Automating Customized Computing

Jason Cong

Chancellor's Professor, UCLA

Director, Center for Domain-Specific Computing

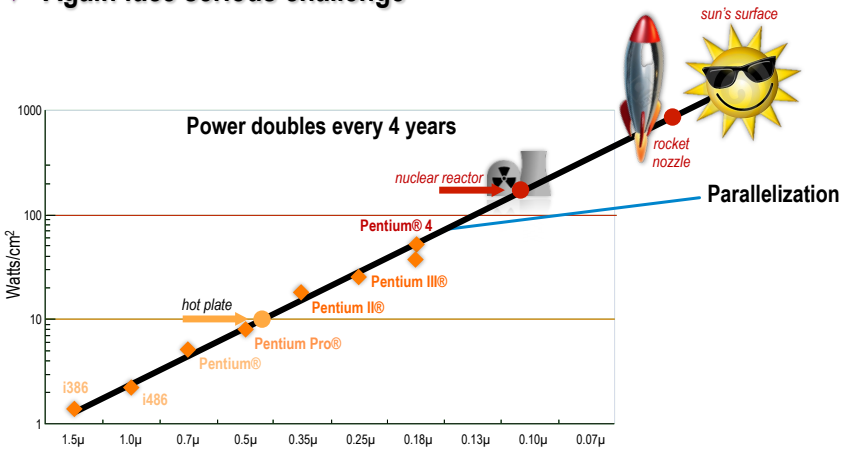
cong@cs.ucla.edu

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

1

Challenge with Processor Design – Power Barrier

- Current solution: *Parallelization*
- Again face serious challenge

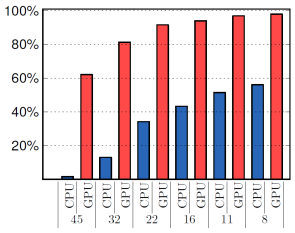


Based on Fred Pollack (Intel) and Michael Taylor (UCSD)

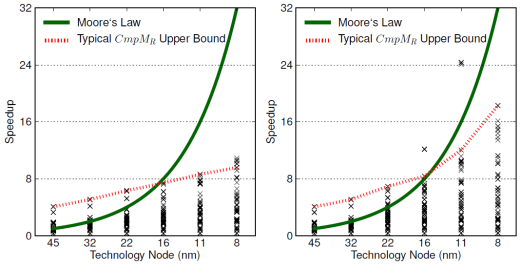
2

Dark Silicon and the End of Multicore Scaling
[G. Venkatesh et.al. ASPLOS'10] [H. Esmaeilzadeh et. al., ISCA'11]

- **Utilization wall [ASPLOS'10]**
  - Assuming 80W power budget, at 45 nm TSMC process, less than 7% of a 300mm^2 die can be switched.
- **Power wall [ISCA'11]**
  - At 22 nm, 31% of a fixed-size chip must be powered off
  - At 8 nm, more than 50%.
- **A significant gap between what is achievable and what is expected by Moore's Law**
  - Due to power and parallelism limitation
  - Speedup gap of at least 22x at 8 nm technology

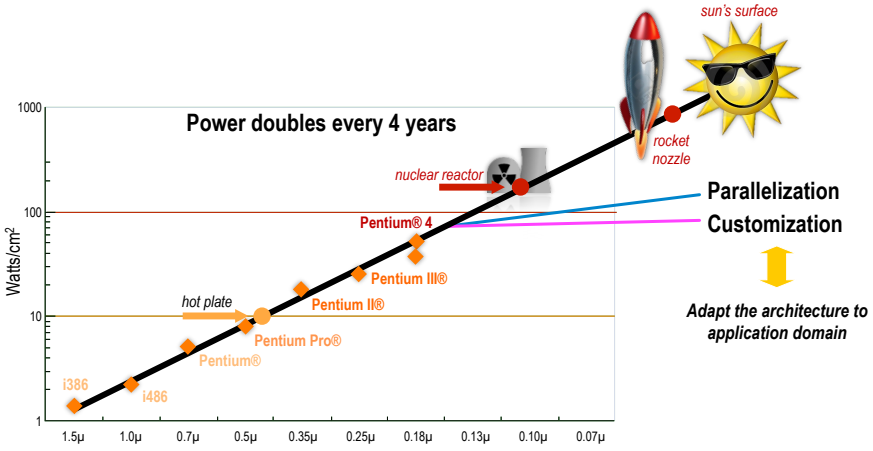


Percent dark silicon: geomean



(a) Conservative Scaling (b) ITRS Scaling

What's Next? Customization and Specialization



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

Potential of 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)

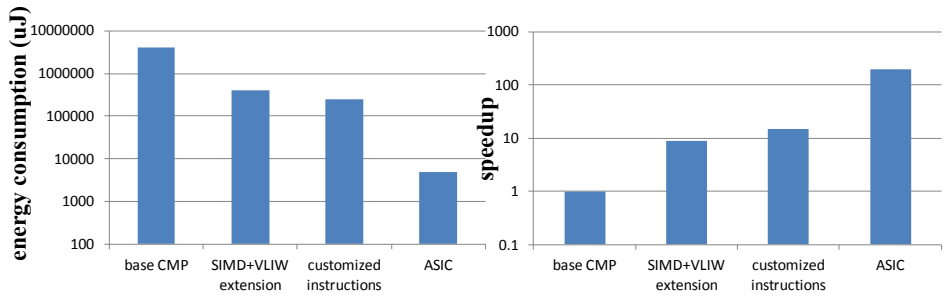
[1] Amphion CS5230 on Virtex2 + Xilinx Virtex2 Power Estimator
[2] Dag Arne Osvik: 544 cycles AES – ECB on StrongArm SA-1110
[3] Helger Lipmaa PIII assembly handcoded + Intel Pentium III (1.13 GHz) Datasheet
[4] gcc, 1 mW/MHz @ 120 Mhz Sparc – assumes 0.25 u CMOS
[5] Java on KVM (Sun J2ME, non-JIT) on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

5

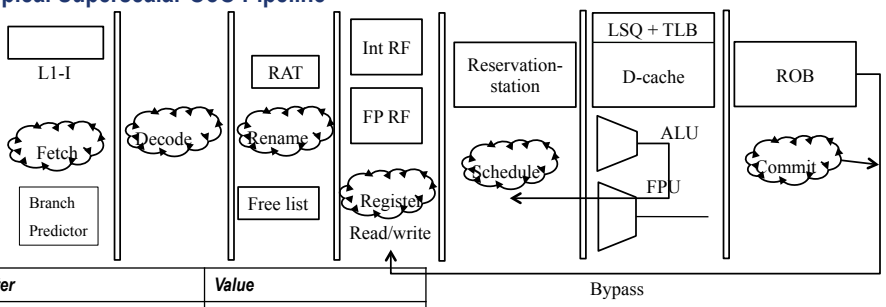
Various Improvements Are Not Enough

- Case study of H.264 [Hameed *et al.*, ISCA'2010]
  - Optimization using SIMD + VLIW → 10x energy efficiency
  - Customized instruction fusion → 1.6x energy efficiency
- Still 50x away from ASICs – Not enough!



Understanding Energy Inefficiency of Processors [DAC'2014]

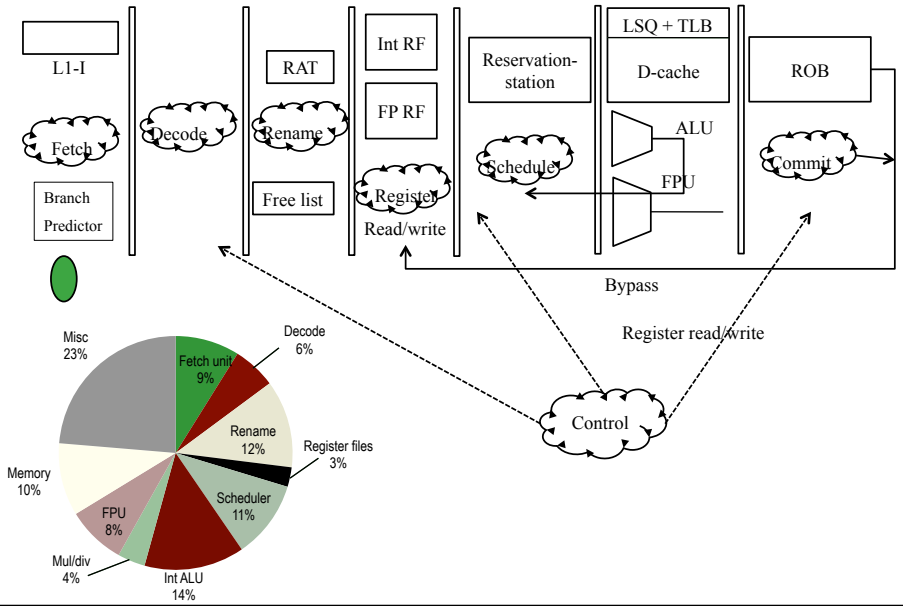
Typical Superscalar OoO Pipeline



Parameter	*Value*
Fetch/issue/retire width	4
# Integer ALUs	3
# FP ALUs	2
# ROB entries	96
# Reservation station entries	64
L1 I-cache	32 KB, 8-way set associative
L1 D-cache	32 KB, 8-way set associative
L2 cache	6 MB, 8-way set associative

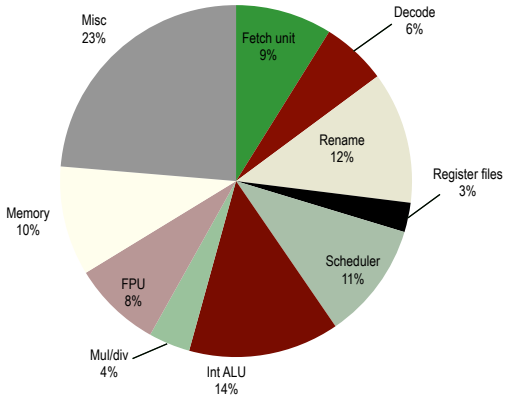
7

Energy Breakdown of Pipeline Components



8

Removing 'Non-Computing' Portions of the Pipeline



- **Remaining**
  - ~ 10% + 26% = 36%
- **Will drop to about 11%**
  - If ALU is replaced by dedicated logic

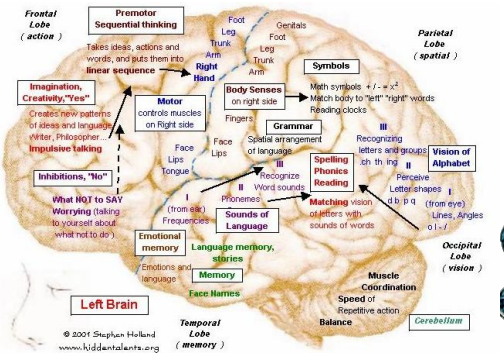
9

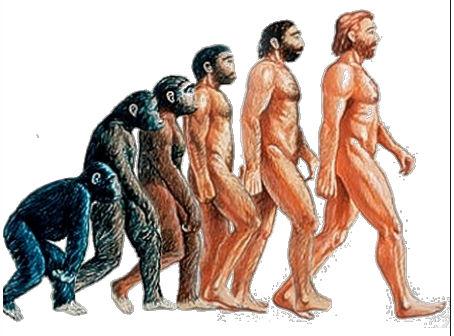
What's Beyond Processors?
Our Proposal – Customized Computing with Accelerator-Rich Architectures

- **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.

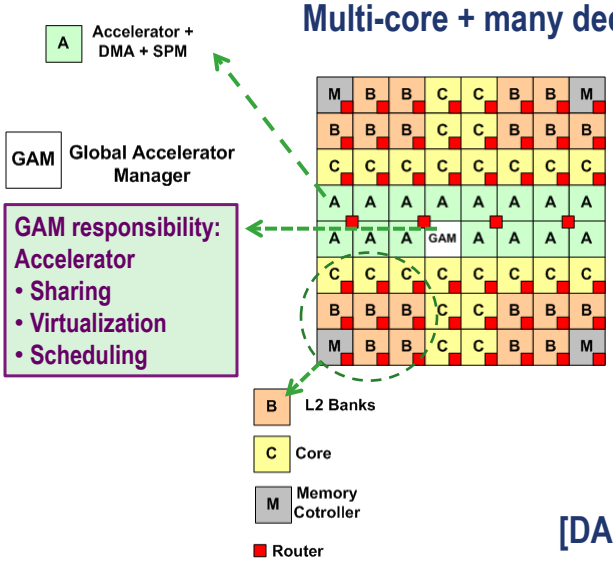
Domain-specific computing holds significant advantages, Cong said. While general-purpose computing relies on computer architecture and languages aimed at any type of application, domain-specific computing utilizes a customizable architecture and custom-oriented, high-level computer languages tailored 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.

Customized Computing at Single-Chip Level

Initial Effort: Accelerator-Rich CMP (ARC) for Domain-Specific Computing

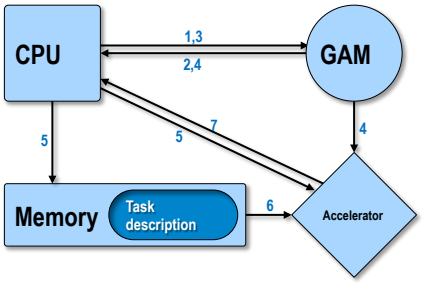
Multi-core + many dedicated accelerators



[DAC'2012]

14

ISA Extension and Overall Workflow in ARC



New ISA
lcacc-req t
lcacc-rsrv id, e
lcacc-cmd id, f, addr
lcacc-free id

1. The core requests for a given type of accelerator (lcacc-req).
2. The GAM responds with a "list + waiting time" or NACK
3. The core reserves (lcacc-rsv) and waits.
4. The GAM ACK the reservation and send the core ID to accelerator
5. The core shares a task description with the accelerator through memory and starts it (lcacc-cmd).
6. The accelerator reads the task description, and begins working
7. When the accelerator finishes its current task it notifies the core.
8. The core then sends a message to the GAM freeing the accelerator (lcacc-free).

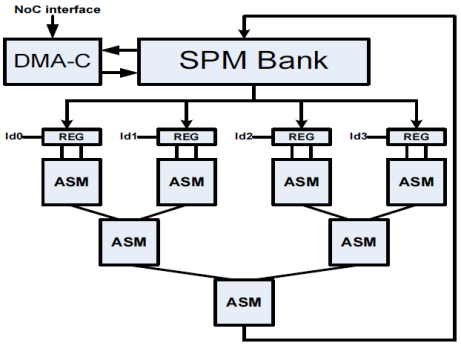
15

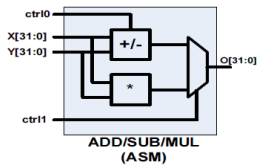
Possibility of Accelerator Composition – Use of Accelerator Building Blocks (ABBs)

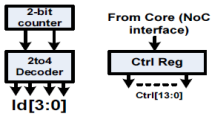
ABBs	Denoise	Deblur	Registration	Segmentation
Float Reciprocal (FInv)	✓	✓		✓
Float Square-Root (FSqrt)	✓	✓	✓	✓
Float Polynomial-16 (Poly16)	✓	✓	✓	✓
Float Divide (FDiv)	✓	✓	✓	✓

Example:

Poly-8 ABB



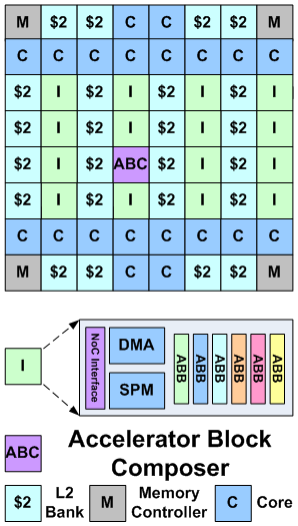




16

Second Effort: Accelerator Composition [ISLPED'12]

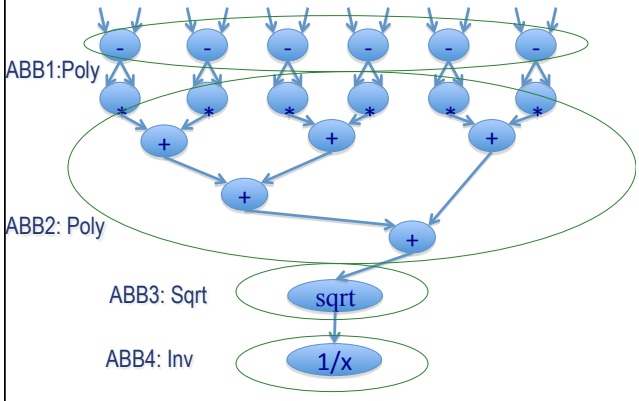
- **ABB**
  - Accelerator building blocks (ABB)
  - Primitive components that can be composed into accelerators
- **ABB islands**
  - Multiple ABBs
  - Shared DMA controller, SPM and NoC interface
- **ABC**
  - Accelerator Block Composer (ABC)
    - Runtime composition of virtual accelerators from ABBs
    - Arbitrate requests from cores
- **Other components**
  - Cores
  - L2 Banks
  - Memory controllers
- **Accelerator composition: Static mapping + dynamic allocation**

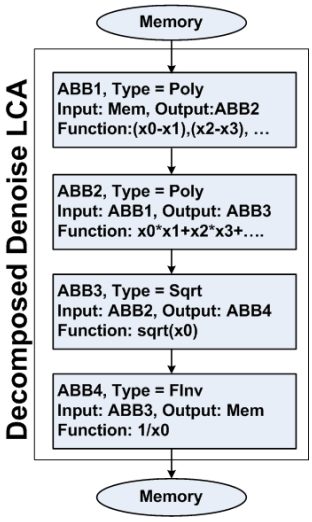


17

Static Mapping/Decomposition into ABBs

$$1 \Big/ \sqrt{\sum_{i=0}^{6} (Xi - Y)^2}$$





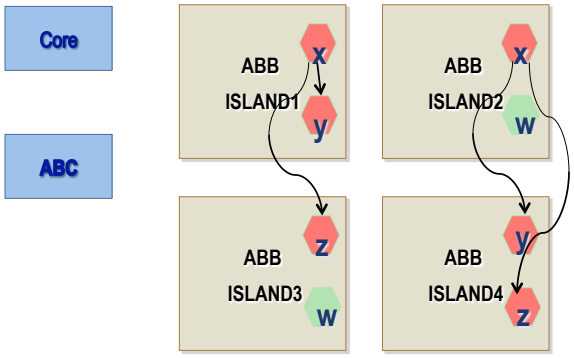
18

Accelerator Composition Process at Runtime

Accelerator cloning

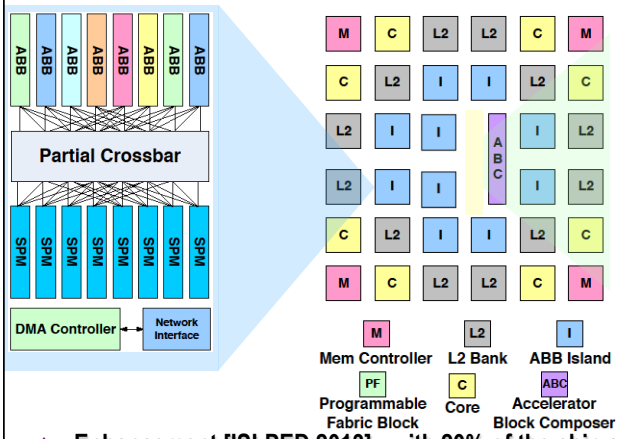
- Repeat to generate more LCAs if ABBs are available

Core ID	ABB Type	ABB ID	Status
1	x	1	Busy
1	y	1	Busy
2	x	1	Busy
2	w	1	Free
3	z	1	Busy
3	w	1	Free
4	y	1	Busy
4	z	1	Busy



19

Latest Effort: Composable Accelerators with Programmable Fabrics [ISLPED'2013]



Dynamic Resource Allocation of ABBs

- **Enhancement [ISLPED 2013]: with 20% of the chip area dedicated to programmable fabric, we can achieve more:**
  - **Flexibility:** An average 8.2x (up to 146x) speedup in other domains, such as commercial, vision and navigation
  - **Longevity:** 22x speedup on a new application within the medical imaging domain

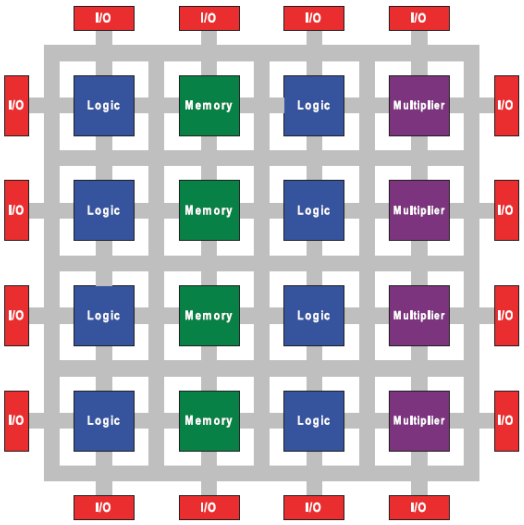
20

Customized Computing at Server Node Level

21

Modern FPGA Architecture – A Simplified View

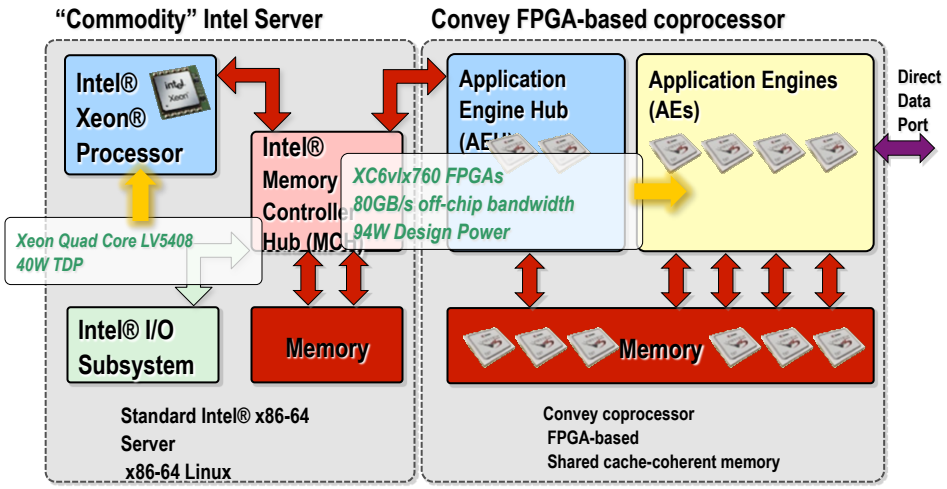
- **Configurable logic block**
  - Lookup table
- **Island-style configurable mesh routing**
- **Dedicated components**
  - Specialization allows optimization
  - Memory,
  - DSP (multiplier) blocks
  - Embedded processor
  - High-speed I/Os
  - Anything that the FPGA architect wants to put in!



Source: I. Kuon, R. Tessier, J. Rose. FPGA Architecture: Survey and Challenges. 2008.

22

Example of CDSC Heterogeneous Computing Platform



23

Server Node-level Acceleration Example : 3D CT-Reconstruction Algorithm (EMTV) [FCCM'2015]

- **Optimization techniques:**
  - Data reuse
  - Prefetching
  - Pipelining
  - Fixed-point optimization
  - Parallelization

Platform	Exec Time	Speedup
CPU	54.6 hours	1
GPU	110 mins	29.8X
FPGA	38.8 mins	84.5X

Sinogram reuse

Sinogram + voxel reuse

Server Node-level CPU + FPGA Integration

- **Press releases**

Xilinx Demonstrates Industry's First QPI 1.1 Interface with FPGAs at Intel Developer Forum

...h Interconnect enables 7 series All Programmable FPGAs; extends ...lines of Intel processor-based systems

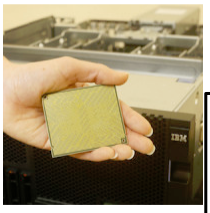
...wire
...RANCISCO
..., 2012

...RANCISCO, Sept. 11, 2012 /PRNewswire/ -- Xilinx, Inc. (NASDAQ: XLNX) today ...ced at the Intel Developer Forum (IDF) the industry's first demonstration of a ...grammable gate array (FPGA) interfacing to an Intel Sandy Bridge Xeon ...sor using the QuickPath Interconnect (QPI) protocol. Xilinx's QPI solution ...rs

April 23, 2014

What Power8 and OpenPOWER Might Mean for HPC

Timothy Prickett Morgan



IBM is making a big play in hybrid computing, seeking to marry its POWER8 processors with various kinds of accelerators and high-speed networking and opening up its chip and system software through the ...OpenPOWER Foundation. (What Open ...

IBM is working with FPGA makers Xilin... running over the CAPI interface, so this... the Impact2014 event, IBM and Xilinx w... being accelerated by FPGAs and show... order of magnitude lower latency. A Mo... machines accelerated by Altera FPGAs... adapter and switch maker Mellanox Te... using Remote Direct Memory Access (... boosted throughput and cut latencies ...

April 10, 2013

Altera Demonstrates Industry's First QPI 1.1 FPGA Home Agent for Enhanced Server Capabilities

Demo Features Stratix V FPGA Configured to Extend Co-processing for Intel Processors

San Jose, Calif., and IDF Beijing, April 10, 2013—Altera Corporation (NASDAQ: ALTR) today announced the industry's first demonstration of an FPGA Home Agent enabled by the Intel QuickPath Interconnect (QPI) protocol 1.1. Connecting to Intel's Sandy Bridge XEON processors, the demonstration leverages an Altera® Stratix® V FPGA configured as the Home Agent, and it supports both the Caching Agent and Home Agent in a Pactron Vigor Development Platform. This solution is ideal for designers of low-latency signal-processing, packet processing and embedded applications, such as high-frequency trading and big data that need higher computation performance-per-watt than traditional CPU configurations can deliver. Altera is demonstrating its QPI 1.1 intellectual property (IP) solution to support both the Caching Agent and Home Agent in a Pactron Vigor Development Platform at the Intel Developers Forum (IDF) Beijing, April 10-11, in Altera's booth #E120.

QPI is the only way to coherently connect to an Intel server processor. The Altera StratixV FPGA transceiver has been qualified to support the Intel QPI electrical specification at 8 Gbps. Developers of low-latency, high-bandwidth systems looking to extend the flexible shared memory model that Intel uses for x86 programming can now efficiently integrate a Stratix V FPGA into their systems. The Home Agent demo addresses 32 GB of memory on the motherboard connected to the socket with support for two channels connecting to four 8 GB RDIMMs.

"Our QPI 1.1 solution provides developers of data centers and high-performance computing applications a platform to significantly increase their compute performance while reducing system cost and power," said David Gamba, director of the compute and storage product line at Altera. "FPGAs deliver a highly effective, efficient way to speed the processing of large data sets through parallel processing and accelerated data transfers."

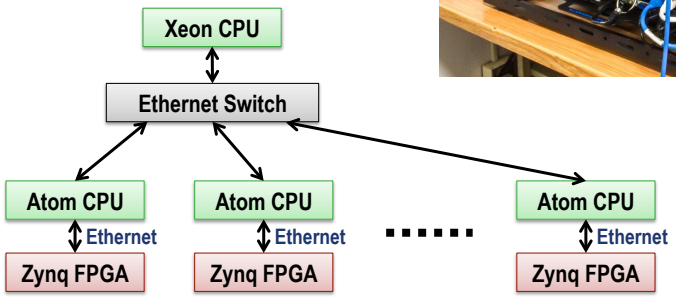
25

Customized Computing at Cluster and Datacenter Levels

26

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

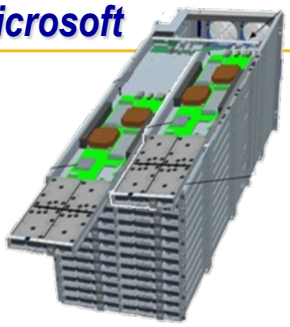
- Deployed in 2013
- Used for research and teaching
  - CS133 (60+ students)
  - CS259 (18 students)



27

Datacenter Level Integration at Microsoft

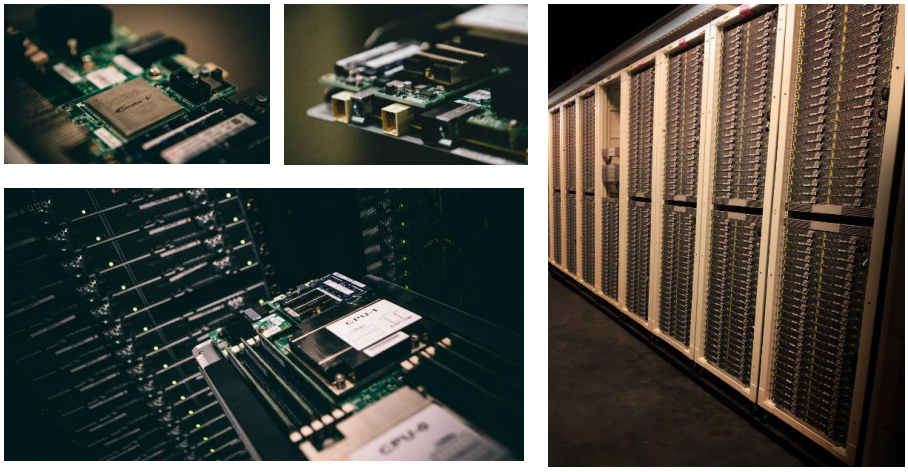
- Microsoft Open Compute Server
- 1U, ½ wide servers
- Enough space & power for ½ height, ½ length PCIe card
- Squeeze in a single FPGA
- Won't fit (or power) GPU



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

28

1632 Server Pilot Deployed in a Production Datacenter

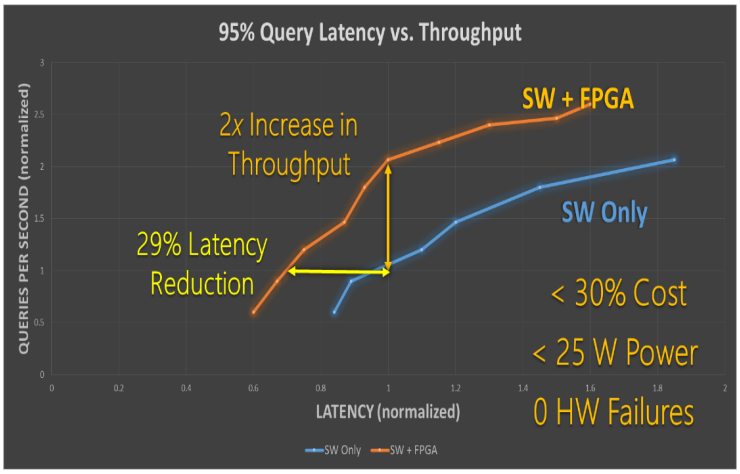


A. Putnam, "*A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services*", ISCA'2014

29

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

30

How to Program Such "Beasts"?

-- "Write Once, Compile Anywhere"

31

UCLA COMPUTER SCIENCE DEPARTMENT

FPT'2005 Keynote

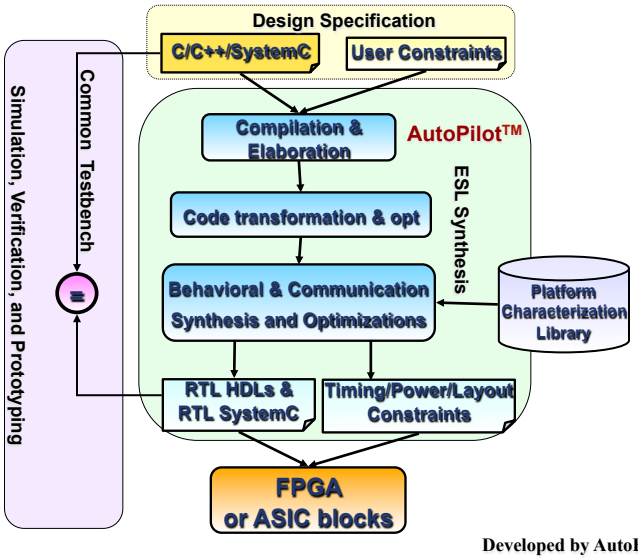
Platform-Based Synthesis for Field-Programmable SOCs

Prof. Jason Cong

cong@cs.ucla.edu

UCLA Computer Science Department

1. C/C++ Based Synthesis for Dedicated 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
- …

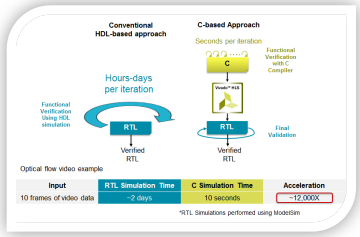
QoR matches or exceeds manual RTL for many designs

Developed by AutoESL, acquired by Xilinx in Jan. 2011

33

Vivado High-Level Synthesis:
Production-Proven and Adopted by 1000+ companies

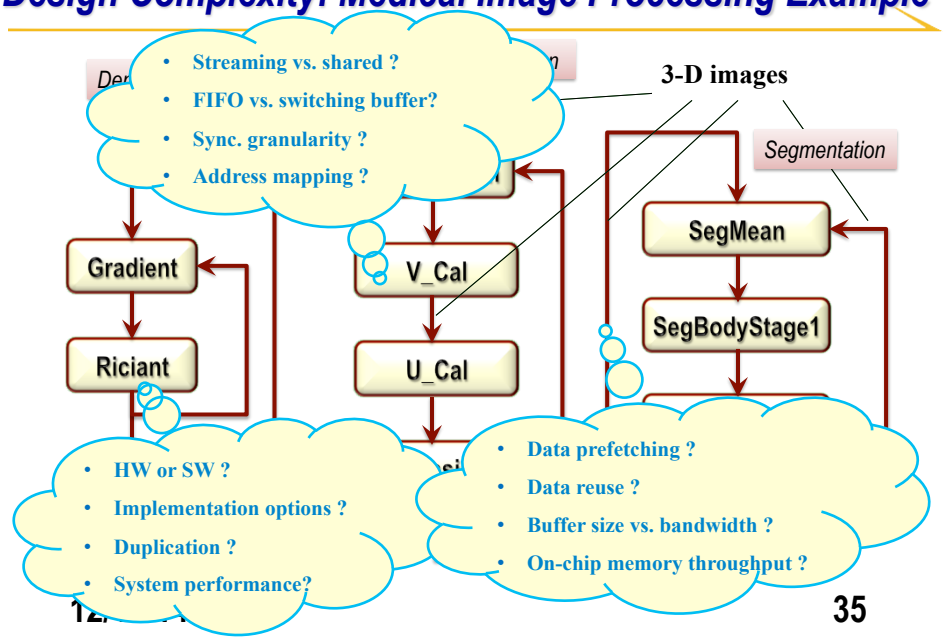
- **C libraries:**
  - Arbitrary precision
  - Floating-point math.h
  - OpenCV video functions
  - DSP
  - Linear algebra
- **Accelerated verification**
  - >100X faster than RTL design
- **Fast compilation and design exploration**
  - Algorithm feasibility
  - Architecture Iteration



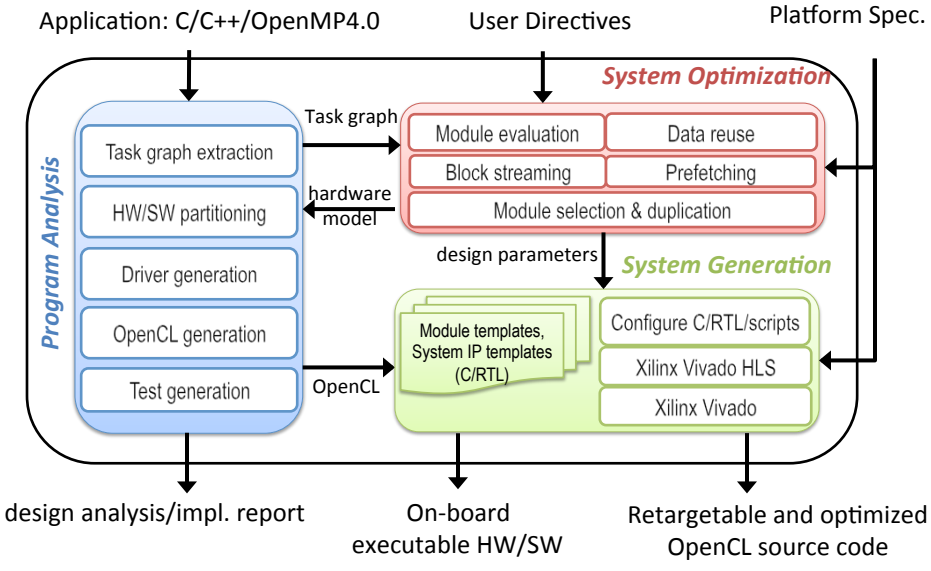
	Hand-coded RTL	Vivado HLS
Design Time (weeks)	12	1
Latency (ms)	37	21
Memory (RAMB18E1)	134 (16%)	10 (1%)
Memory (RAMB36E1)	273 (65%)	138 (33%)
Registers	29686 (9%)	14263 (4%)
LUTs	28152 (18%)	24257 (16%)

Customer proven results

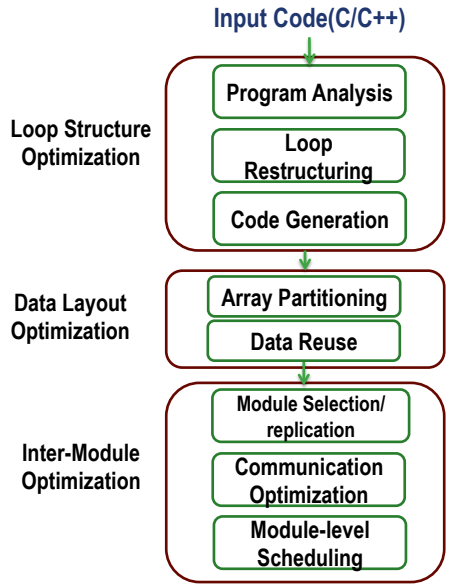
Design Complexity: Medical Image Processing Example



CMOST: Fully Automated Compilation and Mapping Flow



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

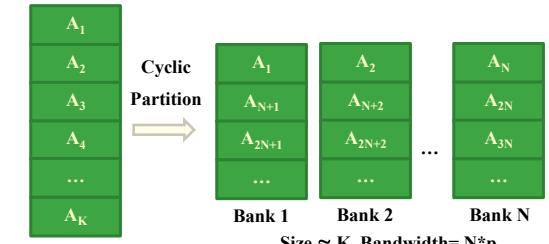
An optimal microarchitecture for stencil computation acceleration based on non-uniform partitioning of data reuse buffers (DAC'14)

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

37

Memory Partitioning for Throughput Optimization

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

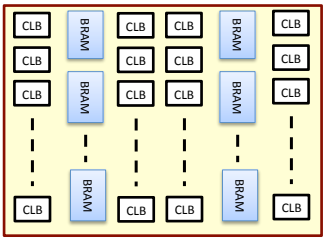


Size = K, Bandwidth = p

p: memory port number

Size ≈ K, Bandwidth= N*p

N: Partition Factor



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

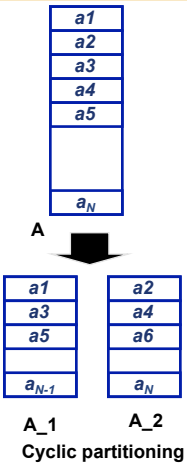
38

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)

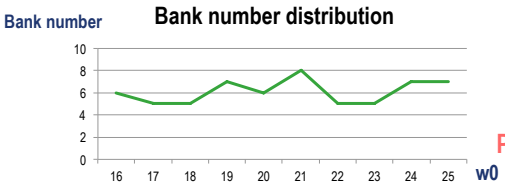
39

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:

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

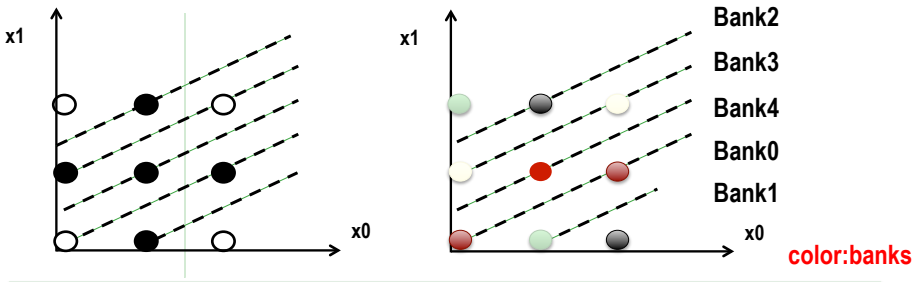
$N \nmid (w0-1)$ $N \nmid (w0+1)$

Partition results are related to array sizes!

40

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



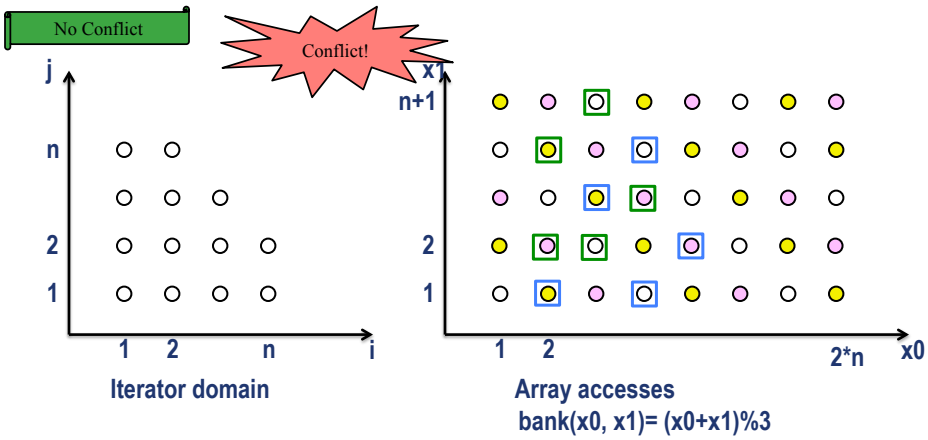
$\forall i,j$, A[a_1*i+b_1][c_1*j+d_1] not conflict with A[a_1*i+b_1][c_1*j+d_1]
$\Leftrightarrow$ gcd((α_1 (a_1−a_2)+α_2 (c_1−c_2)), N) $\nmid$ (α_1 (b_1−b_2)+α_2 (d_1−d_2))

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

41

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]);
```



42

Conflict Polytope

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

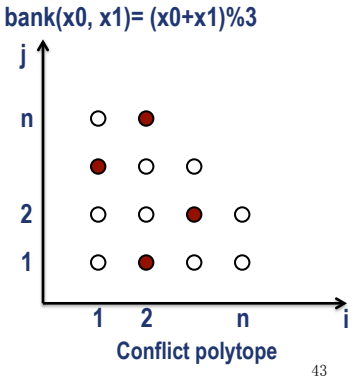
$$\begin{cases} 1 \le i \le n \\ 1 \le j \le n \\ i + j \le n + 2 \\ (i+1) + (j+1) \equiv (2i + j) \bmod 3 \end{cases}$$

```
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

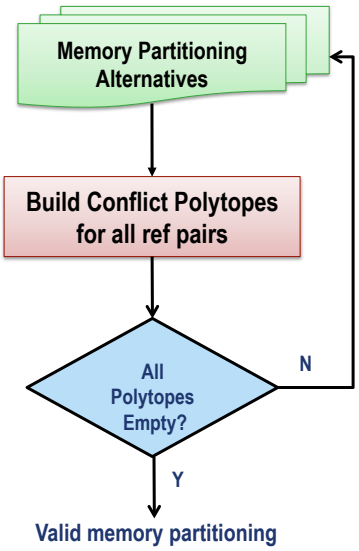
$$(i+1) + (j+1) = (2i + j) + 3k$$

- 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*



43

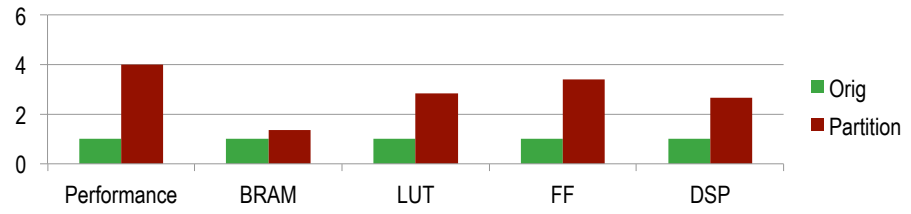
Generalized Memory Partitioning (GMP)



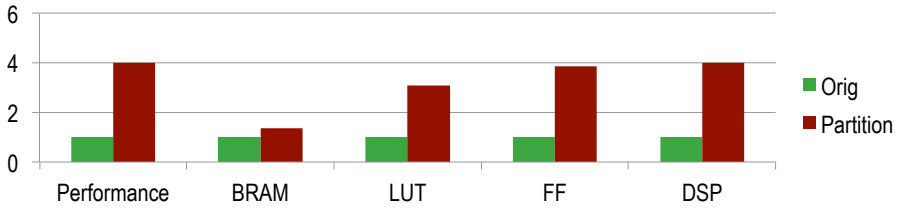
- Complexity independent of
  - Sizes of iteration domain
  - Sizes of array
- Complexity related to
  - Dimensions of iteration domain and array
    - In real case <= 3
  - Number of References
    - In real case <= 100
- Number of memory partitioning alternatives ≤ number of references

44

Experimental Results for Denoise



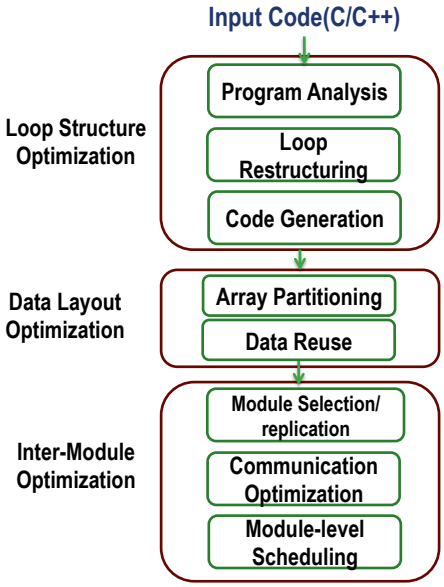
Gradient (II 4->1)



Rician (II 4->1)

45

Automating Customized Computing – Another Example

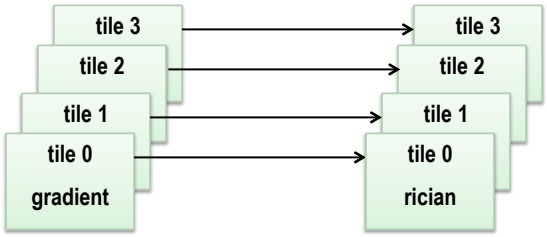


Combining Computation with Communication Optimization in System Synthesis for Streaming Applications, FPGA'14

46

Motivation

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



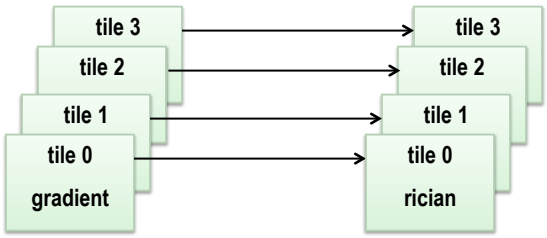
- Which implementation to use for each module?
  - Memory partitioned v.s. Memory non-partitioned

	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

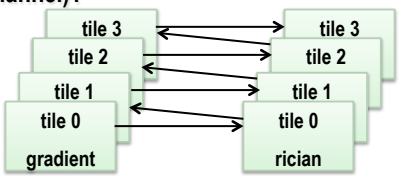
47

Motivation

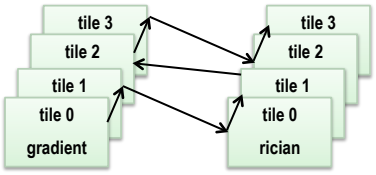
Tile size: 32x32
Image: 64x64, 4 tiles



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



scheduling 0 →1 tile

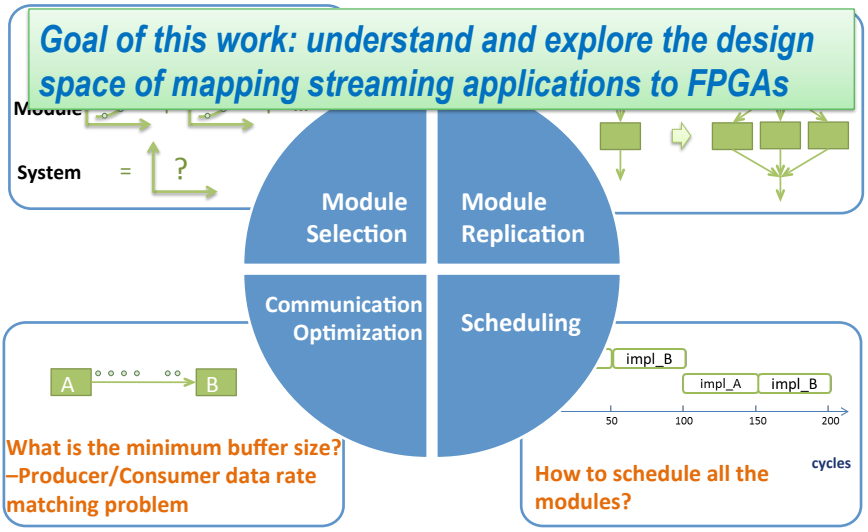


scheduling 0 →2 tiles

48

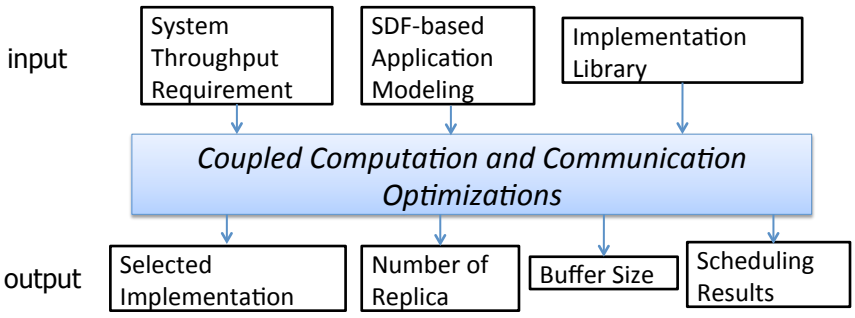
System-Level Synthesis for Streaming Applications

Goal of this work: understand and explore the design space of mapping streaming applications to FPGAs



49

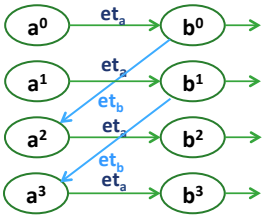
Proposed System Synthesis Framework



50

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



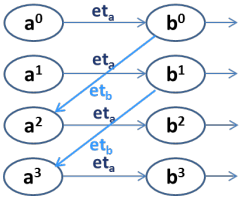
Module latency constraints
et_a: execution time of task a
et_b: execution time of task b

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

51

Formulation (2/2)

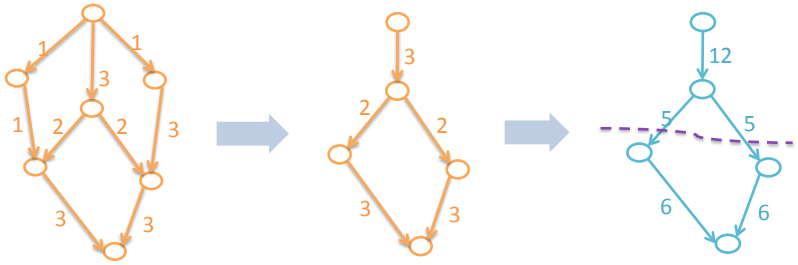
- Associate each node with a scheduling variable
  - $t(b^0) - t(a^0) >= et_a$
  - $t(a^2) - t(b^0) >= et_b$
  - ...
  - Scheduling variables are integer variables



- 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

52

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,*
- or more aggressively, *(1-ε)*maxL*

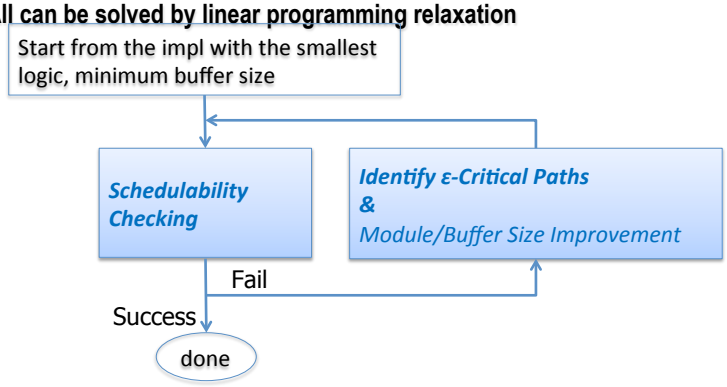
Module Improvement

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

53

Streaming Synthesis (ST-Syn)

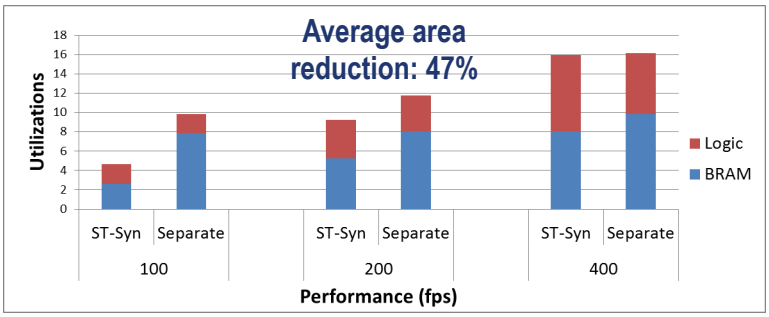
- ***Formulation*** – ***Schedulability checking***
  - System of Difference Constraint Problem
- ***Exploration*** – ***Identify critical path, module/buffer improvement***
  - Find ε-critical paths in the scheduling graph
  - Minimum cut problem
- **All can be solved by linear programming relaxation**



54

Experiments on 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



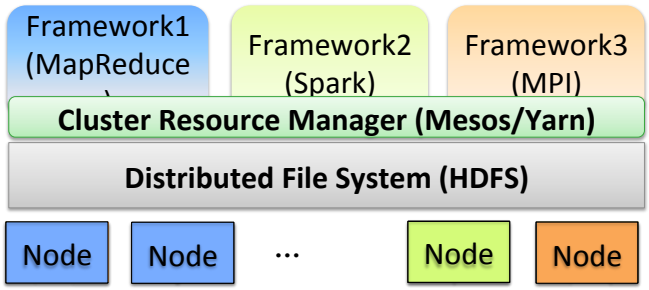
55

BUT HOW DO WE MANAGE SUCH HETEROGENEOUS RESOURCES?

56

Overview of Cluster Resource Management

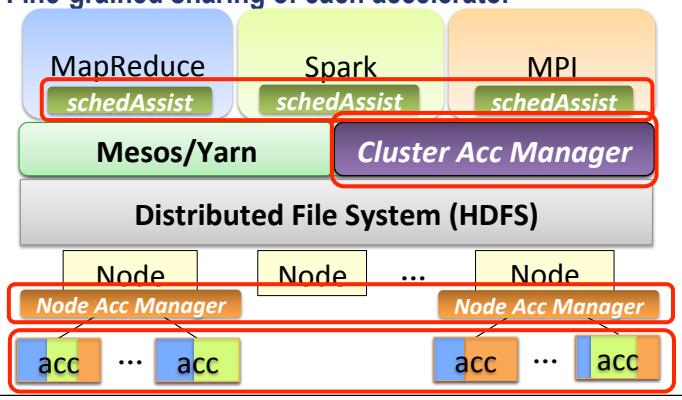
- Cloud computing frameworks targeting different workloads
  - MapReduce, Spark, Storm, MPI, etc.
- Each framework has a finite set of cluster resources
  - CPU, memory, disk bandwidth, etc.
  - Fine-grained sharing enabled by cluster resource manager



57

Our Cluster Accelerator Resource Management

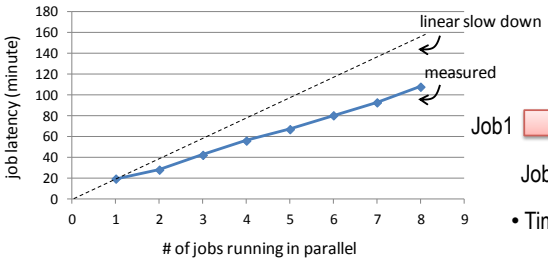
- How to make sure acc resources are efficiently utilized?
  - Each framework get the accelerator resources it deserves
  - Tasks in each framework are assigned to their preferred node
  - Fine-grained sharing of each accelerator

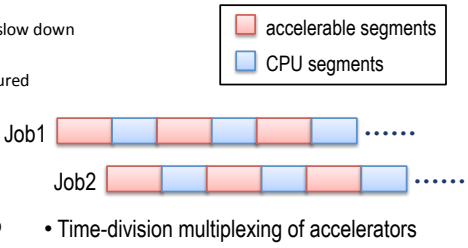


58

Initial Experimental Results

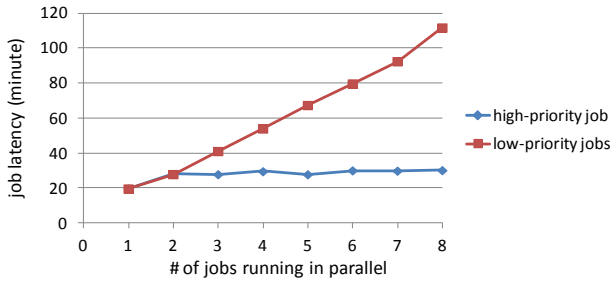
- Running multiple jobs of the same type on a 5-node FPGA cluster





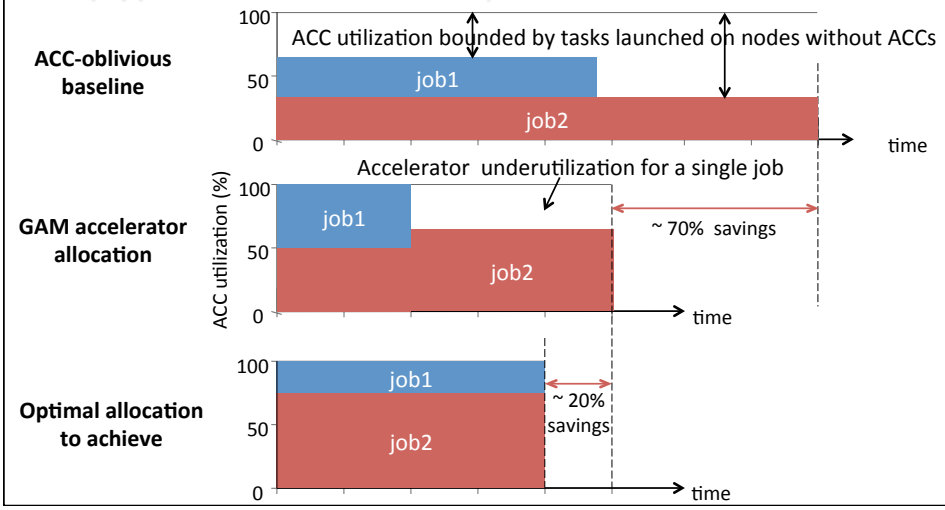
- Time-division multiplexing of accelerators

- Priority-aware accelerator mapping



Initial Experimental Results (Cont'd)

- Running two jobs of different sizes on a 10-node cluster (5 nodes equipped with FPGA accelerators)



Concluding Remarks

- **New era of computing**
  - Future computing platforms will have a sea-of-accelerators
  - With efficient support for customization and specialization
- **Customized computing at all levels**
  - Chip-level
  - Server node level
  - Data center level
- **Software is the key**
  - Programming models
    - Hopefully transparent to the programmer: Hadoop/MapReduce + C/C++, …
  - Compilation support
  - Runtime management

61

Acknowledgements – CDSC and C-FAR

- **Center for Domain-Specific Computing (CDSC) under the NSF Expeditions in Computing Program and 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)

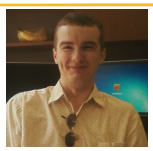

Sarkar (Associate Dir) (Rice)

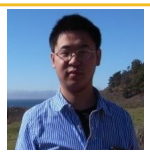

Vese (UCLA)

62

Postdoc and Graduate Students in Collaboration


Mohammad Ali Ghodrat
(UCLA)

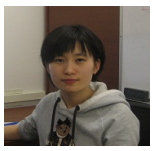

Michael Gill
(UCLA)


Yi Zou
(UCLA)


Beayna Grigorian
(UCLA)


Chunyue Li
(UCLA)


Prof. Deming Chen
(UIUC/ADSC)


Hui Huang
(UCLA)

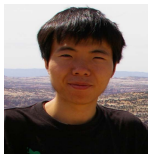

Muhuan Huang
(UCLA)


Dr. Peng Li
(UCLA)


Prof. Louis-Noël Pouchet
(UCLA)


Bo Yuan
(UCLA)


Yuxin Wang
(PKU)


Bingjun Xiao
(UCLA)


Dr. Peng Zhang
(UCLA)


Wei Zuo
(UIUC)

63

Thanks!

64