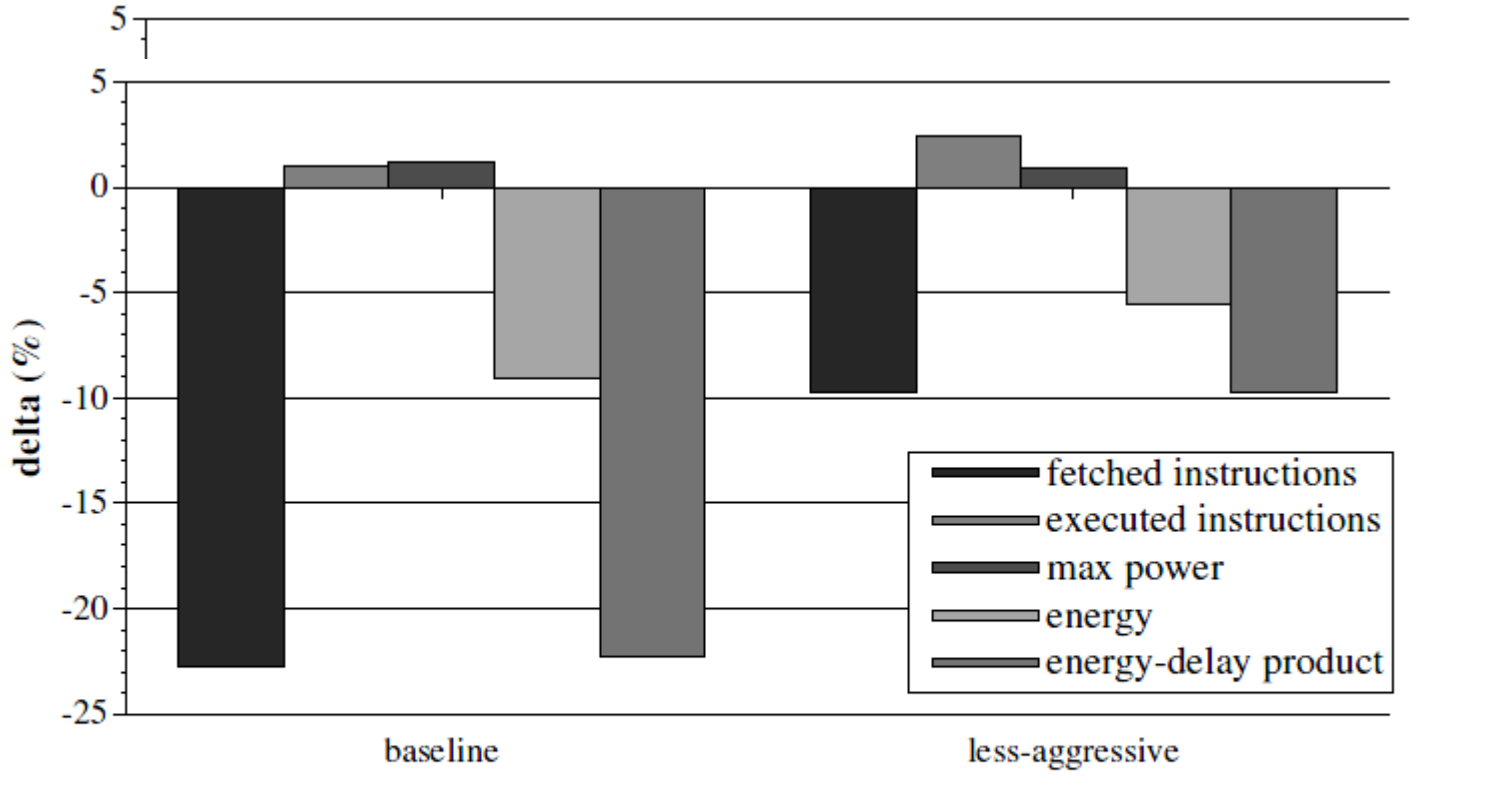


Figure 15. Power consumption comparison of DMP with the baseline processor (left) and less aggressive baseline processor (right)

Experimental Result

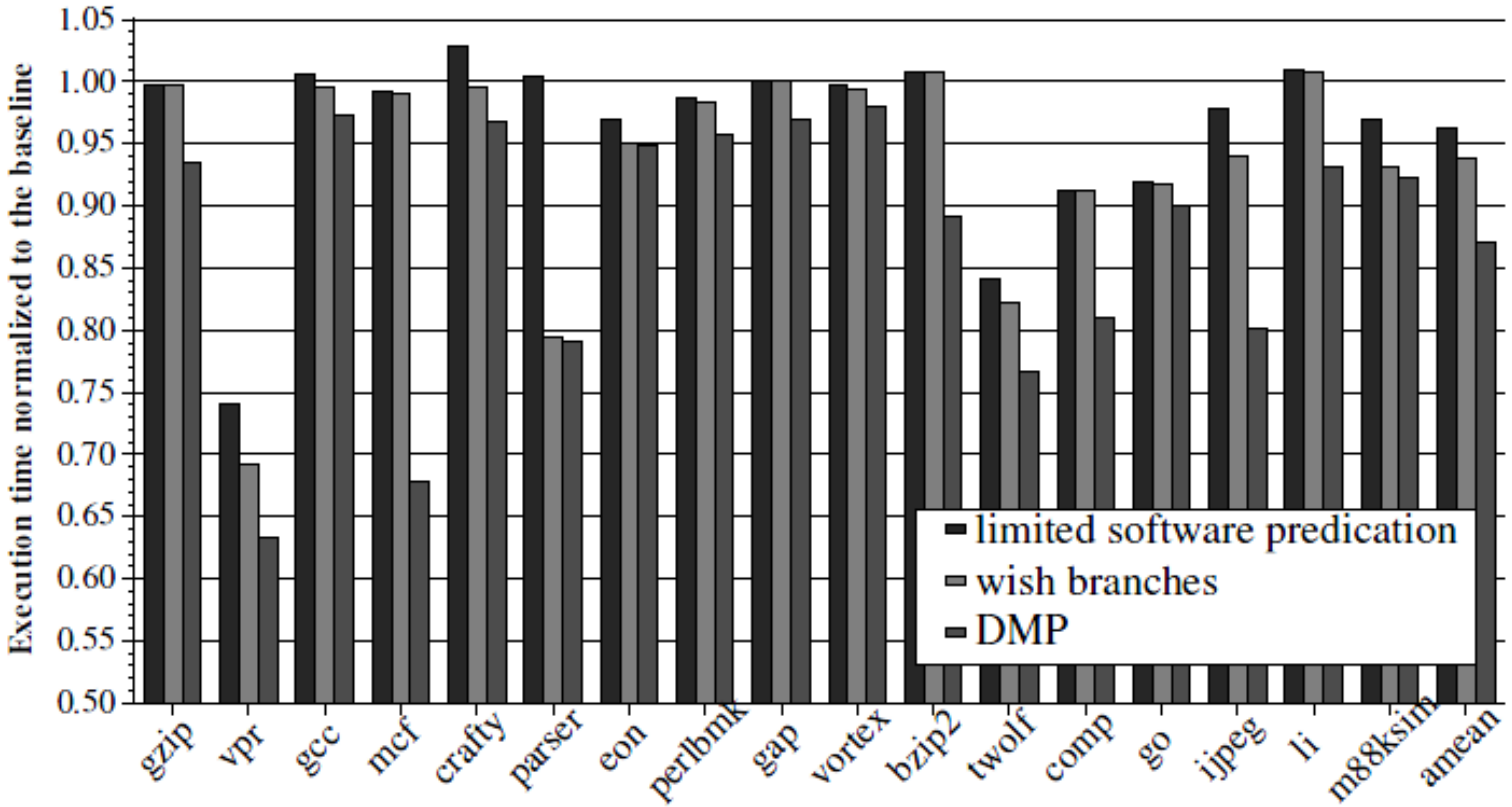
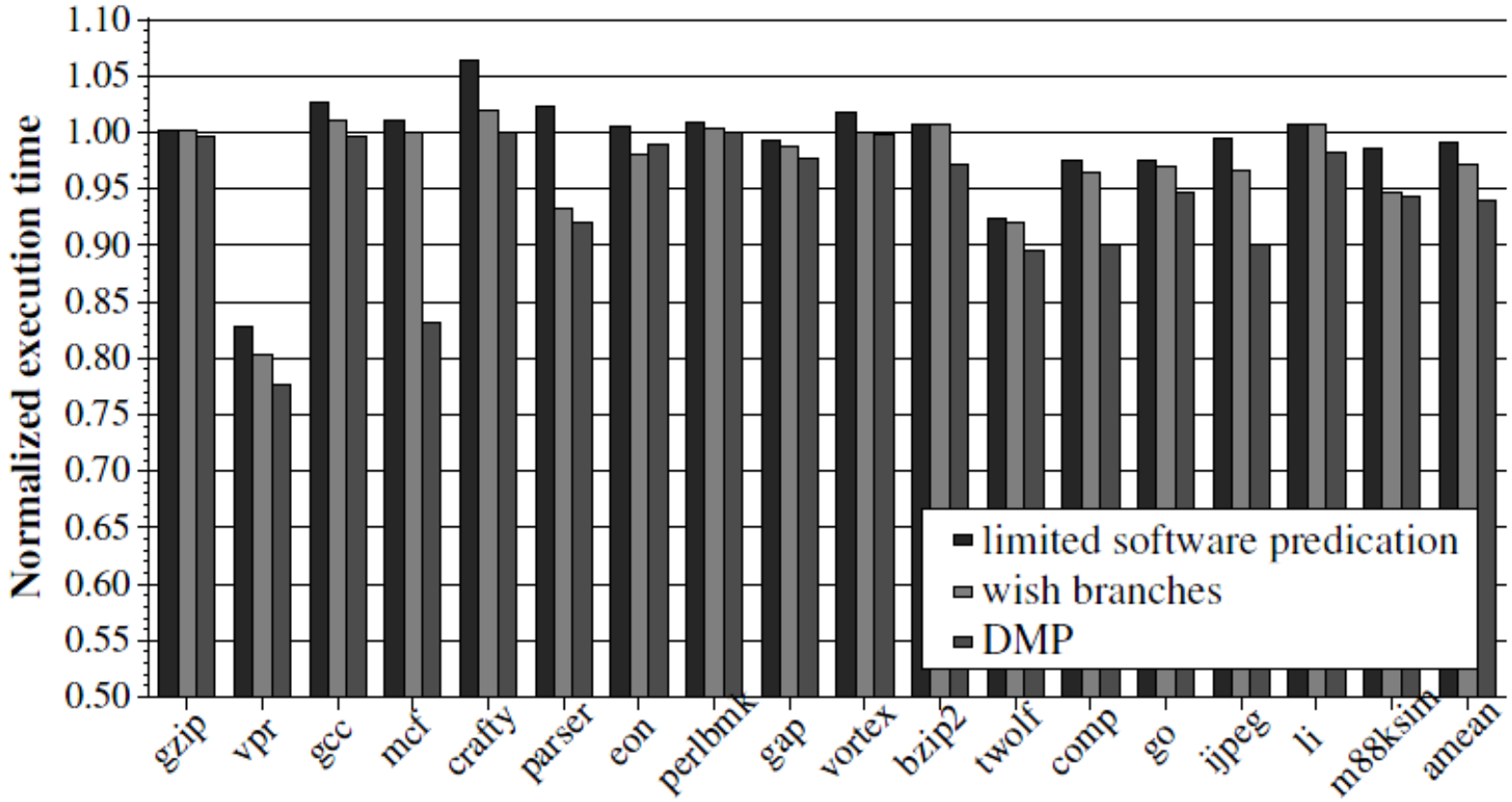


Figure 9. DMP vs. limited software predication and wish branches

Experimental Result



Thank you for listening!