UNIVERSITY *of* CALIFORNIA · IRVINE

Application Specific Compression Scheme for Fast Checkpointing on FPGAs

Ting-Shuo Chou, Chen Huang, Bailey Miller,
Tony Givargis, and Frank Vahid
{tingshuc, givargis}@uci.edu
{chuang, bmillier, vahid}@cs.ucr.edu

ASP-DAC 2013

Outline

- Motivation
- Problem Statement
- Approaches
- Experimental Results
- Conclusions

Cyber-Physical Systems

Q:What is a cyber-physical system?

A: a system with integrations of computation and physical processes

Smart Car

Intelligent Transport System

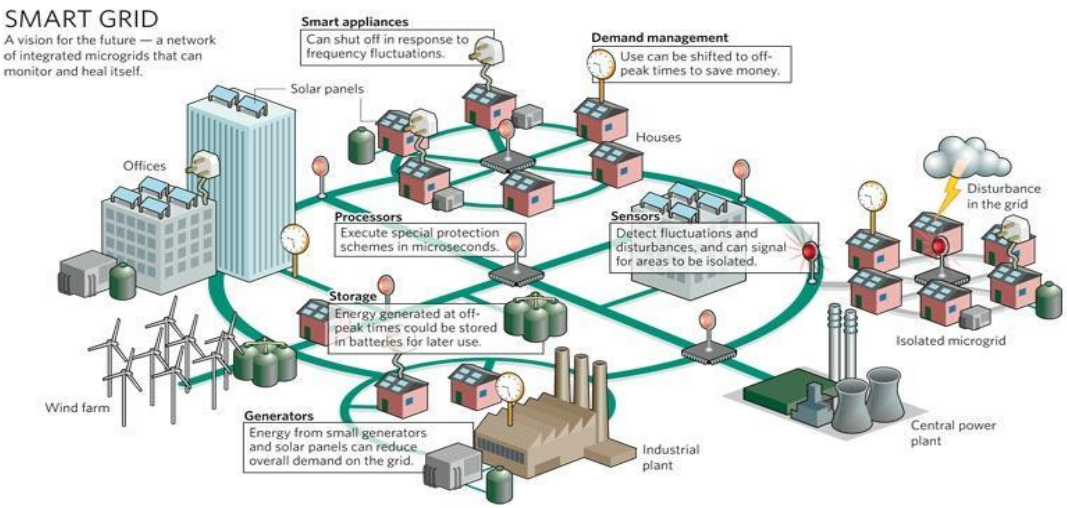
Cyber-Physical Systems

Q:What is a cyber-physical system?

A: a system with integrations of computation and physical processes

Smart Bridge

Smart Grid



Cyber-Physical Systems

Q:What is a cyber-physical system?

A: a system with integrations of computation and physical processes

ASIMO, Service Robot

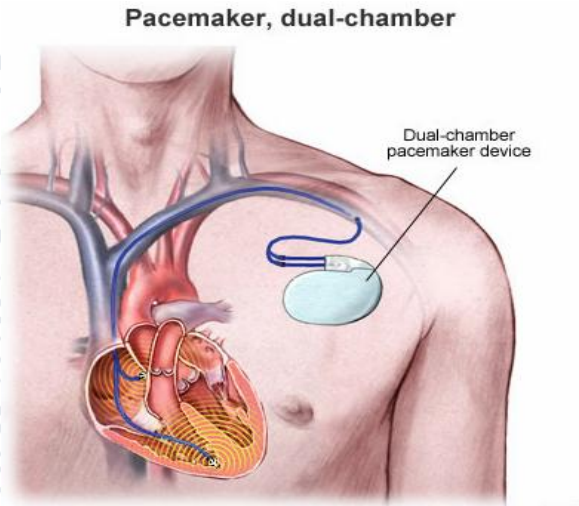
Paro, Therapeutic Robot

Roomba, Household Robot

Cyber-Physical Systems

Q:What is a cyber-physical system?

A: a system with integrations of computation and physical processes

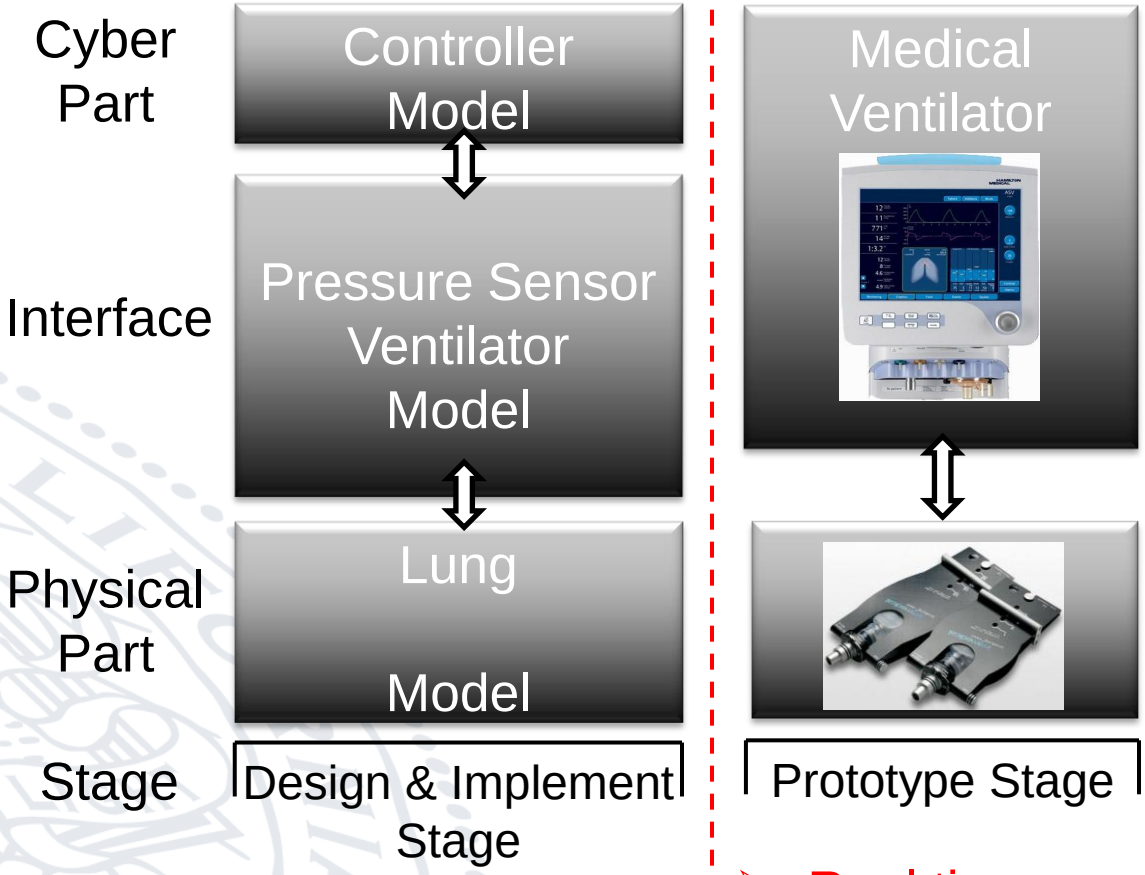


Pacemaker

Medical Ventilator

Infusion Pump

CPS Design and Validation in Medical Area

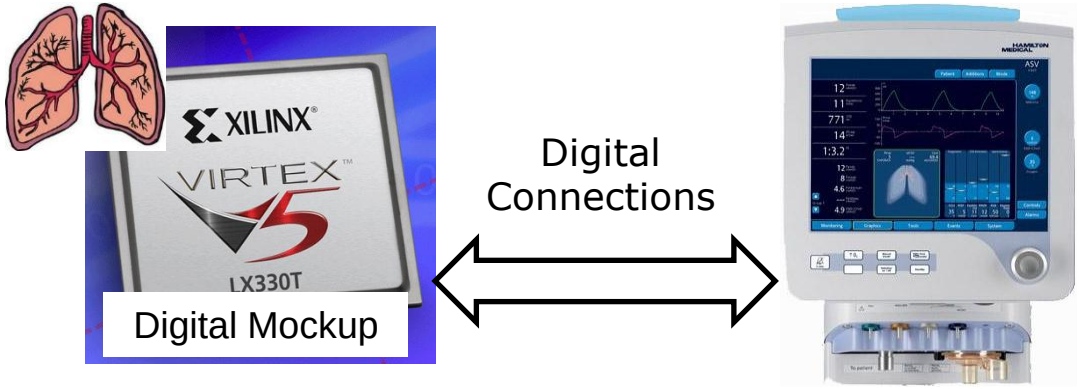


- Real time constraint

References:

- B. Miller, F. Vahid, T. Givargis, "Digital Mockups for the Testing of a Medical Ventilator," ACM SIGHIT Symposium on International Health Informatics (IHI), 2012
- Z. Jiang, M. Pajic, A. T. Connolly, S. Dixit, and R. Mangharam, "Real-time Heart Model for Implantable Cardiac Device Validation and Verification," IEEE Euromicro Conference on Real-Time Systems (ECRTS), 2010

Digital Mockup of Human Lung



- We run lung model on a FPGA in real time
- Advantages:
  - Configurability: adjust parameters of the lung (child and old man)
  - Observability: watch pressure at any point in the lung
  - Time-controllability: move time of the lung model back or forth

References:

- B. Miller, F. Vahid, T. Givargis, "MEDS: Mockup Electronic Data Sheets for Automated Testing of Cyber-Physical Systems Using Digital Mockup," Design Automation and Test in Europe, 2012

Outline

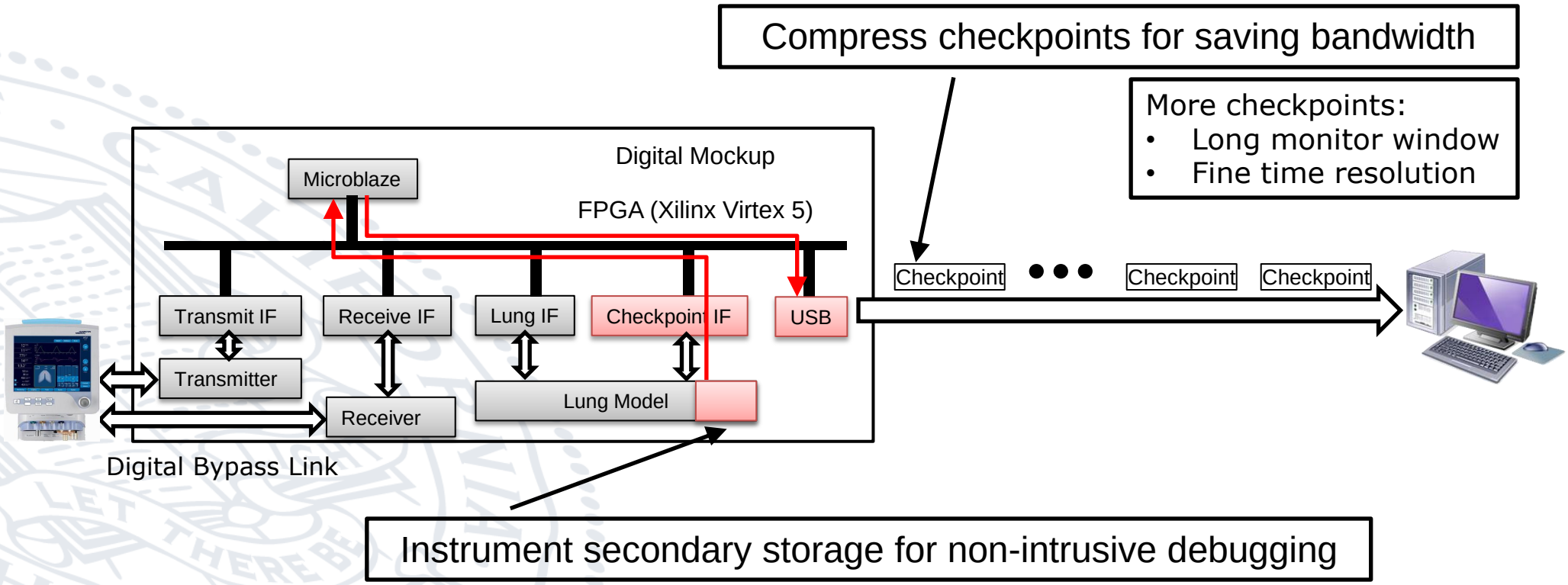
- Motivation
- Problem Statement
- Approaches
- Experimental Results
- Conclusions

Problem Statement

- Problem: Controllability of lung models and digital mockups in general
- Constraint: The lung model runs in real time => it can't be suspended
- Goal: non-intrusive debugging and time-controllability
  - Time-controllability: manipulate temporal state (i.e., time) of the lung model
  - Non-intrusive debugging: not change timing behavior of a digital mockup
- How to do it:
  - We use checkpoints:
    - The FPGA's internal state (contents in RAM or reg.) at a certain clock cycle
  - We manipulate checkpoints through
    1. Instrumenting Secondary Storage (Memory)
    2. Developing Compression Scheme

Testing and Debugging Environment

The digital mockup continuously sends checkpoints to the host machine



Outline

- Motivation
- Problem Statement
- Approaches
  - Weibel Lung Model
  - Design Instrumentation
  - Compression Scheme
- Experimental Results
- Conclusions

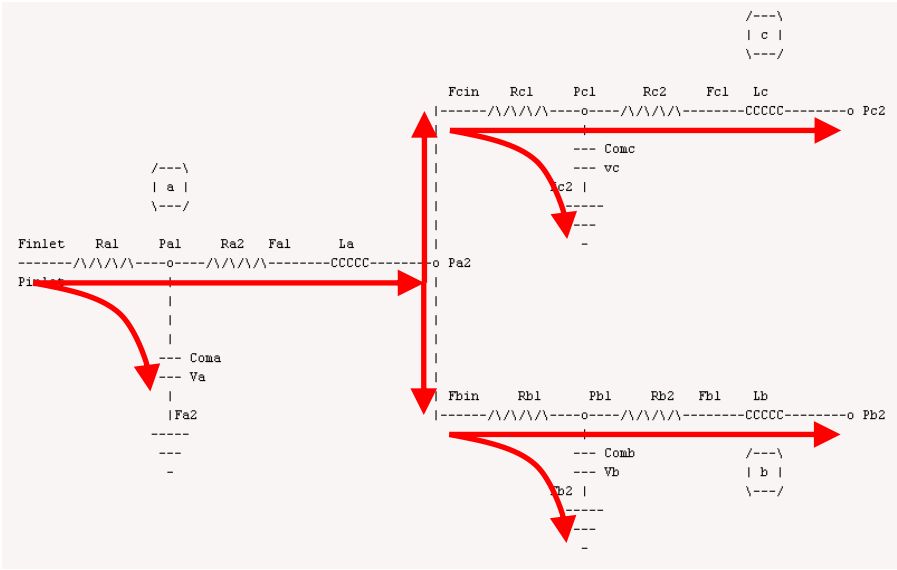
Weibel Lung Model: Equations

- Weibel Lung Model is similar to RC circuit
  - Pressure ⇔ Voltage
  - Flow ⇔ Current
  - Volume ⇔ Electronic Charges

ODEs:

Pa1 = Va/Com_a
Pinlet = (Finlet * Ra1) + Pa1
d(Va)/dt = Finlet - Fa1
d(Fa1)/dt = (Pa1 - Pa2 -(Fa1 * Ra2))/La
Pb1 = Vb/Com_b
Pa2 = (Fbin * Rb1) + Pb1
Fa1 = Fcin + Fbin
d(Fb1)/dt = (Pb1 - Pb2 -(Fb1 * Rb2))/Lb
d(Vb)/dt = Fbin - Fb1
Pc1 = Vc/Com_c
Pa2 = (Fcin * Rc1) + Pc1
d(Fc1)/dt = (Pc1 - Pc2 -(Fc1 * Rc2))/Lc
d(Vc)/dt = Fcin - Fc1

2 levels Weibel Lung Model

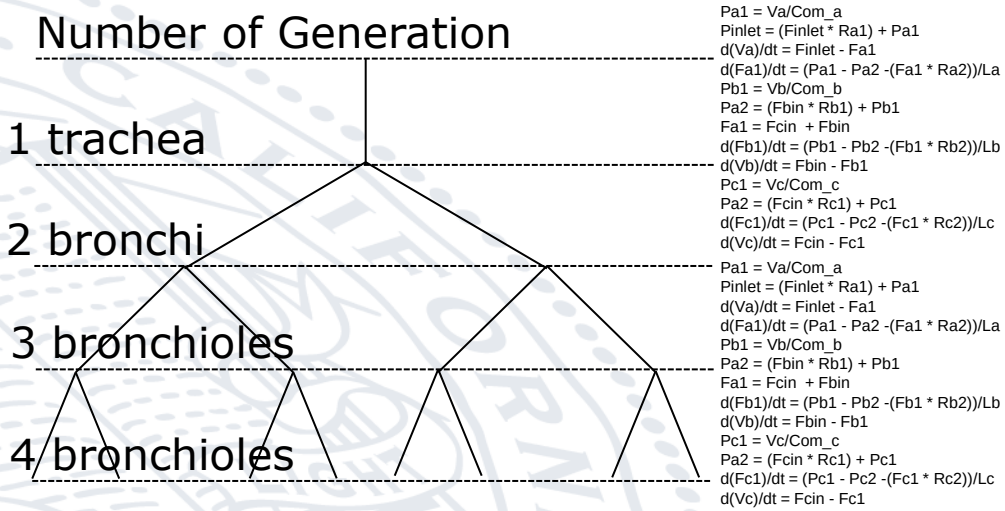


References:

- P. Barbini, et al., "A Dynamic Morphometric Model of the Normal Lung for Studying Expiratory Flow Limitation in Mechanical Ventilation," *Annals of Biomedical Engineering*, Vol. 33, No. 4, May 2005
- P. Barbini, et al., "A Simulation Study of Expiratory Flow Limitation in Obstructive Patients during Mechanical Ventilation," *Annals of Biomedical Engineering, Vol. 34, No. 12, December 2006*
- Electrical circuit for a bifurcating airway (http://www.physiome.org)

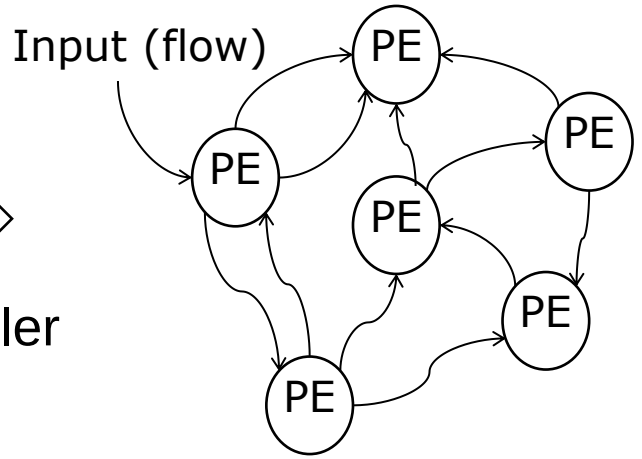
Weibel Lung Model: Mapping

- Process Element: the processing unit that solves ODEs
- Thousands of ODEs are compiled into VHDL code by PE compiler



4 levels Weibel Lung Model

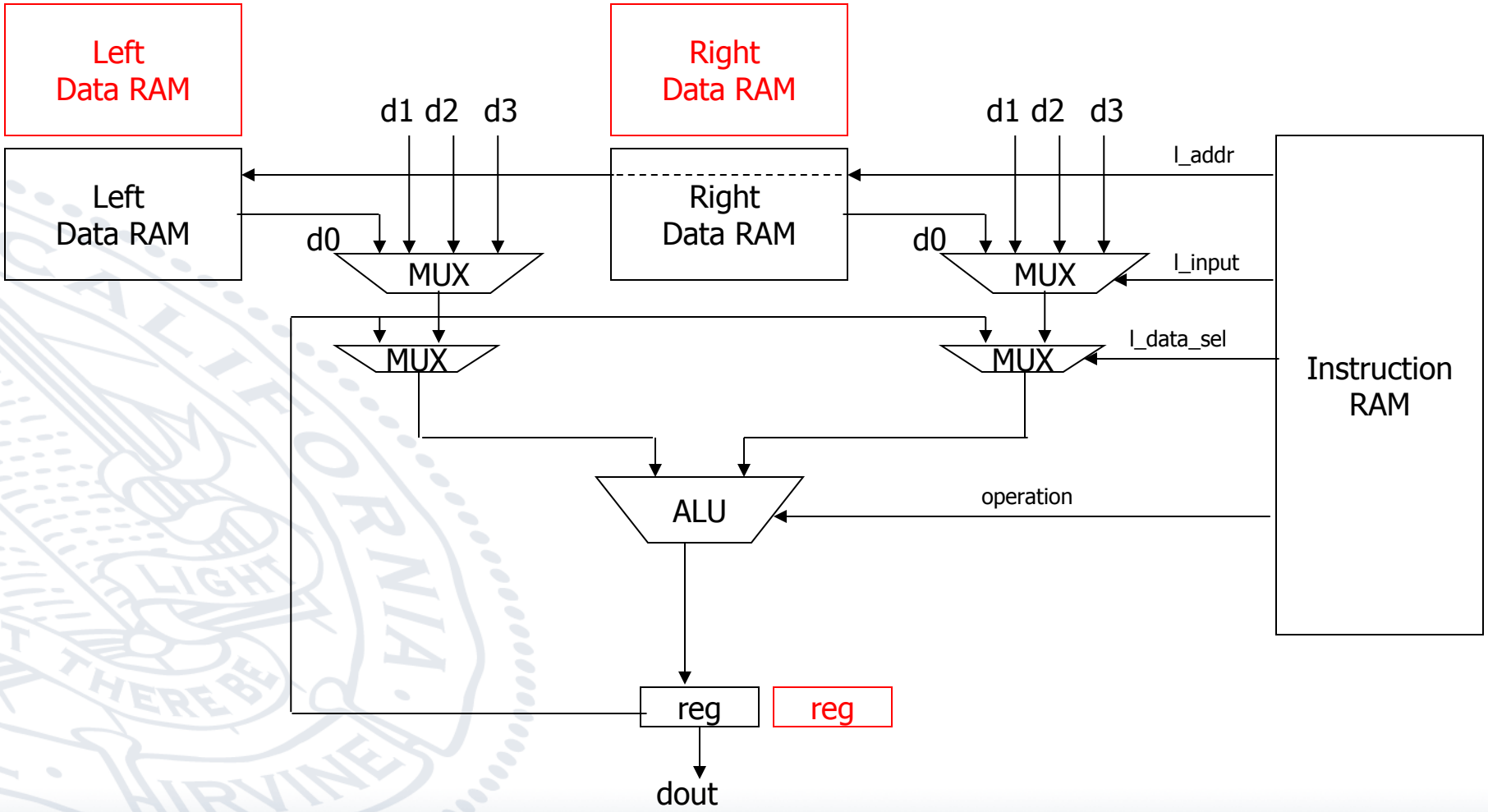
PE compiler



References:

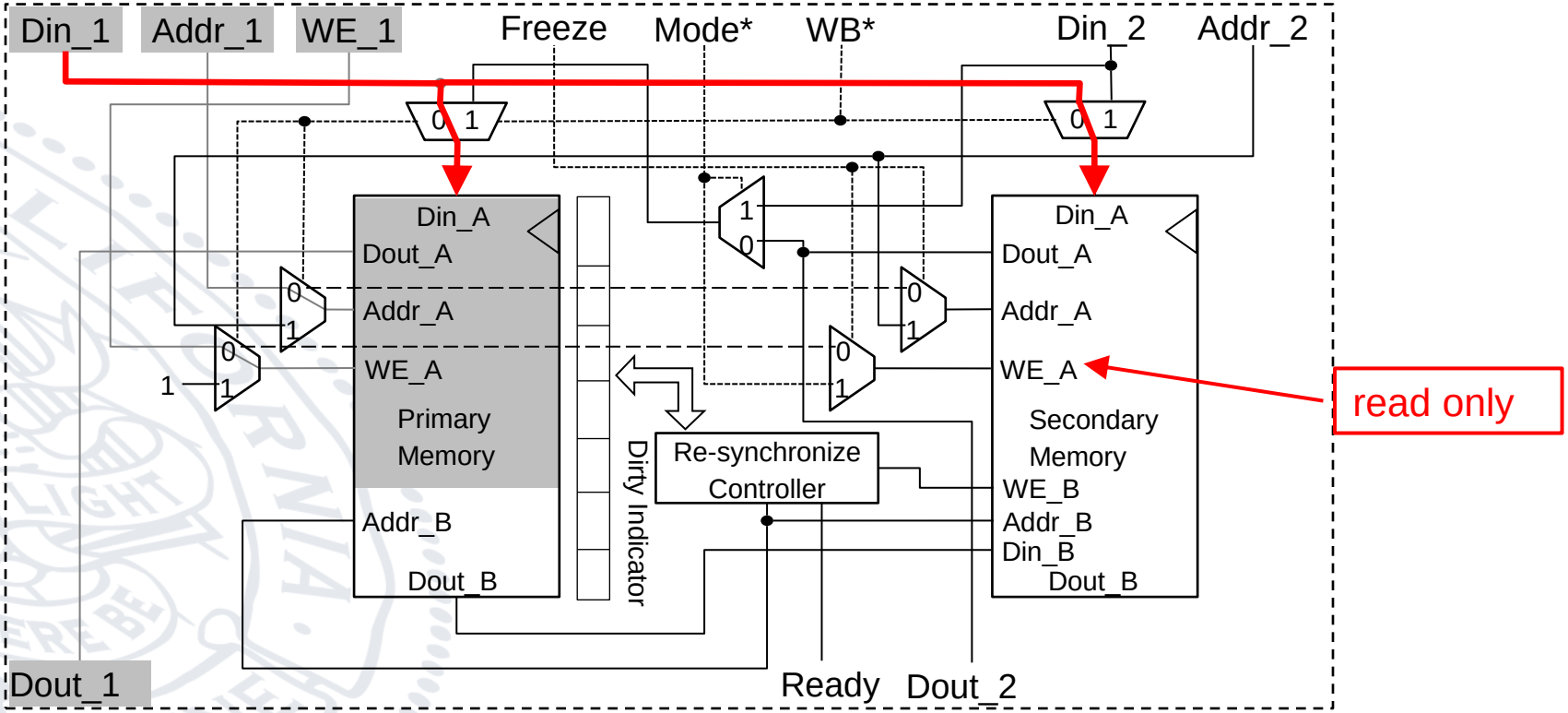
- C. Huang, F. Vahid, and T. Givargis, "A custom FPGA processor for physical model differential equation solving," Embedded Systems Letters, 2011

Design Instrumentation at PE Level



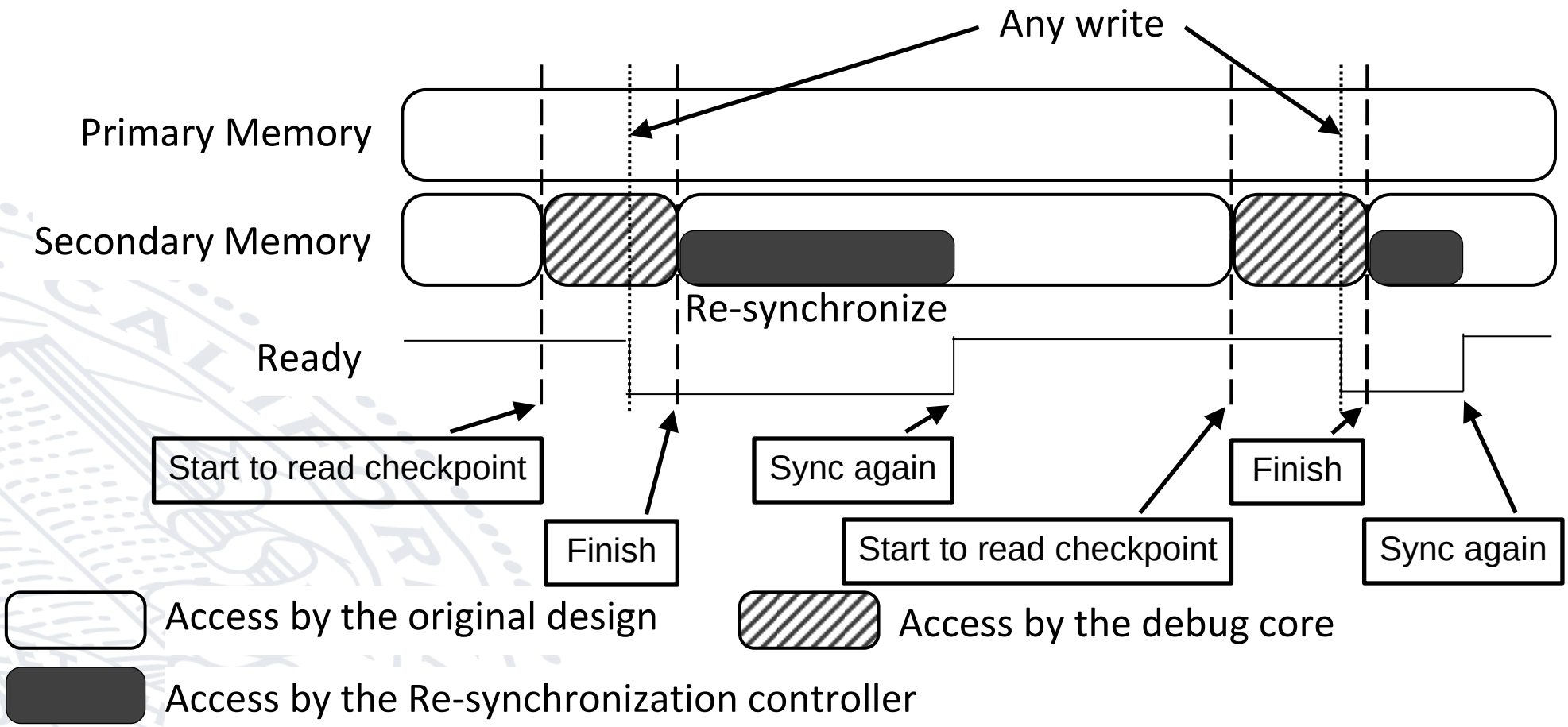
Design Instrumentation

Freeze	WB*	Mode*	Description
0	0	0	Normal use. Data at Din_1 are duplicated
1	0	0	Read data in secondary memory
1	1	0	Write back using data in secondary memory
1	1	1	Write back using data at Din_2



Note: no copy operation is required

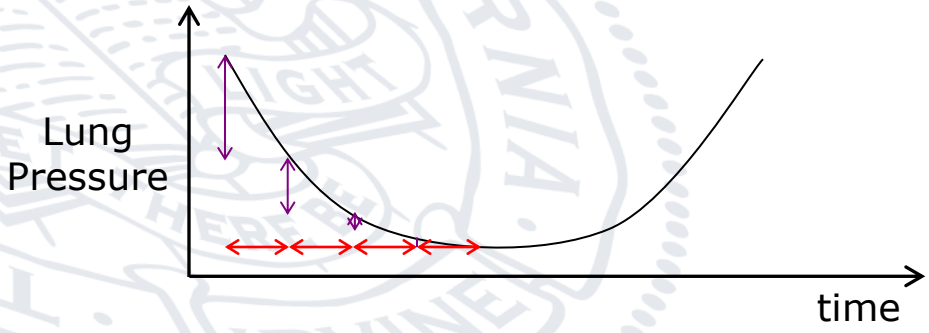
Access Timing Diagram

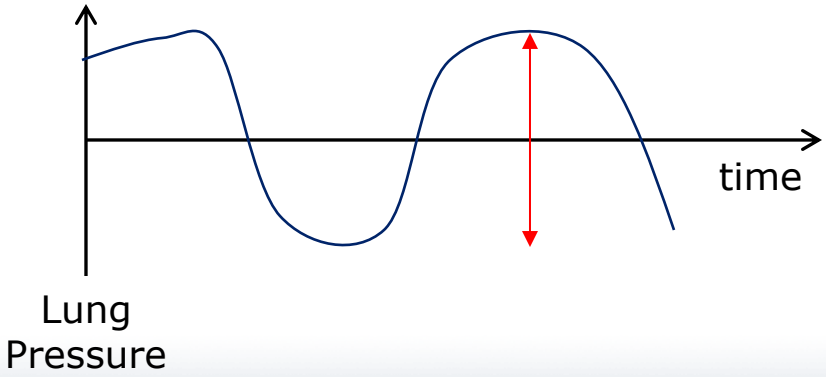


The primary memory is never accessed by a debug core and re-synchronization controller

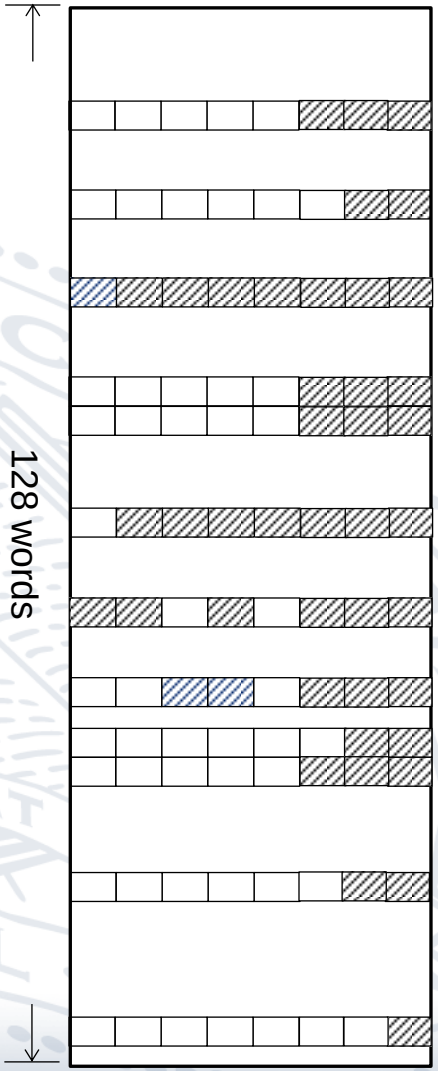
Compression Scheme

- Our compression scheme is application-specific
- Our compression scheme uses data differencing approach
  - use dirty flag to track dirty data
- Observations:
  - the values generated by ODE solvers does not change too much over time
  - these values are restricted to be within a certain range in the case of lung or heart mode





How to Encode the Differences

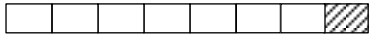


How about Dirty Word Tracking (DWT)?

7 index bits + 1 unchanged nibble + 7 dirty nibble = Good deal

7 index bits + 7 unchanged nibble + 1 dirty nibble = Bad deal

How about Dirty Nibble Tracking (DNT)?



10 index bits + 1 dirty nibble = Not so good, average

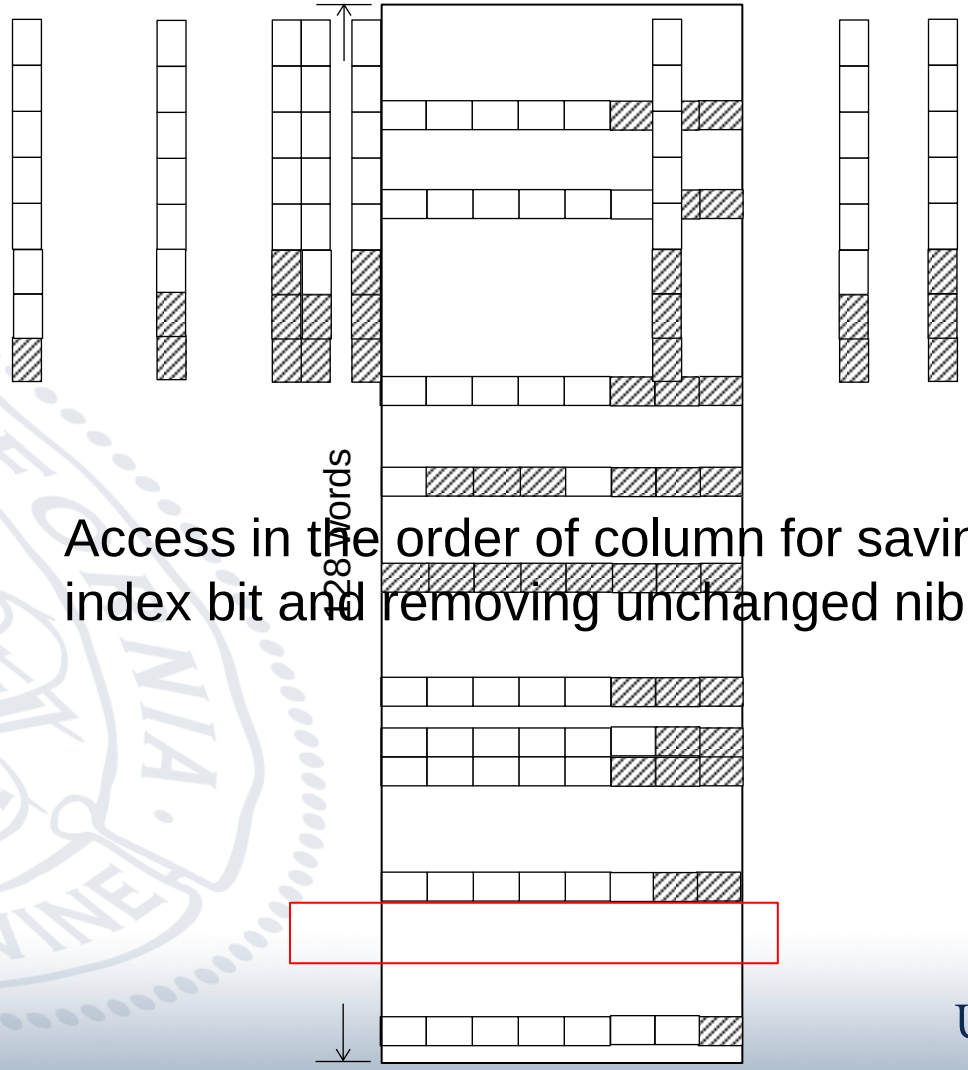
50 index bits + 5 dirty nibble = Not so good

70 index bits + 7 dirty nibble = Not so good

How about hybrid encoding method?

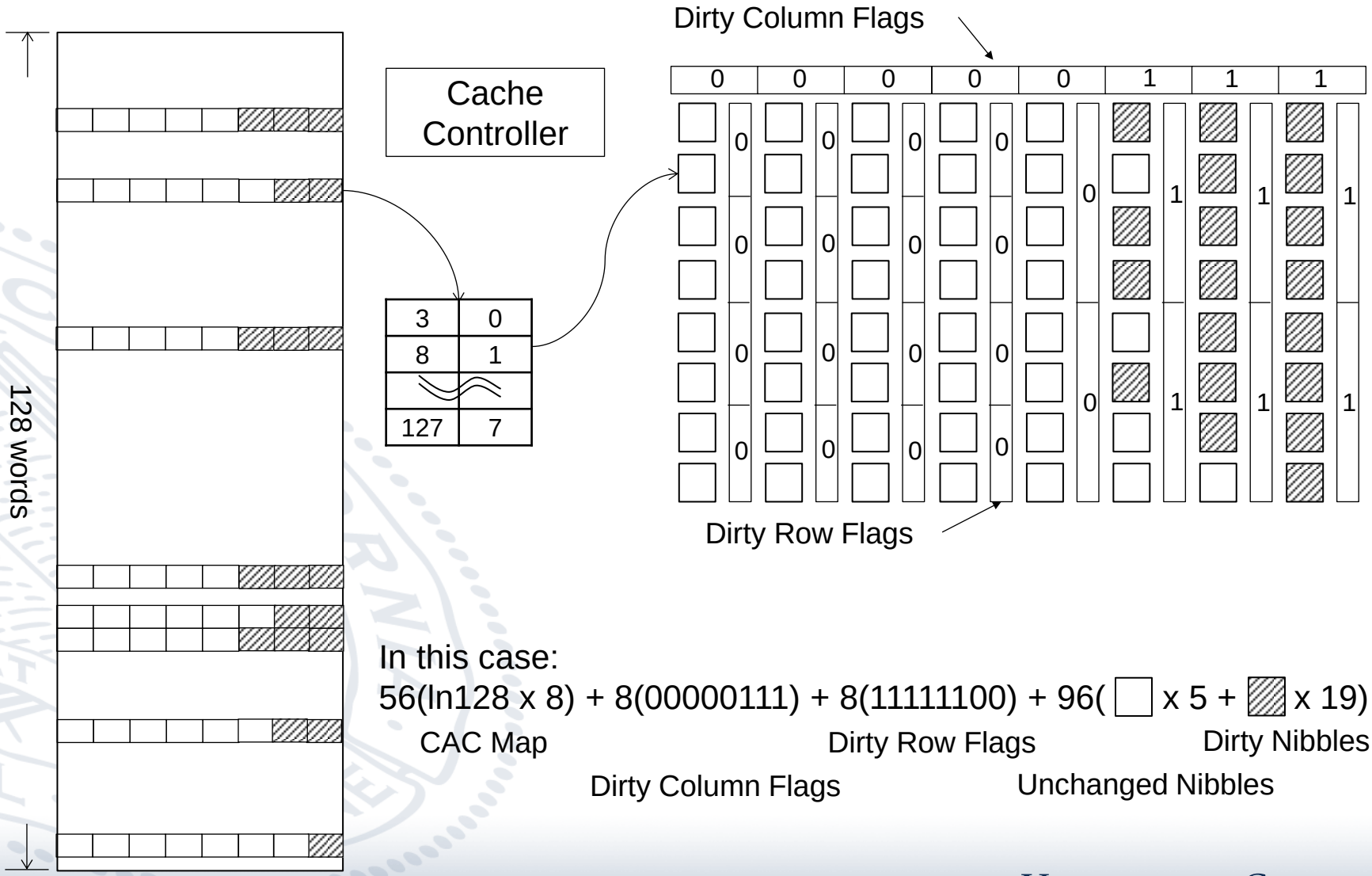
Hybrid Encoding Method

Idea: pack the words with three dirty nibbles at LSB and access the data in the order of columns



Access in the order of column for saving index bit and removing unchanged nibbles

How does Column Accessible Cache Works



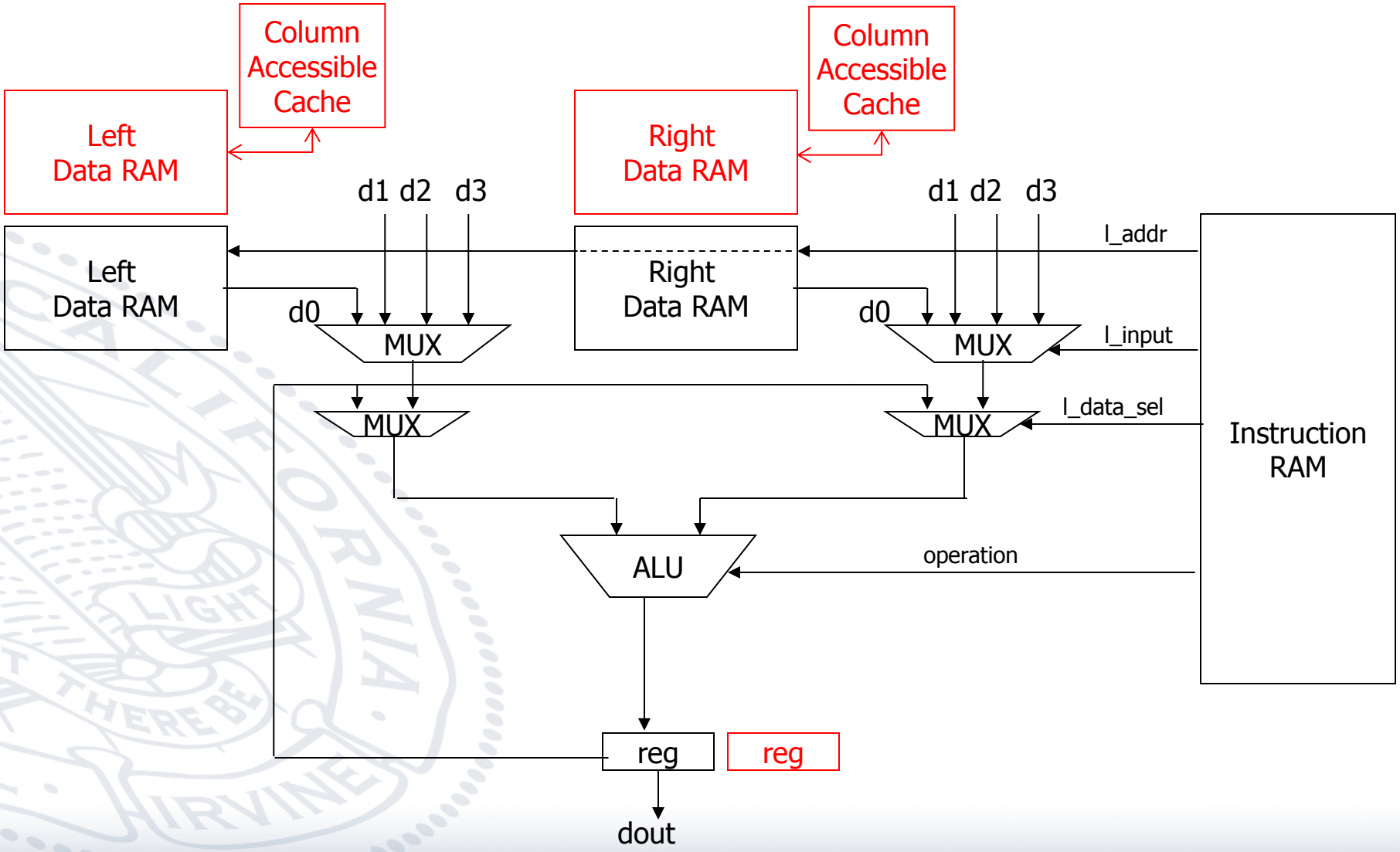
In this case:

56(ln128 x 8) + 8(00000111) + 8(11111100) + 96(☐ x 5 + ▨ x 19) = 168 bits

CAC Map — Dirty Row Flags — Dirty Nibbles

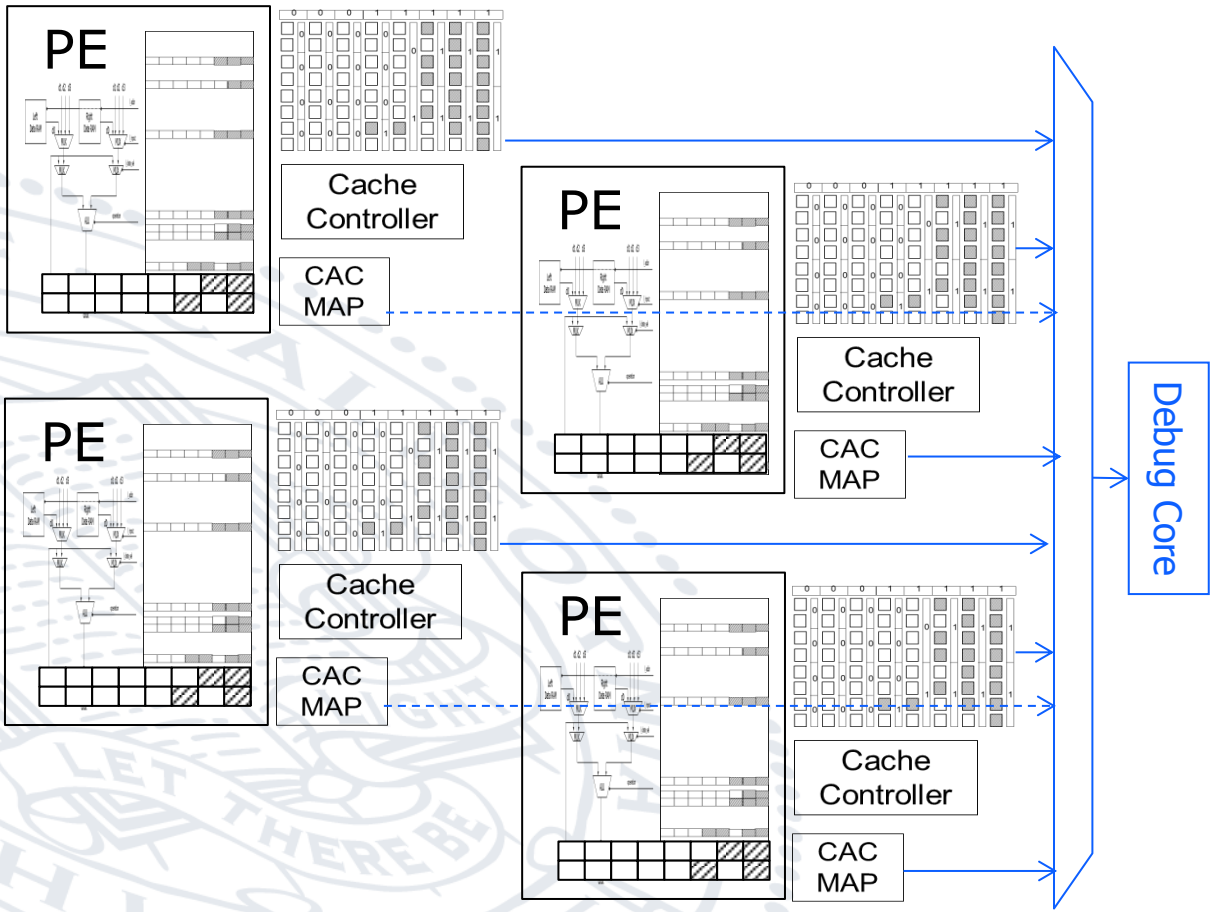
Dirty Column Flags — Unchanged Nibbles

Where are Column Accessible Caches

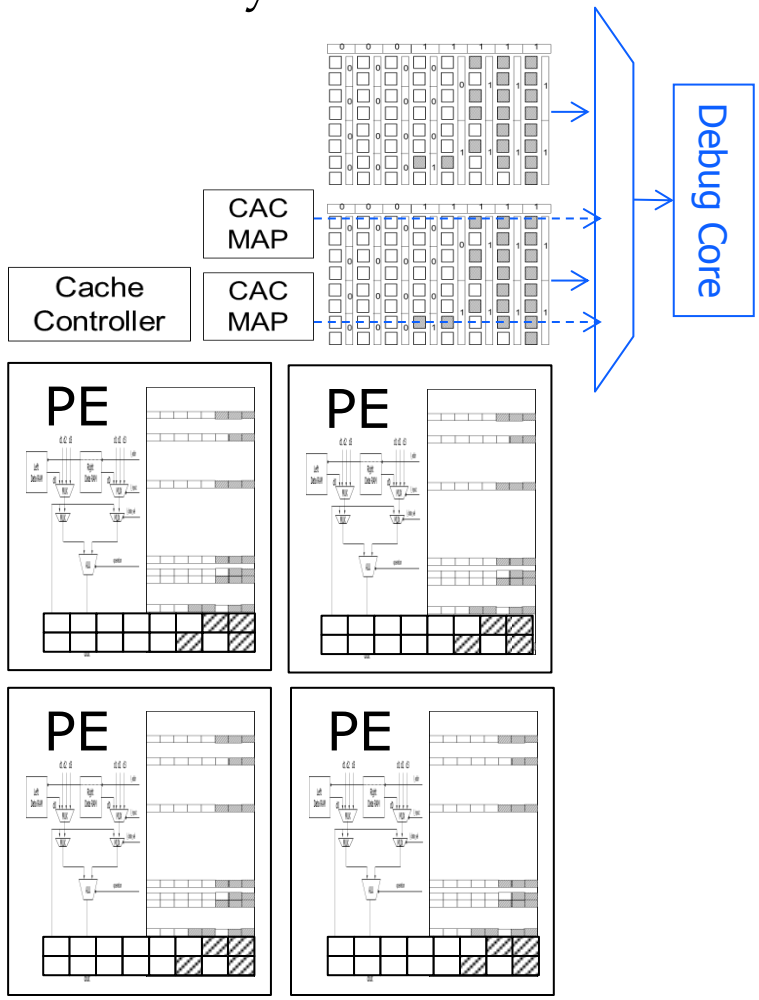


Distributed CACs and Global CAC

Intuitively, we'll have better compression rate when CAC is fully filled with entries



Distributed CACs



Global CAC

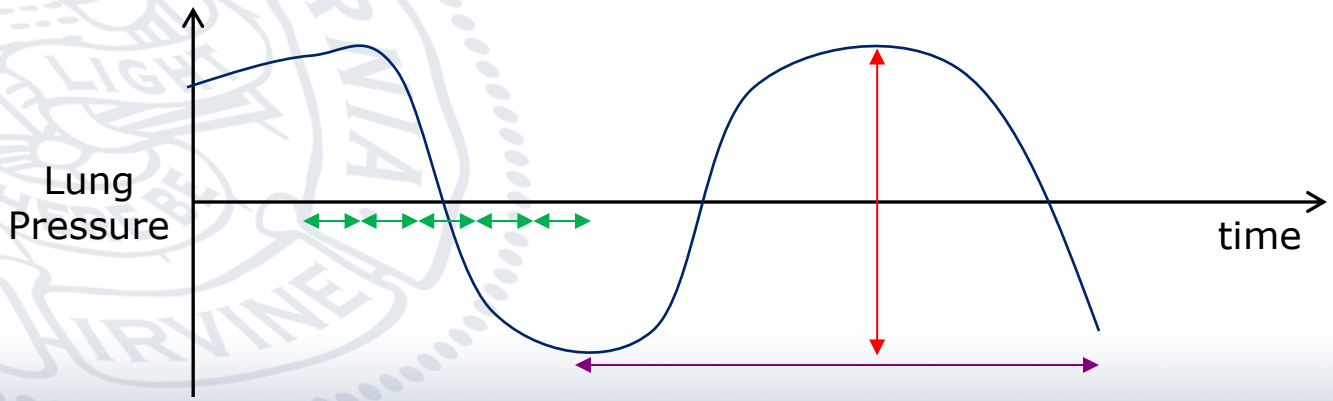
Outline

- Motivation
- Problem Statement
- Approaches
- Experimental Results
- Conclusions

Simulation Environment

To explore the benefits of CACs, we build a customized simulator that cycle-accurately simulates PEs and their network.

Input parameter	Possible value
lung model	we4_pe4, we6_pe8 we8_pe16, we10_pe196
input pressure	-500 ~ 500 mmHg
input oscillation rate	10 ~ 20 times / minute
input shape	square and sine
interval between checkpoints	10 ~ 1000 ms



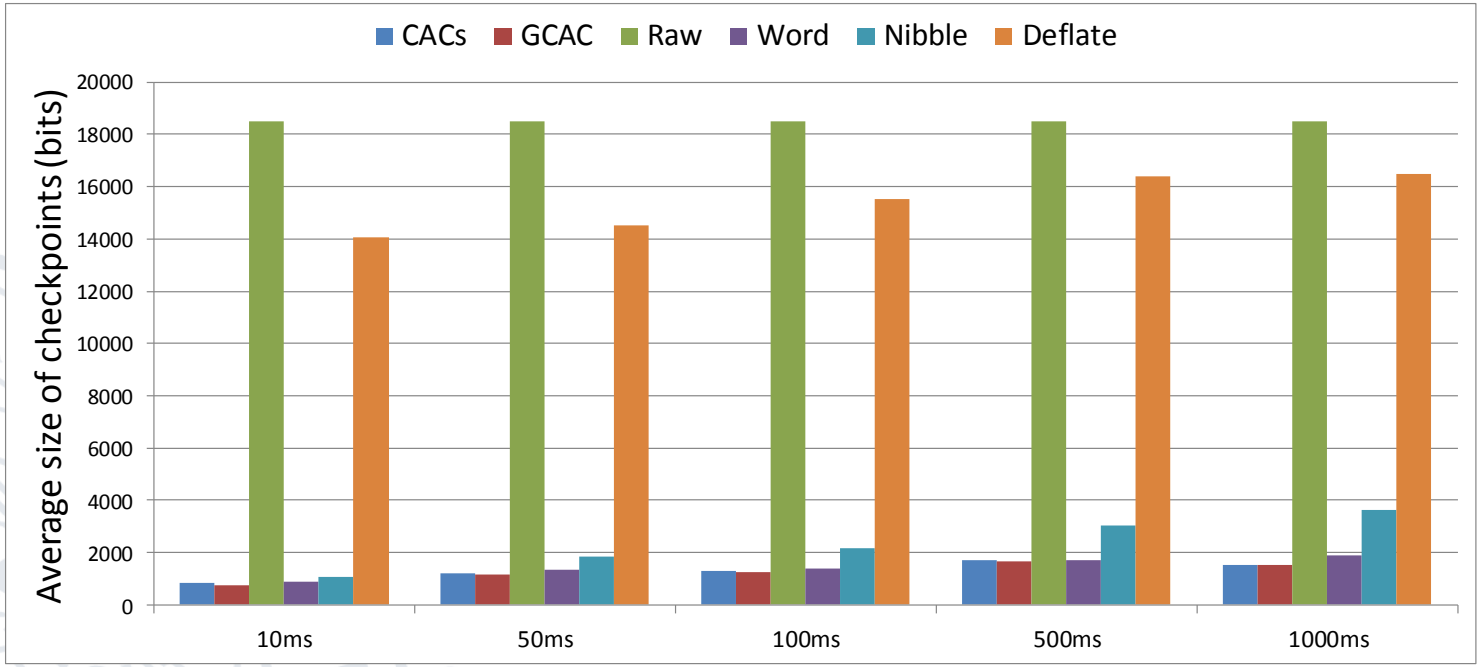
Different Approaches for Comparison

Raw	No compression
Deflate	Compressed by Deflate
DWT	Dirty Word Tracking
DNT	Dirty Nibble Tracking
CACs	Distributed Column Accessible Cache (CACs)
GCAC	Global Column Accessible Cache (CAC)

Note: A checkpoint includes not only the differenced data but also their index

Experimental Results I

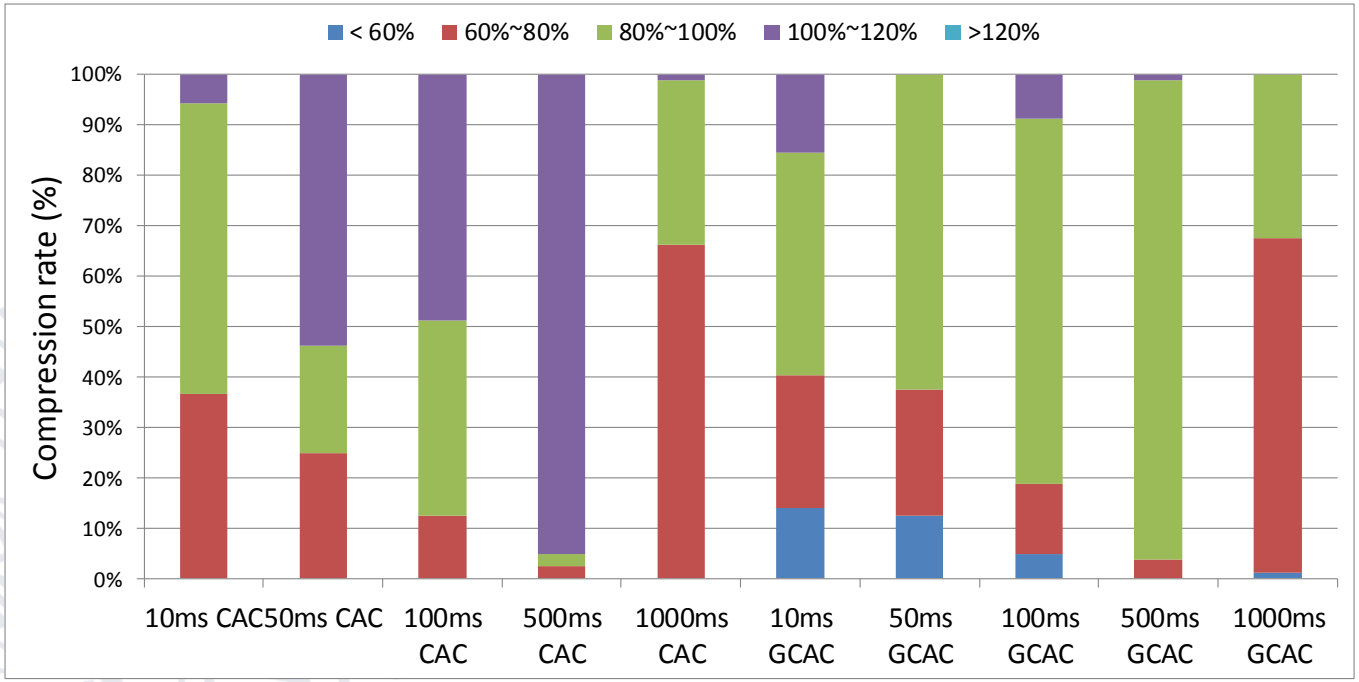
Average size of checkpoints



- we6_pe8 and 15 times/second are used
- We can see the data differencing approaches (i.e., CACs, GCAC, DWT, and DNT) are much better than general compression algorithm in our application

Experimental Results II

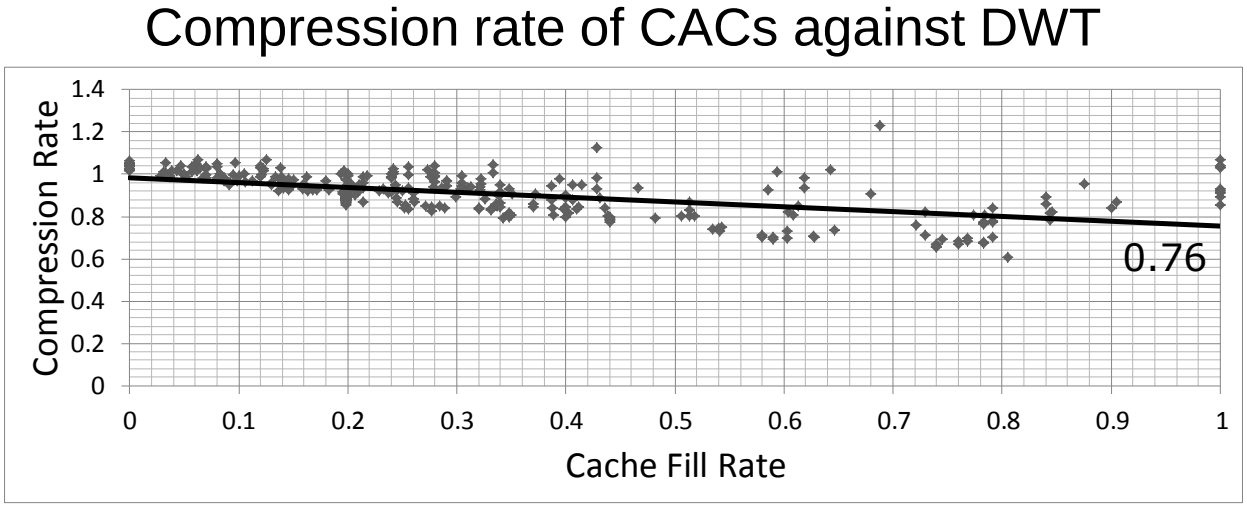
Compression rate distribution against DWT

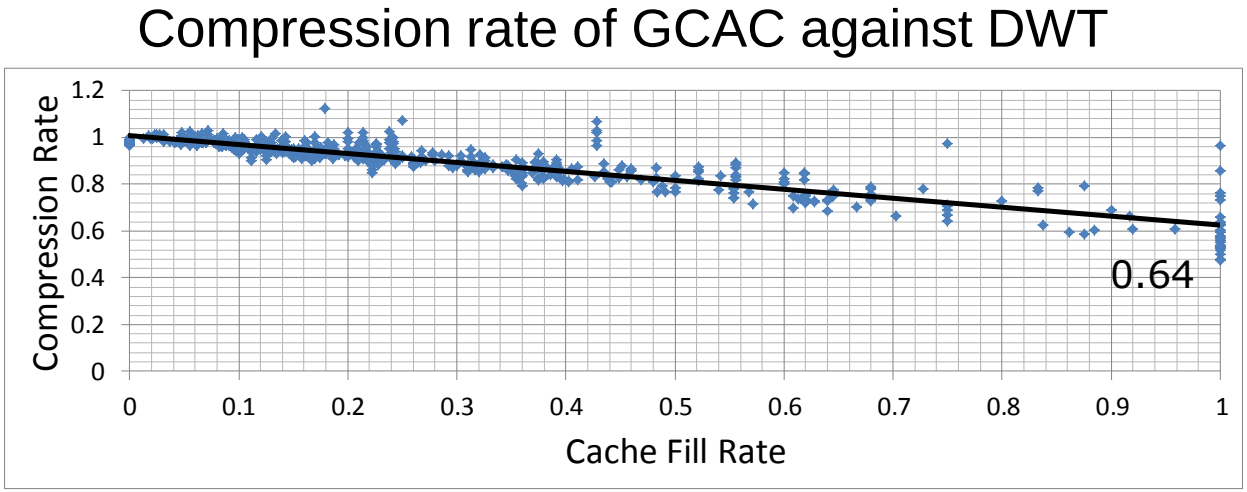


- The top of green bar shows the percentage that CACs or GCAC create smaller checkpoint than DWT-approach
- Data differencing approaches are good (about 90% saving), yet CACs and GCAC are event better (20% more saving)

Experimental Results III

- Line fitting shows the trend
- Statistically, GCAC yields better compression rate when cache fill rate is higher





FPGA Resource Overhead

- The area overhead is introduced by non-intrusive monitoring and creating checkpoint
- The overhead is acceptable

FPGA area overhead of checkpointing and compression engine

	LUTs	LUTs (modified)	Slice	Slices(modified)
BRAM36	0/1	200/2	0	72
RAM128X32S	100	434 (334%)	31	170
RAM64X32S	66	319 (383%)	21	112
RAM32X32S	16	177 (1006%)	4	57
RAM32X16S	8	119 (1387%)	2	54
CAC	320	-	87	-
CAC_32X1	457	-	138	-
CAC_64X1	476	-	147	-
CAC_128X1	484	-	144	-

FPGA area of Weibel lung models

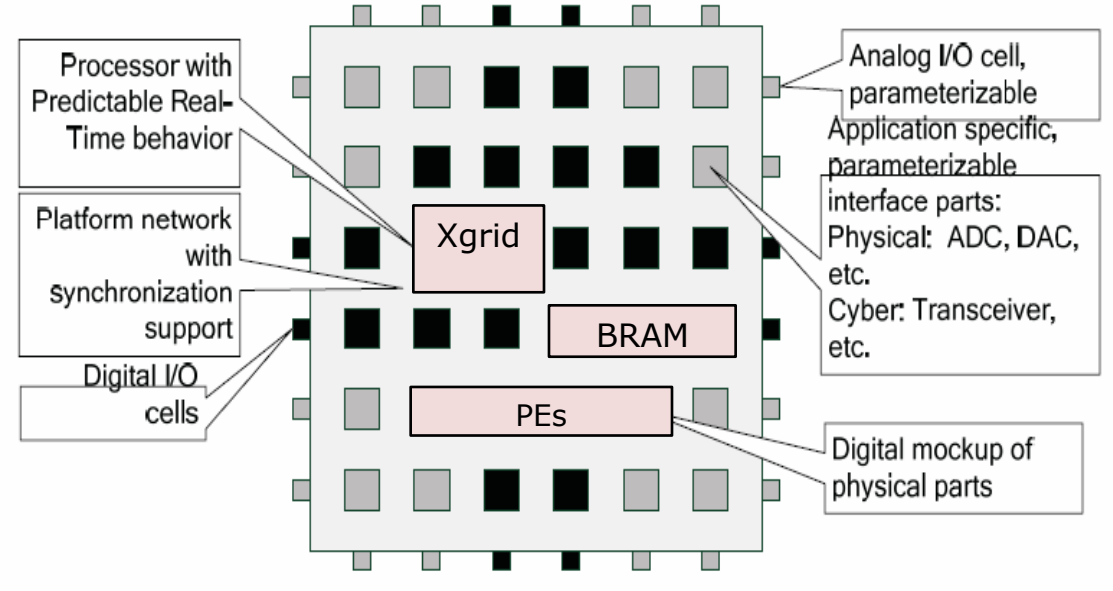
	LUTs	Slices	BRAM
we4_pe4	7317	2832	52
we4_pe4_D	7779(6%)	3132	52 (0%)
we4_pe4_DC	8408(15%)	3363	52 (0%)
we6_pe8	8769	3720	56
we6_pe8_D	9875(13%)	3841	56 (0%)
we6_pe8_DC	11235(28%)	4505	56 (0%)
we8_pe16	12963	4747	64
we8_pe16_D	14431(11%)	5459	64 (0%)
we8_pe16_DC	17621(36%)	6656	64 (0%)

Outline

- Motivation
- Problem Statement
- Approaches
- Experimental Results
- Conclusions

Conclusions and Future Work

- Our design instrumentation provides non-intrusive debugging
  - we trade FPGA resource (area) for it
- Our compression scheme saves 20% more than the basic data differencing technique
- We will apply our technique to High-level CPS platform for real-time debugging and monitoring



Thank You!!
Q & A

Different Approaches for Comparison

A checkpoint includes not only the differenced data but also their index

Raw=∑(NumberPEWidth + PEDataRAMSize)
Deflate=∑(NumberPEWidth + PEDataRAMComressedSize)
DWT=∑(NumberPEWidth + MaxNumberDirtyWordWidth +
(WordEncodingWidth × NumberDirtyWord))
DNT=∑(NumberPEWidth + MaxNumberDirtyNibbleWidth +
(NibbleEncodingWidth × NumberDirtyWord))
CACs=∑(NumberPEWidth + CACEncoding +
MaxNumberDirtyWordWidth+
(WordEncodingWidth × NumberDirtyWordInRAM))
GCAC=GCACEncoding +
∑(NumberPEWidth + MaxNumberDirtyWordWidth +
(WordEncodingWidth × NumberDirtyWordInRAM))

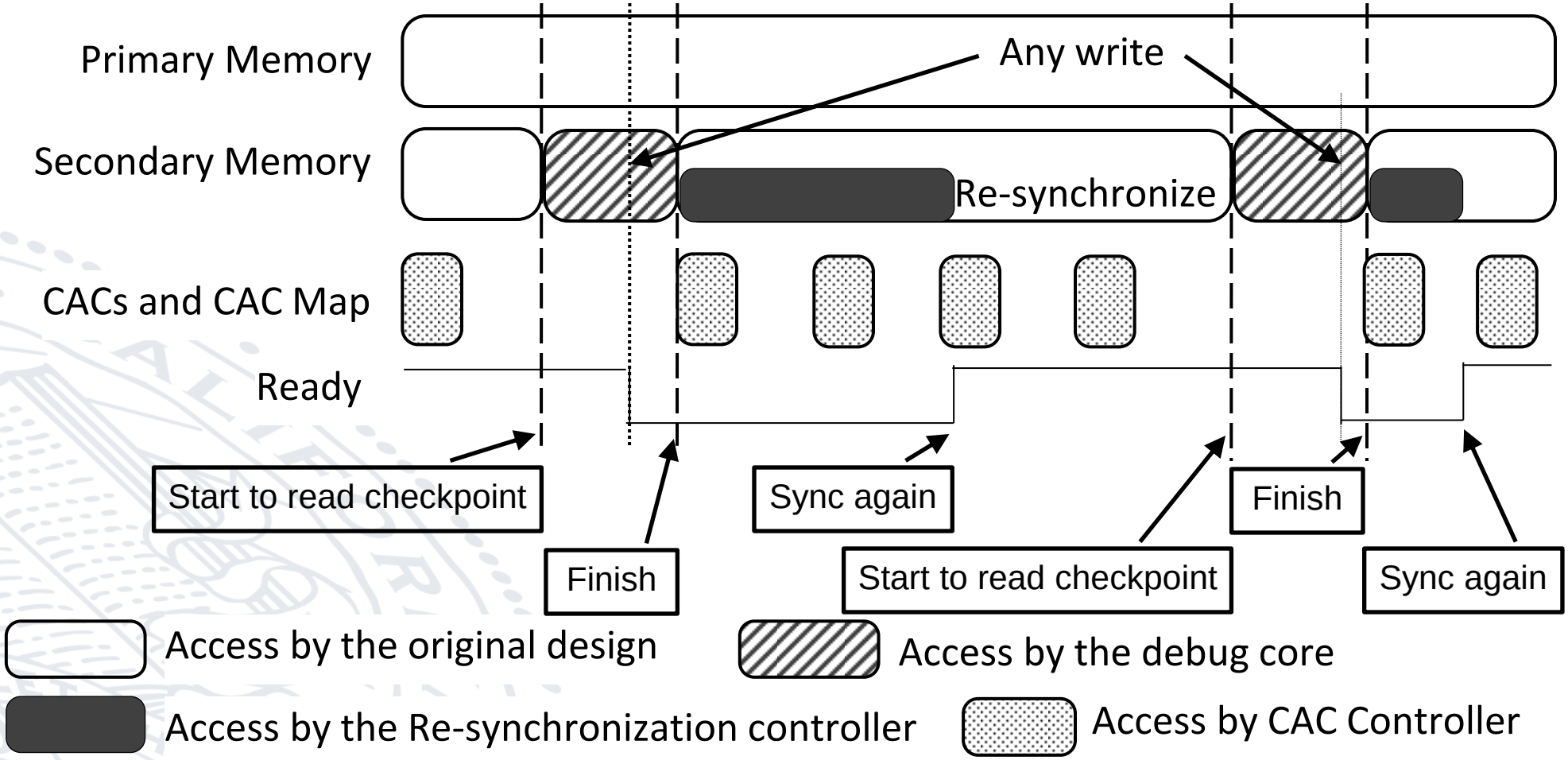
Bandwidth Consumption

UART	Time Resolution
115k	1 s
11.5k	100 ms
1.15k	10 ms
0.15k	1 ms

USB (Full Speed)	Time Resolution
12000k	1 s
1200k	100 ms
120k	10 ms
12k	1 ms

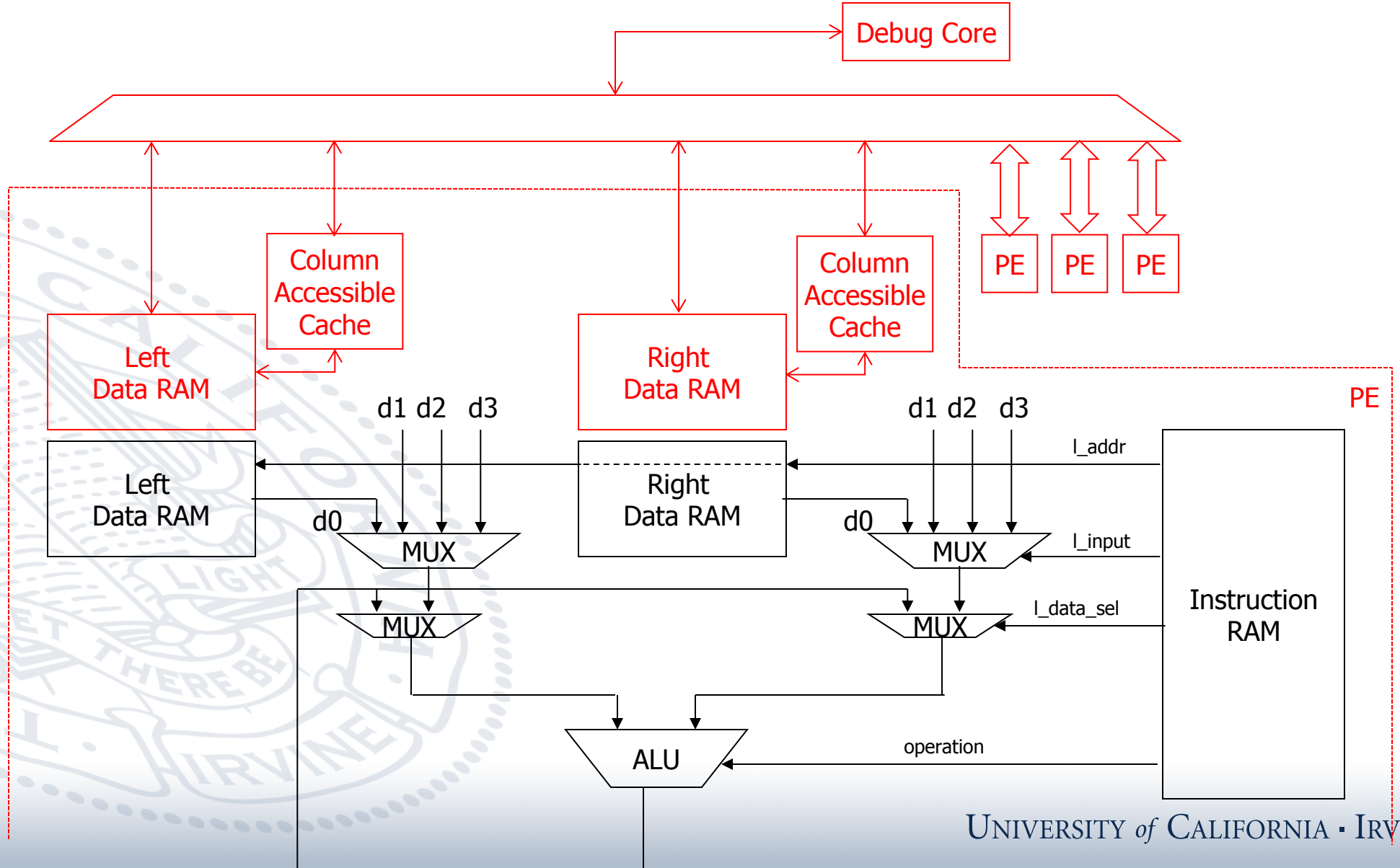
	we6_pe8	we8_pe16	we10_pe196
CAC@1s	1.5k	2.6k	16k
Deflate@1s	16.4k	5.3k	26k
RAW@1s	18.4k	65k	400k

Access Timing Diagram



The primary memory is never accessed by a debug core and re-synchronization controller

How do We Connect Column Accessible Caches?

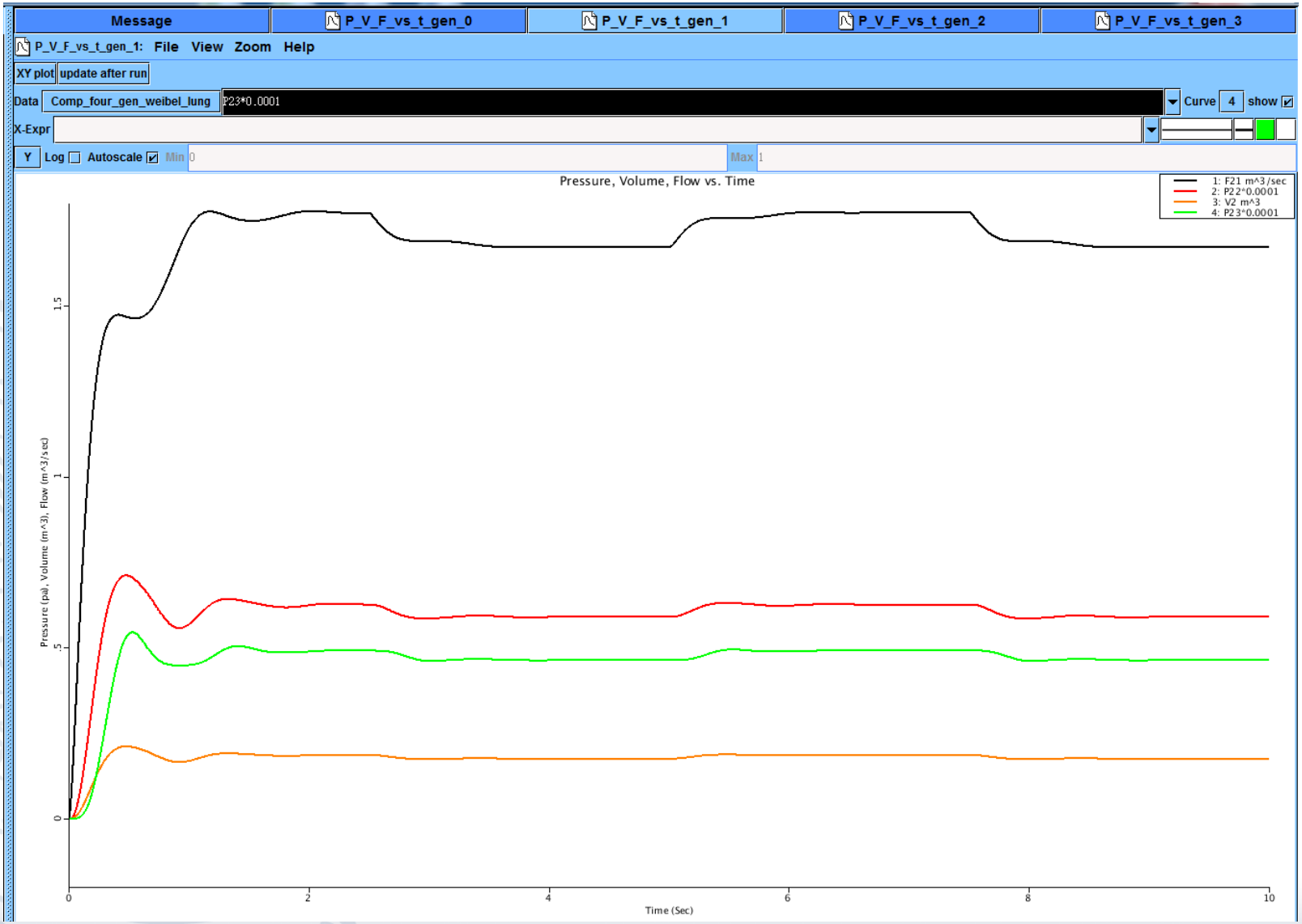


Related Work

Reference:

- T. Wheeler, P. Graham, B. Nelson, and B. Hutchings, “Using Design-Level Scan to Improve FPGA Design Observability and Controllability for Functional Verification,” in FCCM, 2001
- C.L. Chuang and W.H. Cheng, "Hybrid approach to faster functional verification with full visibility", IEEE Design & Test of Computers, pp.154-162, April 2007
- M. A. Khan, R. N. Pittman, and A. Forin, “gNOSIS: A Board-level Debugging and Verification Tool,” in ReConFig, 2010
- Xilinx ChipScope and Altera SiganlTap

Generation 1



Generation 3