The Accelerator Store framework for high-performance, low-power accelerator-based systems

Moozhan Motazedi
• “Accelerator store” a structure for sharing memory between accelerators in accelerator-based systems

• Incrementation of performance and reduction of energy consumption 100-500x compared to general purpose cores
• small accelerators and communicate with other accelerators efficiently
• Accelerator Store, a shared memory structure that simplifies accelerator I/O and reduces chip area

Dark silicon threatens general purpose (GP) multicore processor performance by limiting the number of active cores, in conflict with multicore’s thirst for additional active logic.

The following design, reduces accelerator area demands and simplify accelerator I/O with minimal impact on performance and energy:
• Reduced memory buffers  no large buffer memory required by DMA  
• Memory reuse  AS rededicates memory to other running accelerators  
• Simple accelerator I/O  FIFO simple mechanism to transfer data between AS s  
• Reduced GP-CPU energy  FIFO allows only little or no GP-CPU assistance  
• Reduced memory energy  AS turns off unused SRAM memory blocks by monitoring them  
• Low performance overhead  AS reduces performance by less than 1%
Accelerator store systems contain one or more GP-CPU cores and an accelerator store.
• ASPorts (accelerator store port) rather than a single shared bus, is needed since accelerators often require significant bandwidth
• Data access requests traverse through Asports and Three major components:
  - Priority Table
  - Hadle Table
  - SRAM Collection
Accelerator store handles

- Accelerators can not access accelerator store memory directly, the memory must be allocated to a handle ID number.
- Accelerators send information to each other using FIFO handles.
- In addition to FIFO handles, accelerator store supports Random Access handles.

Benefits provided by handles:

- provide memory protection
- declare the mapping of shared physical memories
- enable automatic SRAM VDD-gating
Accelerator store components

1. **Priority table**
   The priority table arbitrates all handle requests from accelerators

2. **Handle table**
   The handle table stores each handle’s configuration and uses this information to translate handle requests from accelerators into SRAM accesses

3. **SRAM collection**
   The accelerator store contains several SRAMs to increase VDD gating opportunities. If the accelerator store contained one valid word and used one large SRAM, the entire large SRAM would need to remain on.
   - It contains a mix of 2KB and 4KB memories.
CHARACTERIZING ACCELERATOR MEMORY

Memory centralization achieves significant area reductions through memory reuse only if accelerator area is dominated by memory.
Evaluation of a security application

overheads, energy overheads and area savings while running this prototype embedded security application on a simulated accelerator store system

**suspicious activity by:**
- acquiring audio signals from a microphone
- analyzing them with the FFT
- checking if a frequency response corresponding to suspicious activity occurred
- if yes, the camera takes pictures every second
- compresses the photos using the JPEG accelerator
- encrypts the JPEGs with the AES accelerator
- writes the encrypted photos to the SD flash card
• **Cycle accurate simulation**
  - simulating the accelerator store logic
  - reproducing accelerator memory accesses

• **Selecting memories to move into the accelerator store**
  to balance memory size and minimize contention, memories with largest values for “memory capacity/average throughput” are moved to accelerator store from the private accelerator memory

• **Performance & energy overhead**
  The AS performance overhead measured by:
  - *shared memory percentage*: amount of accelerator memory in AS
  - *internal bandwidth*: measured by channels, number of handle requests the accelerator store can process per cycle

• **Area reduction**
  by eliminating I/O buffer memories and dedicating memory to accelerators at runtime
Related work

• Multi-component memory sharing but without full features of AS
• SoCDMMU provides hardware support for malloc() for multicore SoCs but does not support FIFOs, automatic VDD-gating, or handles
• Smart memories feature a GP-CPU in multiple tiles but no target accelerators or support automatic VDD-gating
• Memory sharing within functional units
• server blades
  investigated as well, although these approaches are not designed for accelerators and do not support automatic VDD-gating, handles, or FIFOs
Conclusion

The accelerator store, a shared memory framework for accelerator-based systems was introduced here. We described the accelerator store’s handle-based approach to sharing accelerator memory, characterized accelerator memory to estimate the potential for area reduction and performance degradation, and achieved system area reductions of 30% with less than 1% performance impact and no additional energy in a simulated prototype.