Mestrado em Informática

2010/11

A.J.Proença

Tema

Arquitecturas Paralelas (4)

Adaptado de

EETimes articles and slides from Xilinx, Altera & several universities

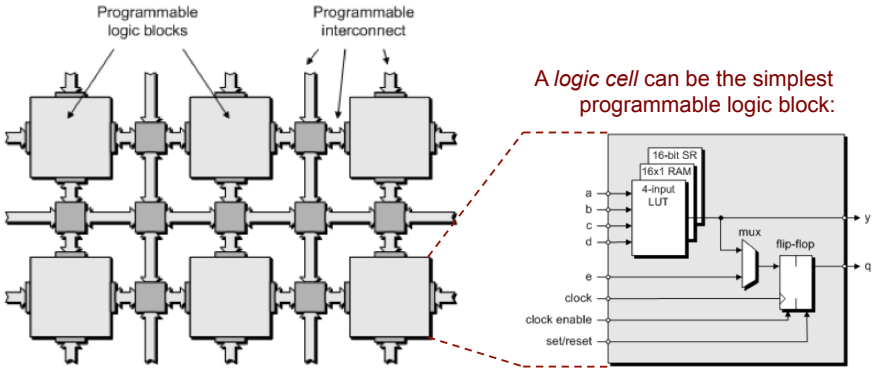
Estrutura do tema AP

1. A evolução das arquitecturas pelo paralelismo
2. Multiprocessadores (SMP e MPP)
3. Data Parallelism: SIMD, Vector, GPU, …
4. Topologias de interligação
5. ISA-free computing units: the FPGA

What is an FPGA (1)

Field-Programmable Gate Arrays (FPGA):

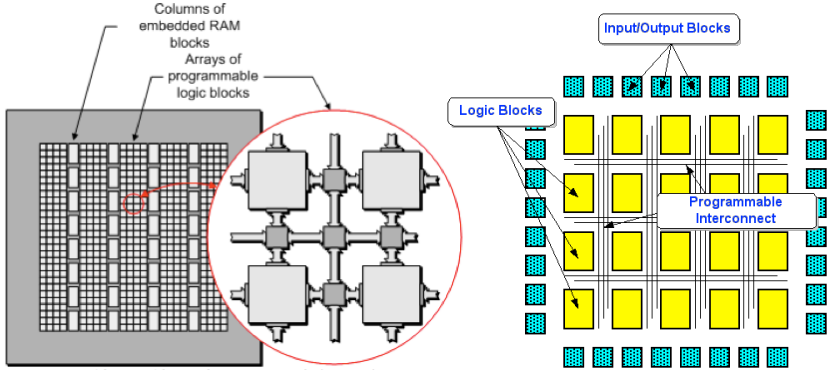
1. A fabric with 1000s of simple *logic cells* with LUTs



What is an FPGA (2)

Field-Programmable Gate Arrays (FPGA):

2. On-chip SRAM, configurable routing and I/O cells

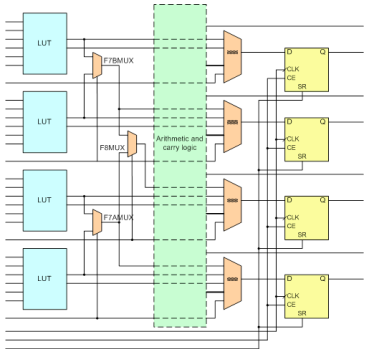


What is an FPGA (3)

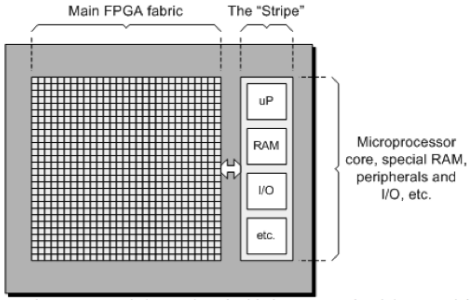
Field-Programmable Gate Arrays (FPGA):

3. Logic blocks may be structured in slices ...

... or pre-defined

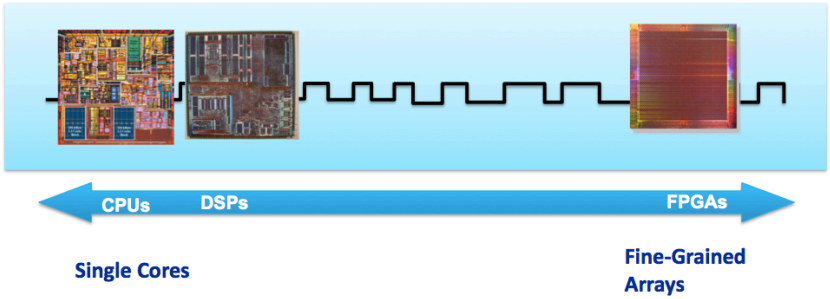


A simplified Xilinx Vertex-5 slice



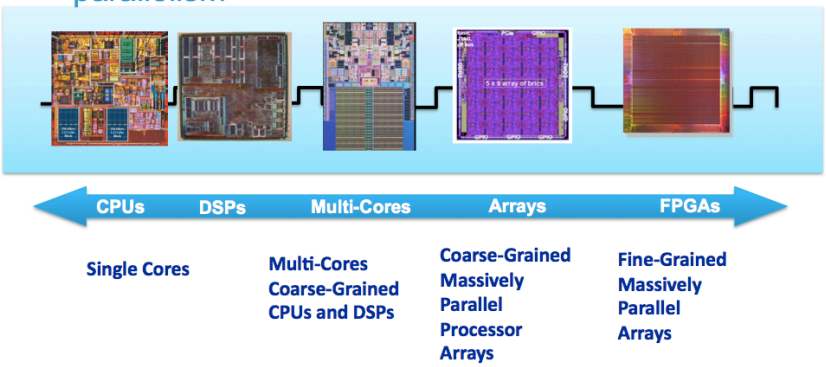
Programmable solutions: 1985-2002

- Technology scaling favors programmability



Programmable solutions: 2002-20xx

- Technology scaling favors programmability and parallelism



Why FPGAs?

- FPGAs have higher computational density per Watt than GPUs and CPUs:
  - 32-bit integer arithmetic: 6X higher
  - 32-bit floating point arithmetic: 2X higher

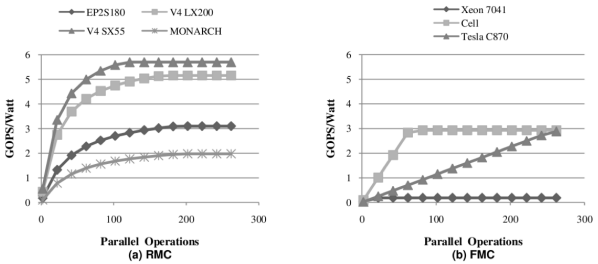


Figure 7. SPFP CDW (90 nm)

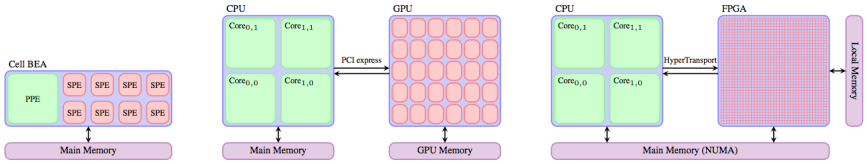
Computational Density: FPGA vs GPU vs CPU

Table 5. CD (in billions of operations per second or GOPs)

Device	Bit-level		16-bit Int.		32-bit Int.		SPFP		DPFP	
	Raw	Sustain.	Raw	Sustain.	Raw	Sustain.	Raw	Sustain.	Raw	Sustain.
Arrix FPOA	6144	6144	384	384	192	192				
ECA-64	2176	2176	13	13	6	6				
MONARCH	2048	2048	65	65	65	65	65	65		
Stratix-II S180	63181	63181	442	442	123	123	53	53	11	11
Stratix-III SL340	154422	154422	933	933	213	213	96	96	26	26
Stratix-III SE260	119539	119539	817	817	204	204	73	73	22	22
TILE64	4608	4608	240	240	144	144				
Virtex-4 LX200	89952	89952	357	116	66	42	68	46	16	16
Virtex-4 SX55	29184	29184	365	110	71	40	31	31	7	7
Virtex-5 LX330T	150163	150163	606	300	131	122	119	116	26	26
Virtex-5 SX95T	48435	48435	599	226	221	92	82	82	15	15
Cell BE	4096	4096	205	205	115	115	205	205	19	19
Tesla C870	5530	5530	346	346	216	216	346	346		
Xeon 7041	1536	1536	42	42	30	30	30	30	24	24
Xeon X3230	4095	4095	128	128	85	85	85	85	64	64

- J. Williams et. al. "Computational density of fixed and reconfigurable multi-core devices for application acceleration", RSSI, 2008.

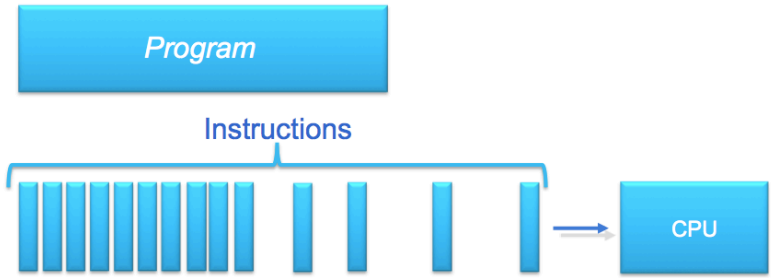
CELL, GPU & FPGA



Composition	CPU	GPU (AMD/NVIDIA)	CBEA	FPGA
	Symmetric Multicore	Symmetric Multicore	Heterogeneous Multicore	Heterogeneous Multicore
Full cores	4	-	1	2
Accelerator cores	0	10 / 30	8	2016 DSP slices
Intercore communication	Cache	None	Mailboxes	Explicit wiring
SIMD width	4	64 / 32	4	Configurable
Additional parallelism	ILP	VLIW / Dual-issue	Dual-issue	Configurable
Float operations per cycle	16	1600 / 720	36	520
Frequency (GHz)	3.2	0.75 / 1.3	3.2	<0.55
Single precision gigaflops	102.4	1200 / 936	230.4	550
Double : single precision performance	1:2	1:5 / 1:12	~1:2	~1:4
Gigaflops / watt	0.8	5.5 / 5	2.5	13.7
Megaflops / USD	70	800 / 550	>46	138
Accelerator Bandwidth (GB/s)	N/A	109 / 102	204.8	N/A
Main Memory Bandwidth (GB/s)	25.6	8	25.6	6.4
Maximum memory size (GiB)	24	2 / 4	16	System dependent
ECC support	Yes	No	Yes	Yes

Programmable and Sequential

- Reaching the limit
  - After four decades of success…



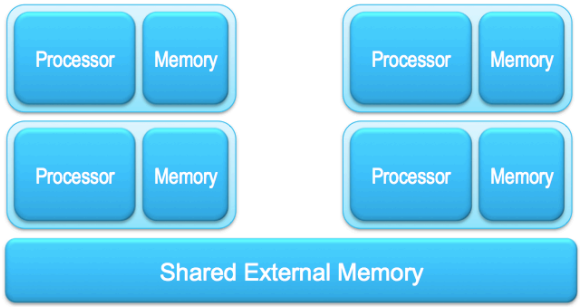
"The End of Denial Architecture"
William J. Dally [DAC'2009]

- Single thread processors are in denial about parallelism and locality
- They provide two illusions:

- Serial execution
  - Denies parallelism
  - Tries to exploit parallelism with ILP – limited scalability
- Flat memory
  - Denies locality
  - Tries to provide illusion with caches – very inefficient when working set doesn't fit in the cache

Programmable and Parallel

- Exploit parallelism on a chip
  - Take advantage of Moore's law
    - Processors not getting faster, just wider
  - Keep the power consumption down
- Use more transistors for information processing



Programming the FPGA with FCUDA

- CUDA
  - popular language for programming Nvidia GPUs
  - describes coarse-grained parallelism
- FCUDA: convert CUDA into annotated C code
- Use high level synthesis (AutoPilot) to convert C into RTL
- Synthesize RTL for an FPGA

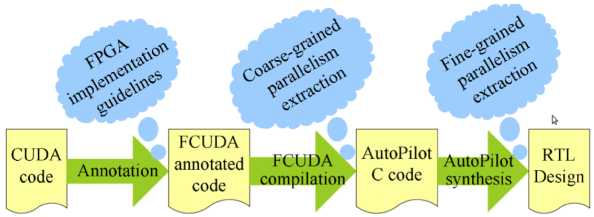
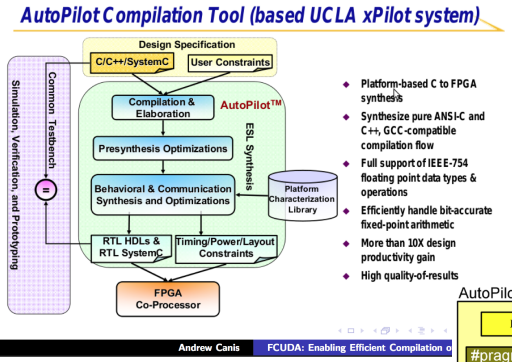


Fig. 1. CUDA-to-FPGA Flow

An overview of AutoPilot



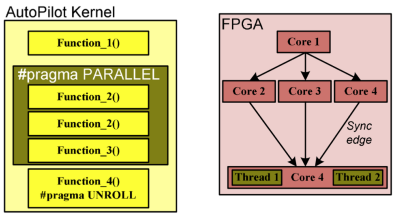
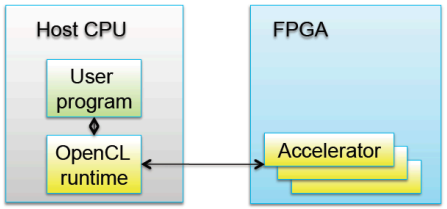


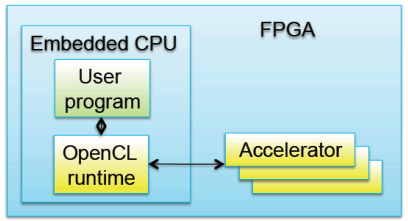
Fig. 3. AutoPilot C Programming Model

Traditional ways to use FPGAs

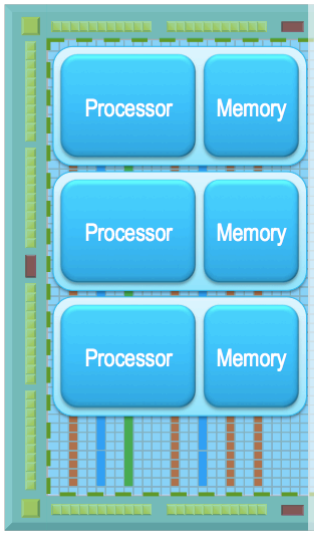
External host



Embedded

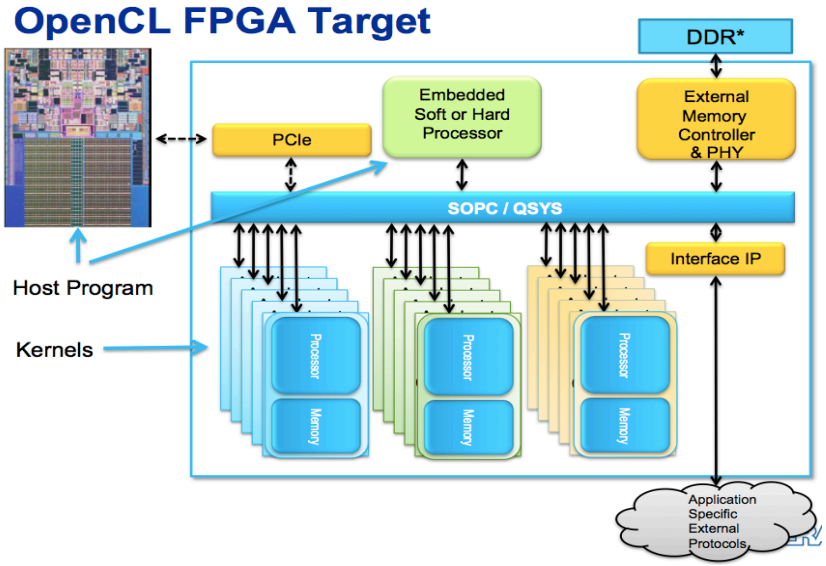


Yet another view: FPGA as just a multiple configurable ISA multicore



- Many coarse-grained processors
  - Different Implementation Options
    - Small soft scalar processor
    - or Larger vector processor
    - or Customized hardware pipeline
  - Each with local memory
- Each processor can exploit the fine grained parallelism of the FPGA to more efficiently implement it's "program"
- Possibly heterogeneous
  - Optimized for different tasks
- Customizable to suit the needs of a particular application

Programming this FPGA with OpenCL



State-of-the-art approaches in heterogeneous computing

TABLE 3. Summary of state-of-the-art approaches in heterogeneous computing. The table is not an exhaustive list, but gives an overview of state-of-the-art problems and techniques.

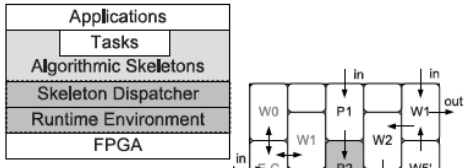
Application	Approach
Miscellaneous	Use of different programming languages (FPGA) [104]; Qualitative comparison (FPGA, CPU, GPU) [101]; Bioinformatics (CBEA) [102]; Restructuring of code for heterogeneous architectures (CBEA) [39]. Molecular Dynamics (FPGA) [105];
Dense Linear Algebra	Achieving peak performance (CBEA) [41, 106, 107]; Matrix multiplication, LU decomposition (FPGA) [108]; Use of registers instead of shared memory (GPU) [109]; Linpack (CBEA) [110]; Mixed precision (FPGA) [52].
Sparse Linear Algebra	Blocking (CBEA) [111]; Data structures (GPU) [112];
Fast Fourier Transform	Auto-tuning (GPU) [113]; Hierarchically decomposed (GPU) [114]; Iterative out-of-place (CBEA) [115]; Communication overheads (CBEA) [116]; Power and resource consumption (FPGA) [117]; Double precision performance (FPGA) [118].
Stencil computations	Cluster implementation (GPU) [119]; Varying stencil weights (CBEA, GPU) [120]; Domain blocking, register cycling (GPU) [121]; Domain blocking, loop unrolling (CBEA) [122]; Auto-tuning (CBEA, GPU) [123]; Time skewing (CBEA) [124, 125]
Random numbers	Sub-word operations (FPGA) [126]; Generating initial state [127]; Stateless (GPU) [128]; Tausworthe (GPU) [129]; Architectural constraints (GPU, CPU, FPGA) [130].
Scan	Up and down sweep (GPU) [131, 132]; Linked lists (CBEA) [72]; Histogram pyramids (GPU) [133, 134].
Sorting	Bitonic sort (GPU) [135]; Hybrid radix-bitonic, out-of-core (GPU) [136]; Radix (GPU) [137]; AA-sort (CBEA) [138].
Image Processing	Canny edge detection (GPU, FPGA) [139, 140]; OpenCV (GPU, CBEA) [141, 142, 143]; Computer Vision library (GPU) [144].

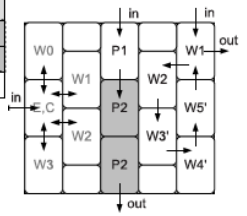
STATE-OF-THE-ART IN HETEROGENEOUS COMPUTING

ANDRÉ R. BRODTKORB[1], CHRISTOPHER DYKEN[2], TROND R. HAGEN[2], JON M. HJELMERVIK[3], AND OLAF O. STORAASLI[2]

This is a draft. Final version will appear in the journal of Scientific Programming, IOS Press.

ISA-free views of FPGAs: with algorithmic skeletons or hardware threads





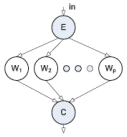


Fig. 3. Farm parallelism

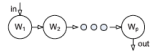


Fig. 4. The pipeline paradigm

Algorithmic Skeletons for the Programming of Reconfigurable Systems

Florian Dittmann

Multithreaded Programming and Execution Models for Reconfigurable Hardware

Enno Lübbert Egge Lübbers