IWOCL

INTERNATIONAL WORKSHOP ON OPENCL

12-13 MAY, 2014

Studying Energy Consumption of an OpenCL Application on Mobile GPU: A Case Study

Elena Barreras, Juan M. Jimenez, Arian Maghazeh, Unmesh Bordoloi

Linköping University

eGPU Compute Applications

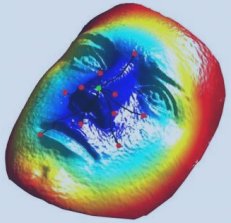
Conventional Domains

Media Codecs

Augmented Reality

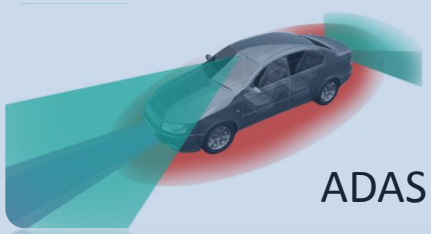
Gaming

Computational Photography



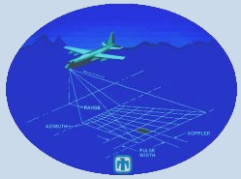
Graphics

Potential Domains



ADAS

Security



Radar Systems: Pattern Detection

Choice of Application

- Aho-Corasick – a pattern matching algorithm
  - Is utilized in security domain among others
  - Relevant for embedded systems – intrusion detection in vehicular systems, mobile devices
  - Not studied for embedded GPUs
  - State of the art parallel implementation available for high-end GPUs

Goal of Our Study

- Study the energy consumption of OpenCL components
- Optimize for embedded GPUs
  - Energy
  - Running times
- Compare with multi-core implementations
  - Tradeoffs

Aho-Corasick (AC)

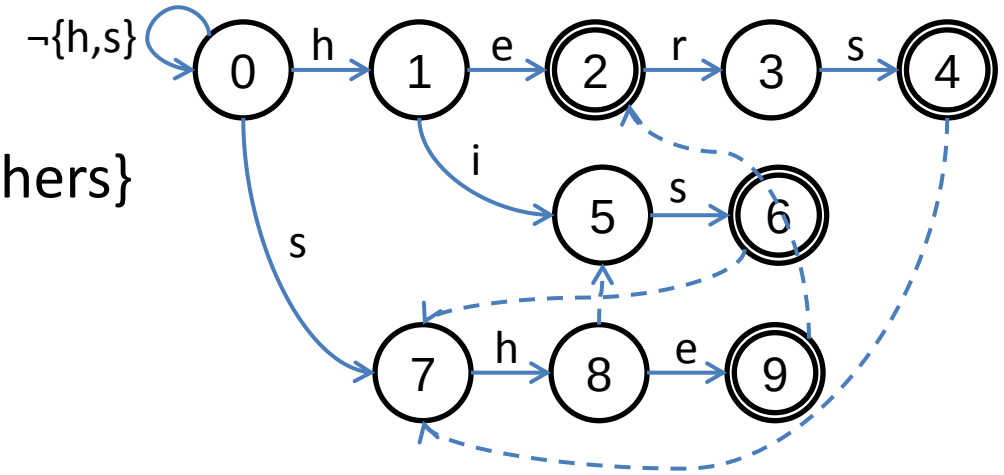
- It locates patterns of strings in an input text

Patterns: {she, he, his, hers}
Input text: ushers
Output: {she, he, hers}

Aho-Corasick (AC)

- How it works:
  - Combines all input patterns (dictionary) and generates a finite state machine
  - Uses the finite machine to find all the matches in the input text in a single traverse
- Open-source implementation available

Patterns: {she, he, his, hers}
Input text: ushers
Output: {she, he, hers}

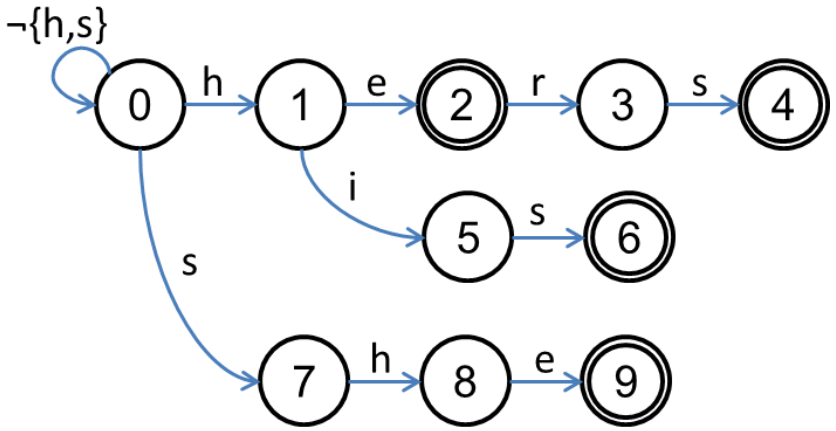


Parallel Failureless AC (PFAC)

- One thread for every input character
  - 10 M threads for a 10 MB input
- Each thread identifies the pattern that begins on that character
- No failure node

Patterns: {she, he, his, hers}
Input text: ushers
Output: {she, he, hers}

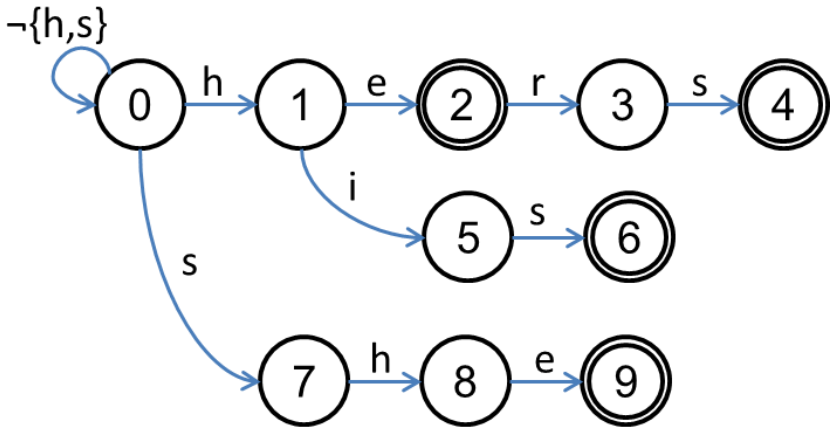
6 threads are launched



PFAC GPU Implementation

- Optimization on high-end GPUs
  - Load input text partially from global to local memory

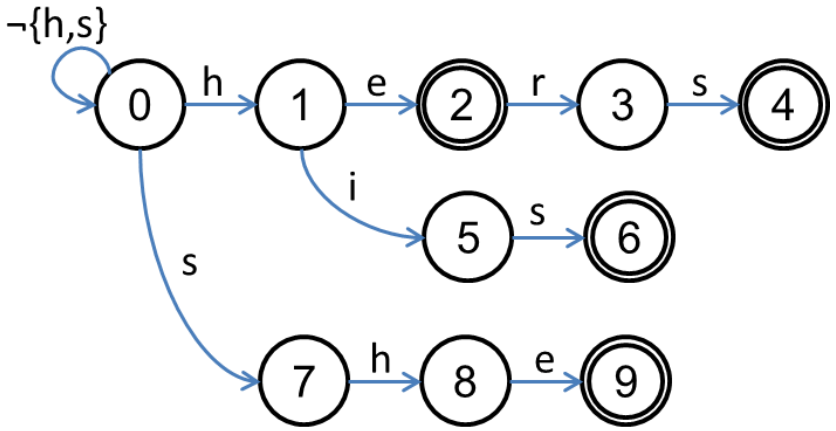
Patterns: {she, he, his, hers}
Input text: ushers
Output: {she, he, hers}



PFAC GPU Implementation

- Optimization on high-end GPUs
  - Load input text partially from global to local memory
  - Uses transition table
  - Load first row of the table into local memory

Patterns: {she, he, his, hers}
Input text: ushers
Output: {she, he, hers}



PFAC GPU Implementation

- Optimization on high-end GPUs
  - Load input text partially from global to local memory
  - Uses transition table
  - Load first row of the table into local memory
  - Convert transition table into an array
  - Store transition table in texture memory (cache optimized)

Optimizations on Embedded GPU

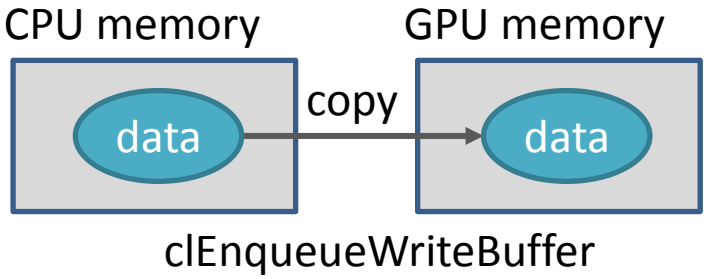
Local memory usage

- Local memory is emulated in global memory
- Using local memory adds an extra overhead
- Exception
  - Adreno 330 has 8KB local memory
  - May benefit only limited applications

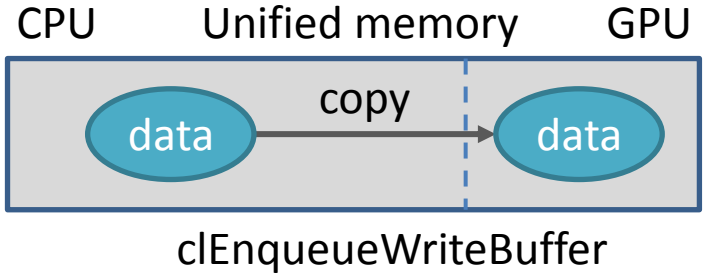
Optimizations on Embedded GPU

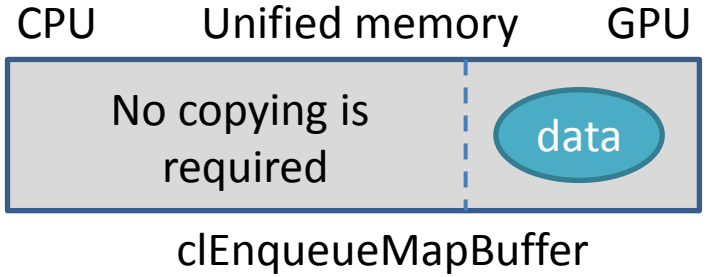
Reduce data communication time

High-end GPU



Embedded GPU

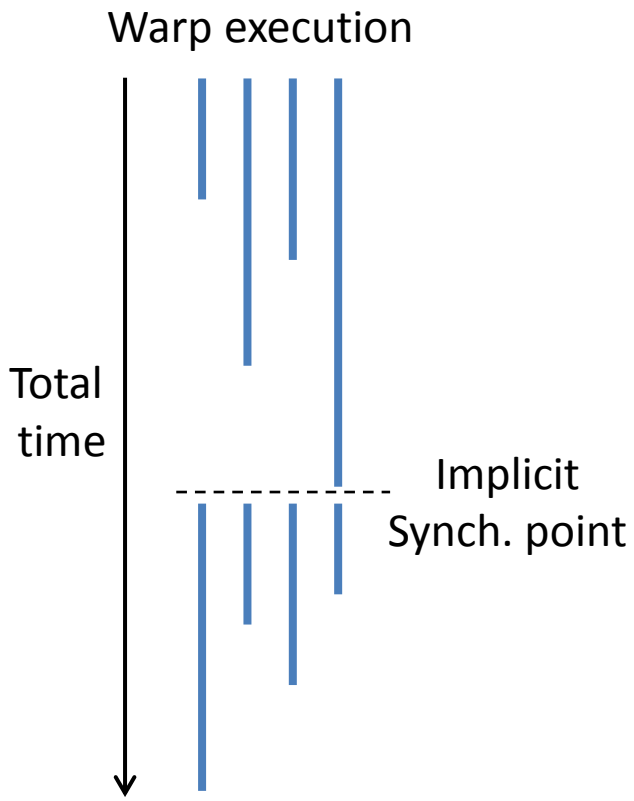




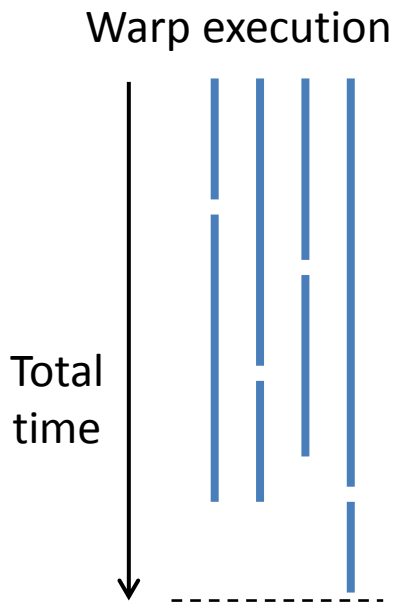
Optimizations on Embedded GPU

Thread granularity

1-char per thread



2-chars per thread



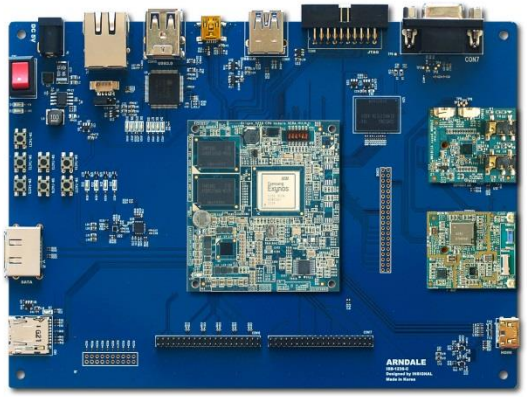
Reduce load imbalance between threads/warps

Implementation Remarks

- *Scalar* variables used in the kernel
- Appropriate work-group sizes chosen
- Kernel included *integer and memory* operations
- Memory bound kernel

Experimental Platforms

Samsung Arndale Board

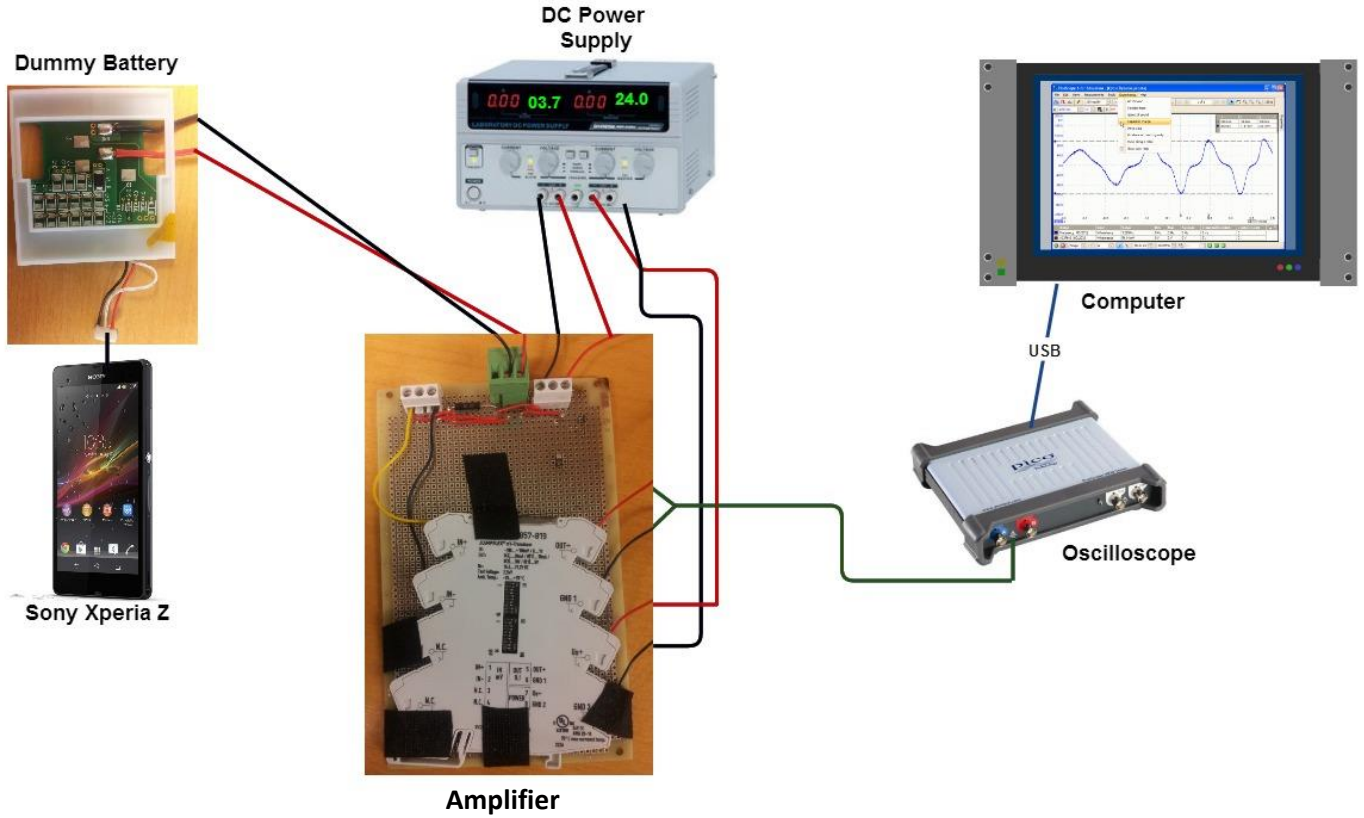


SoC: Exynos 5250
CPU: 1.7 GHz dual-core ARM Cortex-A15
GPU: ARM Mali-T604, 4 cores at 533 MHz, 68 GFLOPS

Sony Xperia Z Ultra

SoC: Snapdragon 800
CPU: up to 2.26 GHz quad-core ARM Cortex-A15
GPU: Adreno 330
4 cores at 450/578 MHz, 115 to 148 GFLOPS

Experimental Setup

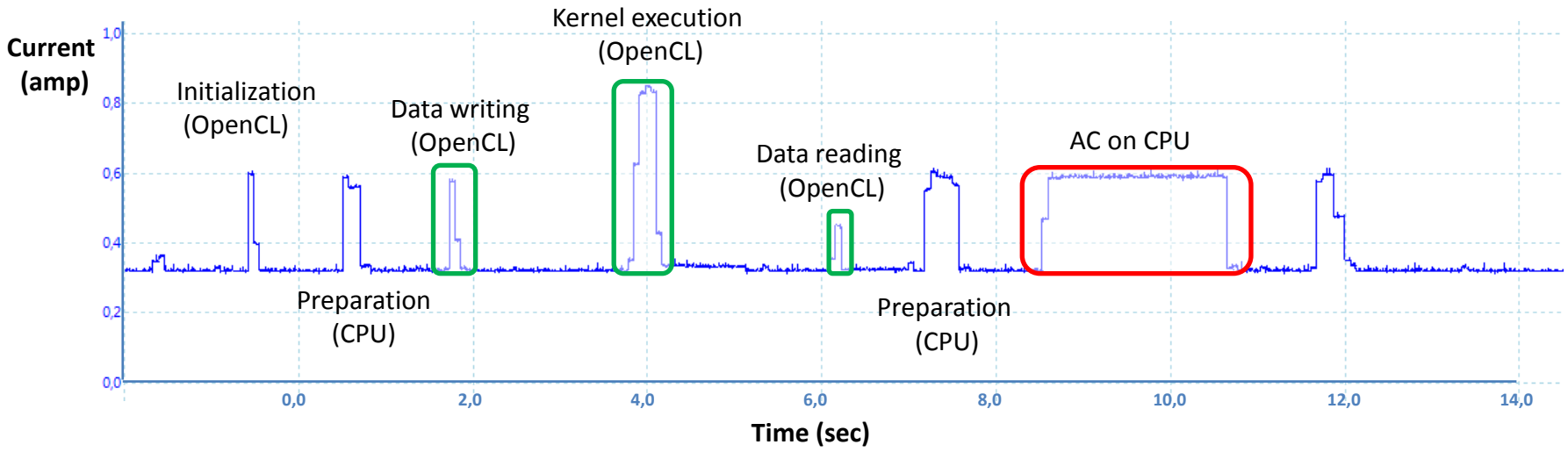


Experimental Input

- Input parameters:
  - 1000 Test patterns with maximum size of 128 characters and input text of size 10 MB
  - Extracted from Snort V2.8
  - FSM included 27570 nodes
  - GPU consumed 44 MB of memory

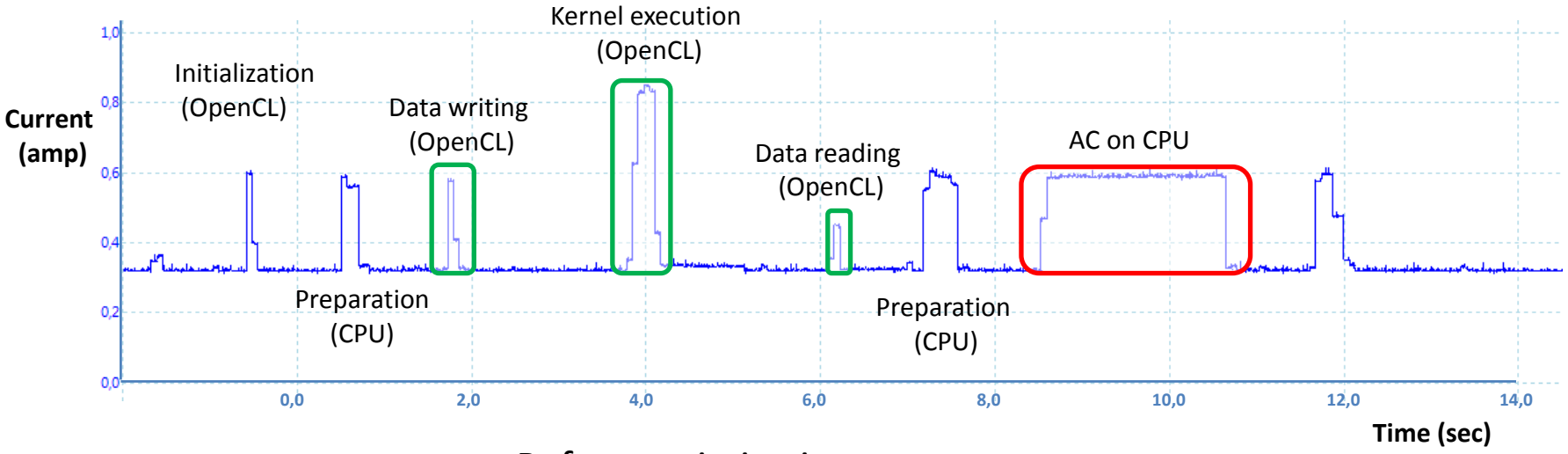
Energy Measurement

- Sample snapshot from the Oscilloscope



Experiment was performed on Arndale board

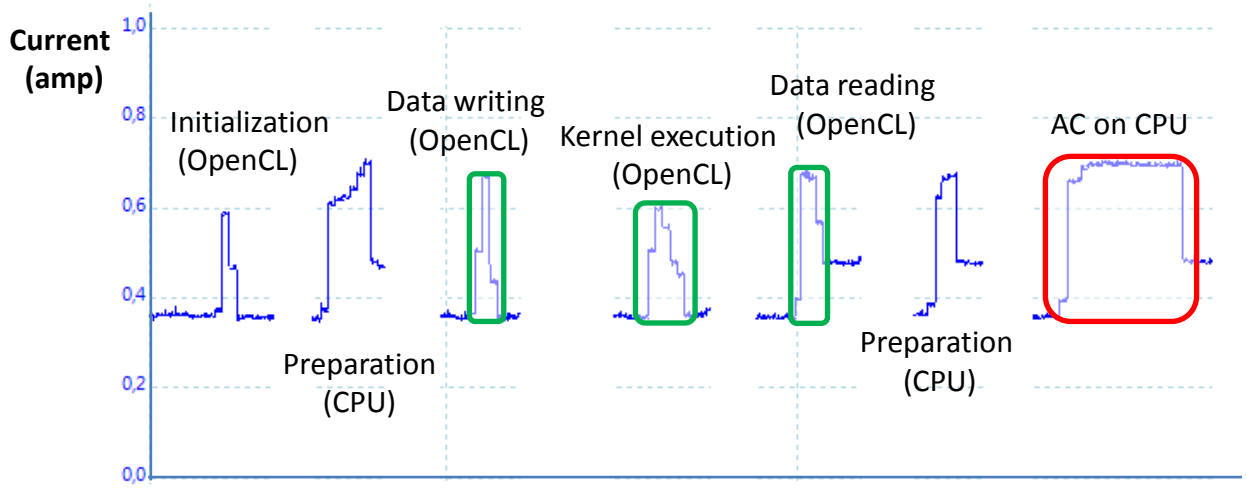
ARNDALE BOARD



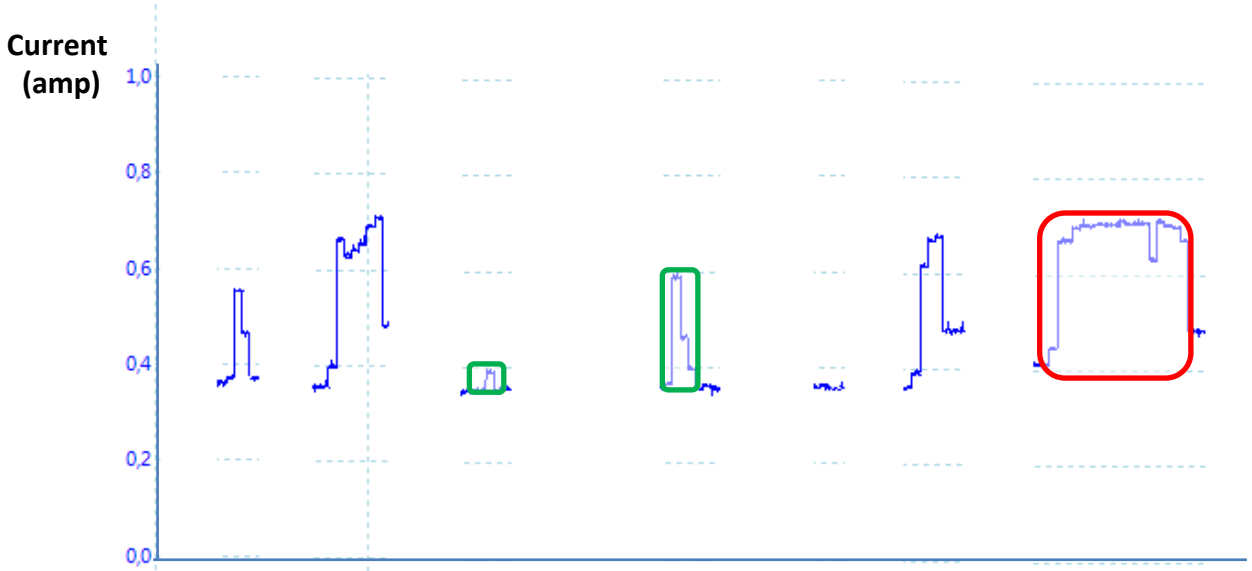
Before optimization

After optimization

SONY XPERIA Z ULTRA



Before optimization



After optimization

Experimental Results

time units are in milliseconds

SONY XPERIA Z ULTRA

OPTIMIZATIONS				RESULTS						
DATA_TX	USE_LOCAL	WG_SIZE	THR_GRAN	WRDEV	KERNEL_EXE	RDDEV	TX_OVH	GPU_TOT	SPEED UP	ENG IMPROV.
no map	yes	128	1	113	208	171	58%	492	2,1	2,7
map	yes	128	1	34	208	0	14%	242	4,3	7,2

Experimental Results

time units are in milliseconds

SONY XPERIA Z ULTRA

OPTIMIZATIONS				RESULTS						
DATA_TX	USE_LOCAL	WG_SIZE	THR_GRAN	WRDEV	KERNEL_EXE	RDDEV	TX_OVH	GPU_TOT	SPEED UP	ENG IMPROV.
no map	yes	128	1	113	208	171	58%	492	2,1	2,7
map	yes	128	1	34	208	0	14%	242	4,3	7,2
map	yes	128	1	34	208	0	14%	242	4,3	7,2
map	no	128	1	34	150	0	18%	184	5,7	10,7

Experimental Results

time units are in milliseconds

SONY XPERIA Z ULTRA

OPTIMIZATIONS				RESULTS						
DATA_TX	USE_LOCAL	WG_SIZE	THR_GRAN	WRDEV	KERNEL_EXE	RDDEV	TX_OVH	GPU_TOT	SPEED UP	ENG IMPROV.
no map	yes	128	1	113	208	171	58%	492	2,1	2,7
map	yes	128	1	34	208	0	14%	242	4,3	7,2
map	yes	128	1	34	208	0	14%	242	4,3	7,2
map	no	128	1	34	150	0	18%	184	5,7	10,7
map	no	64	1	34	168	0	17%	202	5,2	10,0
map	no	128	1	34	150	0	18%	184	5,7	10,7
map	no	256	1	34	140	0	20%	174	6,0	11,3

Experimental Results

time units are in milliseconds

SONY XPERIA Z ULTRA

OPTIMIZATIONS				RESULTS						
DATA_TX	USE_LOCAL	WG_SIZE	THR_GRAN	WRDEV	KERNEL_EXE	RDDEV	TX_OVH	GPU_TOT	SPEED UP	ENG IMPROV.
no map	yes	128	1	113	208	171	58%	492	2,1	2,7
map	yes	128	1	34	208	0	14%	242	4,3	7,2
map	yes	128	1	34	208	0	14%	242	4,3	7,2
map	no	128	1	34	150	0	18%	184	5,7	10,7
map	no	64	1	34	168	0	17%	202	5,2	10,0
map	no	128	1	34	150	0	18%	184	5,7	10,7
map	no	256	1	34	140	0	20%	174	6,0	11,3
map	no	256	4	34	99	0	26%	133	7,9	13,3
map	no	256	8	34	80	0	30%	114	9,2	15,1
map	no	256	12	34	202	0	14%	236	4,4	10,6
map	no	256	16	34	198	0	15%	232	4,5	10,7

Experimental Results

SONY XPERIA Z ULTRA

OPTIMIZATIONS				RESULTS						
DATA_TX	USE_LOCAL	WG_SIZE	THR_GRAN	WRDEV	KERNEL_EXE	RDDEV	TX_OVH	GPU_TOT	SPEED UP	ENG IMPROV.
no map	yes	128	1	113	208	171	58%	492	2,1	2,7
map	yes	128	1	34	208	0	14%	242	4,3	7,2
map	yes	128	1	34	208	0	14%	242	4,3	7,2
map	no	128	1	34	150	0	18%	184	5,7	10,7
map	no	64	1	34	168	0	17%	202	5,2	10,0
map	no	128	1	34	150	0	18%	184	5,7	10,7
map	no	256	1	34	140	0	20%	174	6,0	11,3
map	no	256	4	34	99	0	26%	133	7,9	13,3
map	no	256	8	34	80	0	30%	114	9,2	15,1
map	no	256	12	34	202	0	14%	236	4,4	10,6
map	no	256	16	34	198	0	15%	232	4,5	10,7

ARNDALE BOARD

OPTIMIZATIONS				RESULTS						
DATA_TX	USE_LOCAL	WG_SIZE	THR_GRAN	WRDEV	KERNEL_EXE	RDDEV	TX_OVH	GPU_TOT	SPEED UP	ENG IMPROV.
no map	yes	128	1	91	295	60	34%	446	4,7	3,3
map	yes	128	1	91	295	6	25%	392	5,4	3,6
map	yes	128	1	91	295	6	25%	392	5,4	3,6
map	no	128	1	91	150	6	39%	247	8,5	8,2
map	no	64	1	91	155	6	38%	252	8,3	8,2
map	no	128	1	91	150	6	39%	247	8,5	8,2
map	no	256	1	91	143	6	40%	240	8,8	8,3
map	no	256	4	91	114	6	46%	211	10,0	9,0
map	no	256	8	91	104	6	48%	201	10,4	9,3
map	no	256	12	91	101	6	49%	198	10,6	9,3
map	no	256	16	91	97	6	50%	194	10,8	9,5

GPU vs. Multi-core

- PFAC implemented on multi-core with OpenMP

SONY Z ULTRA	PFAC OPENMP				MOST OPTIMIZED on GPU
	1 CORE	2 CORE	3 CORE	4 CORE	
TIME (ms)	348	175	118	89	GPU_KERNEL TIME = 80 (ms) GPU_OVERALL TIME = 114 (ms)
KERNEL SPEED UP	4,4	2,2	1,5	1,1	
OVERALL SPEED UP	3,0	1,5	1,0	0,8	
ENERGY IMPROV.	5	4	4	4	

ARNDALE	PFAC OPENMP		MOST OPTIMIZED on GPU
	1 CORE	2 CORE	
TIME (ms)	680	620	GPU_KERNEL TIME = 97 (ms) GPU_OVERALL TIME = 194 (ms)
KERNEL SPEED UP	7,0	6,4	
OVERALL SPEED UP	3,5	3,1	
ENERGY IMPROV.	3,3	4,9	

Takeaways

- Embedded GPUs – alternative to save energy
- Nonconventional applications may benefit from GPU computing in embedded systems
- Micro-architecture specific optimizations are required to get efficient performance

Aknowledgments

- Sony Mobile Lund

- Adrian Horga

Questions?