Synthesis for Customized Computing

Jason Cong

Chancellor's Professor, UCLA

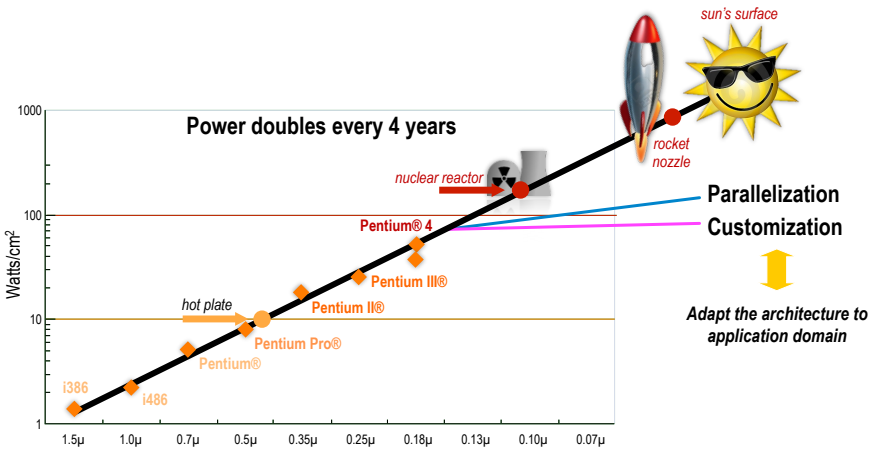
Director, Center for Domain-Specific Computing

cong@cs.ucla.edu

http://cadlab.cs.ucla.edu/~cong

1

Our Research Focus Since 2008: Customized Computing for Better Efficiency



Based on Fred Pollack (Intel) and Michael Taylor (UCSD)

2

Why Customized Computing?

AES 128bit key 128bit data	Throughput	Power	Figure of Merit (Gb/s/W)
0.18mm CMOS	3.84 Gbits/sec	350 mW	11 (1/1)
FPGA [1]	1.32 Gbit/sec	490 mW	2.7 (1/4)
ASM StrongARM [2]	31 Mbit/sec	240 mW	0.13 (1/85)
ASM Pentium III [3]	648 Mbits/sec	41.4 W	0.015 (1/800)
C Emb. Sparc [4]	133 Kbits/sec	120 mW	0.0011 (1/10,000)
Java [5] Emb. Sparc	450 bits/sec	120 mW	0.0000037 (1/3,000,000)

[1] Amphion CS5230 on Virtex2 + Xilinx Virtex2 Power Estimator
[2] Dag Arne Osvik: 544 cycles AES – ECB on StrongArm SA-1110
[3] Helger Lipmaa PIII assembly handcoded + Intel Pentium III (1.13 GHz) Datasheet
[4] gcc, 1 mW/MHz @ 120 Mhz Sparc – assumes 0.25 u CMOS
[5] Java on KVM (Sun J2ME, non-JIT) on 1 mW/MHz @ 120 MHz Sparc – assumes 0.25 u CMOS

Source: *P Schaumont and I Verbauwhede, "Domain specific codesign for embedded security," IEEE Computer 36(4), 2003*

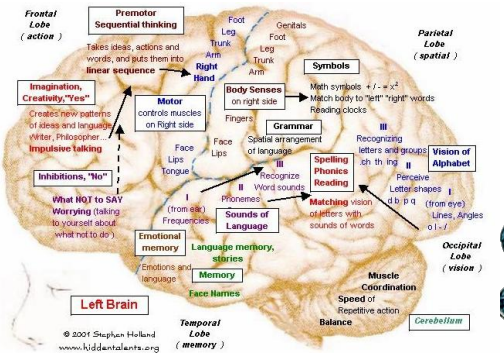
3

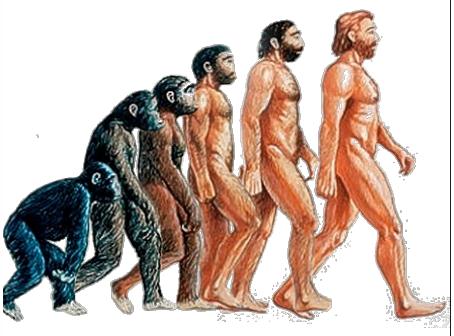
How to Improve the Efficiency? Our Proposal – Customized Computing with Accelerator-Rich Architectures

- A customizable heterogeneous platform (CHP)
  - With a sea of dedicated and composable accelerators
  - Most computations are carried on accelerators – not on processors!
- A fundamental departure from von Neumann architecture
- Why now?
  - Previous architectures are device/transistor limited
  - Von Neumann architecture allows maximum device reuse
    - One pipeline serves all functions, fully utilized
- Future architectures
  - Plenty of transistors, but power/energy limited (dark silicon)
  - Customization and specialization for maximum energy efficiency
- A story of specialization

Lessons from Nature: Human Brain and Advance of Civilization

- **High power efficiency (20W) of human brain comes from specialization**
  - Different region responsible for different functions
- **Remarkable advancement of civilization also from specialization**
  - More advanced societies have higher degree of specialization





UCLA Newsroom

Home

All Stories

All Stories
Featured News
News Releases
Advisories
Images
Multimedia

Research
Health Sciences
Arts & Humanities
Student Affairs
Academics & Faculty
Campus News
Media Contacts

Images
Video
Blogs

For the Media
Contacts
News releases
Advisories
About UCLA

UCLA Newsroom > All stories > News Releases

NSF awards UCLA $10 million to create customized computing technology

By Wileen Wong Kromhout| 8/11/2009 9:45:00 AM

The UCLA Henry Samueli School of Engineering and Applied Science has been awarded a $10 million grant by the National Science Foundation's Expeditions in Computing program to develop high-performance, energy efficient, customizable computing that could revolutionize the way computers are used in health care and other important applications.

In particular, UCLA Engineering researchers will demonstrate how the new technology, known as domain-specific computing, could transform the role of medical imaging and hemodynamic simulation, providing more cost-effective and convenient solutions for preventive, diagnostic and therapeutic procedures and dramatically improving health care quality, efficiency and patient outcomes.

"This significant award is another testament to the world-class faculty here at UCLA who continue to push the envelope to solve society's most pressing issues," said UCLA Chancellor Gene Block. "We are grateful to the NSF, which has repeatedly provided crucial funding to our faculty, helping to place the university among the nation's top five in research funding."

In an effort to meet ever-increasing computing needs in various fields, the computing industry has entered an "era of parallelization," in which tens of thousands of computer servers are connected in warehouse-scale data centers, said Jason Cong, the Chancellor's Professor of Computer Science and director of the new UCLA Center for Domain-Specific Computing (CDSC), which will oversee the research. But these parallel, general-purpose computing systems still face serious challenges in terms of performance, energy, space and cost.

Domain-specific computing holds significant advantages, Cong said. While general-purpose computing relies on computer architecture and languages aimed at any type of application, domain-specific computing utilizes a customizable architecture and custom-oriented, high-level computer languages tailored to a particular application area or domain — in this case, medical imaging and hemodynamic modeling. This customization ultimately results in much less energy consumption, faster results, lower costs and increased productivity.

The goal of the new UCLA center, Cong said, is to look beyond parallelization and focus on domain-specific customization to bring significant power-performance efficiency improvement to important application domains.

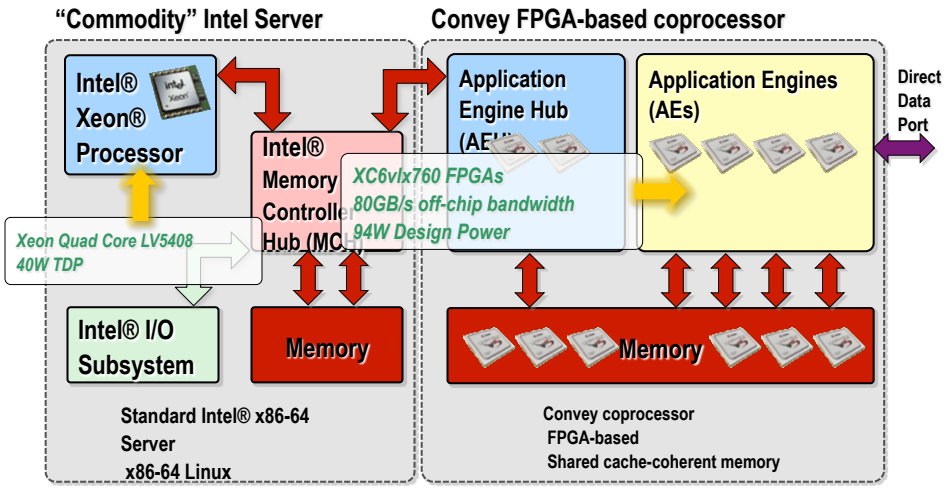
Levels of Customization

- Single-chip level
  - Require new processor designs, e.g. using composable accelerators [ISLPED' 12, DAC'14]
- Server node level
  - Host CPU + FPGA via PCI-e or QPI connections
- Data center level
  - Clusters of heterogeneous computing nodes

Levels of Customization

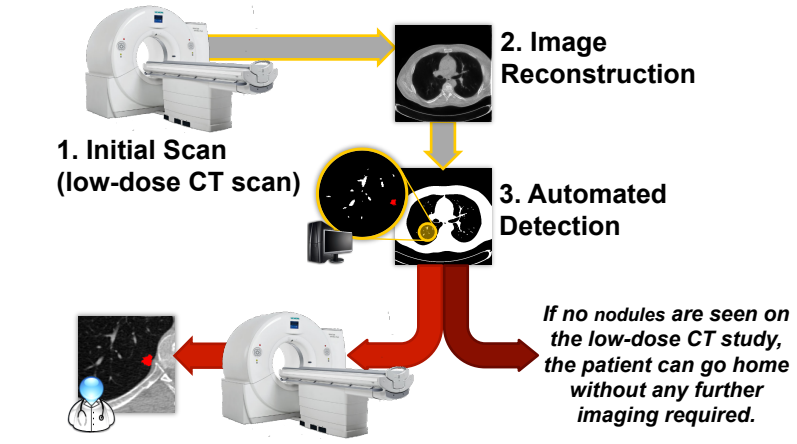
- Single-chip level
  - Require new processor designs, e.g. using composable accelerators [ISLPED' 12, DAC'14]
- Server node level
  - Host CPU + FPGA via PCI-e or QPI connections
- Data center level
  - Clusters of heterogeneous computing nodes

Example of CDSC Heterogeneous Computing Server



9

A Success Application: Low-Dose Adaptive CT Scan



If nodules are detected, a diagnostic follow-up scan is performed during the same visit.

Center for Domain Specific Computing · UCLA Health RADIOLOGY · CFAR

5 Years of Accelerating Medical Image Processing

	2010	2013	2015 (Today)
CT image reconstruction	18 hours Single thread CPU	20 minutes FPGA acceleration on Convey	6 minutes 4 Virtex-6 FPGAs on Convey w/data reuse
Denoising	5 minutes Single thread CPU	15 seconds NVidia GPU	3 seconds Core i7 Haswell, OpenMP, stencils
Registration	10 minutes Single thread CPU	2 minutes NVidia GPU	30 seconds Core i7 Haswell, OpenMP, stencils
Segmentation	20 minutes Single thread CPU	4 minutes Multithread CPU	1 minute Core i7 Haswell, OpenMP, stencils
Analysis	45 minutes Single thread CPU	18 minutes Multithread CPU	5 minutes* Core i7 Haswell, OpenMP

* New detection method w/improved accuracy

Workstation | CPU + GPU | FPGA, CPU platform

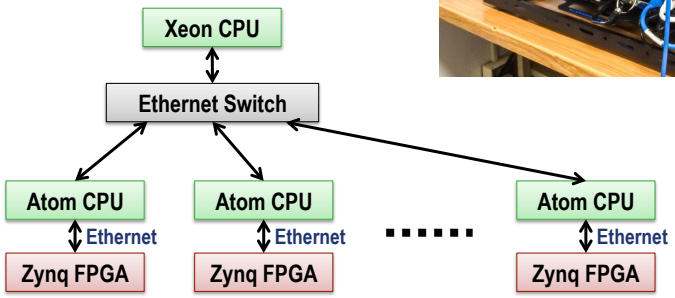
Levels of Customization

- **Single-chip level**
  - Require new processor designs, e.g. using composable accelerators [ISLPED' 12, DAC'14]
- **Server node level**
  - Host CPU + FPGA via PCI-e or QPI connections
- **Data center level**
  - Clusters of heterogeneous computing nodes
  - How about programming at data center level?

12

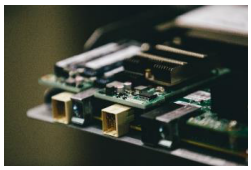
FPGA "FARM" at UCLA – A Small Multi-FPGA Rack

- Deployed in 2013
- Used for research and teaching
  - CS133 (60+ students)
  - CS259 (18 students)



13

Datacenter Level Integration at Microsoft

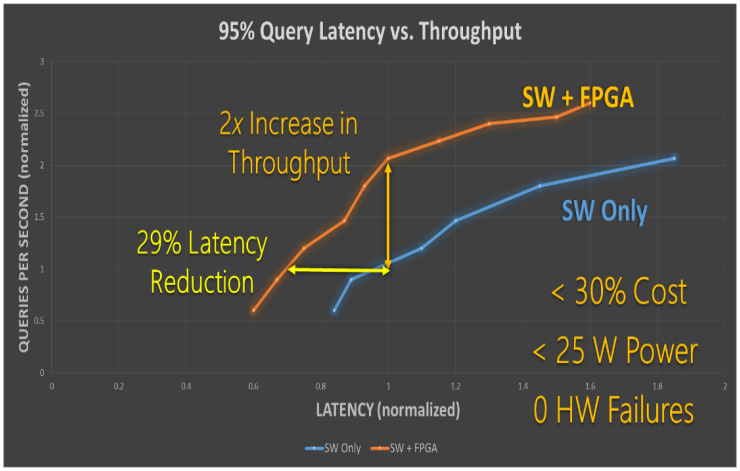


A. Putnam, "*A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services*", ISCA'2014

14

Accelerating Large-Scale Services – Bing Search

1,632 Servers with FPGAs Running Bing Page Ranking Service (~30,000 lines of C++)

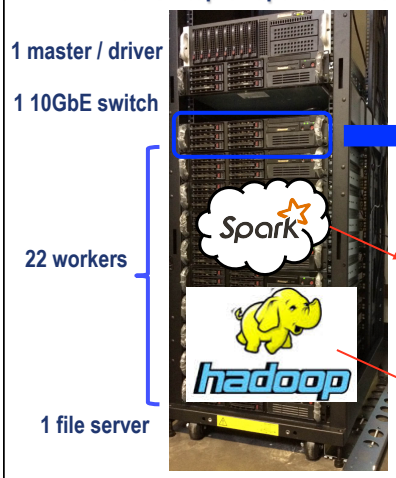


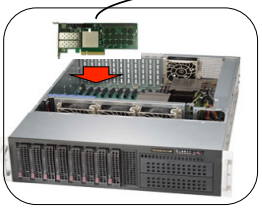
A. Putnam, "*A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services*", ISCA'2014

15

CDSC FPGA-Enabled Cluster

- **A 24-node cluster with FPGA-based accelerators**
  - **Run on top of Spark and Hadoop (HDFS)**





Alpha Data board:
1. Virtex-7 FPGA
2. 16GB on-board RAM

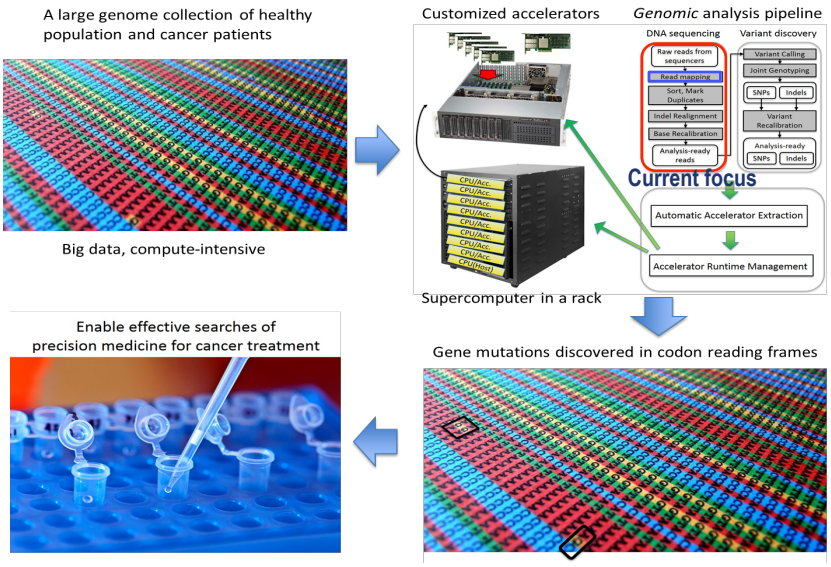
Each node:
1. Two Xeon processors
2. One FPGA PCIe card (Alpha Data)
3. 64 GB RAM
4. 10GBE NIC

Spark:
- Computation framework
- In-memory MapReduce system

HDFS:
- Distributed storage framework

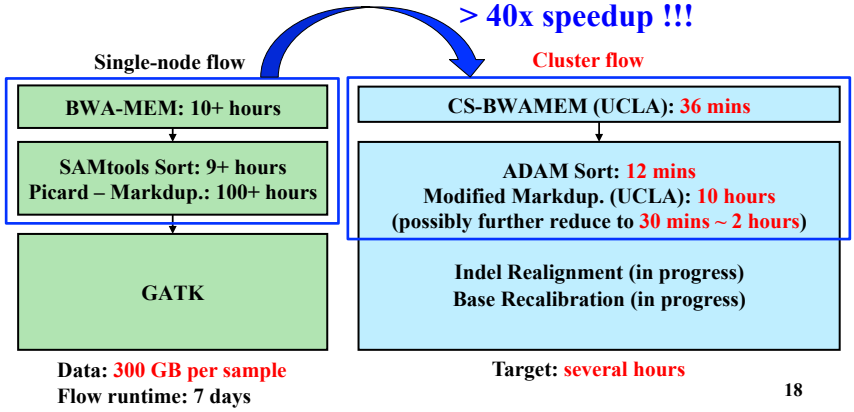
16

Example Application: Personalized Cancer Treatment



Current Progress on Whole-Genome Sequencing

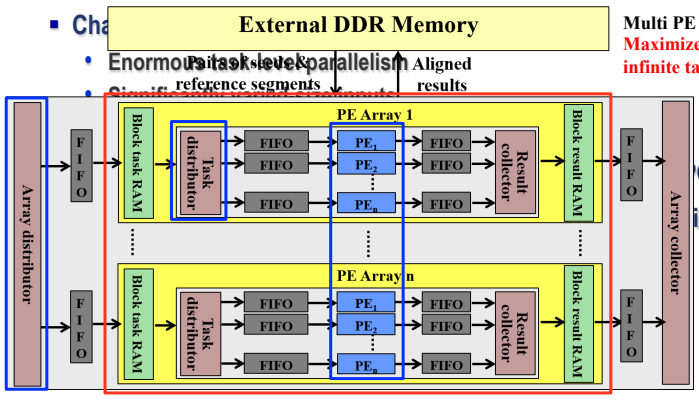
- GATK data cleanup flow
- Milestone
  - Whole-genome alignment in 36 minutes; whole-exome alignment in 7.5 minutes



18

Hardware Acceleration in CS-BWAMEM

- CS-BWAMEM acceleration
  - Accelerate the Smith-Waterman kernel ($O(n^2)$ time complexity)
    - ~50% of total program runtime
    - Batch a group of data and send them FPGA to reduce communication overhead
  - Cha
    - Enormous task-level parallelism
    - Sig



Multi PE arrays:
Maximize the throughput for infinite task-level parallelism

Two-level task distribution:
Shorten the critical path
PE-based architecture:
Decent resource utilization for varied-size inputs
26X speedup over 24=thread Xeon processor
Compatible to the state-of-the-art tool BWA-MEM:
Can be integrate into real DNA sequencing pipeline

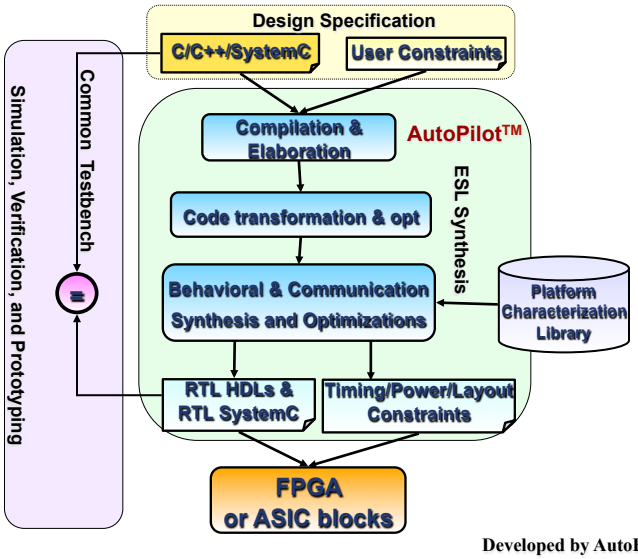
19

How to Program Such "Beasts"?

-- "Write Once, Compile Anywhere"

20

C/C++ Based Synthesis for Accelerators
xPilot (UCLA) -> AutoPilot (AutoESL) -> Vivado HLS (Xilinx)



- Platform-based C to RTL synthesis
- Synthesize pure ANSI-C and C++, GCC-compatible compilation flow
- Full support of IEEE-754 floating point data types & operations
- Efficiently handle bit-accurate fixed-point arithmetic
- SDC-based scheduling
- Automatic memory partitioning
- …

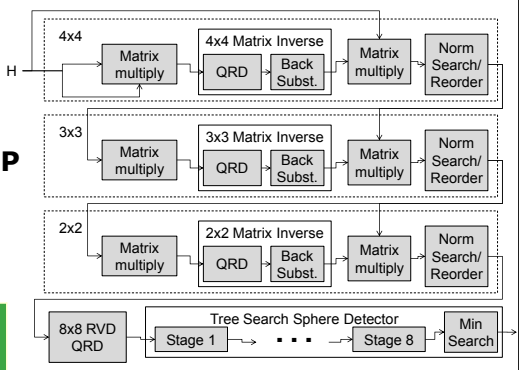
QoR matches or exceeds manual RTL for many designs

Developed by AutoESL, acquired by Xilinx in Jan. 2011

21

AutoPilot Results: Sphere Decoder (from Xilinx)

- **Wireless MIMO Sphere Decoder**
  - ~4000 lines of C code
  - Xilinx Virtex-5 at 225MHz
- **Compared to optimized IP**
  - 11-31% better resource usage



Metric	RTL Expert	AutoPilot Expert	Diff (%)
LUTs	32,708	29,060	-11%
Registers	44,885	31,000	-31%
DSP48s	225	201	-11%
BRAMs	128	99	-26%

TCAD April 2011 (keynote paper)
"High-Level Synthesis for FPGAs: From Prototyping to Deployment"

22

AutoPilot Results: Optical Flow (from BDTI)

- **Application**
  - Optical flow, 1280x720 progress scan
  - Design too complex for an RTL team
- **Compared to high-end DSP:**
  - 30X higher throughput, 40X better cost/fps

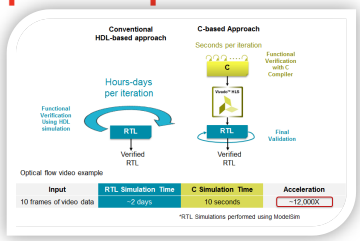
	Chip Unit Cost	Highest Frame Rate @ 720p (fps)	Cost/performance ($/frame/second)
Xilinx Spartan3ADSP XC3SD3400A chip	$27	183	$0.14
Texas Instruments TMS320DM6437 DSP processor	$21	5.1	$4.20

BDTi evaluation of AutoPilot
http://www.bdti.com/articles/AutoPilot.pdf

23

Vivado High-Level Synthesis: Accelerated IP Development and Design Space Exploration

- **C libraries:**
  - Arbitrary precision
  - Floating-point math.h
  - OpenCV video functions
  - DSP
  - Linear algebra
- **Accelerated verification**
  - >100X faster than RTL design
- **Fast compilation and design exploration**
  - Algorithm feasibility
  - Architecture Iteration



	Hand-coded RTL	Vivado HLS
Design Time (weeks)	12	1
Latency (ms)	37	21
Memory (RAMB18E1)	134 (16%)	10 (1%)
Memory (RAMB36E1)	273 (65%)	138 (33%)
Registers	29686 (9%)	14263 (4%)
LUTs	28152 (18%)	24257 (16%)

Customer proven results

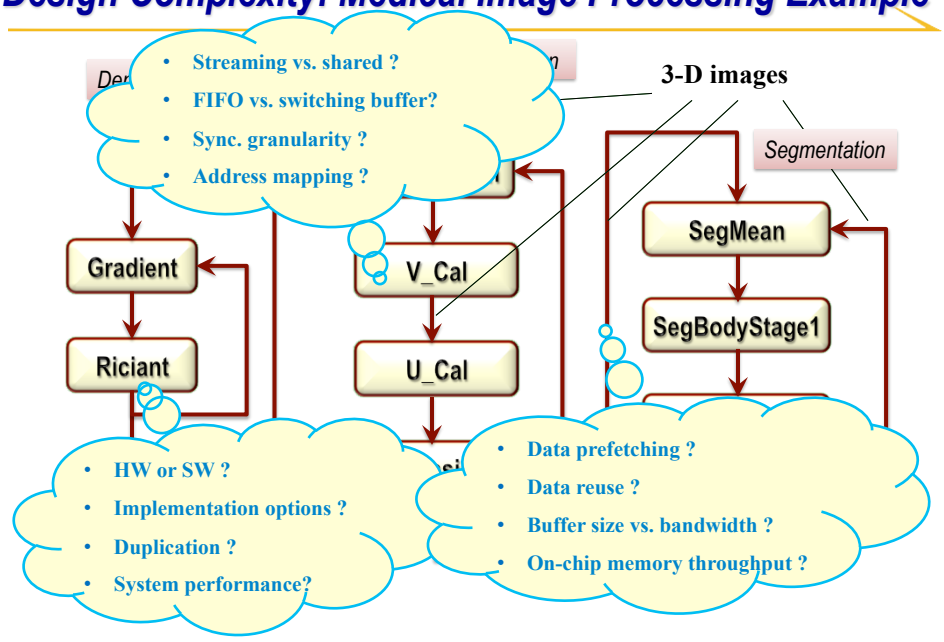
Production-Proven and Adopted by 1000+ companies

- "Vivado® HLS enabled easy and fast implementation of 768x768 QRD single precision floating-point design. We like this tool QoR, productivity and flexibility and will deploy in to more production designs."
- "Vivado HLS made it possible for DSP software engineers to implement LTE layer 1 switch on Zynq® SoCs by enabling us to target more than 500K lines of C code."
- "We developed C++ DSP functions using Vivado HLS and the results met size/speed goal for commercial platform deployment on Virtex®-7."

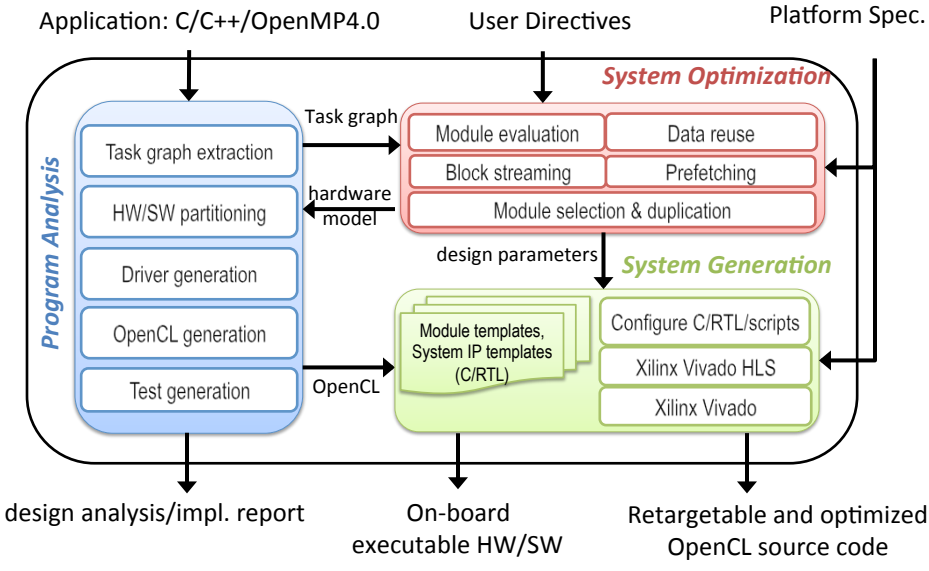
HLS Alone Is Not Enough for Customized Computing

- **A large design space for software and hardware co-design**
- **Also need automated source code transformation for HLS friendly C/C++ code to enable**
  - Concurrent memory access
  - Data reuse and on-chip buffer generation
  - Data prefetching
  - …

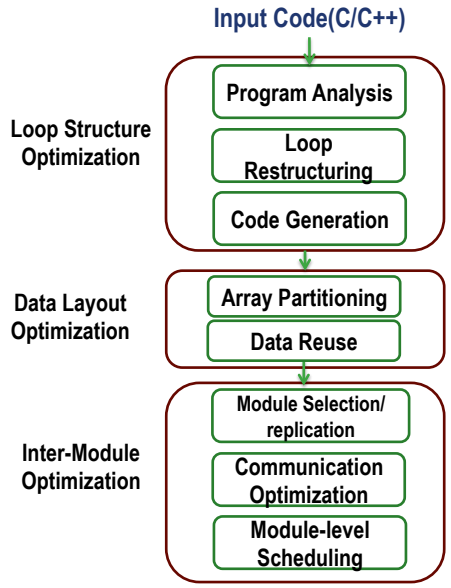
Design Complexity: Medical Image Processing Example



CMOST: Fully Automated Compilation and Mapping Flow [DAC 2015]



Latest Research - Optimizations Beyond HLS



Polyhedral-Based Data Reuse Optimization for Configurable Computing FPGA'13 Best Paper Award

Improving Polyhedral Code Generation for High-Level Synthesis CODES-ISSS'14 Best Paper Award

Theory and Algorithm for Generalized Memory Partitioning in High-Level Synthesis, FPGA'14

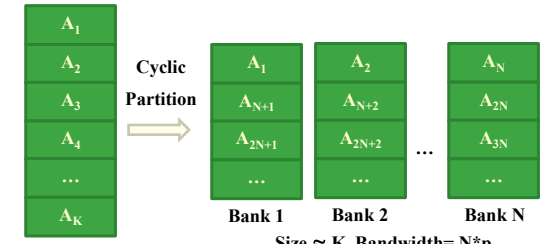
An optimal microarchitecture for stencil computation acceleration based on non-uniform partitioning of data reuse buffers (DAC'14)

Combining Computation with Communication Optimization in System Synthesis for Streaming Applications, FPGA'14

29

Memory Partitioning for Throughput Optimization

- Memory is still a bottleneck
  - Data intensive applications: image/video
  - Loop unrolling/tiling/pipelining
- Memory partitioning

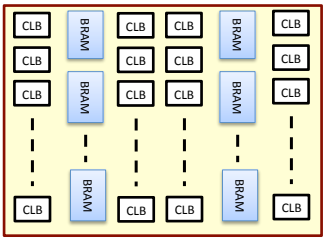


Size = K, Bandwidth = p

p: memory port number

Size ≈ K, Bandwidth= N*p

N: Partition Factor



Challenge: generate conflict-free memory partitioning for a given program

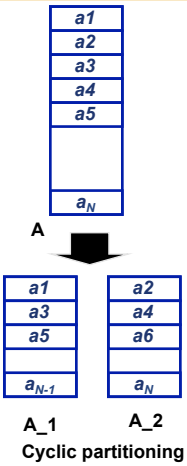
30

Memory Partitioning for HLS*

- **Cyclic partitioning**
  - Easy to implement
  - Very effective in practice
- **Example**
  - Will i and (3*i+1) go to the same memory bank?
- **Theorem**

  $\forall i,\ a_1*i+b_1 \neq a_2*i+b_2 \bmod N$

  $\Leftrightarrow \gcd(a_1-a_2, N) \nmid (b_1-b_2)$



*J. Cong, W. Jiang, B. Liu and Y. Zou. ACM TODAES 2011 (Best Paper Award)

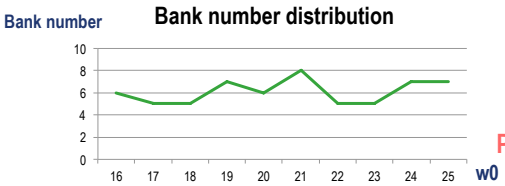
31

Memory Partitioning for Multidimensional Arrays

- **Flatten-Based Partitioning**
  - Flatten multidimensional array
  - Partition flattened single dimensional array

```
for (j=0; j<w1; j++)
 for (i=0; i<w0; i++)
  foo(A[j][i], A[j][i-1], A[j-1][i ], A[j+1][i], A[j][i+1]);
```

```
foo(A[w0 *j+i], A[w0 *j+i-1], A[w0 *j+i- w0], A[w0 *j+i+ w0], A[w0 *j+i +1]);
```



Conflict free conditions:

$N \nmid 2$ $\quad N \nmid w0$

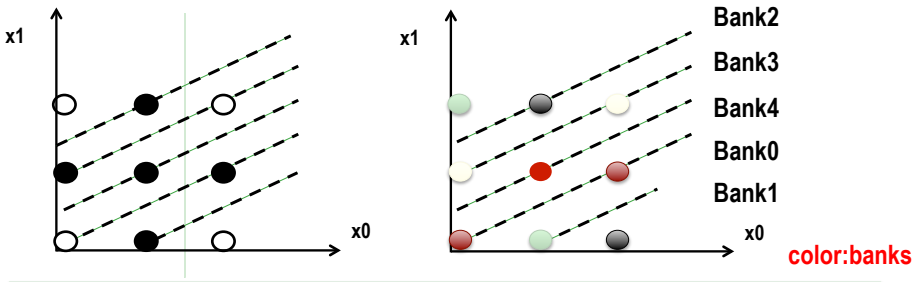
$N \nmid (w0-1)$ $\quad N \nmid (w0+1)$

Partition results are related to array sizes!

32

Linear-Transformation-Based Partitioning*

- Linear transformation-based approach
  - Multidimensional address $\vec{x}$ linearization: $L(\vec{x}) = \vec{\alpha} \cdot \vec{x}$
  - Bank mapping: bank$(\vec{x})$= $L(\vec{x})$ mod N (Cyclic)
- Example: denoise



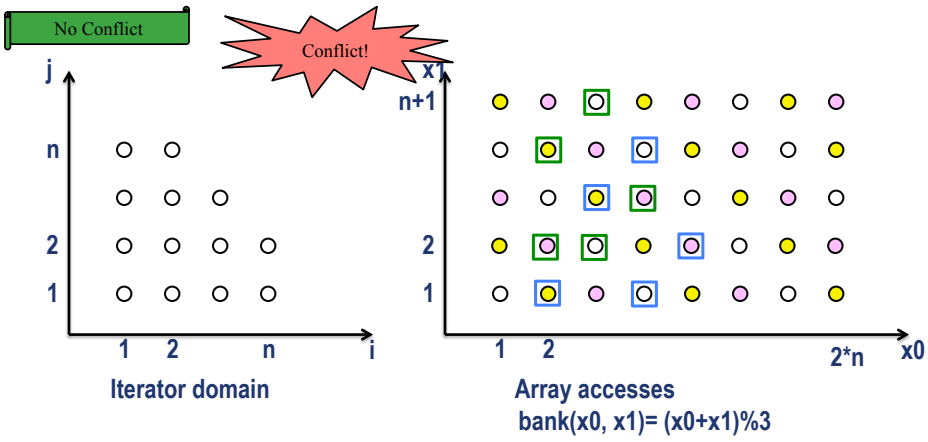
$\forall i,j$, A[a_1*i+b_1][c_1*j+d_1] not conflict with A[a_1*i+b_1][c_1*j+d_1]
$\Leftrightarrow$ gcd((α_1 (a_1−a_2)+α_2 (c_1−c_2)), N) $\nmid$ (α_1 (b_1−b_2)+α_2 (d_1−d_2))

*Y. Wang, P. Li, P. Zhang, C. Zhang and J. Cong. DAC 2013

33

Access Conflict

```
for (i = 1; i <= n; i++)
  for(j = 1; j <= min(n,n-i+2); j++)
    foo(A[j+1][i+1], A[j][2*i]);
```



34

Conflict Polytope

- *Conflict polytope* of two references is a subset of iteration domain where the two references are mapped on the same bank

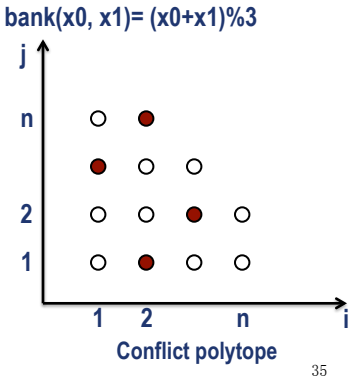
$$\begin{cases} 1 \le i \le n \\ 1 \le j \le n \\ i + j \le n + 2 \\ (i+1) + (j+1) \equiv (2i + j) \bmod 3 \end{cases}$$

```
for (i = 1; i <= n; i++)
  for(j = 1; j <= min(n,n-i+2); j++)
    foo(A[j+1][i+1], A[j][2*i]);
```

- Insert an extra variable *k* to linearize

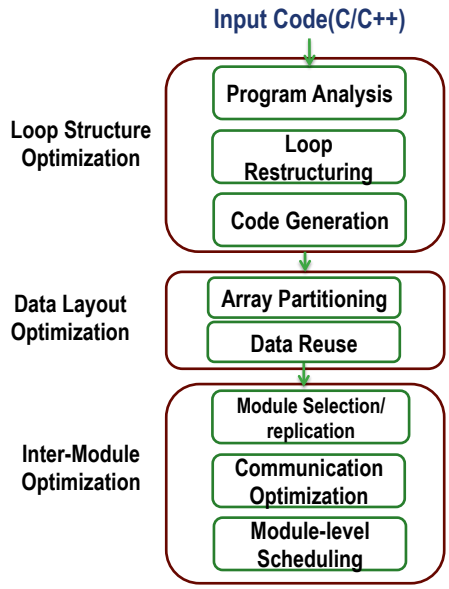
$$(i+1) + (j+1) = (2i + j) + 3k$$

- Fourier–Motzkin Algo. (Fourier 1826, Motzkin1936)
  - Test the emptiness of the conflict polytope
  - Algorithm complexity: $O(m^{2^t})$
    - m: number of inequalities (m=4 in the example)
    - t: number of of variables (t=3 in the example)
    - independent of iteration domain size *n*



35

Latest Research - Automating Customized Computing



Polyhedral-Based Data Reuse Optimization for Configurable Computing FPGA'13 Best Paper Award

Improving Polyhedral Code Generation for High-Level Synthesis CODES-ISSS'14 Best Paper Award

Theory and Algorithm for Generalized Memory Partitioning in High-Level Synthesis, FPGA'14

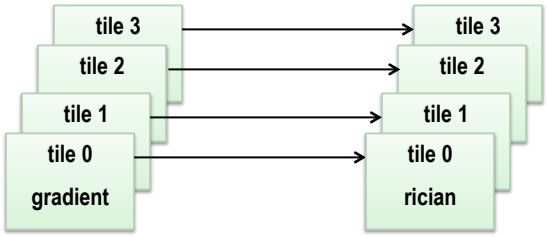
An optimal microarchitecture for stencil computation acceleration based on non-uniform partitioning of data reuse buffers (DAC'14)

Combining Computation with Communication Optimization in System Synthesis for Streaming Applications, FPGA'14

36

Motivation

Tile size: 32x32
Image: 64x64, 4 tiles



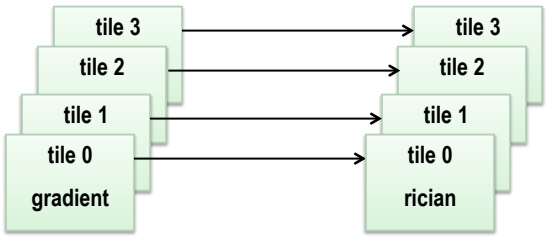
- Which implementation to use for each module?
  - Memory partitioned v.s. non-memory-partitioned

	BRAM	DSP	FF	LUT
non-partitioned gradient	128	21	2511	2125
partitioned gradient	176	56	7147	7262
partitioned rician	128	22	4692	3991
non-partitioned rician	176	88	14475	15537

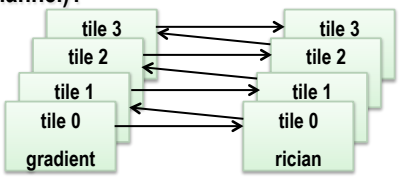
37

Motivation

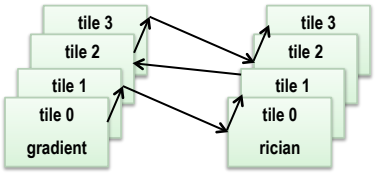
Tile size: 32x32
Image: 64x64, 4 tiles



- How many number of replicas?
- Scheduling and Communication cost (number of tiles in the communication channel)?



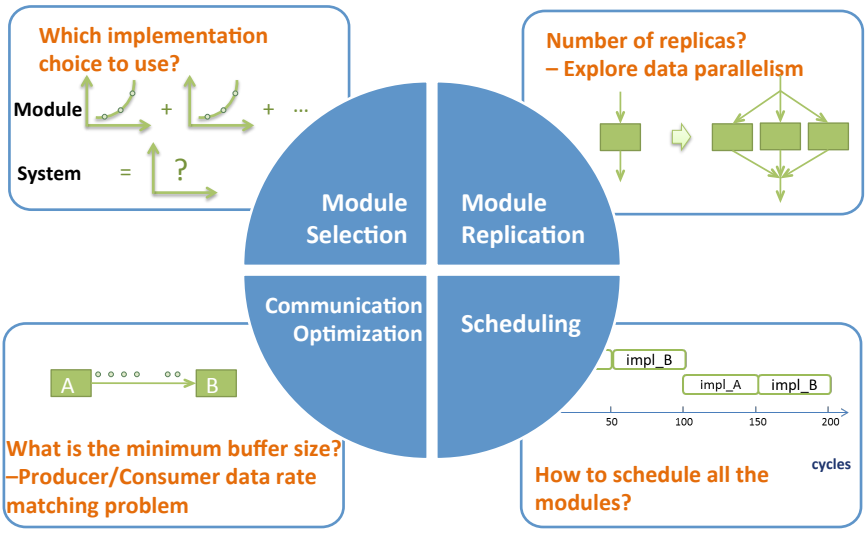
scheduling 0 →1 tile



scheduling 0 →2 tiles

38

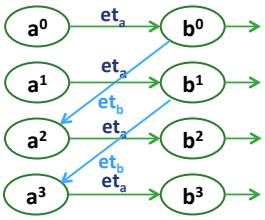
A Rich Design Space: System-Level Synthesis with HLS for Streaming Applications



39

Formulation (1/2)

- Derive a scheduling graph
  - Associate each node with a time variable, denoting the starting time of the node
  - Scheduling graph: delineates all the scheduling constraints
    - Module latency, Module replication, System throughput requirement, Buffer constraints



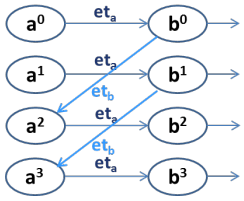
Module latency constraints
et_a: execution time of task a
et_b: execution time of task b

Buffer Constraints
If buffer size between a and b is 2, then add edges: $b^0 \rightarrow a^2$ $b^1 \rightarrow a^3$

40

Formulation (2/2)

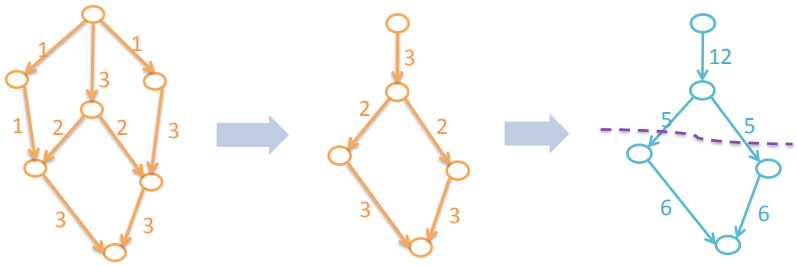
- Associate each node with a scheduling variable
  - $t(b^0) - t(a^0) >= et_a$
  - $t(a^2) - t(b^0) >= et_b$
  - ...
  - Scheduling variables are integer variables



- Schedulability checking problem is a ***System of Difference Constraints (SDC)*** problem
  - It can be solved optimally *in polynomial time* by linear programming relaxation
  - And the solution is guaranteed to be integers

41

Exploration



Scheduling Graph

- Find the length of longest path *(maxL)*
- In this example, *maxL* = 8

Find critical paths

- Find all the paths whose lengths are *maxL,*
- **o**r more aggressively, *(1-ε)*maxL*

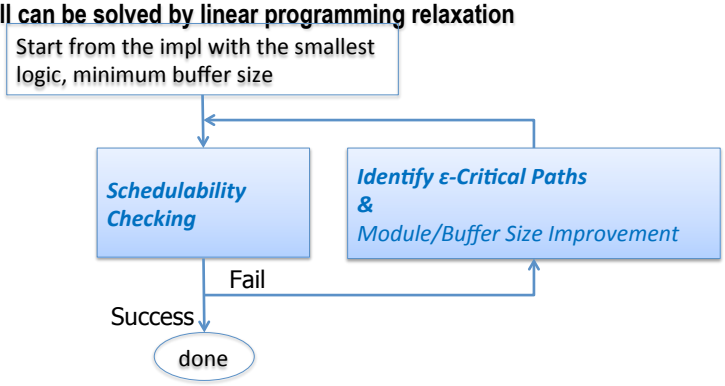
Module Improvement

- Associate each edge a new weight – the area penalty to remove this edge from the critical paths
- Find a minimum cut on the graph

42

Streaming Synthesis (ST-Syn)

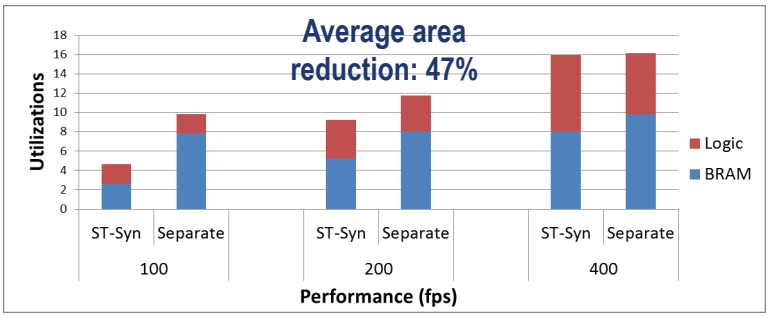
- *Formulation – Schedulability checking*
  - System of Difference Constraint Problem
- *Exploration – Identify critical path, module/buffer improvement*
  - Find ε-critical paths in the scheduling graph
  - Minimum cut problem
- All can be solved by linear programming relaxation



43

Experiments on Example Denoise

- Our methodology: ST-Syn
  - computation & communication co-optimization
- Separate:
  - separate computation opt. + communication opt.
- → Communication and computation should be considered in a unified framework



44

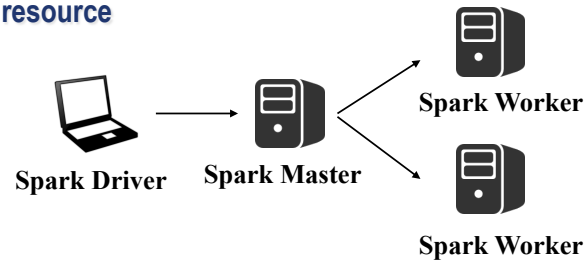
More is Needed for Data Center Level Deployment

45

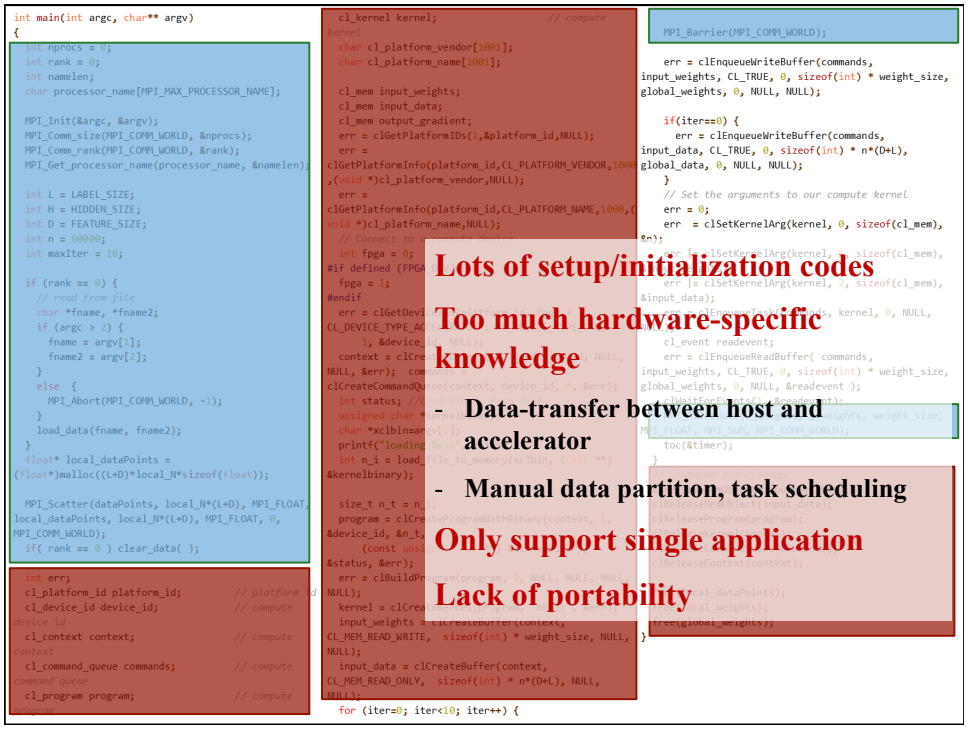
Scalable Big-Data Programming

- **Simplified programming models**
  - MapReduce, Dataflow
- **User-transparent Runtime**
  - Distributed computing
  - Scheduling and resource management
  - Fault-tolerance

```
val points = sc.textfile().cache()
for (i <- 1 to ITERATIONS) {
  val gradient = points.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x)))
    - 1) * p.y * p.x
    ).reduce(_ + _)
  w -= gradient
}
```

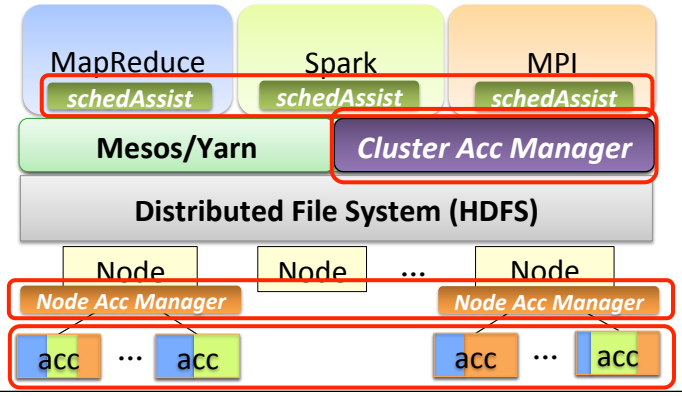


46



More on Cluster Accelerator Resource Management

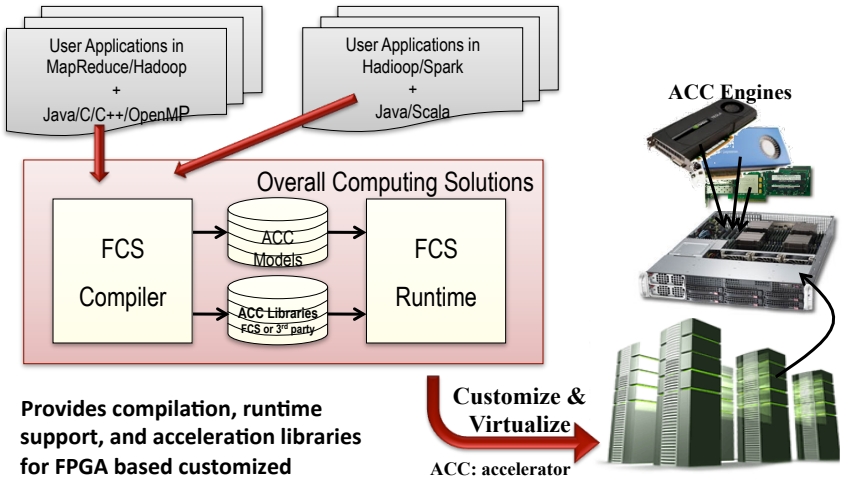
- **How to make sure acc resources are efficiently utilized?**
  - Each framework get the accelerator resources it deserves
  - Tasks in each framework are assigned to their preferred node
  - Fine-grained sharing of each accelerator



48

Falcon Computing Solutions, Inc

http://falcon-computing.com



- Provides compilation, runtime support, and acceleration libraries for FPGA based customized computing in datacenters
- Another UCLA spin-off

49

Challenges and Opportunities for Logic Synthesis

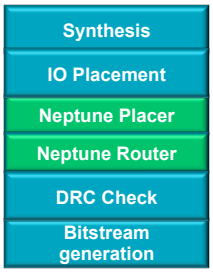
-- Ultra Fast Synthesis

50

Example Neptune Inc. (2011-2013)

Fast FPGA Place/Route Solution Provider

Vision: Enable Ultra Fast FPGA Compilation



- Delivered Placer 20x speedup vs. Xilinx tool
- Tightly integrated with Xilinx ISE and Vivado environment
- Supported advanced Virtex-6 and Virtex-7 Architecture
- Triggered tremendous interest in emulation and reconfigurable computing
- Acquired by Xilinx in 2013

XILINX®

Can We Do the Same or Better for Logic Synthesis?

Goal: Synthesize 1M Cells Per Minute?

- **Utilizing cloud computing, GPU/FPGA acceleration, etc?**

52

Concluding Remarks

- New era of computing
  - Accelerator-centric computing
  - Need efficient support for customization and specialization
- Customization at all levels
  - Chip-level
  - Server node level
  - Data center level
- Data center level customization holds great promise
  - That's where workload aggregates

53

Concluding Remarks (Cont'd)

- Software support is critical
  - Programming models
    - Hadoop/MapReduce or SPARK (+ C/C++), OpenMP, OpenCL,, …
  - Fast and simple compilation
  - Runtime management
- Need critical mass for FPGA acceleration libraries

54

Acknowledgements

- Center for Domain-Specific Computing (CDSC) under the supports NSF Expeditions in Computing Program, Fujitsu, Intel, and Mentor Graphics under the NSF InTrans Program
- C-FAR Center under the STARnet Program
- CDSC faculty:


Aberle (UCLA)


Baraniuk (Rice)


Bui (UCLA)


Chang (UCLA)


Cheng (UCSB)


Cong (Director) (UCLA)


Palsberg (UCLA)


Potkonjak (UCLA)


Reinman (UCLA)


Sadayappan (Ohio-State)


Sarkar (Associate Dir) (Rice)

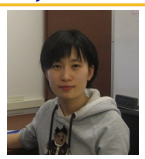

Vese (UCLA)

55

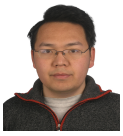
More Acknowledgements -- Postdocs, Graduate Students, and Collaborators

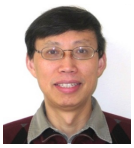

Prof. Deming Chen (UIUC/ADSC)


Yuting Chen (UCLA)


Hui Huang (UCLA)


Muhuan Huang (UCLA/Falcon Comp)

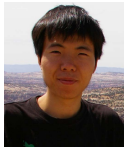

Dr. Peng Li (UCLA)


Dr. Peichen Pan (Falcon Comp)


Prof. Louis-Noël Pouchet (UCLA)

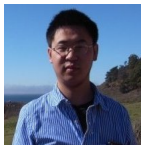

Yuxin Wang (PKU)


Di Wu (UCLA/Falcon Comp)


Bingjun Xiao (UCLA)


Hao Yu (UCLA)


Dr. Peng Zhang (Falcon Comp.)


Yi Zou (UCLA)


Wei Zuo (UIUC)

56