Lecture

9

ECE M216A

Brief Intro to CGRAs

Prof. Dejan Marković

ee216a@gmail.com

Coarse-Grained Reconfigurable Array (CGRA)

Classification of CGRAs

Architecture	Year	Programming model*	Computation model	Execution model**	Specifications
Xputer [8]	1991	D	SCSD	SSE	
PADDI [9]	1992	I	SCSD	SSE	
PADDI-2 [67]	1993	D	SCMD	DSD	
RAW [68]	1997	I	MCMD	DSD	*More like a multicore processor*
PipeRench [69]	1998	D	SCMD	SSE	
Morphosys [11]	2000	I	SCMD	SSE	
Wavescalar [62, 70]	2003	I	MCMD	DDD	*Dataflow-driven ISA*
PACT-XPP [13]	2003	I/C	SCSD	DSD	
DRP [26]	2004	I	SCSD	SSE	*programmable FSM controller*
ADRES [10]	2004	I	SCSD	SSE	*VLIW controller*
ASH [57]	2004	D	SCSD	SSD	
TRIPS [12]	2004	I	MCMD	DSD	*Dataflow-driven ISA*
CCA [52]	2004	*transparent*	SCSD	SSE	*Runtime-generated configurations*
Tartan [60]	2006	I	MCMD	DSD	*Asynchronous circuit*
TFlex [71]	2007	I	MCMD	DSD	*Dataflow-driven ISA*
RICA [72]	2008	I	SCSD	SSE	
PPA [54]	2009	I	SCSD	SSE	*Polymorphic configurations*
TCPA [50]	2009	D	SCSD	SSE	
C-Cores [73]	2010	I	SCSD	SSE	*Targeted reconfigurability, ASIC-like*
DySER [47]	2012	I	SCSD	SSD	
REMUS [30]	2013	I	SCSD	SSE	
Triggered Inst. [61]	2013	D	MCMD	DSD	
T3 [74]	2013	I/C	MCMD	DSD	*Dataflow-driven ISA*
SGMF [75]	2014	I	MCMD	DDD	
FPCA [76]	2014	I	SCSD	SSE	
DynaSPAM [53]	2015	*transparent*	SCMD	SSD	*Based on PipeRench*
NDA [77]	2015	-	SCSD	SSE	*Process-in-memory*
HARTMP [78]	2016	I	SCMD	DSD	
DORA [51]	2016	*transparent*	SCSD / SCMD	SSD	*Based on DySER*
HRL [79]	2016	D/I	SCSD	SSE	*Process-in-memory, mix-grained*
HReA [16]	2017	I	SCSD	SSD	*General-purpose*
Plasticine [19]	2017	D	SCMD / MCMD	SSD	*Parallel-pattern-based programming*
Stream-dataflow [20]	2017	I	SCSD	DSD	*Vector memory interface*
CGRA-ME [80]	2017	I	SCSD	SSE	*ADRES-like*
Wave DPU [18]	2017	I/C	SCSD	SSD	*Commercial product for DNN*
PX-CGRA [81]	2018	-	SCSD	SSE	*Approximate PEs*
i-DPs CGRA [82]	2018	-	SCMD	SSE	*Double-ALU/Reg. in each PE*
Parallel-XL [83]	2018	I/C	SCMD/MCMD	DDD	*Intel Cilk & work stealing*
dMT-CGRA [84]	2018	I/C	MCMD	DDD	*Based on SGMF*

*I–imperative programming model, D–declarative programming model, C–parallel/concurrent (imperative) programming model, "*transparent*" means that CGRA-related programming is not required, "-" means that programming is not mentioned in that work.
**SSE–static-scheduling sequential-execution, SSD–static-scheduling static-dataflow-execution, DSD–dynamic-scheduling static-dataflow-execution, DDD–dynamic-scheduling dynamic-dataflow-execution.

Required model features

- Programming: I/C (Imperative, concurrent)
- Computation: MCMD (multi-config, multi-data)
- Execution: DDD (dynamic-scheduling, dynamic-dataflow)

Only recent designs have the desired features

- [83] is an FPGA prototype and simulations based
- [84] focuses on the narrow aspect of inter-thread communication (point-to-point), extensions to CUDA

Great need, many open challenges

- No efficient programming paradigm for CGRAs
- More complicated Hw than CPU due to 2D scheduling
- High-level abstraction provides coarse-grain parallelism, which is insufficient to fulfill the hardware potential
- Performance depends on applications; the need for application oriented extensions to the programming model
- Reconfig. speed down to pipeline level (10's of cycles)

L. Liu, et al., "A Survey of Coarse-Grained Reconfigurable Architecture and Design: Taxonomy, Challenges, and Applications," ACM Computing Surveys, Oct. 2019.

Partial FPGA Reconfig. Small-Size & Very Slow

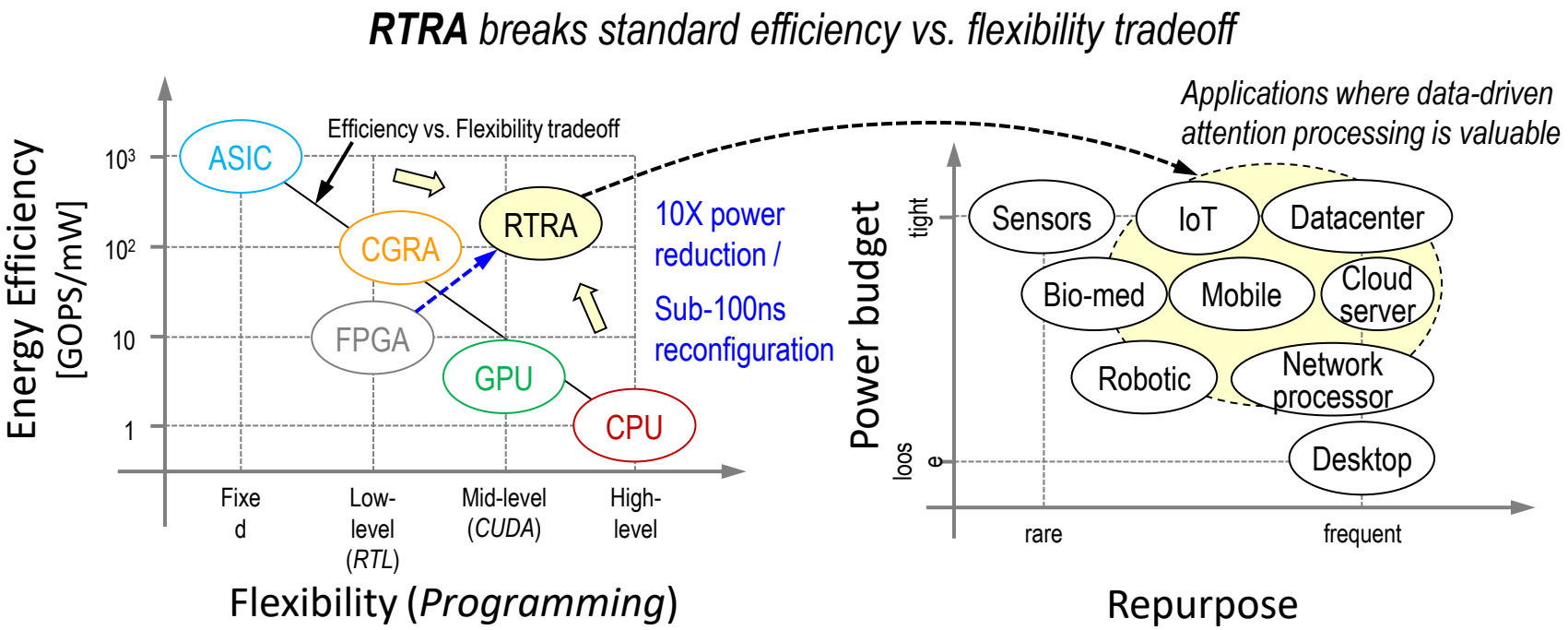
- **FPGA time to dynamic partial reconfigure depends on [1]:**
  - The size of the config. bit-stream ($BitStr_{size}$) – usually in KB
  - The reconfig. path throughput ($RP_{throughput}$) – usually in MB/s

$$T_{dyn-rec} = \frac{BitStr_{size}}{RP_{throughput}}$$

- **Dynamic partial reconfiguration controllers go up to 400MB/s [2]**
- **Usually, a large number of Clk cycles is required for a small amount of logic**
  - 130k Clk cycles to reconfigure 1.5k slices of logic [3] | 0.4ms @ 300MHz Clk
- **An SDR pipeline on a Zynq FPGA uses 3.2k slices of logic, 4-region partition**
  - Largest partial bit-stream size for a region is 324 KB [4]
  - Worst execution time for dynamic partial reconfig. of this region is 1.08ms

[1] G. Valente et al., "Dynamic partial reconfiguration profitability for realtime systems," IEEE Embedded Systems Letters, pp. 1–1, 2020.
[2] S. D. Carlo, P. Prinetto, P. Trotta, and J. Andersson, "A portable open-source controller for safe dynamic partial reconfiguration on Xilinx FPGAs," in Proc. of the 25th International Conference on Field Programmable Logic and Applications (FPL), 2015, pp. 1–4.
[3] L. Pezzarossa, A. T. Kristensen, M. Schoeberl and J. Sparso, "Can real-time systems benefit from dynamic partial reconfiguration?," 2017 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC), Linkoping, 2017, pp. 1-6, doi: 10.1109/NORCHIP.2017.8124984.
[4] A. Kamaleldin et al., "A reconfigurable hardware platform implementation for software defined radio using dynamic partial reconfiguration on Xilinx Zynq FPGA," 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, 2017, pp. 1540-1543, doi: 10.1109/MWSCAS.2017.8053229.

Runtime Reconfig. for Data-Driven Processing

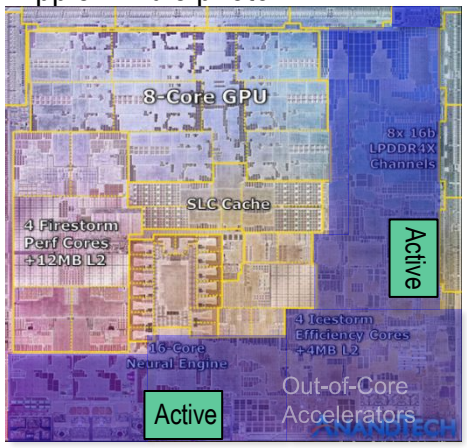


- **Opportunistically repurpose unutilized processor arrays**
  - Multi-step compilation (Sw + Hw) avoids complete program recompile
  - Requires array's network symmetry (for polygon translation/rotation/flip)
  - Support for Sw compilation from Python/C++ base

An Alternative to Accelerators?

- **~7% of *entire* SoC area is active, TSMC 45nm node [*]**
- **Depending on domain specialization, RTRA can be within 2x-10x in area and power vs accelerator**
  - **Note:** system/platform power will not be 2-10x higher; much less (~30-50%)
    - e.g. iPad battery life with H.264 on CPU (3 hours) vs accelerator (10 hours), a 3x system impact with a 1,000x accelerator gain
- **Feasible for new and/or evolving architectures, SDR, etc.**

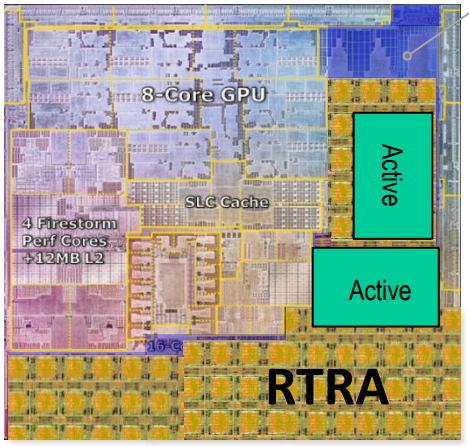
Apple M1 die photo



Replace "UnCore" with RTRA

No area increase (or even saving)

~2 active programs
>10x area overhead



potentially saved area

~2 active programs
Can accommodate more

Algorithm updates w/o chip respin

[*] G. Venkatesh, et al., ACM SIGARCH Computer Architecture News, Volume 38Issue 1 March 2010 pp 205–218 https://doi.org/10.1145/1735970.1736044