How can hardware skip execution of large amount of instructions ?

Yasuhiko NAKASHIMA, Kyoto Univ./PRESTO,JST

Modern commercial microprocessors have successfully achieved high performance with superscalar or VLIW technique, which extracts ILP (Instruction Level Parallelism) in programs. But these widely used techniques are addressing serious problems such as power dissipation and memory barrier. Consequently, it becomes impossible to gain more performance on conventional programs with only ILP in the next decade. The sample program shown in the right side has a loop and a recursion that are complexly nested each other, so general parallelization technique can not be applied. This research focuses on how hardware can skip execution of large amount of instructions while producing correct results. In fact, the main processor can get correct results of the sample program by executing only a tenth part of instructions that should be executed on normal processors. This proposal employs dynamic early-computation and reuse of regions, and makes the point that neither binary-annotation nor recompilation is required.

```
T(j): loop  if (F(i, j)) {
              k = P(i, j);
              if (T(k) || k == 0 ) return true;
              else R(i, j);
            }
        return false;
```

1. Overview

ABI (Application Binary Interface) helps enlarging regions:

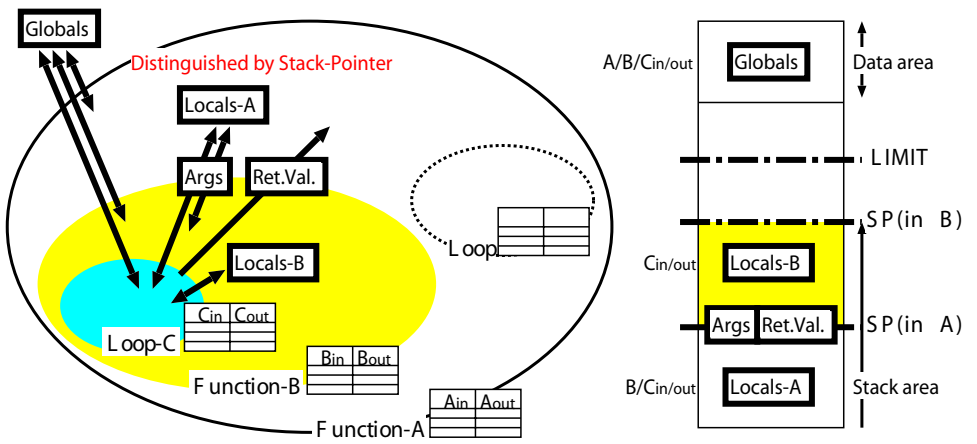
Local variables can be excluded from the relations between inputs and outputs of each region with the help of ABI. Consequently, the number of input variables to be verified and the number of output variables to be written are respectively minimized without additional special purpose instructions.

Multilevel memoizing helps enlarging regions:

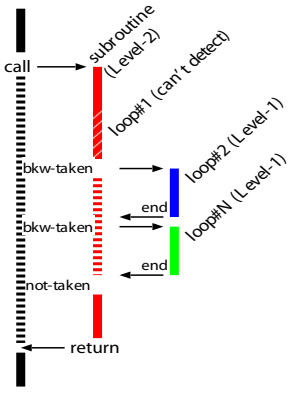
Essential inputs and outputs of nested inner regions are simultaneously memoized while executing outer region, and promising regions are selected for future reuse and early-computation.

Dedicated main thread is free from speculative execution:

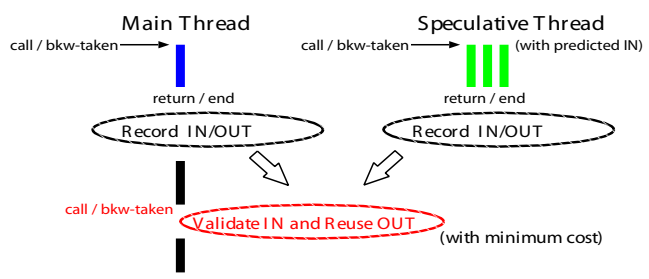
Speculative threads execute promising regions with predicted inputs and maintain reusable sets of input and output in a tree structure. Main thread can reuse these sets with minimum input test.



(a) Functions and Loops (b) Memory Map



Regions detected by Main Thread



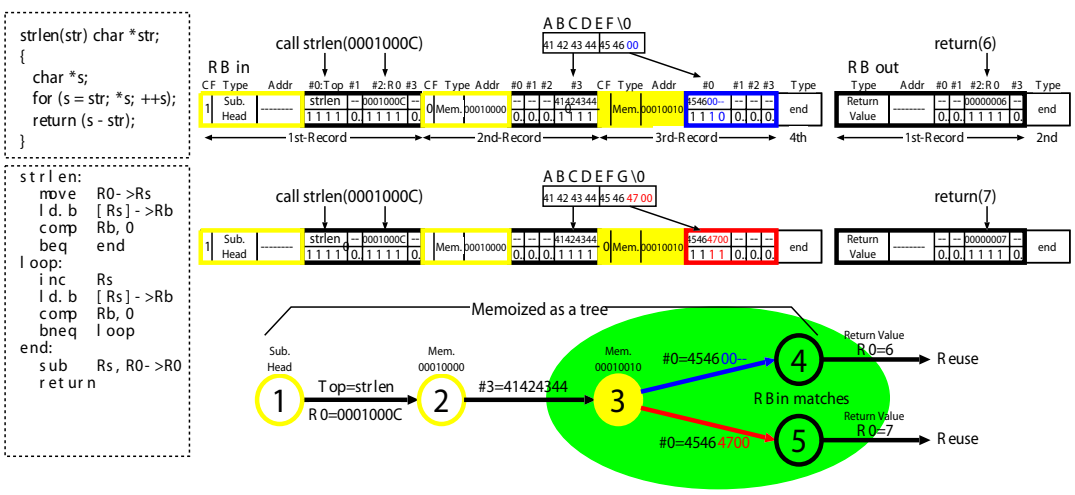
Multithreading and Region-Reuse

2. Execution Model

Combination of input-history, input-prediction, speculation, memoization and region-reuse

Each thread memoizes the input and output of each region derived from the top address of the initially assigned region into a linear list (16 bytes with mask * 64 records) corresponding to each of the multilevel regions.

When a region is finished, corresponding linear list is folded into an associative memory (16 bytes with mask * 64 lines) that represents several paths as a tree.



Linear list (Pat. 2004-258905, 355397)

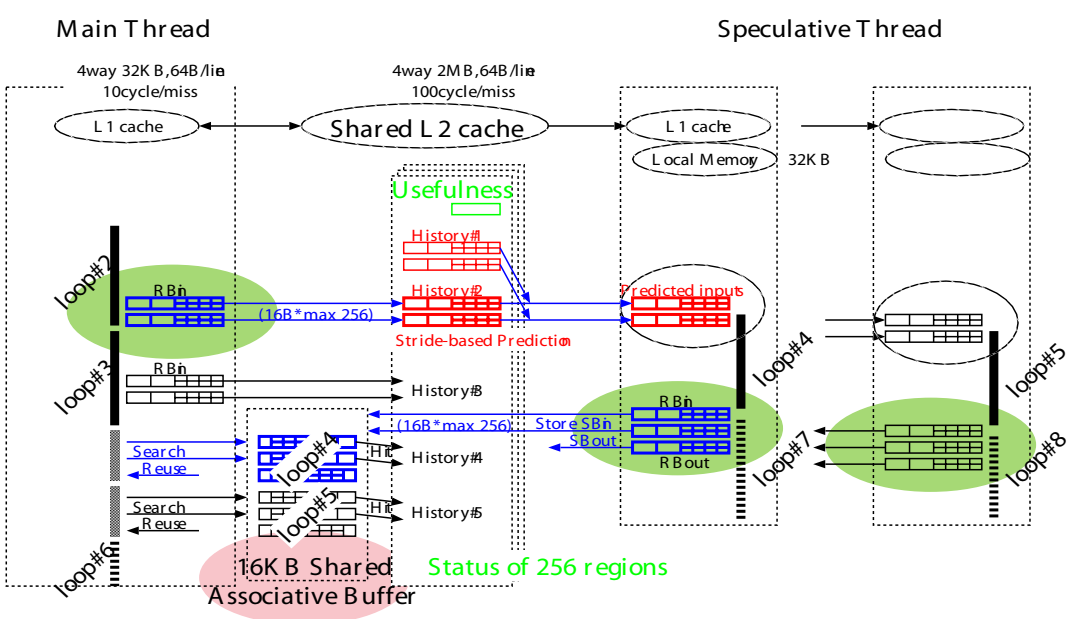
Reg./mem. read before written is memorized as input.
Reg./mem. read after written is ignored.
Reg./mem. written is memoized as output.
Local reg./mem. is ignored. (determined by stack-pointer)

Shared associative memory (Pat. 2005-92354)

Hashing technique is not applicable because valid-byte-mask is specified by valid-data itself. The edges hold valid-byte-mask and valid-data, and the nodes hold the block address to be referred next.

Hybrid structure of CAM and RAM (Pat. 2004-176140, 266056)

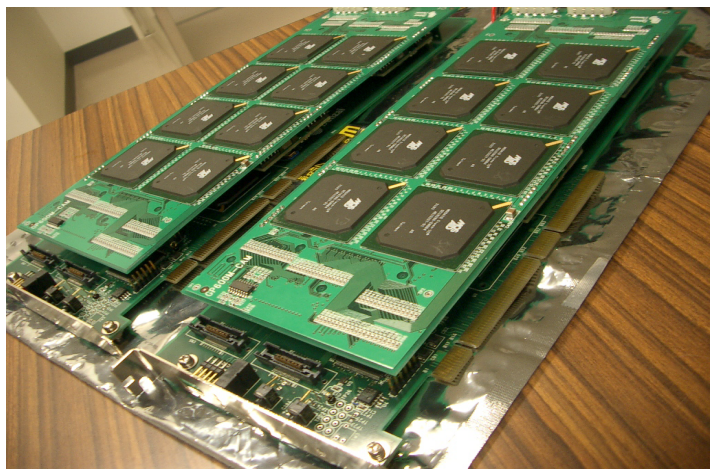
The edges are held in CAM, and the nodes are held in RAM. If a traversal encounters at a point where the tree structure has no more branch, further reg./mem. are guaranteed as not-overwritten. The path can be reused without further traversal.



3. Evaluation with Real CAM

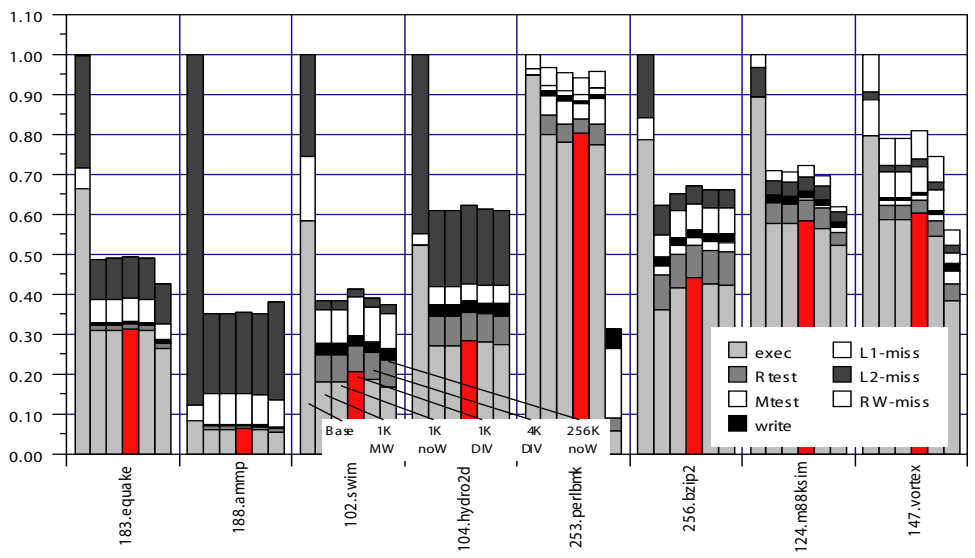
Large-scale accelerator for simulating associative search

288/144 bits with masks * 256K/512K entries
PCI bus interface, 1 usec/search
900 times faster than software
It accelerates the performance simulator of this proposal by 90 times.



	spec95	spec2k	
1-thread max./ave.	28%/ 3%	52%/ 6%	speedup
4-thread max./ave.	158%/28%	150%/20%	speedup

Promising region contains average hundreds of instructions.
The hit ratio of shared L2-cache is greatly improved.



Relaying mechanism between speculative threads
(Pat. 2004-324348, 347124)
Partitioning of associative memory for high-speed and low-power
(Pat. 2005-234806)
Base: no SPs, 2-way SS SPARC
1K-DIV: 3 SPs, 1KB associative memory * 16 blocks (red bar)
4K-DIV: 3 SPs, 4KB associative memory * 16 blocks
256K: 3 SPs, 4MB associative memory (upper bound)

(Normalized by cycles with neither reuse nor speculative threads)

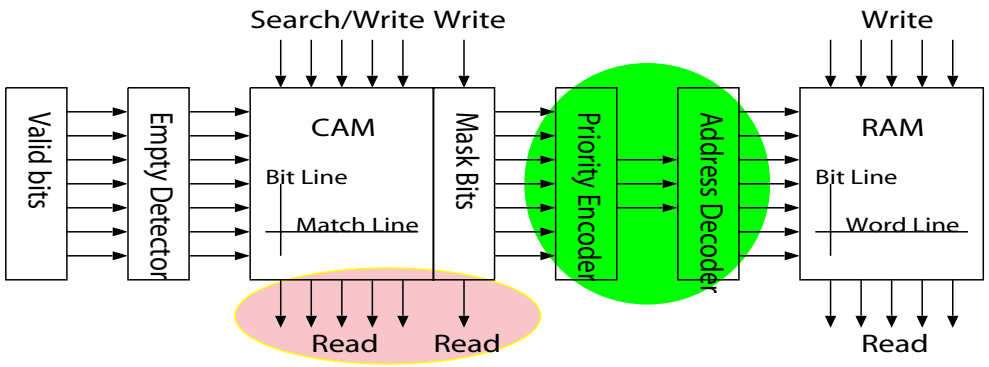
4. Feasibility Study

Special purpose CAM for high-frequency (Pat. 2005-234806)

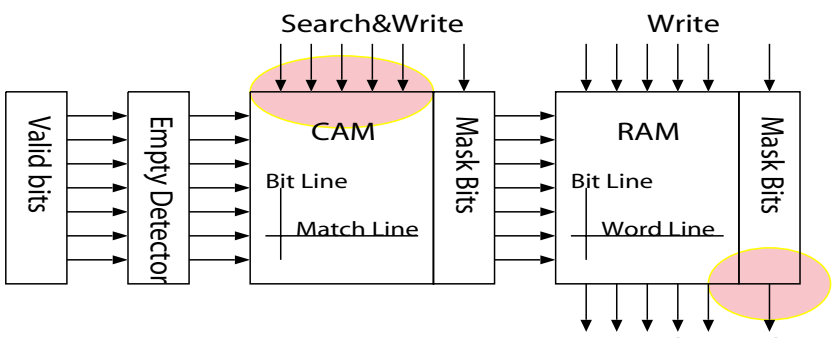
Search & write function guarantees that at most one match-line is asserted. Consequently, priority encoder and address decoder can be omitted. Search & write is accelerated by sharing bit-lines between comparand-data and write-data.

Search & read function used for logging matched data is accelerated by duplicating mask-bits between CAM part and RAM part.

Above special purpose 1KB associative memory is designed with 0.18um rule, and operates in 1.6ns under HSPICE simulator.



Reuse Buffer with General Purpose CAM (One cycle for Search, Write, and Read)



Reuse Buffer with Special Purpose CAM (One cycle for Search&Write and Search&Read)