 CONTENTS:

- Company Overview
- Why Accelerators?
- Processor + Accelerator Architectures
- CGRA Accelerators Architecture
- Our CGRA Solution Overview
GENERAL INFORMATION:

- **Company Name:** AXELERAware LLC
- **Registration Country:** GEORGIA, Democratic Republic of
- **Registration Date:** 2019, August 19
- **Registration No:** 422736864
- **Address:** b9, 75a, Nutsubidze St., Tbilisi, Georgia
- **Tel:** +995 591 091993
- **Web:** www.axeleraware.com
- **Email:** support@axeleraware.com
We are a startup company that focuses on CGRA acceleration solutions for Desktop, Mobile, GPU, IOT Processors and Embedded SOCs, developing CGRA based general-purpose accelerators as a solution for speeding up a wide range of computation intensive applications without adding large area and power overhead to the processor and so to be beneficiary while being so cost efficient to implement for our customers.
Why Accelerators?

- Transistors are not getting much smaller. (Due to the ending of Moore’s Law)

- The peak power per mm² of chip is increasing. (Due to the end of Dennard scaling) but the power budget per chip is limited. (Due to electromigration, mechanical and thermal limits)

- We have already played the multicore card (limited by Amdahl’s Law).

- We have already played all our cards in Computer Performance.

- Now CPU performance limits are reached!

- In “Hardware Side”, we have only Accelerators option.
Why Accelerators?

Slowing down performance improvement of Processors:

- **End of Dennard Scaling** \(\Rightarrow\) Multicore 2X/3.5 years (23%/year)
- **CISC** 2X/2.5 years (22%/year)
- **RISC** 2X/1.5 years (52%/year)
- **End of the Line** \(\Rightarrow\) 2X/20 years (3%/yr)
- **Amdahl’s Law** \(\Rightarrow\) 2X/6 years (12%/year)
Processor + Accelerator Simple Future SOC Architecture:

- Processor
- RAM
- AC (FPGA or CGRA)
- Other Logics

*AC = Accelerator (Either FPGA or CGRA)
Processor + Accelerator Architectures

(A) FPGA as a AC Card:

Processor chip \(\rightarrow\) AC chip \(\rightarrow\) Board

Physical bus (e.g. Custom Bus, PCI)

(B) CPU + FPGA on a Single Chip:

Processor \(\rightarrow\) AC

Single Chip

(C) Processor + CGRA:

Register File \(\rightarrow\) Memory \(\rightarrow\) ALU \(\rightarrow\) AC

Inside Processor

(D) Our Accelerator Solution:

Registers File

ALU \(\rightarrow\) AC
(A) FPGA as a AC Card: (Intel FPGA AC Card)
(B) CPU + FPGA on a single Chip:
(Intel Xeon Scalable processor with integrated FPGA)
**Processor + Accelerator Architectures**

**(C) Processor + CGRA**: *(Recent Intel x86 Processors)*

**SPECIFICATIONS:**
- Intel Core i7
- Out of Order
- CGRA:
  - Huge Area Overhead
  - Huge Power Overhead
  - Large Cache
- Multi Level Execution:
  - Up to 60 (15-30-60) levels
- Instant configuration change is impossible
- Dynamic Detection
- Software Transparent

**COSTS:**
- Huge Area Overhead
- Huge Power Consumption
- Complexity of Hardware
Our SOLUTION OVERVIEW:

- General Purpose AC (GPA)
- Low Area Overhead
- Low Power Consumption
- Cheap to Implement
- Software Transparency
- Dynamic Detection
- Power Saving Options

- Even with lower performance a light GPA could be applicable and reasonable
- ...

(D) Our Accelerator Solution:
CGRA Accelerators Architecture

- CGRA is simply an array of Processing Elements (PEs) interconnected by a network.
- Each PE consists of an ALU-like functional unit.
- CGRA could provide ILP speedup in single core processors as well as multicore processors.

CGRA Accelerators Technology Position:
Accelerators Speedup Efficiency and Limitations:

Effective Speed up for Embedded and computation intensive Applications

Low efficiency for Desktop and Memory intensive Applications
Typical CGRA (4x4 Architecture):
Typical CGRA (CCA: Configurable Computation Array):
Typical CGRA (3x3 Architecture):
Typical CGRA (2 Levels, 2x4 Architecture):

- 16 Processor CLUSTER: a full custom tiled GDSII block
- Fully-Connected PE Quads with fan-out
- 8 DPU Arithmetic Units
  - Per-cycle grouping into 8, 16, 24, 32, 64-b Operations
  - Pipelined MAC Units with (un)Signed Saturation
  - Support for floating point emulation
  - Barrel Shifter, Bit Processor
  - SIMD and MIMD instruction classes
  - Data driven
- 16KB Data RAM
- 16 Instruction RAMs
- Full custom semi-static digital circuits
- Robust PVT insensitive operation
  - Scalable to low voltages
  - No global signals, no global clocks
CGRA Accelerators Architecture

**Typical CGRA** (n Levels, 3x4 Architecture):
Our CGRA Solution Overview

SOLUTION FEATURES / BENEFITS:

- Innovative Dynamic Mapping Method
- Focus only on Integer Processing
- Adaptable to any RISC Pipeline
- Fully Software Transparent
- Low Power and Area Overhead
- Coupling AC to ALU
- Fast Configuration
- Fast Dynamic Detection
- Power Saving Option (Static and Smart)
- Sensor for detecting integer parts of running programs
- Efficient for Bitwise, Integer and Fix Point Approximated Applications (AI, Signal processing, Mining and ...)
Our CGRA Solution Overview

Why Focus on Integer Operations?

- Lots of integer operations in Embedded and Computation Intensive applications:
  - More than 50% of operations in average, according to Mibench. (Only in integer applications.)

- In addition to integer applications, in many floating point applications such as AI and Signal Processing, due to process simplification and performance improvement, fix point approximation and integer processing is used finally.
Our CGRA Solution Overview

Integer Process Percentage in Mibench Applications:
Our CGRA Solution Overview

Basic Block Size in Mibench Applications:
Our CGRA Solution Overview

Integer Percentage (Int%) and Estimation of Speedup Percentage (SP%) of our CGRA accelerator:

Done by Simulation software in some Mibench Applications.

<table>
<thead>
<tr>
<th>APP</th>
<th>Int%</th>
<th>SP%</th>
</tr>
</thead>
<tbody>
<tr>
<td>BasicMath</td>
<td>62%</td>
<td>19%</td>
</tr>
<tr>
<td>Bitcount</td>
<td>78%</td>
<td>18%</td>
</tr>
<tr>
<td>jpegDecoder</td>
<td>50%</td>
<td>22%</td>
</tr>
<tr>
<td>jpegEncoder</td>
<td>55%</td>
<td>23%</td>
</tr>
<tr>
<td>mad</td>
<td>58%</td>
<td>15%</td>
</tr>
<tr>
<td>patricia</td>
<td>59%</td>
<td>10%</td>
</tr>
<tr>
<td>sha</td>
<td>68%</td>
<td>24%</td>
</tr>
<tr>
<td>pgpEncode</td>
<td>63%</td>
<td>12%</td>
</tr>
<tr>
<td>pgpDecode</td>
<td>63%</td>
<td>13%</td>
</tr>
<tr>
<td>adpcmDecode</td>
<td>77%</td>
<td>21%</td>
</tr>
<tr>
<td>adpcmEncode</td>
<td>82%</td>
<td>22%</td>
</tr>
<tr>
<td>gsmDecode</td>
<td>64%</td>
<td>20%</td>
</tr>
<tr>
<td>gsmEncode</td>
<td>52%</td>
<td>20%</td>
</tr>
<tr>
<td>aesEncode</td>
<td>44%</td>
<td>17%</td>
</tr>
<tr>
<td>aesDecode</td>
<td>47%</td>
<td>18%</td>
</tr>
</tbody>
</table>
Our CGRA Solution Overview

Integer Percentage in Mibench Applications:
Our CGRA Solution Overview

Estimation of Speedup Percentage in Mibench Applications:

![SpeedUp Chart](chart.png)
Our CGRA Solution Overview

CGRA Implementation in MIPS32 Core:
CGRA Implementation in ARM7 Core:

Our CGRA Solution Overview
Our CGRA Solution Overview

CGRA Implementation in RISC-V Core: (Ongoing)
Performance Simulation Results based on HDL Implementation in ARM Core:

![Performance Speedup Graph](image)
THANKS FOR YOUR CONSIDERATION

www.axeleraware.com