On-the-Fly Load Data Value Tracing in Multicores

Mounika Ponugoti, Amrish K. Tewar, Aleksandar Milenković
University of Alabama in Huntsville, Huntsville, Alabama, U.S.A.

ABSTRACT

Software testing and debugging of modern multicore-based embedded systems is a challenging proposition because of growing hardware and software complexity, increased integration, and tightening time-to-market. To find more bugs faster, software developers of real-time embedded systems increasingly rely on on-chip trace and debug resources, including hefty on-chip buffers and wide trace ports. However, these resources often offer limited visibility of the system, increase the system cost, and do not scale well with a growing number of cores. This paper introduces mlvCFiat, a hardware/software mechanism for capturing and filtering load data value traces in multicores. It relies on first-access tracking in data caches and equivalent modules in the software debugger to significantly reduce the number of trace events streamed out of the target platform. Our experimental evaluation explores the effectiveness of the proposed technique as a function of cache sizes, encoding mechanism, and the number of cores. The results show that mlvCFiat significantly reduces the total trace port bandwidth. The improvements relative to the existing Nexus-like load data value tracing range from 15 to 33 times for a single core and from 14 to 20 times for an octa core.

BACKGROUND AND MOTIVATION

Trends in Embedded Systems

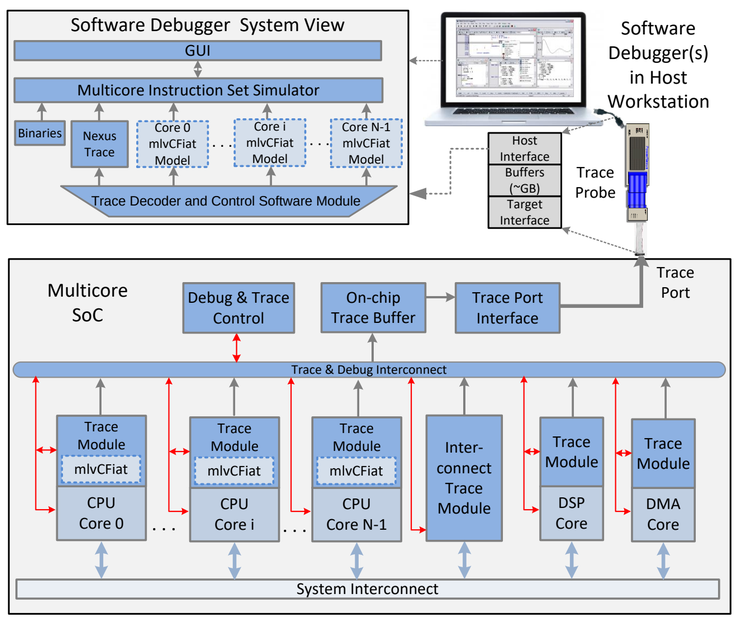
- Indispensable in many aspects of many life
- Trends: more complex and smaller, running sophisticated software
- SW development cost exceeds 80% of the total cost
- Debugging SW is hard and time-consuming: developers spend 50%-75% of time in debugging; increases as we transition to multicores
- Estimated cost of SW bugs and glitches: $20-$60 billion annually
- Need for better tools to help SW/FW developers find bugs faster

Tracing and Debugging Challenges

- What is my system doing now? Limited visibility of internal signals
  - High system complexity + high operating frequencies
  - Limited bandwidth for debugging
- Traditional approach to debugging: stop the processor & examine or change the system state
  - Slow and expensive
  - Not practical to use breakpoints in real-time embedded systems
  - May change the sequence of events on target platform
  - Does not scale well to multicores
- Include dedicated on-chip trace and debug infrastructure

Tracing and Debugging in Multicores

- IEEE Standard: Nexus 5001
  - Class 1: Traditional run-control debugging
  - Class 2: Captures control-flow tracing in near real time
  - Class 3: Captures data tracing in near real time
  - Class 4: Emulating memory and I/O through a trace port



- What do we need to replay a program offline?
  - Instruction Set Simulator + program binary
  - Initial platform state (GPRs + SPRs)
  - Exceptions Trace + *Load Data Value Trace* captured on the platform

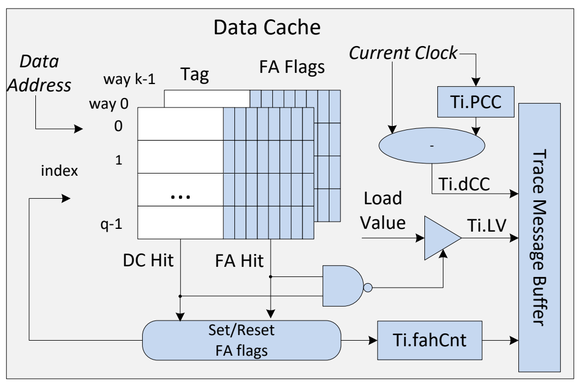
Data Tracing: Why Is It Challenging?

- Metrics
  - $Average\ bpi = \frac{Total\ trace\ size\ in\ bits}{Total\ number\ of\ executed\ instructions}$
  - $Average\ bpc = \frac{Total\ trace\ size\ in\ bits}{Execution\ time\ measured\ in\ clock\ cycles}$

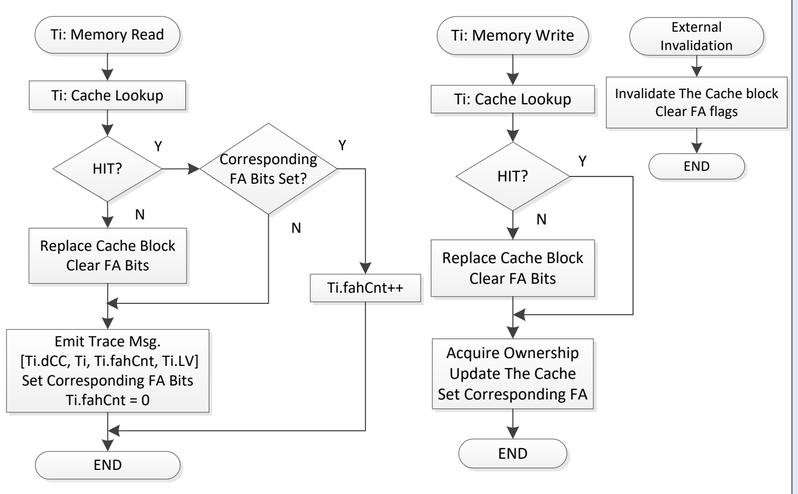
mlvCFiat

- mlvCFiat – multicore load value cache first access tracking
  - Hardware/software mechanism for filtering load data value traces
- Target platform: L1D caches are extended to include first-access tracking bits (FA bits)
- Software debugger maintains software copies of L1D caches
  - Same updating policies, same cache organization as in hardware
  - Instruction set simulator requires load data values from the target platform only for reads that cannot be inferred

Hardware Structures on Target Platform

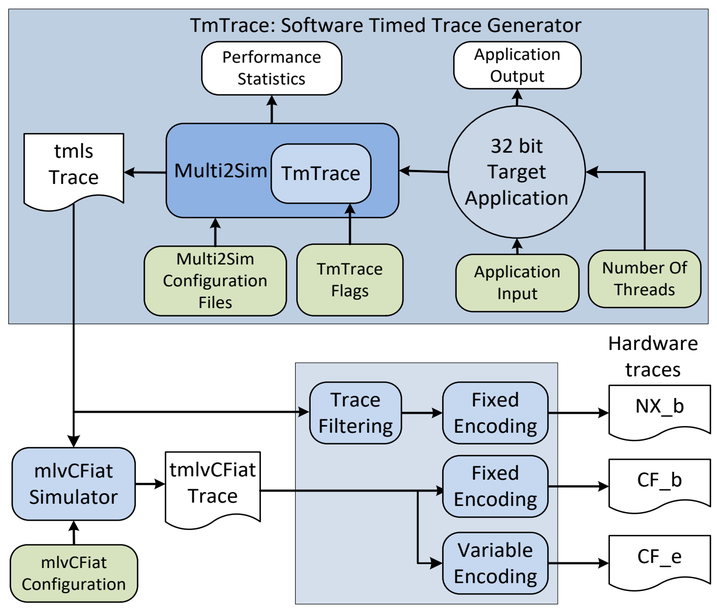


Operations on Target Platform

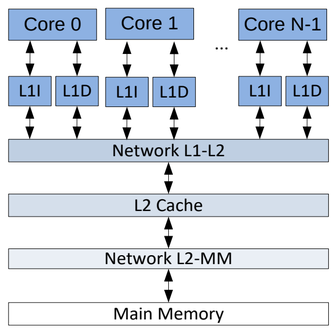


Operations in Software Debugger

EXPERIMENTAL ENVIRONMENT



Multicore Model

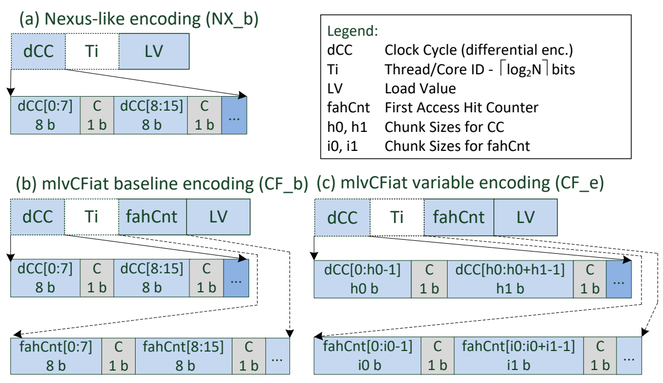


CS16:
L1D/L1I cache size: 16 KB
L2 cache size: N*64 KB

CS32:
L1D/L1I cache size: 32 KB
L2 cache size: N*128 KB

L1D/L1I hit time: 4 cc
L1D/L1I associativity: 4-way
L2 hit time: 12 cc
L2 associativity: 16-way
Cache block size: 32 B
First-access granularity: 4 B
Memory latency: 100 cc

Format of Trace Messages

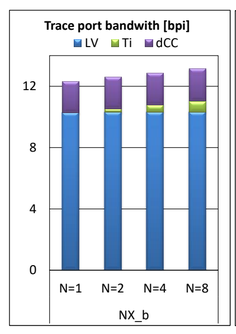


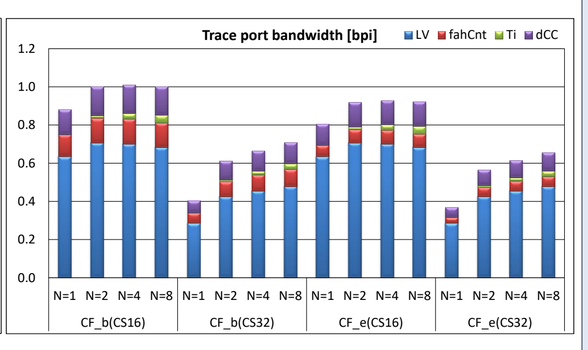
- In mlvCFiat length of LV field depends on granularity
- Encoding parameters (h0,h1) = (4, 2), (i0, i1) = (2, 2)

RESULTS

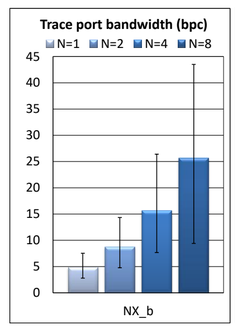
Trace Port Bandwidth [bpi]

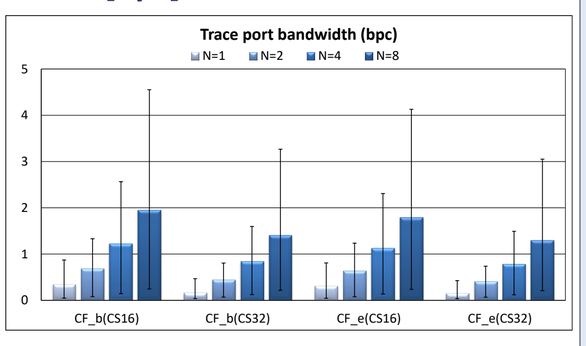
# Cores	N=1			N=2			N=4			N=8		
Mechan	NX_b	CF_e	CF_e	NX_b	CF_e	CF_e	NX_b	CF_e	CF_e	NX_b	CF_e	CF_e
Config	-	CS16	CS32	-	CS16	CS32	-	CS16	CS32	-	CS16	CS32
barnes	15.03	2.21	0.79	15.31	2.30	1.16	15.59	2.39	1.47	15.86	2.44	1.77
cholesky	15.35	1.86	0.75	16.21	1.30	0.79	15.87	0.95	0.62	15.59	0.61	0.44
fft	10.65	2.57	1.48	10.84	2.62	1.50	11.02	2.65	1.52	11.19	2.67	1.54
fmm	8.82	0.36	0.23	8.96	0.37	0.24	9.14	0.38	0.25	9.33	0.38	0.26
lu	11.88	0.58	0.57	12.07	0.61	0.57	12.27	0.62	0.58	12.47	0.65	0.45
radiosity	12.11	0.25	0.09	12.36	0.55	0.45	12.59	0.56	0.44	12.58	0.63	0.54
radix	13.41	0.75	0.54	13.75	1.64	1.44	14.09	1.73	1.52	14.54	1.79	1.57
raytrace	15.17	1.06	0.34	15.45	1.28	0.61	15.73	1.38	0.71	16.01	1.54	0.90
water-ns	10.64	0.49	0.22	10.81	0.52	0.25	10.98	0.56	0.39	11.15	0.56	0.42
water-sp	11.38	0.07	0.05	11.55	0.07	0.06	11.73	0.08	0.07	11.90	0.09	0.08
Total	**12.34**	**0.80**	**0.37**	**12.63**	**0.92**	**0.57**	**12.89**	**0.93**	**0.61**	**13.17**	**0.92**	**0.66**





Trace Port Bandwidth [bpc]





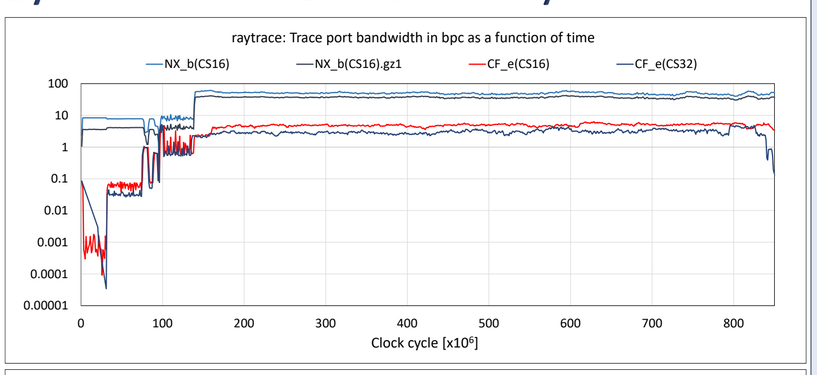
Speedup TPB(NX_b)/TPB(compressed NX_b)

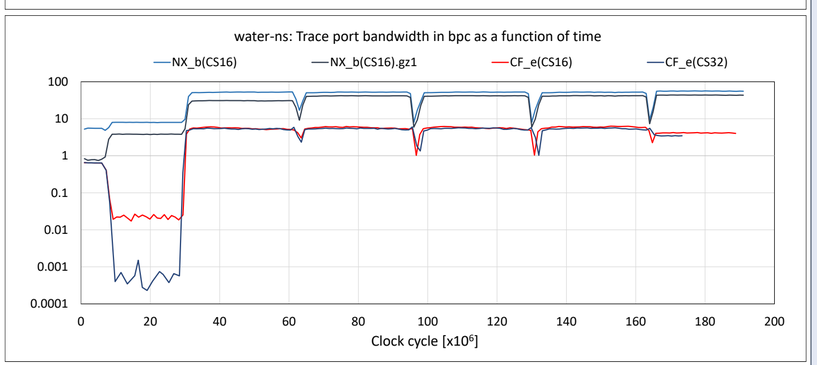
# Cores	N=1		N=2		N=4		N=8	
Config	Split	Unified	Split	Unified	Split	Unified	Split	Unified
barnes	2.10	1.38	1.78	1.30	1.66	1.24	1.58	1.27
cholesky	6.73	1.74	3.85	1.67	3.53	1.85	4.13	2.50
fft	1.93	1.39	1.79	1.36	1.68	1.30	1.66	1.37
fmm	4.95	1.95	3.73	1.85	3.06	1.58	2.75	1.59
lu	5.93	1.56	3.58	1.54	3.14	1.42	3.06	1.77
radiosity	3.86	1.63	2.50	1.54	2.10	1.37	1.95	1.48
radix	4.23	2.02	3.05	1.81	2.08	1.46	1.96	1.42
raytrace	3.88	1.51	2.61	1.47	2.26	1.32	2.08	1.38
water-ns	2.69	1.41	2.08	1.38	1.94	1.27	1.87	1.35
water-sp	3.03	1.37	2.40	1.36	2.11	1.26	2.02	1.39
Total	**3.33**	**1.54**	**2.52**	**1.49**	**2.21**	**1.38**	**2.18**	**1.52**

Speedup TPB(NX_b)/TPB(CF_e)

# Cores	N=1		N=2		N=4		N=8	
Config	CS16	CS32	CS16	CS32	CS16	CS32	CS16	CS32
barnes	6.8	19.1	6.7	13.1	6.5	10.6	6.5	9.0
cholesky	8.2	20.4	12.5	20.4	16.7	25.5	25.7	35.1
fft	4.2	7.2	4.1	7.2	4.2	7.3	4.2	7.3
fmm	24.2	37.8	24.1	37.2	24.2	36.7	24.4	36.6
lu	20.5	20.7	19.8	21.3	19.8	21.2	19.2	27.6
radiosity	47.7	128.8	22.3	27.6	22.6	28.4	19.8	23.4
radix	17.9	24.8	8.4	9.6	8.1	9.3	8.1	9.3
raytrace	14.3	44.2	12.1	25.4	11.4	22.1	10.4	17.9
water-ns	21.7	47.4	20.8	42.8	19.7	28.0	20.0	26.6
water-sp	168.1	210.3	158.5	189.4	147.3	170.9	136.9	156.2
Total	**15.3**	**33.4**	**13.7**	**22.3**	**13.9**	**21.0**	**14.3**	**20.1**

Dynamic Trace Port Bandwidth Analysis





CONCLUSIONS

- Need for trace modules that can guarantee
  - Unobtrusive program tracing in real-time
  - High compression (low trace port bandwidth)
  - Low complex implementation: narrow trace ports & small buffers
- mlvCFiat
  - Trace out only data cache read misses or read hits that occur for the first time
  - Relatively low-complexity
  - Improvement: 15-33 times when N=1 & 14-20 times when N=8
- Variable encoding improves TPB 8-9% relative to base encoding

ACKNOWLEDGMENT

This work is supported in part by an NSF grant CNS-1217470.