Coarse-Grained Reconfigurable Architectures

•••

Architecture, Applications and Challenges

Bharathwaj Suresh, Chandrakanth, Mark Kubiak,
Sreya Banerjee, Utkarsh Sharma, Yuchong Luo

Agenda

- **CGRA introduction**
  - CGRA concept
  - CGRA vs. FPGA
- **CGRA Architectures**
  - SRP
  - Plasticine
  - Multi-CGRA
- CGRA Tools
  - OpenCGRA
  - Stanford AHA

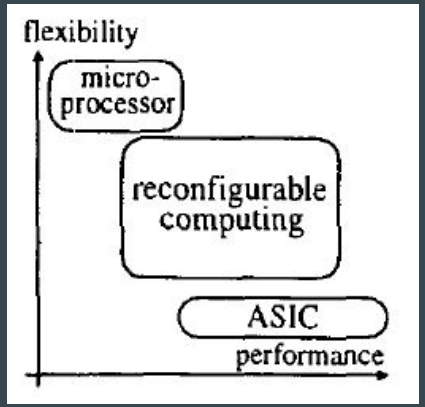
CGRA Introduction

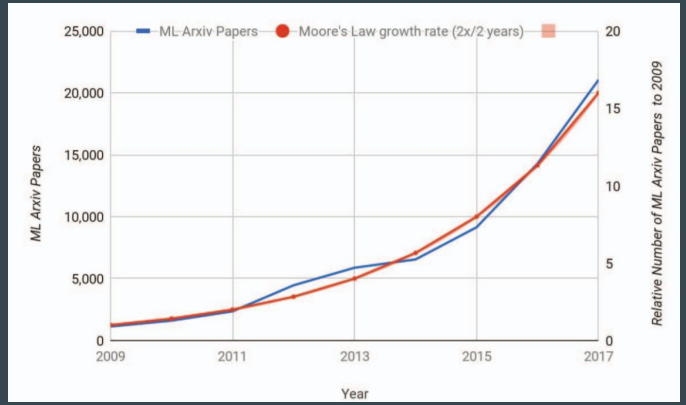
...

Background

- Algorithmic and technological innovations placed great demands on computing systems
- ASICs have high performance but no flexibility.
- General purpose processors have good flexibility but low power efficiency
- FPGAs suffer from programming inefficiencies

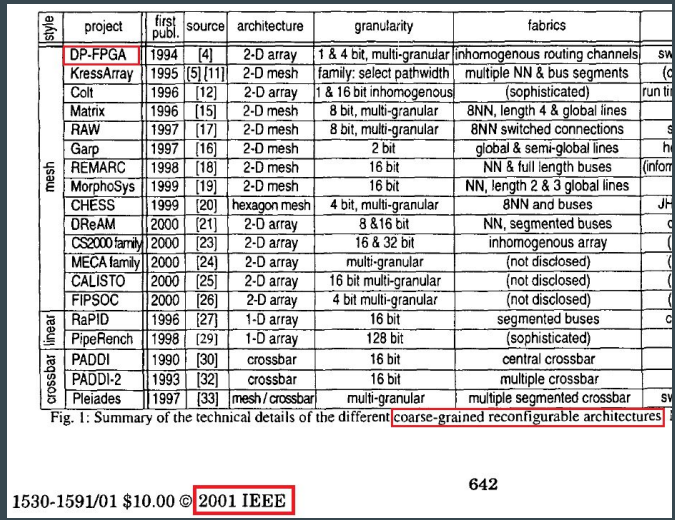
-> Therefore, new architecture is needed





Coarse-grained Reconfigurable Architectures(CGRA)

- The architecture was first introduced in 1990s
- Before being classified as CGRA, they were treated as optimized FPGA
- Around 2000, the terminology ‘CGRA’ appeared, and it is widely used since then



style	project	first publ.	source	architecture	granularity	fabrics	
mesh	DP-FPGA	1994	[4]	2-D array	1 & 4 bit, multi-granular	inhomogenous routing channels	sw
	KressArray	1995	[5] [11]	2-D mesh	family: select pathwidth	multiple NN & bus segments	(c
	Colt	1996	[12]	2-D array	1 & 16 bit inhomogenous	(sophisticated)	run ti
	Matrix	1996	[15]	2-D mesh	8 bit, multi-granular	8NN, length 4 & global lines	
	RAW	1997	[17]	2-D mesh	8 bit, multi-granular	8NN switched connections	s
	Garp	1997	[16]	2-D mesh	2 bit	global & semi-global lines	h
	REMARC	1998	[18]	2-D mesh	16 bit	NN & full length buses	(inform
	MorphoSys	1999	[19]	2-D mesh	16 bit	NN, length 2 & 3 global lines	
	CHESS	1999	[20]	hexagon mesh	4 bit, multi-granular	8NN and buses	JH
	DReAM	2000	[21]	2-D array	8 &16 bit	NN, segmented buses	c
	CS2000 family	2000	[23]	2-D array	16 & 32 bit	inhomogenous array	(
	MECA family	2000	[24]	2-D array	multi-granular	(not disclosed)	(
	CALISTO	2000	[25]	2-D array	16 bit multi-granular	(not disclosed)	(
	FIPSOC	2000	[26]	2-D array	4 bit multi-granular	(not disclosed)	(
linear	RaPID	1996	[27]	1-D array	16 bit	segmented buses	c
	PipeRench	1998	[29]	1-D array	128 bit	(sophisticated)	
crossbar	PADDI	1990	[30]	crossbar	16 bit	central crossbar	
	PADDI-2	1993	[32]	crossbar	16 bit	multiple crossbar	
	Pleiades	1997	[33]	mesh / crossbar	multi-granular	multiple segmented crossbar	sw

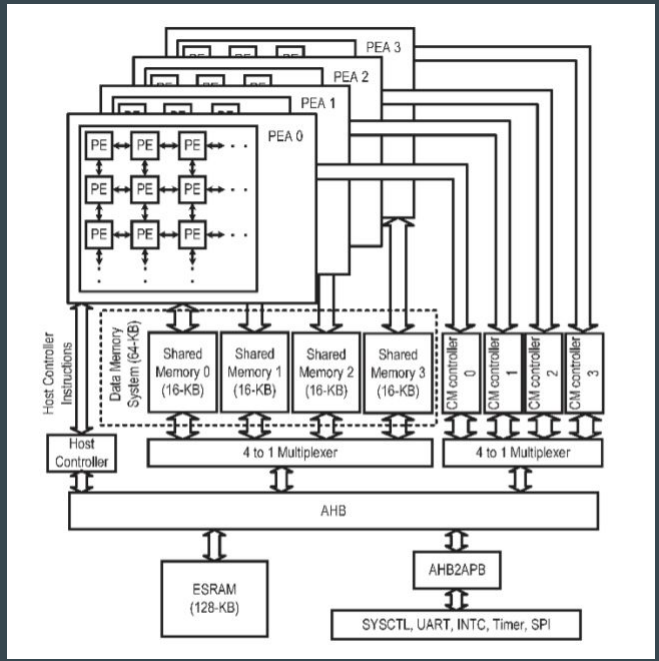
Fig. 1: Summary of the technical details of the different coarse-grained reconfigurable architectures

642

1530-1591/01 $10.00 © 2001 IEEE

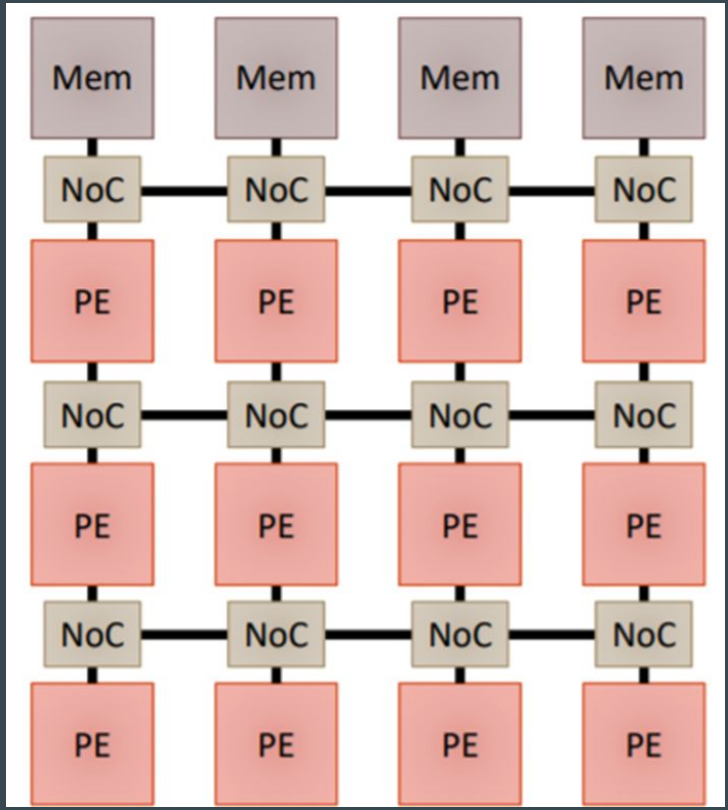
CGRA Feature

- Domain-Specific Flexibility
- Capability of performing data-intensive computation
- Short reconfigured time (High programmability)
- Data-driven architecture



Reconfigurable computing

- We are trying to bridge the gap between general purpose CPUs and accelerators.
- **Basic Idea:** A spatial array of processing elements(PE) and memories with a configurable network.
- We map the computation spatially onto the array.
- Goal is to strike a balance between performance and programmability.

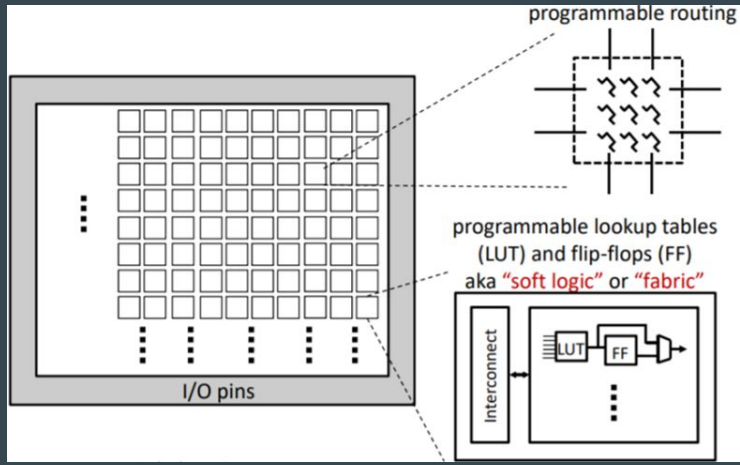


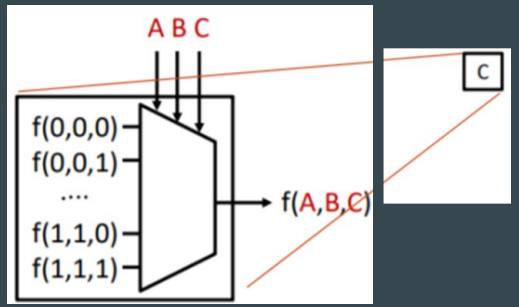
FPGA: Predecessor to CGRA

- Routing is super expensive. ~50% area.
- Each CLB includes LUTs and registers.
- LUTs implement arbitrary logic.
- Smaller LUTs are faster but need multiple LUTs for wide logic(more routing).
- Big LUTs may involve wastage of HW if unused.

Limitations/Reasons to move to CGRA:

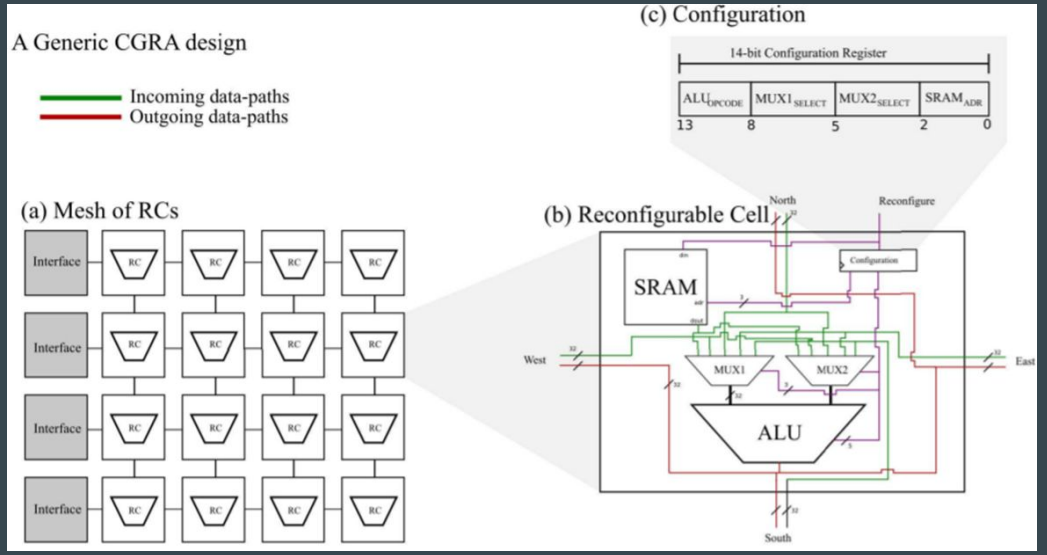
- Low level coding in hardware(Verilog, VHDL).
- Synthesis can take many hours. For eg: In emulation, when a peripheral IP is being given for build, it takes 3hrs, while its sub-system takes at-least 12 hrs.
- Do you need per bit control? In FPGAs you literally need control signals for individual bits to be routed.
- Have a specific application on mind? All bits of word operations are routed from same source to same destination.
- Application specific operations such as frequent additions, multiplications etc can be much more efficiently implemented directly than LUTs.
- Compilation can be easier with CGRAs.
- Lower operating frequencies.
- Also certain logic and pin interfaces may not map well onto an FPGA



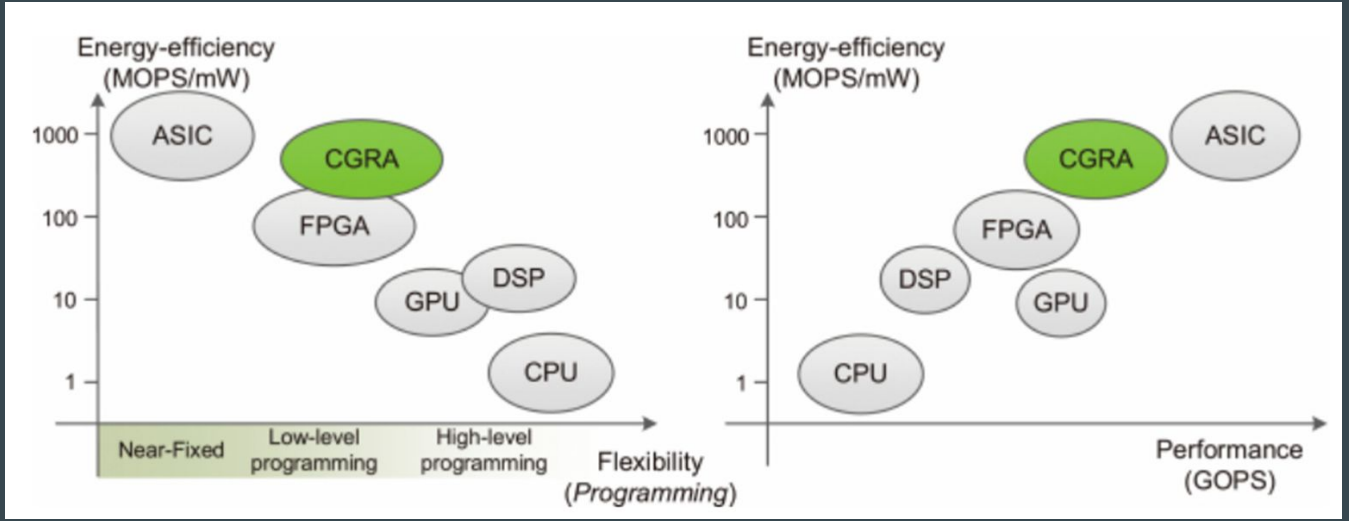


Generic CGRA Architecture

- Increased native word size from 1 bit → multibit.
- Clearly a tradeoff exists between word size and configurability.
- Reduces compilation and reconfiguration times substantially.
- By increasing the granularity larger, specialized units can be built thus operating on higher frequencies.
- Coarser nature helps map complex operations and timing related issues can be resolved.



CGRA vs ASIC, FPGA, CPUs



- ASIC has highest efficiency with least flexibility and highest performance.
- CPUs have least efficiency with high level of flexibility and low performance.
- FPGAs and CGRAs try to bridge this gap, with CGRAs being closer to ASIC in terms of efficiency and performance.

CGRA and other fabrics

Table 1. Comparisons Between CGRAs and Important Computing Fabrics

Architecture	Flexibility	Computation form		Execution mechanism			
		Temporal	Spatial	Reconfiguration time[4]	Configuration-driven	Dataflow-driven	Instruction-driven
CGRA	Domain	√	√	ns-μs	√	√	×
FPGA	General	×[1]	√	ms-s	√	√	×
ASIC	Fixed	×[2]	√	×	×[3]	√	×
In-Order Processor/ VLIW	General	√	×	ns	×	×[5]	√
Out-of-Order Processor	General	√	×	ns	×	√	√
Multicore	General	√	√	ns	×	×[5]	√

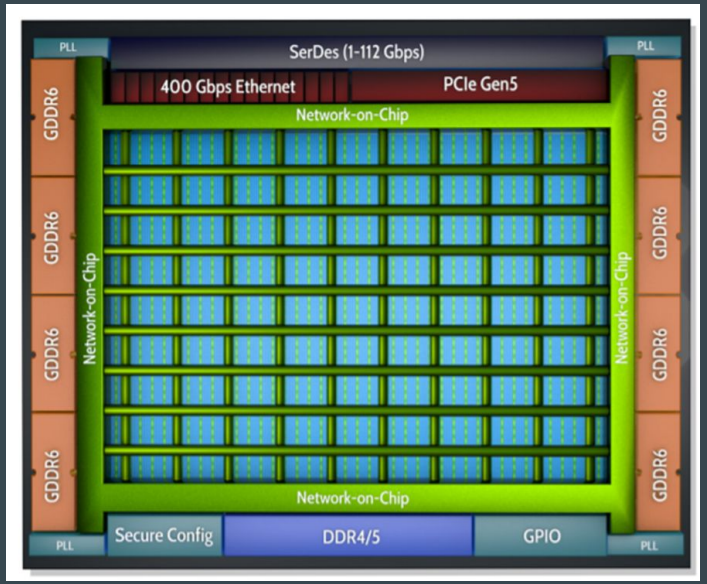
- Combining spatial and temporal computations gives CGRAs high energy/area efficiency

Modern FPGAs:

- FPGAs getting coarser.
- Hardened logic in LUTs.
- DSP blocks available to implement ops efficiently.

Speedster:

- 7nm chip.
- 2.6M 6-input LUT
- 2D NOC which supports > 20Tbps. à Blending FPGA programmability with ASIC like routing and compute engines.
- Contain machine learning processor blocks(MLP).
- MLPS are configurable, contain 32-bit MAC operators, integer and floating point support.
- MLPS have integrated memory blocks to ensure ML algorithms run at maximum performance.



Speedster

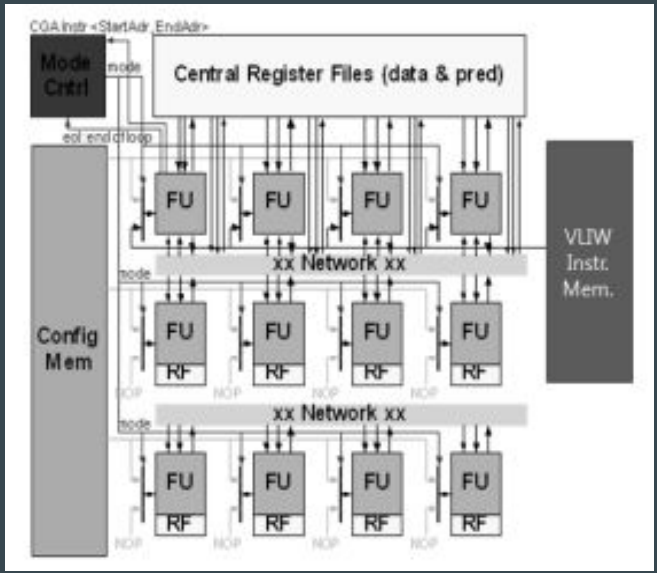
CGRA Architectures

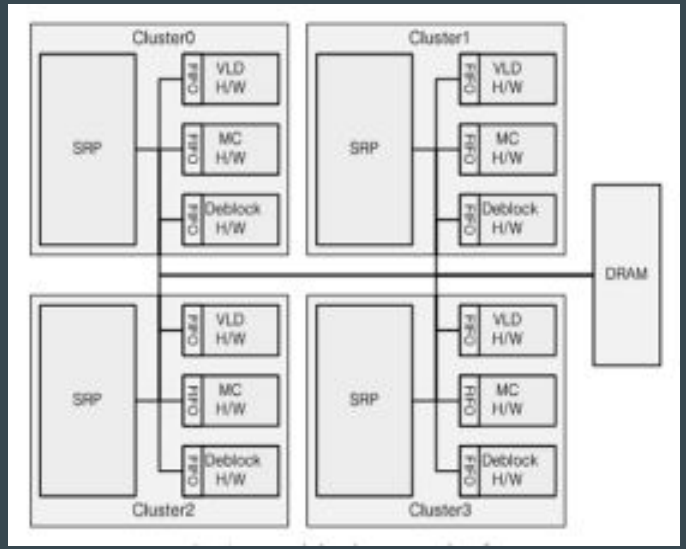
...

Samsung Reconfigurable Processor (SRP)

- H.264/AVC decoder for UHD applications
- Hybrid parallelization (DLP + TLP)

[S. Lee, ICIP 2011]

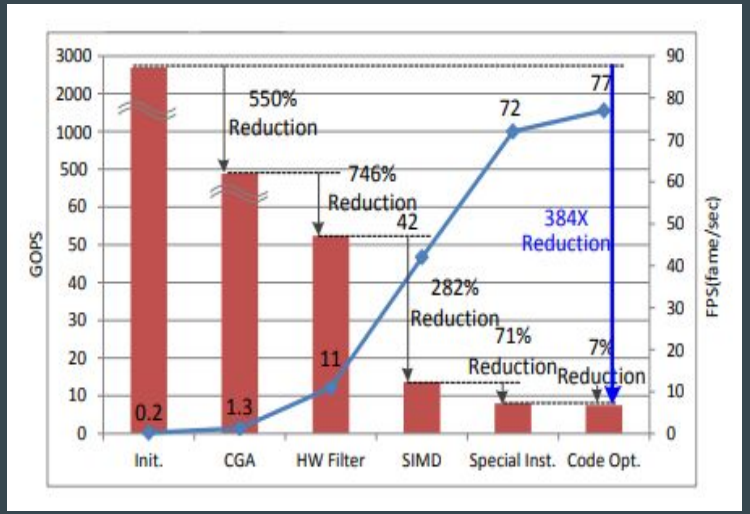




SRP's Architecture & Performance

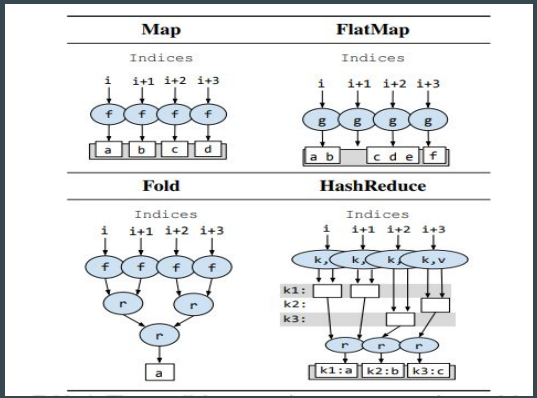
- Two modes:
  - VLIW mode: 2 Scalar, 2 Vector FUs
  - CGRA mode: 8 Scalar , 8Vector FUs (127GOPS)
  - 650 MHz @ 28nm Samsung Tech.
  - 16 Channel DMA controller

[S. Lee 2011]

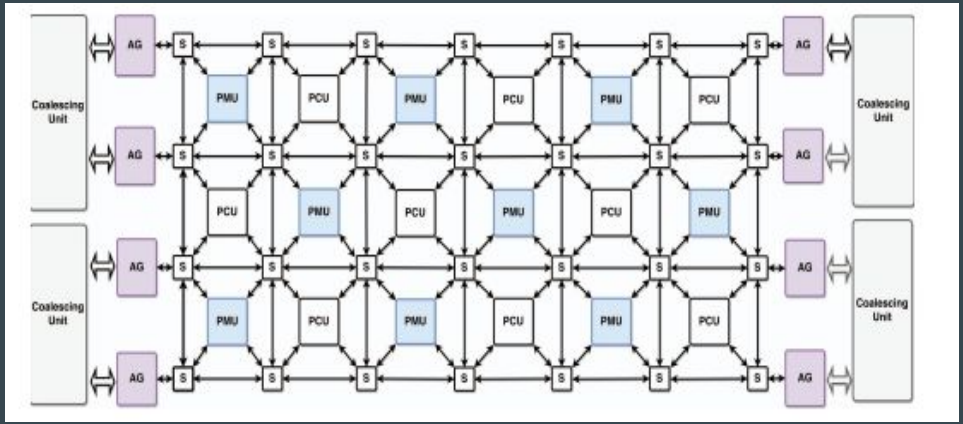


Plasticine

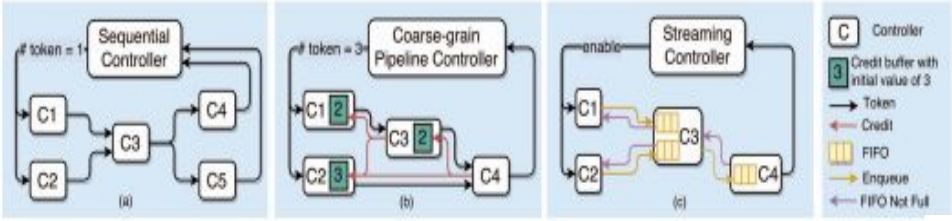
- Co-design the Architecture with high-level programming abstractions.
- Efficiently execute Parallel Patterns
- Reconfigurable Units:
  - PCUs
  - PMUs
  - AGs and CUs



[R. Prabhakar, Micro. 2018]



Plasticine's Control Flow & Performance



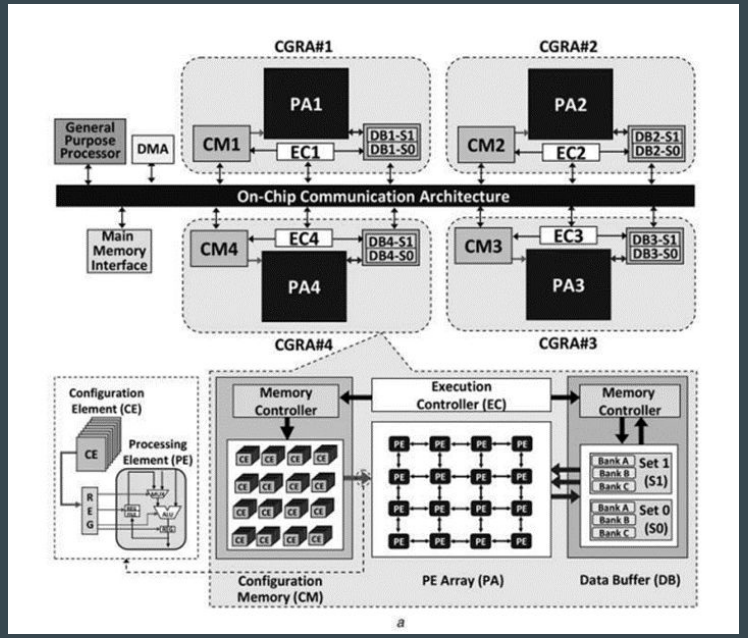
[R. Prabhakar, Micro. 2018]

	Utilization (%)							Power (W)		Plasticine / FPGA		
	FPGA		Plasticine					FPGA	Plas-ticine			
Bench-mark	Logic	Memory	PCU	PMU	AG	FU	Reg			Power	Perf	Perf/W
Inner Product	24.3	33.5	17.2	25.0	47.1	69.8	10.2	21.8	18.9	0.9	1.4	1.6
Outer Product	38.2	71.4	15.6	46.9	88.2	21.6	12.8	24.4	23.9	1.1	6.7	6.1
Black-scholes	68.9	100.0	65.6	21.9	41.2	25.1	53.4	28.3	24.7	0.9	5.1	5.8
TPC-H Q6	24.3	33.4	28.1	25.0	47.1	70.8	20.2	21.7	20.5	0.9	1.4	1.5
GEMM	40.4	94.8	34.4	68.8	97.1	56.0	8.6	25.6	34.6	1.4	33.0	24.4
GDA	53.6	96.8	89.1	87.5	44.1	8.1	11.2	26.5	41.0	1.5	40.0	25.9
LogReg	28.4	73.4	51.6	68.8	8.8	30.2	12.3	22.9	28.6	1.2	11.4	9.2
SGD	60.1	58.2	6.3	9.4	8.8	34.3	7.2	25.6	10.7	0.4	6.7	15.9
Kmeans	42.1	65.4	10.9	17.2	8.8	35.5	10.9	23.9	12.9	0.5	6.1	11.3
CNN	86.8	99.0	48.9	98.4	100	62.5	25.0	34.4	42.6	1.2	95.1	76.9
SMDV	27.3	31.0	43.8	15.6	29.4	10.4	2.7	21.5	19.3	0.9	8.3	9.3
Page Rank	31.3	33.4	28.1	20.3	20.6	3.9	1.9	21.9	17.1	0.8	14.2	18.2
BFS	25.3	45.9	18.8	15.6	11.8	3.1	1.5	21.9	14.0	0.6	7.3	11.4

Kim, Yoonjin. IET 2016

Multi-CGRA

- CGRA based multi core architecture
  - Speed up entire application by running several kernels at the same time - KLP - Kernel-Level Parallelism
  - Parallel kernels may be independent or interdependent
- Limitations
  - Traditional single CGRA core is optimized for parallelized computations in one kernel at a time
  - Multi-CGRA architecture involves linear aggregation of single CGRAs
  - Existing multi-CGRA does not adapt quickly or efficiently to support diverse KLP
  - Multi-core architecture with dynamically reconfigurable array processors is more flexible than industry implemented CGRAs because shared data-memory banks are connected to all processing cores through crossbar switches
  - In Multi-CGRA communication is restricted to the on-chip bus
  - Causes performance bottleneck and high power consumption when number of cores increase.



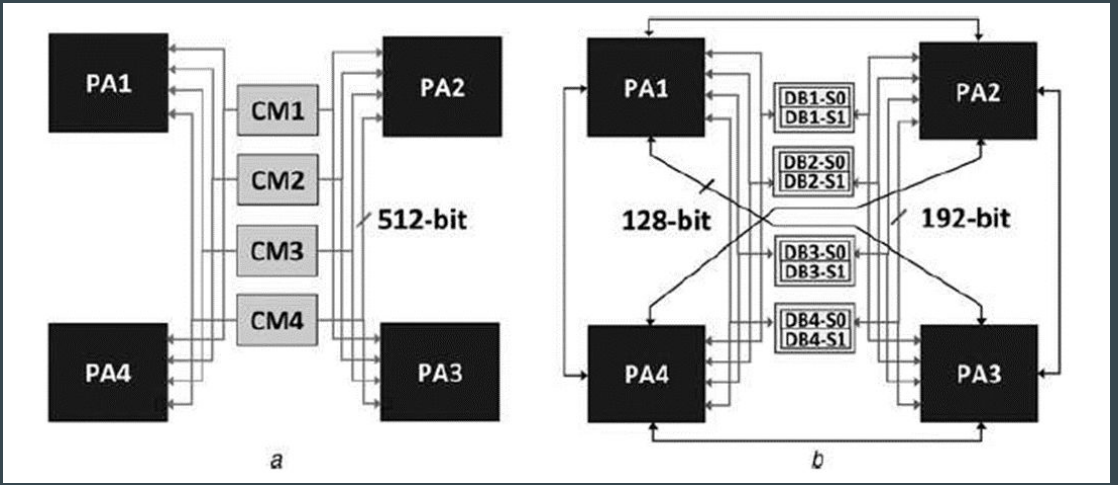
Base Architecture

Consists of General Purpose Processors (GPPS), Direct Memory Access (DMA), 4 CGRA and On chip communication bus.
Each CGRA consists of PE array, data buffer, configuration memory, execution controller.

CCF (Ideal)

Completely Connected Fabric

- Reconfigurable multi CGRA

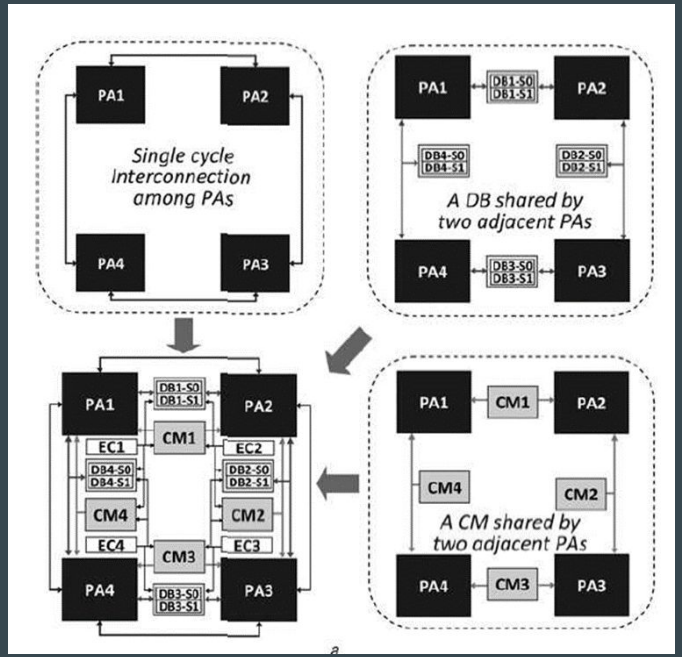


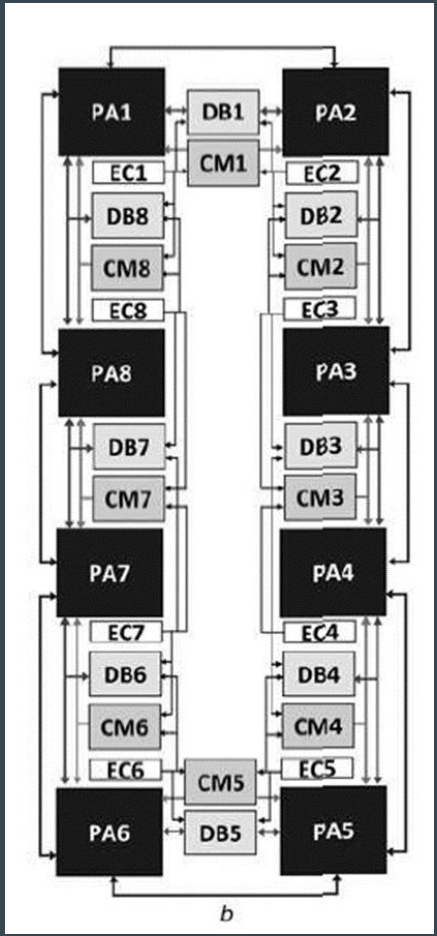
- Flexible connection with adjacent resources as required by applications
- Keeping within system capacity but efficiently using components
- Figure shows a CCF that can enable any combination of mapping between all of the components.
- Limitations
  - full connectivity causes significant area and power overhead with increasing number of CGRAs

RSF

Ring Based Sharing Fabric

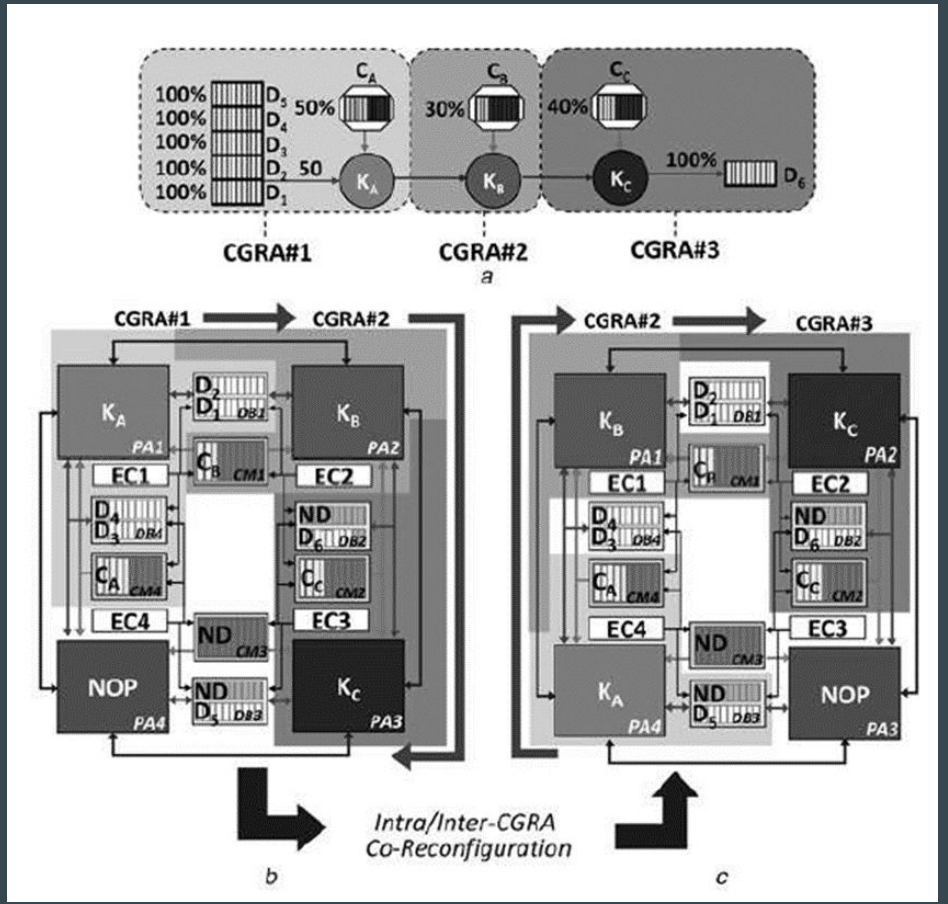
- Component Sharing between adjacent PA
- Trivial Overhead
- Easily expanded





RSF - Reconfigureable

- Pipelined kernel stream for 50 iterations
- CGRA configuration
  - PA1 utilizes D1 and D2 and D3 and D4 for 40 iterations
  - D5 for remaining 10 iterations
- Shifting the configurations of PAs from ‘PA1 to PA2 to PA3’ to ‘PA4 to PA1 to PA2’.
  - PA1 and PA2 reconfigured twice



Metrics

CGRA	CCF	RSF
Area	10%	8%
Delay	14.1%	10%
Power	11 %	5%

CGRA	CCF	RSF
Area	42%	19%
Delay	25.3%	10%
Power	92%	22%

- Tables show % increased compared to BASE for 4 (left) and 16 (right) core implementation
- RSF shows performance improvement **upto 88.8% compared with BASE** and 11.1% compared with CCF
- RSF shows **reduction in energy consumption of upto 87.5% compared with BASE** and 48.2% with CCF

CGRA Tools

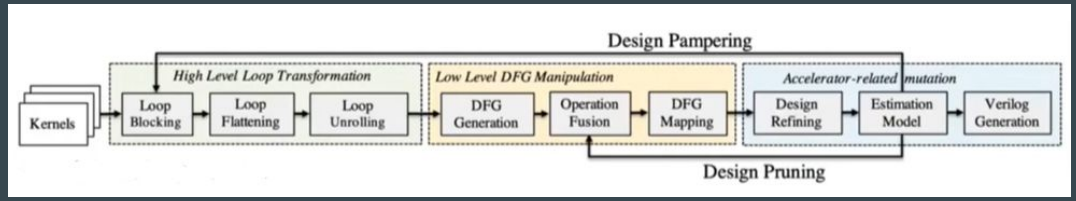
...

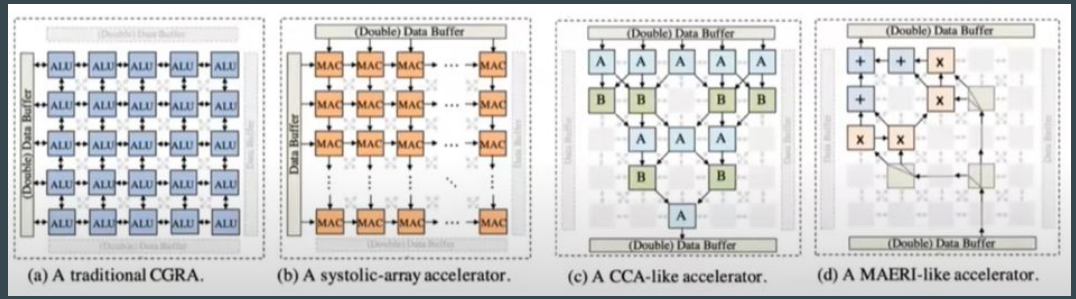
OpenCGRA - https://github.com/pnnl/OpenCGRA

- Open-source, unified framework for modelling, testing, and evaluating domain-specific customized CGRA designs
- Helps non-hardware experts design application specific CGRA designs
  - Conventional CGRA design entails a significant amount of HW/SW engineering effort.

Design Space Exploration (DSE)

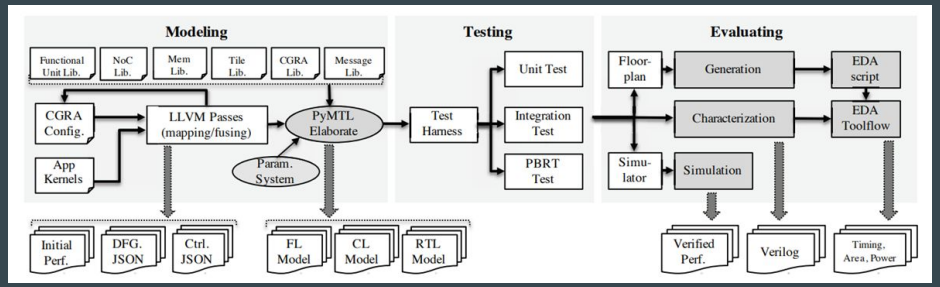
- A generic architectural template is used
  - This can be synthesized to different targets based on design space exploration
    - Communication direction
    - Type of tile
    - Topology
- DSE Steps:
  - Applying loop transformations
  - Specializing the PE
  - Refining the network topology
  - Deciding data memory size





Modelling, Testing and Evaluation

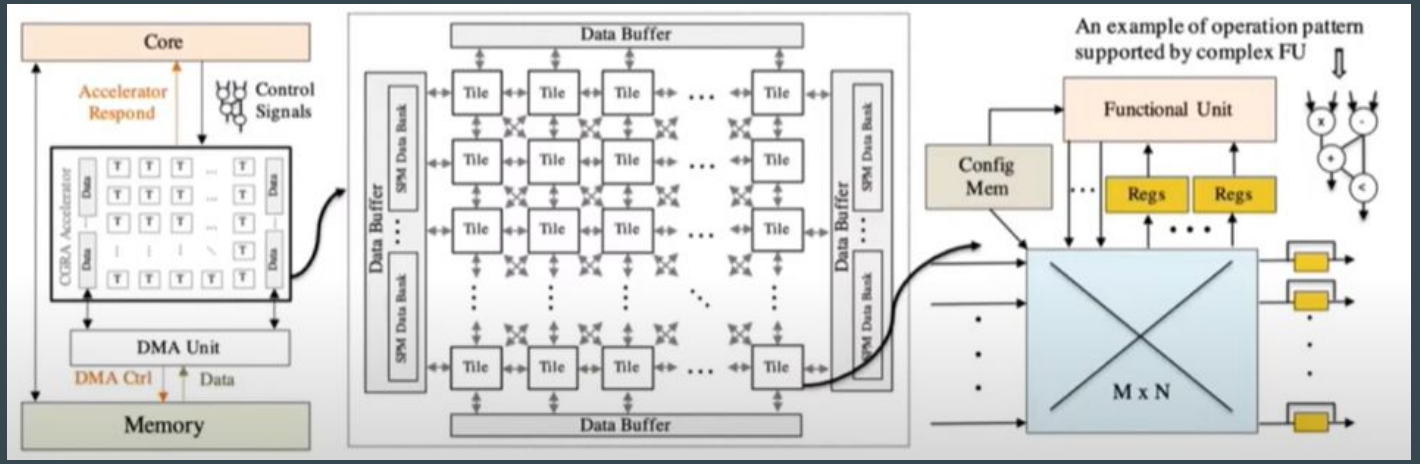
- Uses **PyMTL3** backend to generate Verilog
- Multilevel modelling
  - Functional level
    - Used for verification too
  - Cycle level
    - Cycle level accuracy
  - Register transfer level
    - Most accurate, synthesizable code



- Users only need to specify top level parameters through a python interface
  - Tile count
  - Functionality of tiles (for heterogeneous designs)
- **hypothesis** is an open source python package that is used for testing
  - Includes automatic input case generation
- Evaluation – area, power, timing report (mflowgen used)
  - Script to invoke open source / commercial EDA tools for synthesis

OpenCGRA - Summary

- The goal is to democratise the CGRA domain
  - Software designers can implement reconfigurable hardware solutions quickly
- OpenCGRA combines Design Space Exploration with model generation and evaluation to provide an end-to-end solution to design CGRA

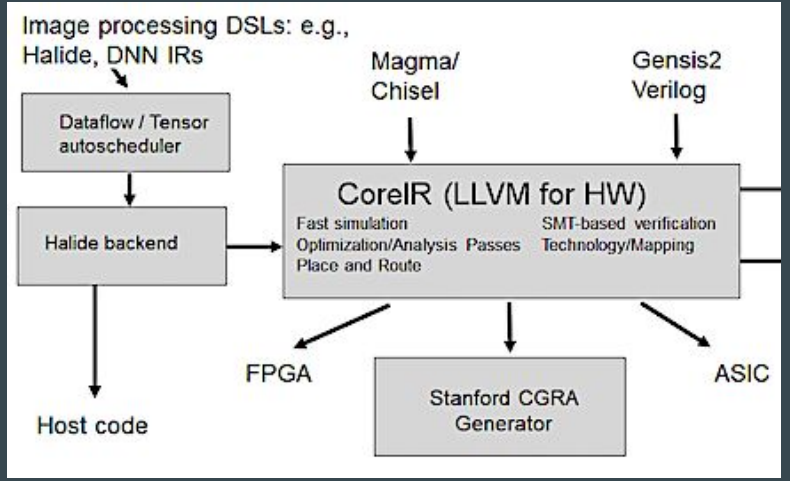


Stanford AHA - Agile Hardware Project

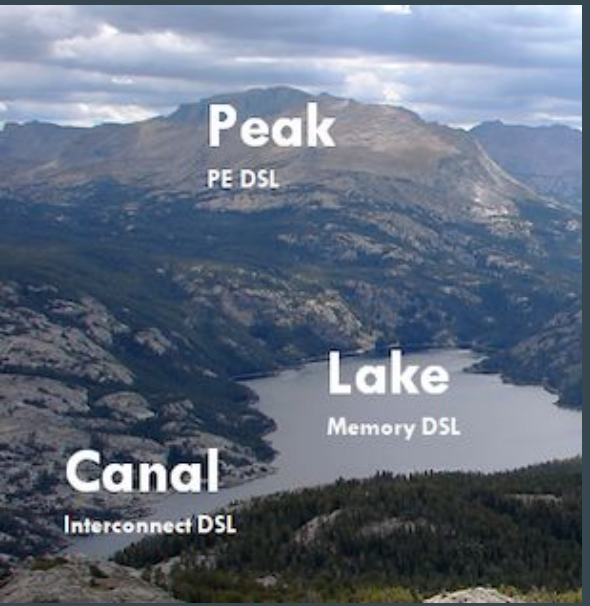
- Stanford research group broadly focused on “agile” hardware development
  - Seeks to develop tools and SoCs in dramatically less time than current hardware processes
  - “Agile” phrasing borrowed from software world. Agile flows seek frequent design iterations.
- Tool development spans FPGAs, CGRAs, and ASICs
- Focused on image processing and computer vision applications with plans to expand
- Most designs assume hard ARM CPU on-chip

AHA - Algorithm Definition

- Algorithm Definition
  - Halide DSL frequently used across projects
  - See: Halide-To-Hardware flow
- Algorithms mapped to AHA's CoreIR
  - "LLVM for hardware" - standardized intermediate representation of hardware
  - Passed to commercial FPGA/ASIC tools or AHA's CGRA tooling
  - Example primitives: reg, shl, eq, gle, mux, concat
- Designs assume availability of a CPU to control FPGA/CGRA/ASIC



AHA - CGRA Hardware Generation

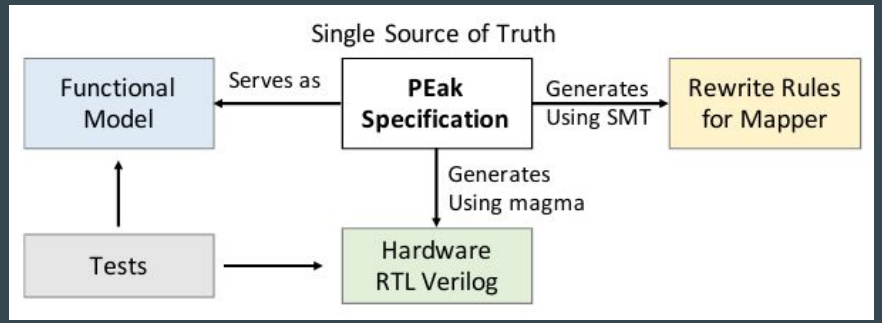


From the AHA webpage
What does it mean??

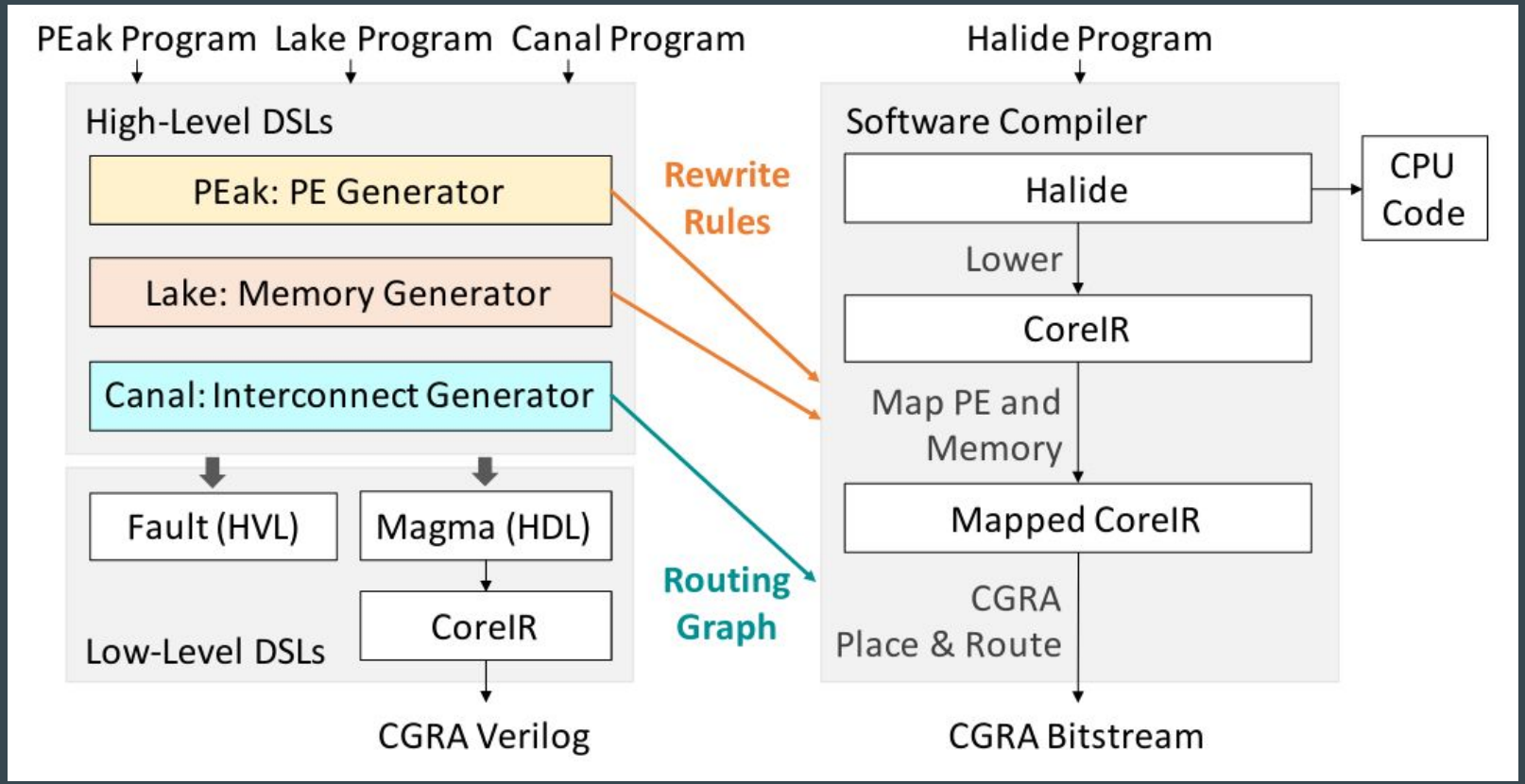
- Set of high-level domain-specific languages (DSLs)
  - All are written by AHA members
  - Outputs of all tools are CoreIR representations
  - CoreIR representations are converted to Verilog modules
- PEak: DSL for Processing Element (PE) specification
  - Discussed thoroughly in reading of Homework 1
- Lake: DSL that maps memory model to SRAM architecture and control
- Canal: DSL to define interconnect switch blocks/routing
- Hardware is finally generated by passing these products to cgra_pnr and providing CGRA grid dimensions

AHA - CGRA Configuration Generation

- Can begin generating configuration after algorithm mapped to CoreIR and PE/Mem/Interconnect generated
- Map CoreIR graph to equivalent graph using PE/Mem/Interconnect blocks
- Graph passed (back) to DSLs to generate configurations
  - PE specs passed back to PEak, Mem specs passed back to Lake
  - Graph connections passed to Canal for routing definitions
- Configurations merged to generate configuration bitstream



AHA - Putting it All Together



Limitation

- Sparse connectivity put a huge burden on the compiler
- The immaturity of CGRA technology and the diversity of application fields makes it difficult to formulate standards

References

- https://dl.acm.org/doi/10.1145/3357375
- Prabhakar R., Zhang Y., Olukotun K. (2020) Coarse-Grained Reconfigurable Architectures. In: Murmann B., Hoefflinger B. (eds) NANO-CHIPS 2030. The Frontiers Collection. Springer, Cham.
- Kim, Yoonjin, Hyejin Joo, and Sohyun Yoon. “Inter-Coarse-Grained Reconfigurable Architecture Reconfiguration Technique for Efficient Pipelining of Kernel-Stream on Coarse-Grained Reconfigurable Architecture-Based Multi-Core Architecture.” *IET circuits, devices & systems* 10.4 (2016): 251–265. Web.

Thank you

Question?

Wave DPU

- Data Flow Computing
- Local Synchronization circuits
- 32x32 clusters

[C. Nicol, WC WhiteP 2017]