A Novel Instruction Scratchpad Memory Optimization Method based on Concomitance Metric

Andhi Janapsatya, Aleksandar Ignjatovic,
Sri Parameswaran
School of Computer Science and Engineering

UNSW
THE UNIVERSITY OF NEW SOUTH WALES
SYDNEY • AUSTRALIA

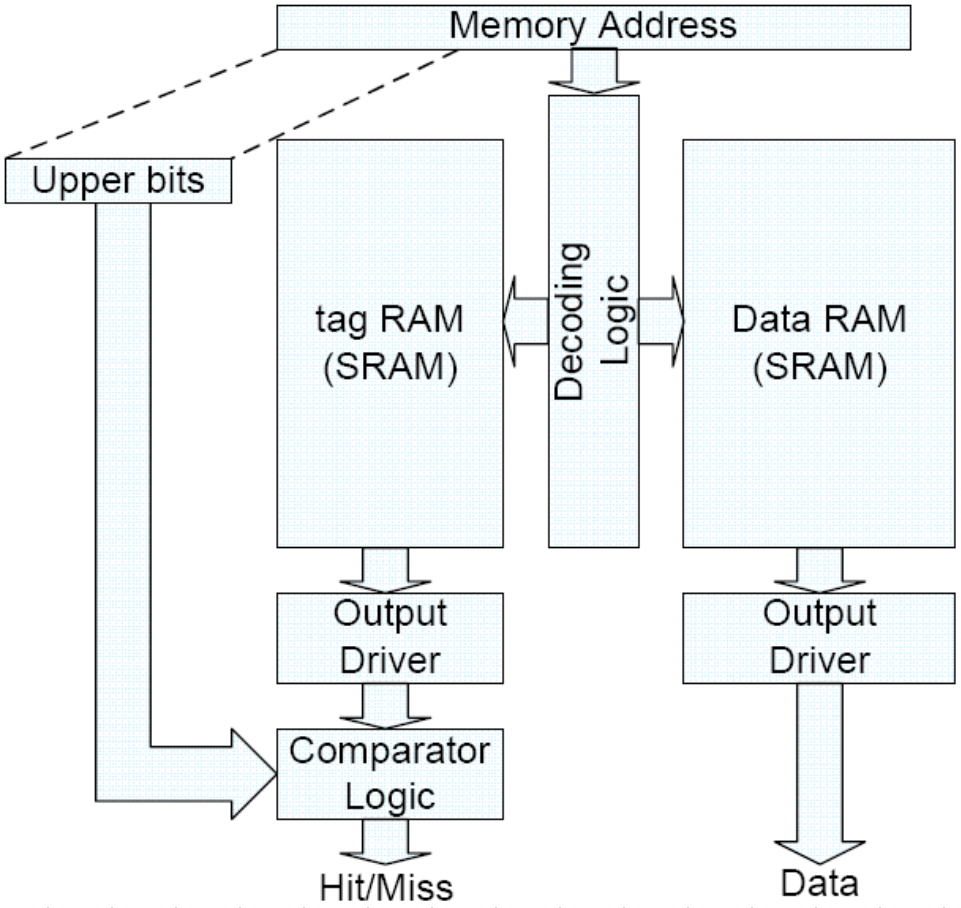
Outline

- Scratchpad Memory vs Cache Memory
- Motivation and Goal
- Existing Work
- Allocation Algorithm
- Experiment and Results
- Conclusion

Cache Memory

- Cache memory uses SRAM cells.
- Hardware tag checking mechanism to control content of cache memory.
- Software does not need to control content of the cache memory.

Cache Architecture

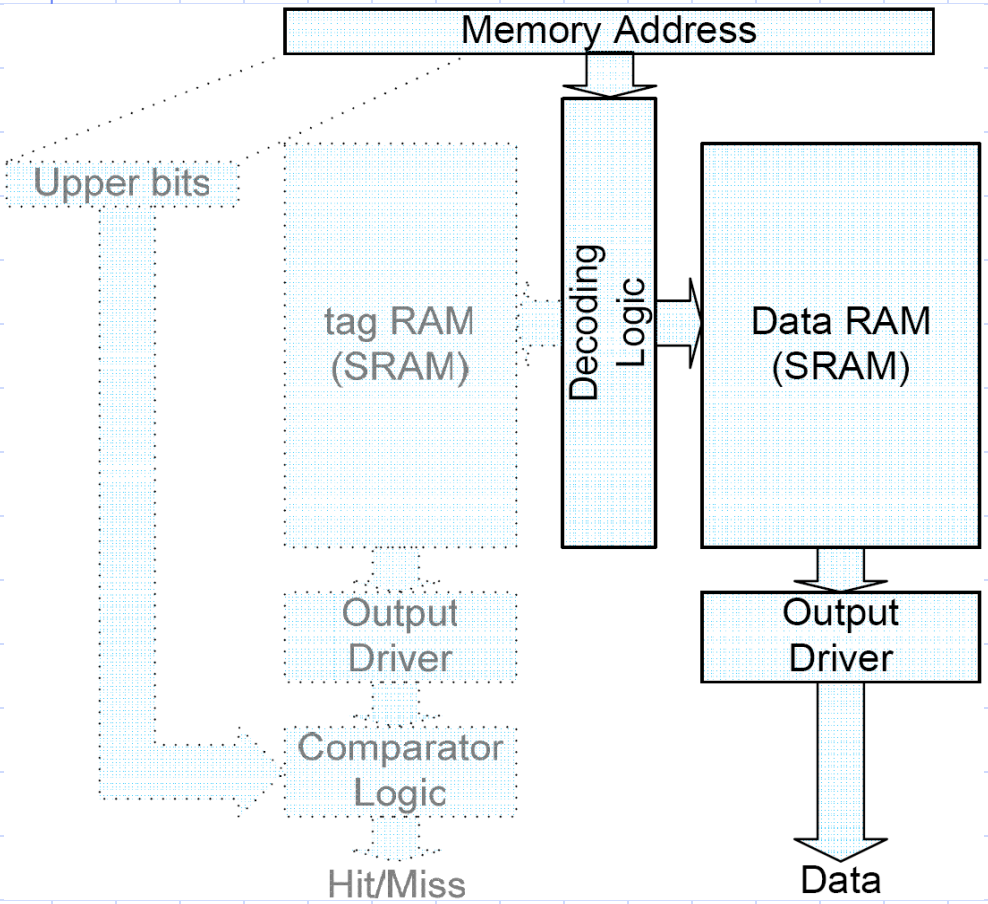


- Automatic checking of cache hit/miss using tag-checking process.
- Every instruction fetch has to go via cache.

Scratchpad Memory

- Scratchpad memory uses SRAM cells.
- No hardware tag checking mechanism.
- Software controlled.
- Smaller total cells area compared to cache memory due to non-existent of tag-RAM.
- Less energy consumption compared to cache memory.

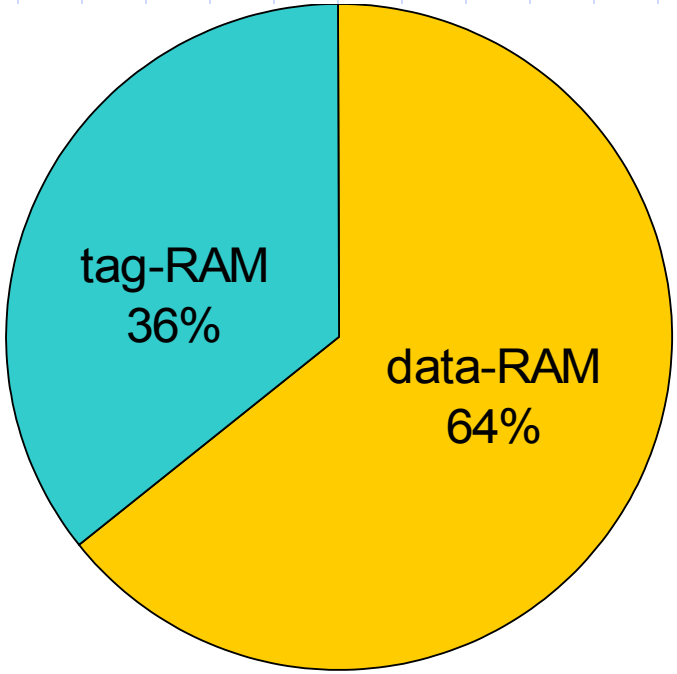
Scratchpad Memory



- No Tag-RAM.
- Smaller chip area per bits compared to cache memory.

Cache Memory

- Energy consumption breakdown for instruction cache (direct-mapped).



- CACTI energy estimation tools shows that 36% of direct-mapped cache access energy is due to tag-RAM access. (0.18 μm)

Scratchpad vs Cache

Size (bytes)	Cache acc. time(ns)	SPM acc. time(ns)	ratio	Cache acc. energy(nJ)	SPM acc. energy(nJ)	ratio
512	1.19	0.74	1.61	1.37	0.18	7.61
1024	1.24	0.78	1.59	1.37	0.19	7.21
2048	1.30	0.83	1.57	1.39	0.20	6.95
4096	1.31	0.88	1.49	1.42	0.23	6.17
8192	1.34	1.05	1.28	1.49	0.29	5.14
16384	1.64	1.21	1.36	1.55	0.36	4.31

Motivation

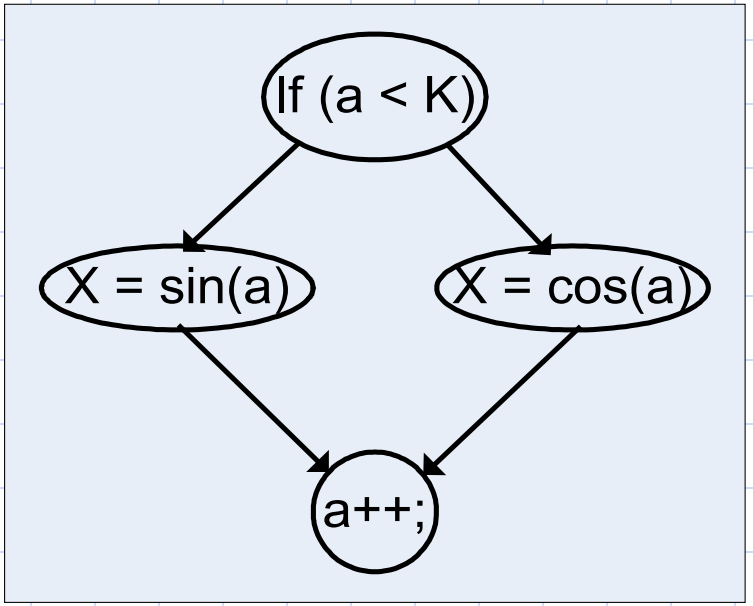
- Embedded application and embedded processor are known prior to execution.
- Profiling can identify the hot-spots in applications.
- We aim to take advantage of profiling information, knowledge of the application, and the processor architecture for system optimization.
- We propose the use of instruction scratchpad memory (SPM) as a replacement of the instruction cache.

Motivation

- Existing scratchpad memory methodologies utilize loop analysis to identify objects that are to be executed from the scratchpad memory.
- Loop analysis can be a complex procedure to execute.
- Precise structure of a loop is irrelevant for deciding which object is to be executed from the scratchpad memory.

Example

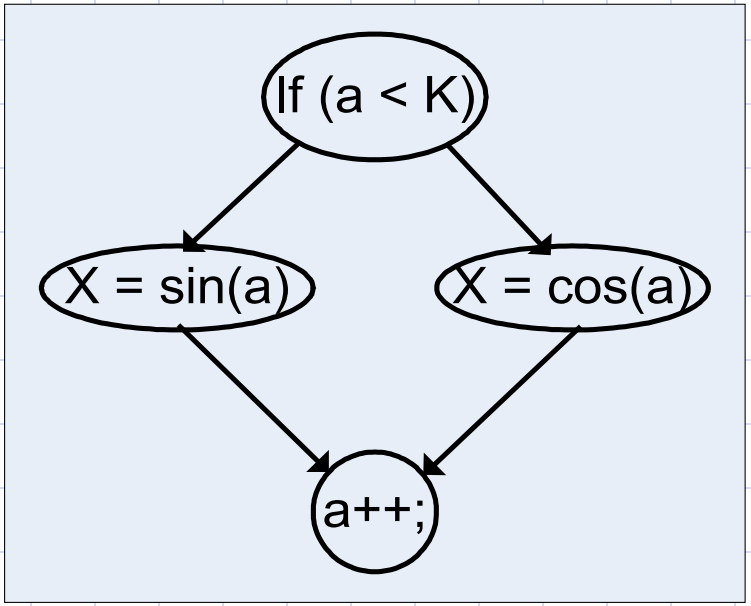
```
for (a=0; a < M; a++) {
   If ( a < K)
      x = sin(a);
   else
      x = cos(a);
}
```



- K = 100; M=200;
- Temporal information can record the time difference in the execution of the two paths.

Example

```
for (a=0; a < M; a++) {
   if (a == even) //if ( a < K)
      x = sin(a);
   else
      x = cos(a);
}
```



- Replace “if (a < K)” with “if (a is even)”
- Loop analysis cannot differentiate the two cases.

Goal

- Utilize scratchpad memory as a replacement for the instruction cache memory.
- Use temporal information to analyze the temporal proximity of different instruction block objects.

Existing Work

- Existing work on scratchpad memory utilization can be classified into:
  - Static management of instruction memory
  - Dynamic management of instruction memory
  - Static management of data memory
  - Dynamic management of data memory

Existing Work

- Static management refers to scheme where the content of the scratchpad memory does not change during the execution of a program.
- And dynamic management refers to architecture that can change the content of the scratchpad memory during execution.

Existing Work

- Panda [1997] and Avissar[2002] presented schemes for static management of data SPM.
- Kandemir[2002] introduced dynamic management of SPM for data memory.
- Udayakumaran[2003] improve the dynamic management scheme for data SPM.

Existing Scratchpad Systems

- Steinke [2002] and Angiolini [2003] presented scheme for static instruction scratchpad memory.
- Janapsatya [2004] presented a dynamic management scheme for instruction scratchpad memory.

Existing Work

- Janapsatya [2004] identify frequently executed code segments within the program and chose these code segments to be executed from the scratchpad memory.
- They perform loop analysis to identify start of loops as the point in the program to copy instruction into the SPM.

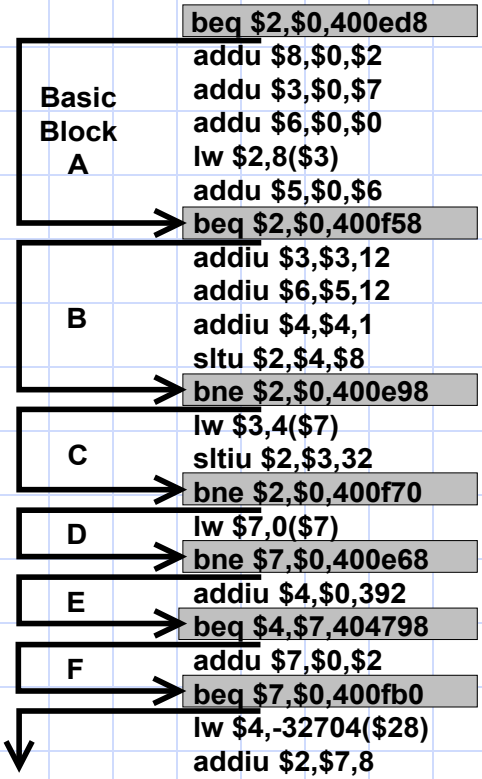
Concomitance

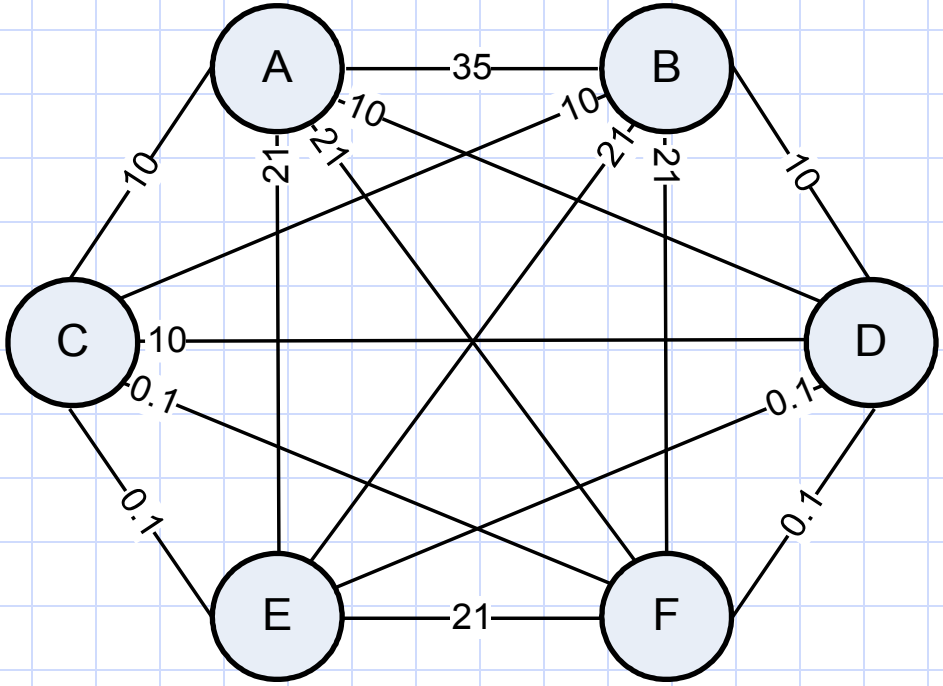
- Concomitance is a quantitative measure on the time correlation of two basic blocks.
- Concomitance is given by the following equation: (please refer to paper for further details)

$$\tau(a,b,T) = \sum_{\substack{b\in[e(a),e'(a)] \\ e(a)\in T}} W(d[e(a),e'(a)]) + \sum_{\substack{a\in[e(b),e'(b)] \\ e(b)\in T}} W(d[e(b),e'(b)])$$

Concomitance

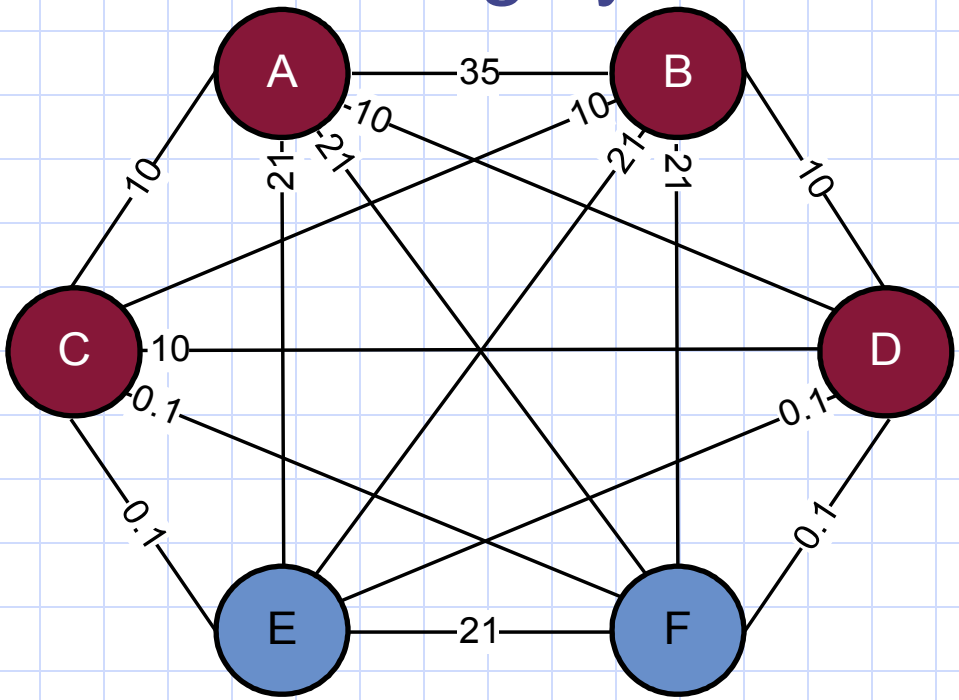
- The following is an example of a concomitance graph.





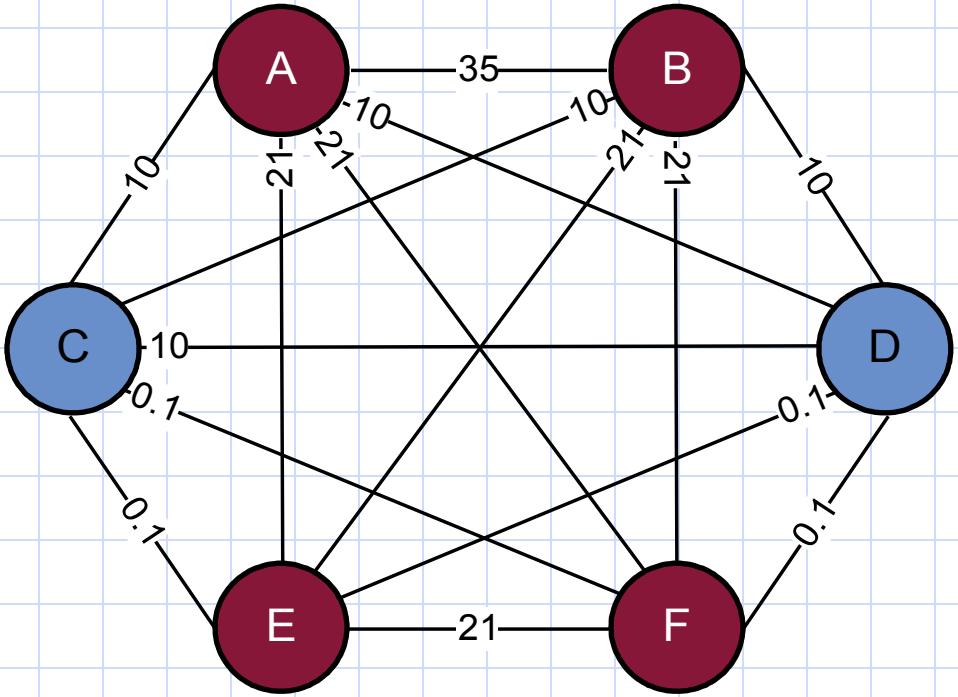
Concomitance

- The concomitance value indicates that block A, B, C, and D are highly correlated.



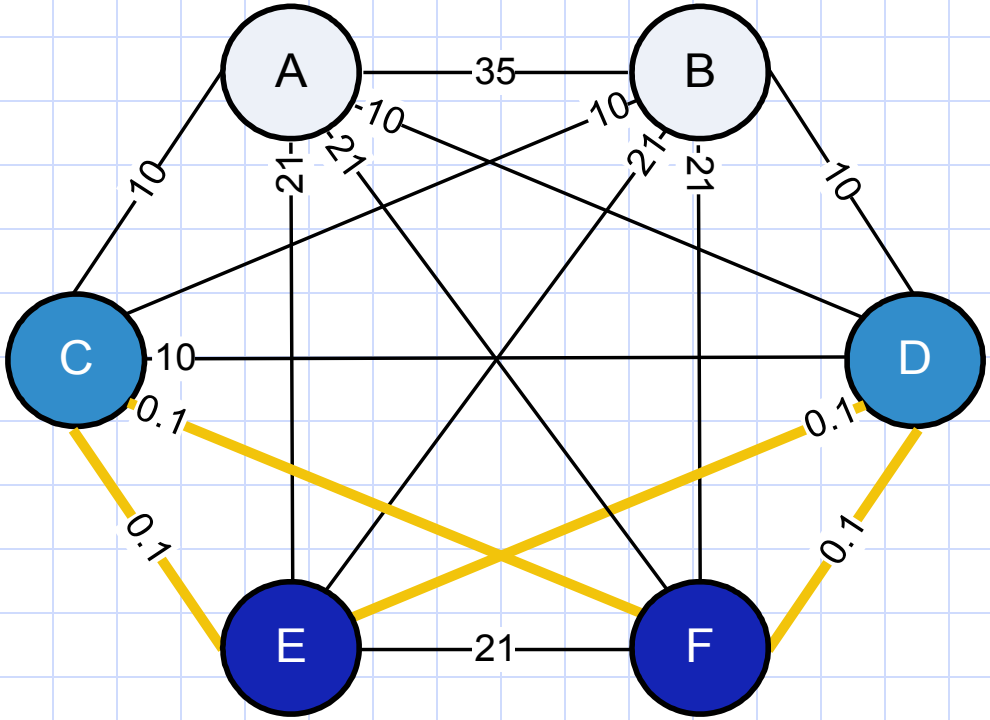
Concomitance

- Also block A, B, E, and F are highly correlated.

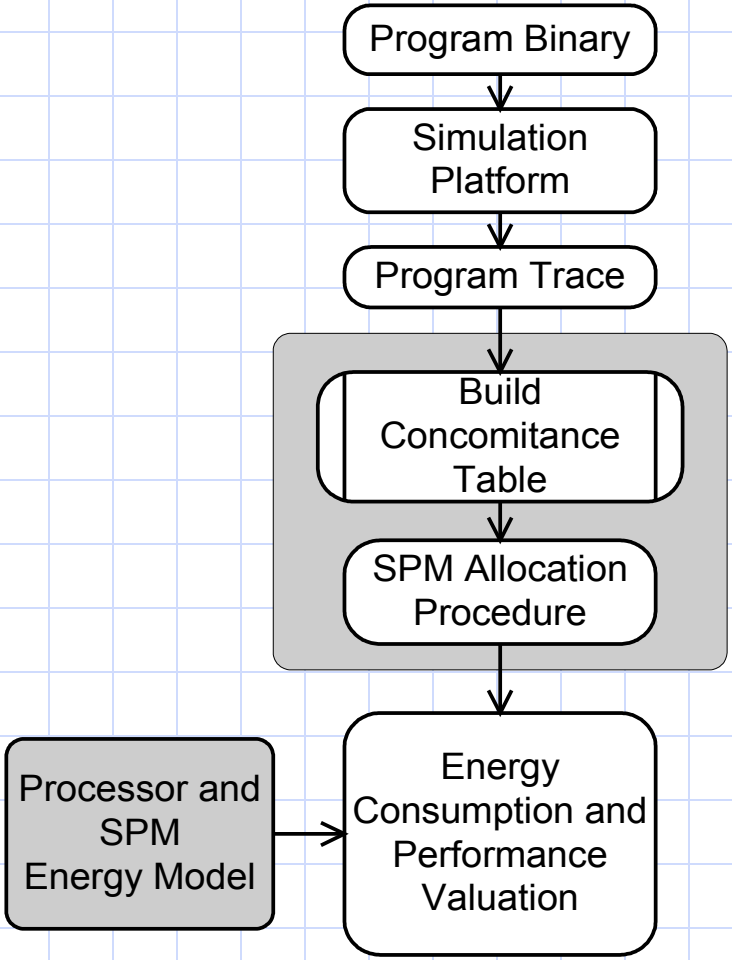


Concomitance

- Also block A, B, E, and F are highly correlated.
- Indicate blocks C, D may overlap with E, F in the memory



Allocation Algorithm



- Concomitance table is build to provide information on the ranking of different basic block.
- SPM allocation procedure is to identify points in the program to insert the copy instruction.

Allocation Algorithm

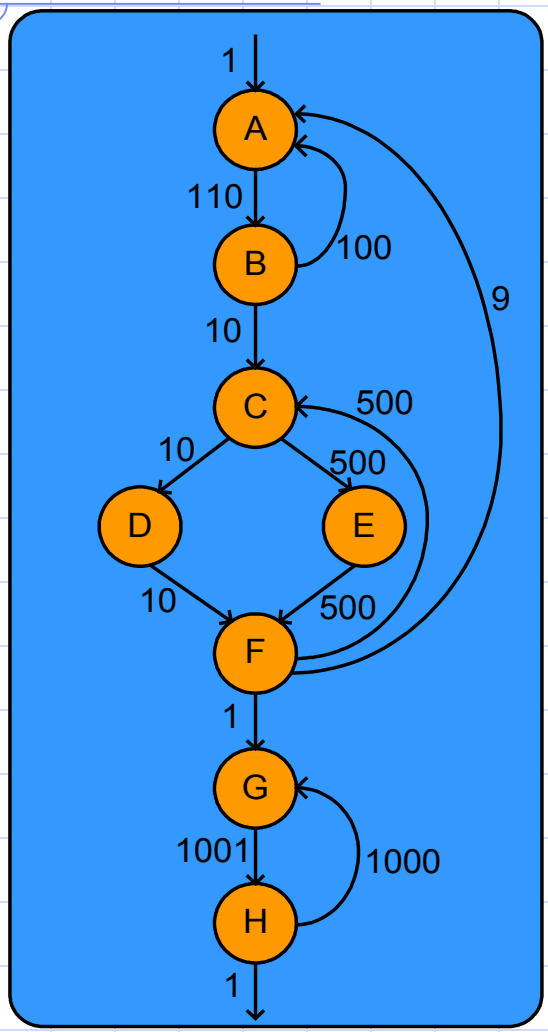
Construct one group containing all the vertices in S;
Sort the *concomitance* values of all pairs $\{a, b\}$ of basic blocks in ascending order;
Start from lowest *concomitance* value;
For each pair of basic blocks $\{a, b\}$ {
    if the resulting group size is larger than SPM size
        Cut the edge connecting a and b
}

- Edge-cut procedure is applied to the concomitance graph.

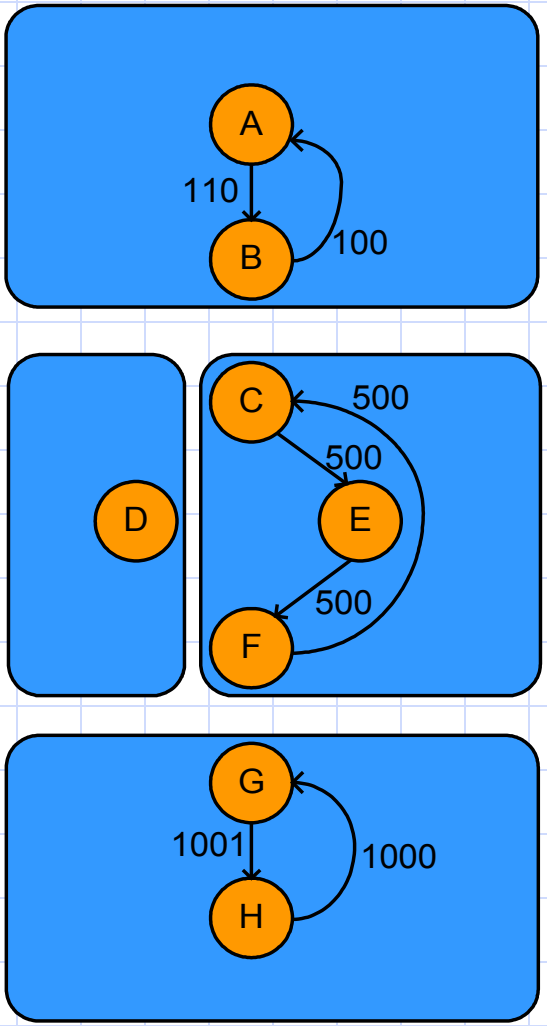
Allocation Algorithm

- Edge-cut procedure is used to determine the sub-graphs for allocation of basic blocks into the scratchpad memory.
- The CFG is then used to identify locations for inserting the SMI (new instruction).

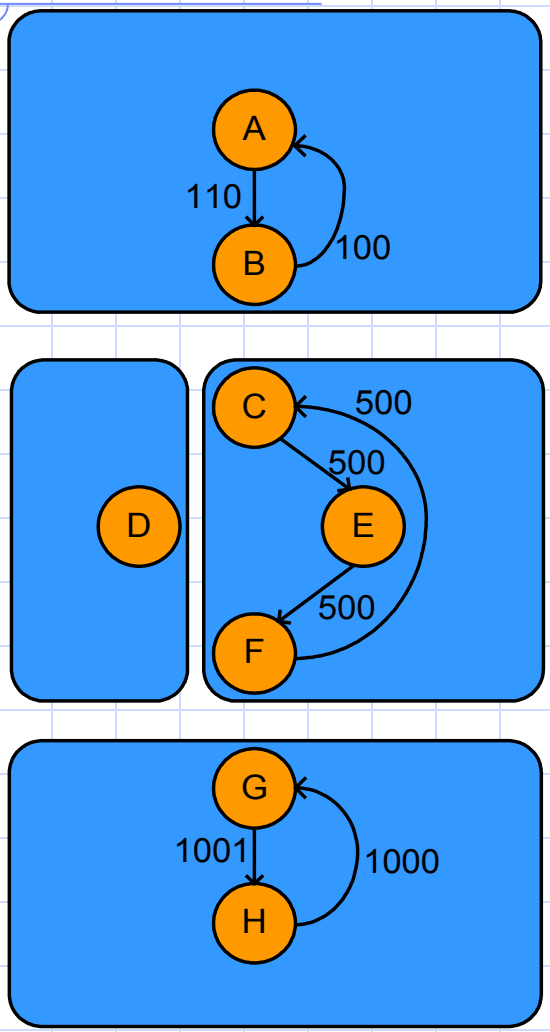
Allocation Algorithm



Edge-cut procedure determine the subgraphs.

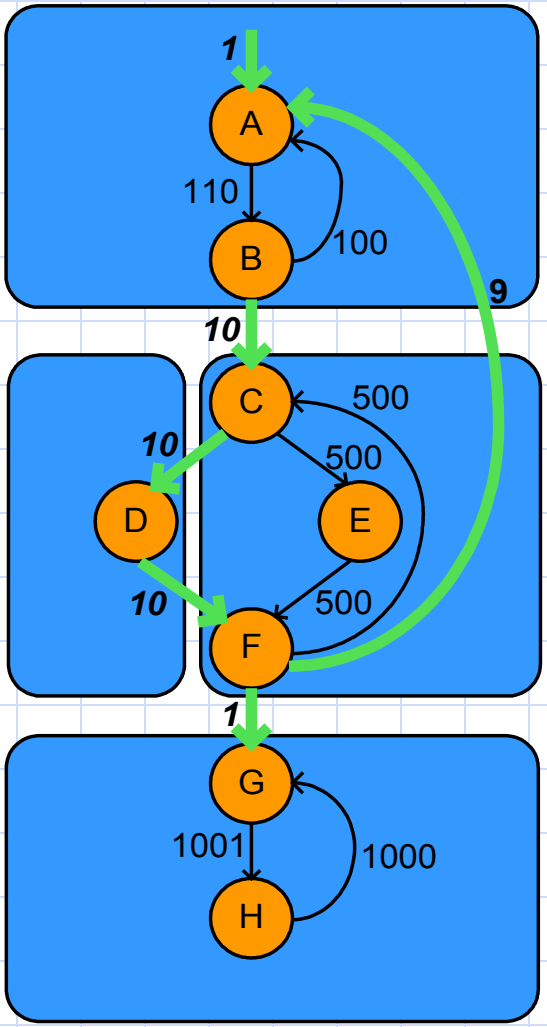


Allocation Algorithm



Analyze the CFG to determine locations of SMI

Green arrows show location of SMI.



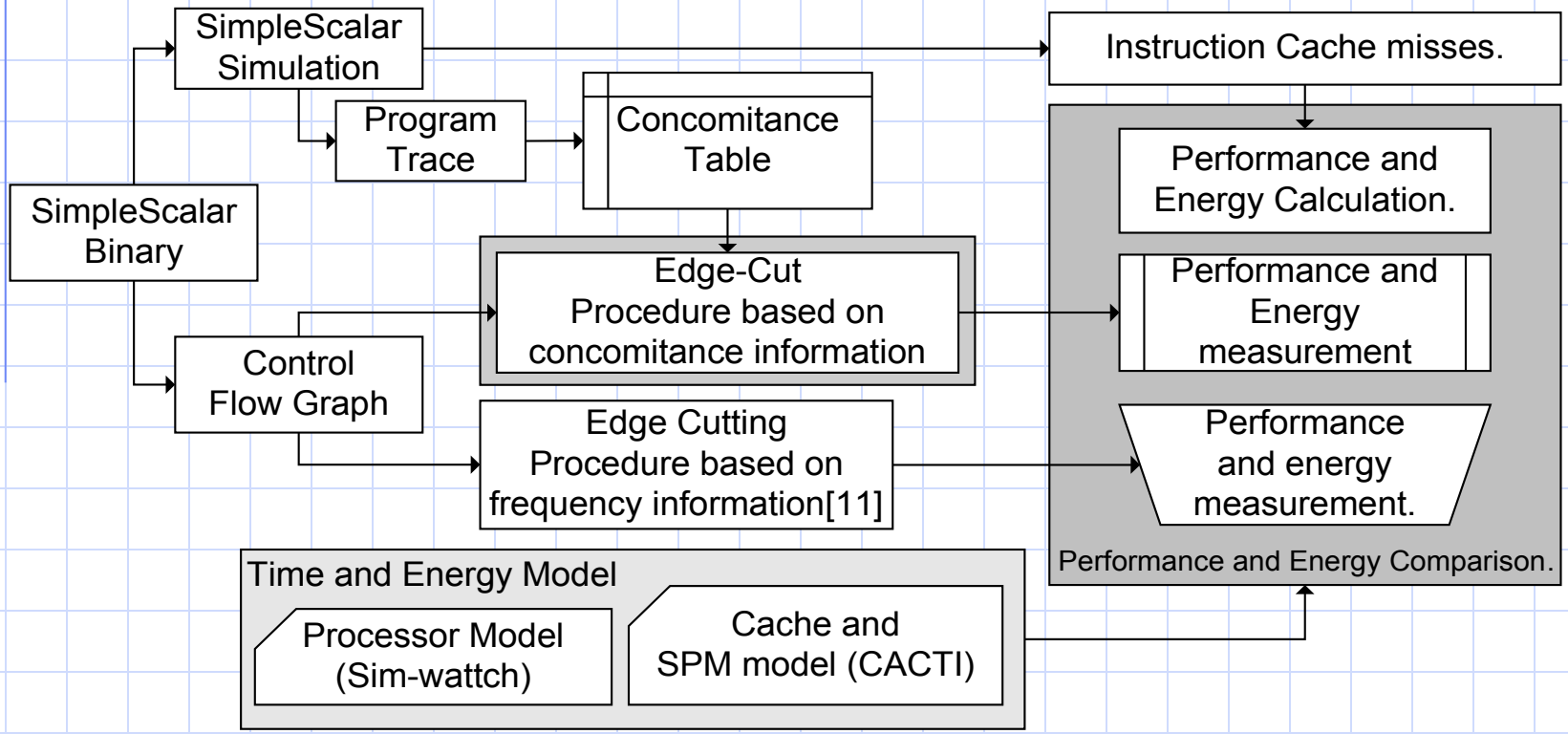
Experimental Setup

- Program traces were generated using SimpleScalar 3.0d [Burger, 1997]
- Benchmarks were taken from Mediabench suite [Lee, 1997]

Experimental Setup

- Cache memory and scratchpad memory energy model derived from CACTI [Shivakumar, 2001] energy model.
- Processor energy model derived from wattch [Brooks, 2000]

Experimental Setup



Results

App.	Prog. size	Total no. of insn. Exec.	Copy insn. inser-ted	Avg. no. of insn. copied into SPM	SMI added [11]	Avg. no. of Insn. copied into SPM[11]
rawcaudio	9182	6689768	4.6	1247	30.4	741972
rawdaudio	9384	12414463	4.6	1263	29.7	2704463
g721enc	11052	314594475	6.2	33751706	63	130722589
g721dec	11066	302967631	4.8	12844416	58	87833725
mpeg2enc	26808	1134231679	23.4	4385032	272.86	197787388

Results

App.	Prog. size	Total no. of insn. Exec.	Copy insn. inser-ted	Avg. no. of insn. copied into SPM	SMI added [11]	Avg. no. of Insn. copied into SPM[11]
rawcaudio	9182	6689768	4.6	1247	30.4	741972
rawdaudio	9384	12414463	4.6	1263	29.7	2704463
g721enc	11052	314594475	6.2	33751706	63	130722589
g721dec	11066	302967631	4.8	12844416	58	87833725
mpeg2enc	26808	1134231679	23.4	4385032	272.86	197787388

Results

App.	Prog. size	Total no. of insn. Exec.	Copy insn. inser-ted	Avg. no. of insn. copied into SPM	SMI added [11]	Avg. no. of Insn. copied into SPM[11]
rawcaudio	9182	6689768	4.6	1247	30.4	741972
rawdaudio	9384	12414463	4.6	1263	29.7	2704463
g721enc	11052	314594475	6.2	33751706	63	130722589
g721dec	11066	302967631	4.8	12844416	58	87833725
mpeg2enc	26808	1134231679	23.4	4385032	272.86	197787388

Results

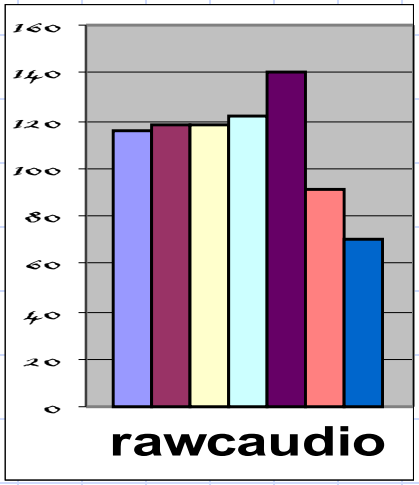
App.	Prog. size	Total no. of insn. Exec.	Copy insn. inser-ted	Avg. no. of insn. copied into SPM	SMI added [11]	Avg. no. of Insn. copied into SPM[11]
rawcaudio	9182	6689768	4.6	1247	30.4	741972
rawdaudio	9384	12414463	4.6	1263	29.7	2704463
g721enc	11052	314594475	6.2	33751706	63	130722589
g721dec	11066	302967631	4.8	12844416	58	87833725
mpeg2enc	26808	1134231679	23.4	4385032	272.86	197787388

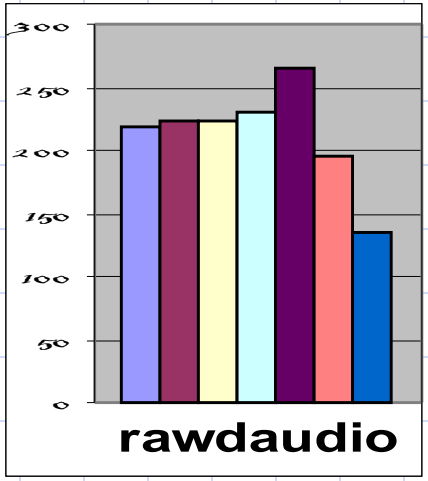
Results

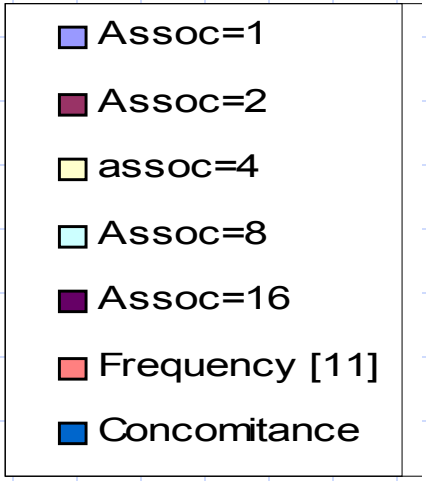
App.	Prog. size	Total no. of insn. Exec.	Copy insn. inser-ted	Avg. no. of insn. copied into SPM	SMI added [11]	Avg. no. of Insn. copied into SPM[11]
rawcaudio	9182	6689768	4.6	1247	30.4	741972
rawdaudio	9384	12414463	4.6	1263	29.7	2704463
g721enc	11052	314594475	6.2	33751706	63	130722589
g721dec	11066	302967631	4.8	12844416	58	87833725
mpeg2enc	26808	1134231679	23.4	4385032	272.86	197787388

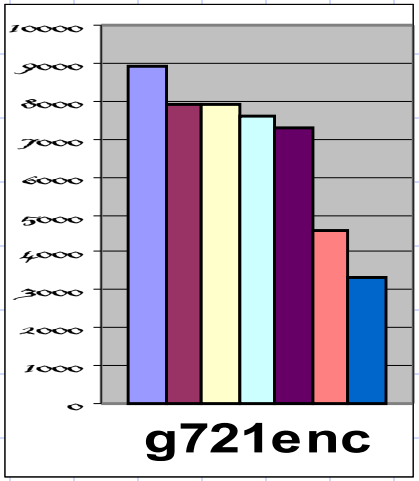
Results

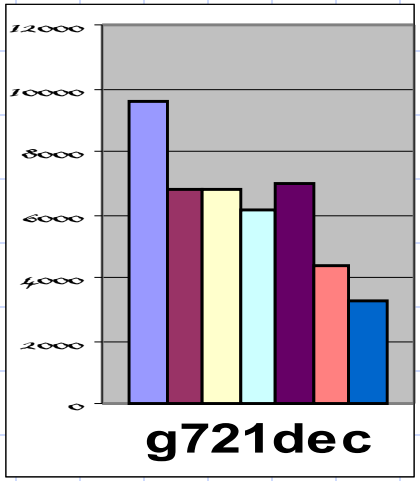
- y-axis is energy in mJ

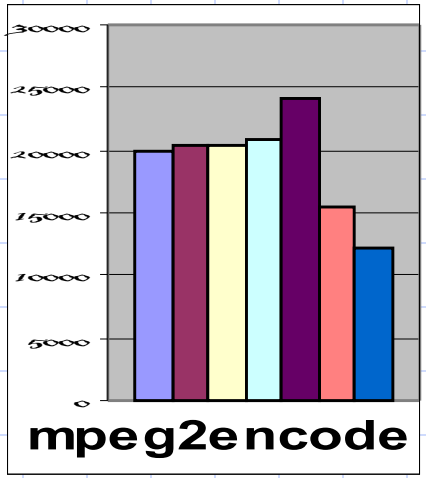










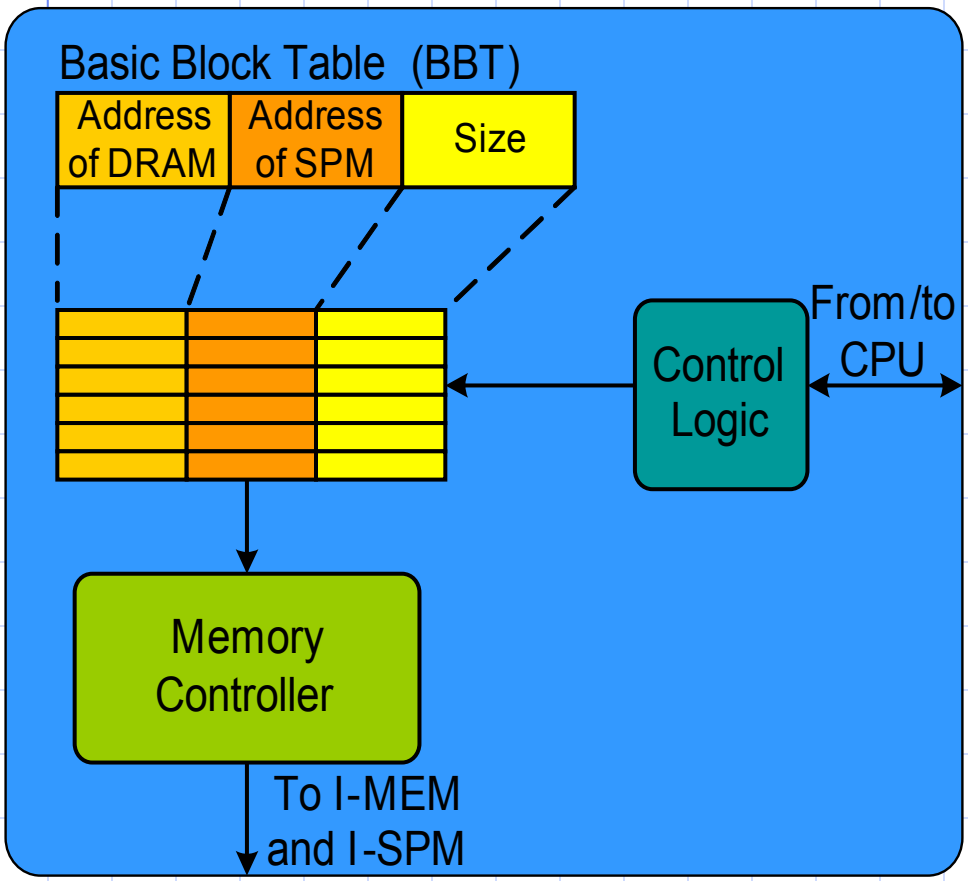


Conclusion

- Utilize scratchpad memory as a replacement of the instruction cache memory.
- Concomitance information is used to select the basic blocks that are to be executed from the scratchpad memory.
- Performance improvement and energy reduction is achieved compared to system with cache memory.

Thank You

Hardware Modification



- SMI initiates the SPM Controller
- BBT stores information of where to copy in I-SPM.