Improving Scalability of CMPs with Dense ACCs Coverage

Nasibeh Teimouri, Hamed Tabkhi and Gunar Schirner

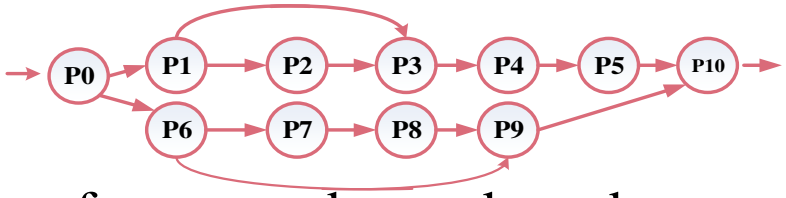
Embedded System Lab. (ESL)
Department of Electrical and Computer Engineering
Northeastern University, Boston (MA), USA

Northeastern

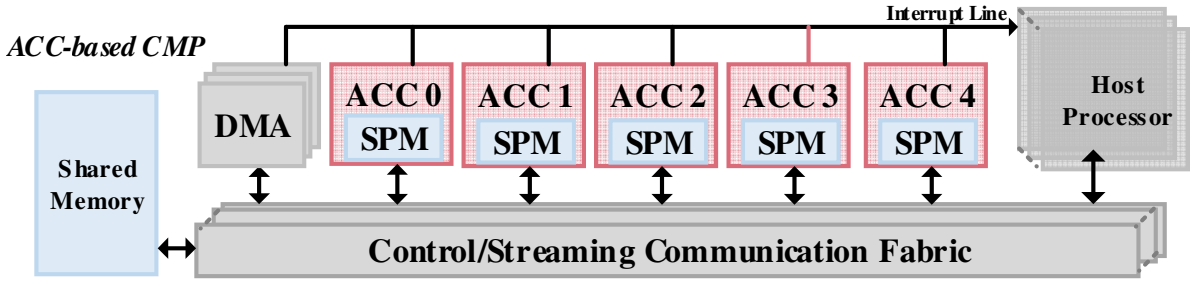
Embedded Systems Laboratory

Context:

- Embedded performance-demanding streaming applications
  - Vision, software-defined radio, multimedia



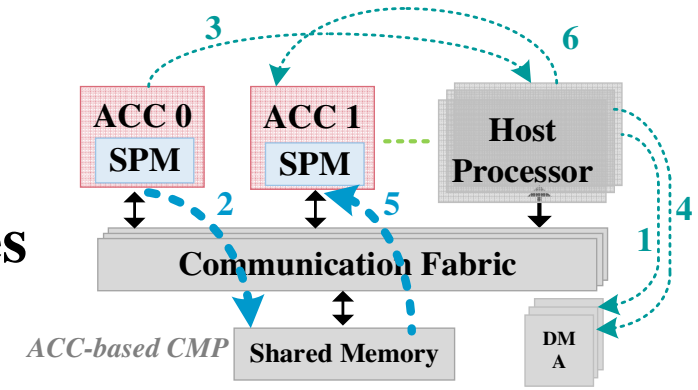
- Heterogeneous implementation to meet performance demands under stringent constraints
- Accelerator-based Chip Multi-Processors (ACMPs)

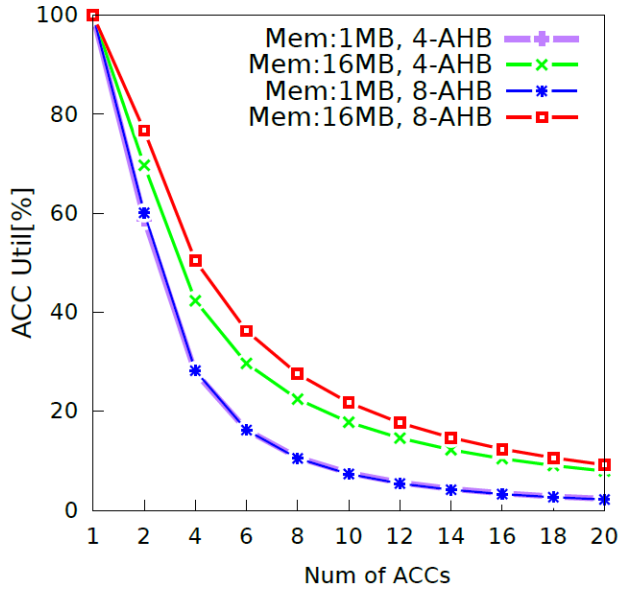


  - Trends (among others):
    - Increasing ACC coverage
    - Increasing density
      - Adjacent nodes in HW

Challenges with Denser ACCs Coverage

- **Processor-centric view**
  - System orchestration by processor
  - Processor becomes bottleneck
- **High contention on shared resources**
  - Memory: local/shared data
  - System Communication Fabric: ACC-to-ACC traffic
  - Processor
- **Unclear ACC comm. semantic**
  - Rely on processor interaction
- **Scalability severely limited with denser ACCs coverage *[DAC'15]***
  - → **System bottlenecks**
  - → **ACCs underutilized**





Problem Formulation and Contribution

1. Define semantics of ACC communication / interaction
    - Foundation for direct ACC-to-ACC communication
2. Transparent Self-Synchronizing (TSS) Architecture template
    - Realizes semantics
        - Mitigate system bottlenecks
    - Peer view between processor and ACCs

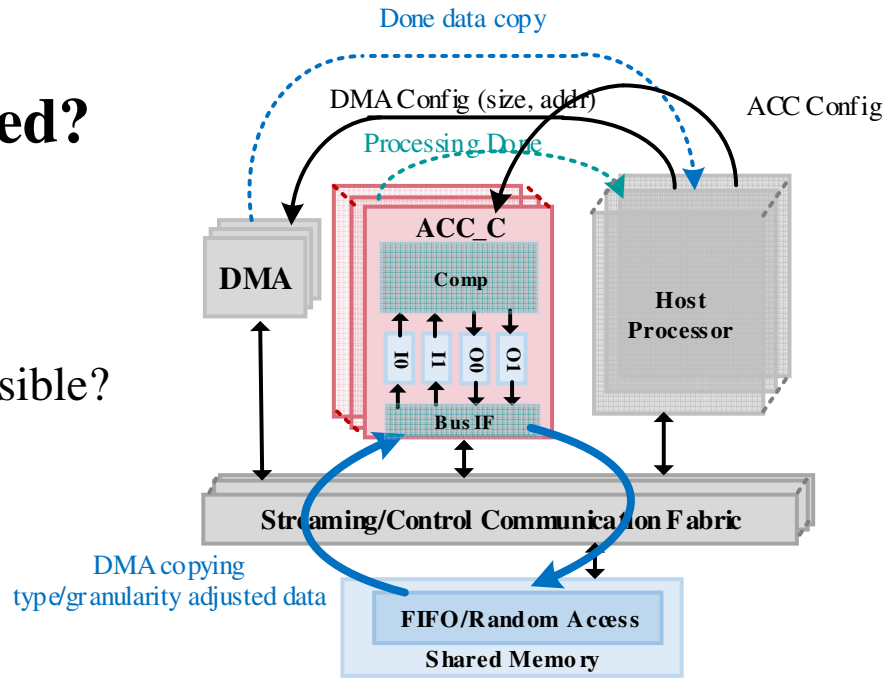
Outline

- Trend: Increasing ACC Coverage and Density
  - Motivation and challenges
- Problem Definition
- ACC-to-ACC Communication Semantics
- Transparent Self-Synchronizing (TSS) Architecture Template
- Experimental Results
- Conclusions

ACC Communication Aspects

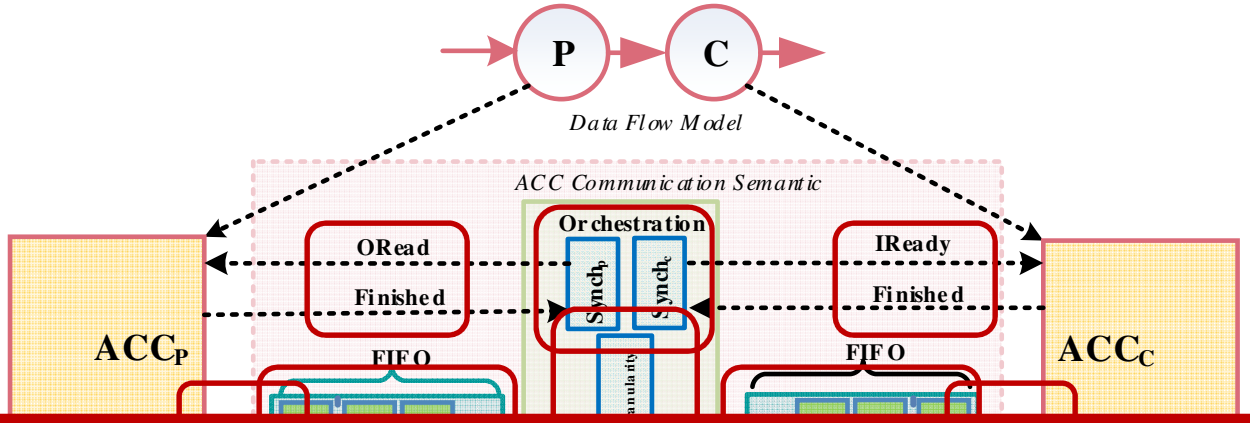
- **Orchestration by processor**
  - Data preparation according to the ACC job size, data type
  - Synchronization of the ACCs and DMA
    - **DMA** data transfer from/to ACC's **memory** through **communication fabric**

- **Which aspects need to be defined?**
  1. **Granularity of processing**:
     ACC job size?
  2. **Data access model**:
     When and which memory region accessible?
  3. **Marshaling / Data Representation**:
     Adjust data type for ACC input
  4. **Synchronization**:
     Start, stop, flow control?



ACC Communication Semantic

- **Synchronization / Control**
  - Initializing ACC for each computation and managing FIFO access
    - Synchronization signals “Iready”, “Oread” and “Finished”
- **Data access model**
  - Double buffering
  - More general: FIFO with head/tail Random Access (RA)
- **Granularity and marshaling management**
  - Data type/size adjustment of input/output of input/output data



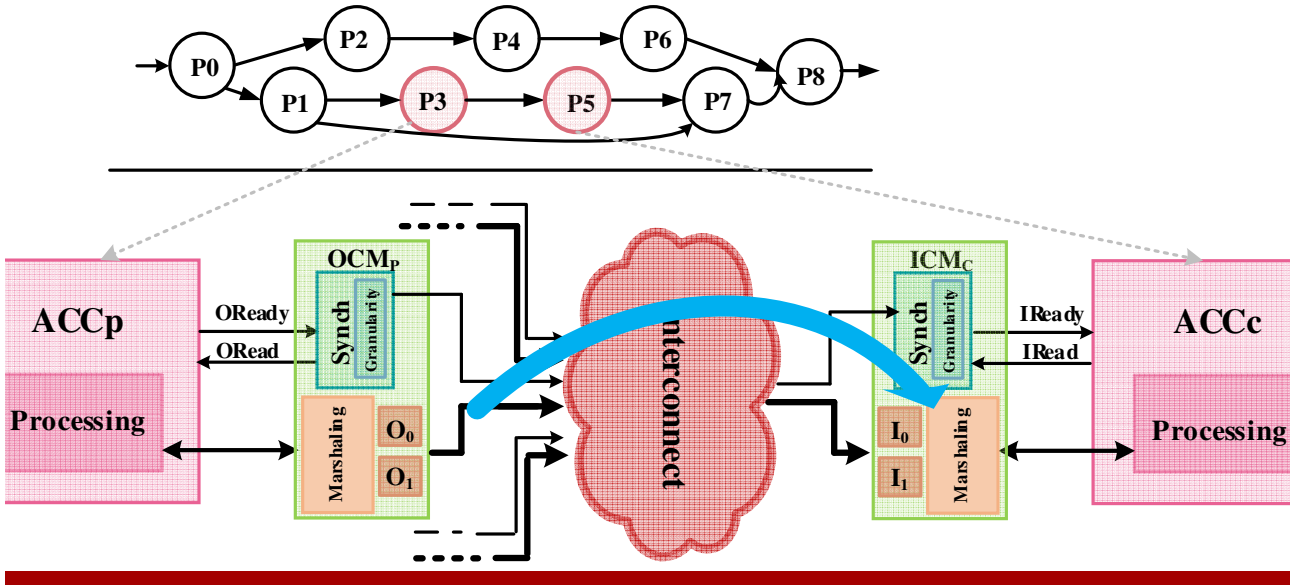
- **All semantic aspects currently involve processor!**
- Even for ACC-to-ACC communication

Outline

- Trend: Increasing ACC Coverage and Density
  - Motivation and challenges
- Problem Definition
- ACC-to-ACC Communication Semantics
- Transparent Self-Synchronizing (TSS) Architecture Template
- Experimental Results
- Conclusions

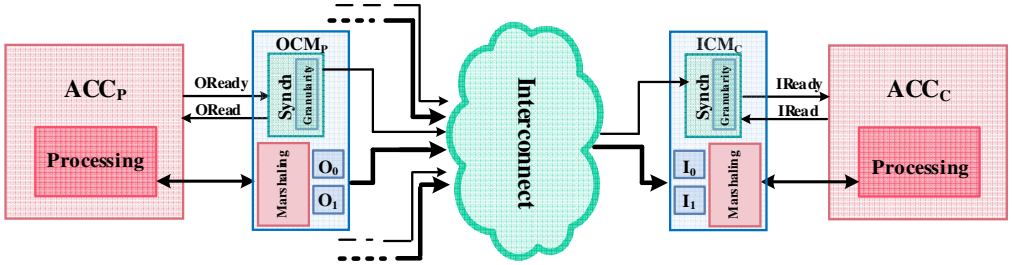
TSS: ACC-to-ACC Communication

- **Separation of computation and communication**
  - Input Control Mgmt (**ICM**) and Output Control Mgmt (**OCM**)
    - Efficient realization of the comm. semantics
      - Data access (I/O buffer)
      - Synchronization, data granularity management
      - Data marshalling
- **Local interconnect across the ACCs**
  - Hides ACC-to-ACC traffic from system bus

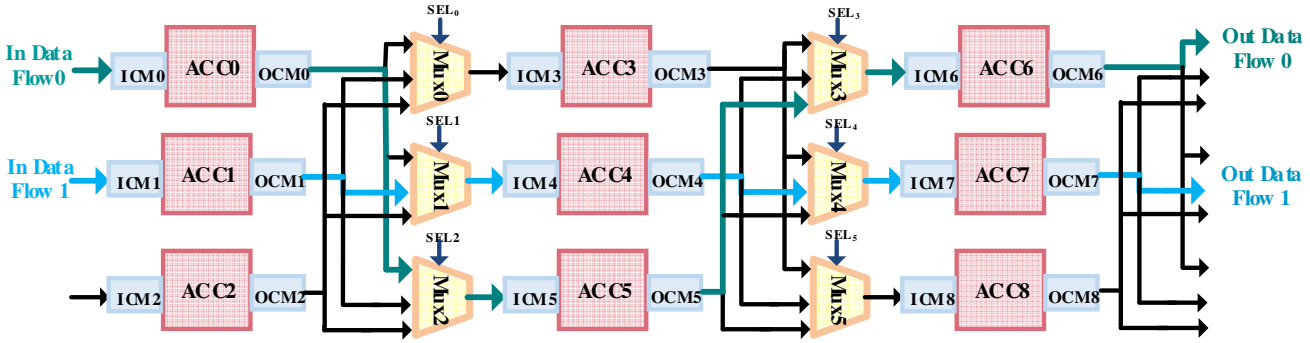


TSS: Interconnect Network

- **Interconnection network**
  - Many options: MUX, NoC, Bus
  - Full connectivity not needed (only feasible connections)
  - Depends on domain

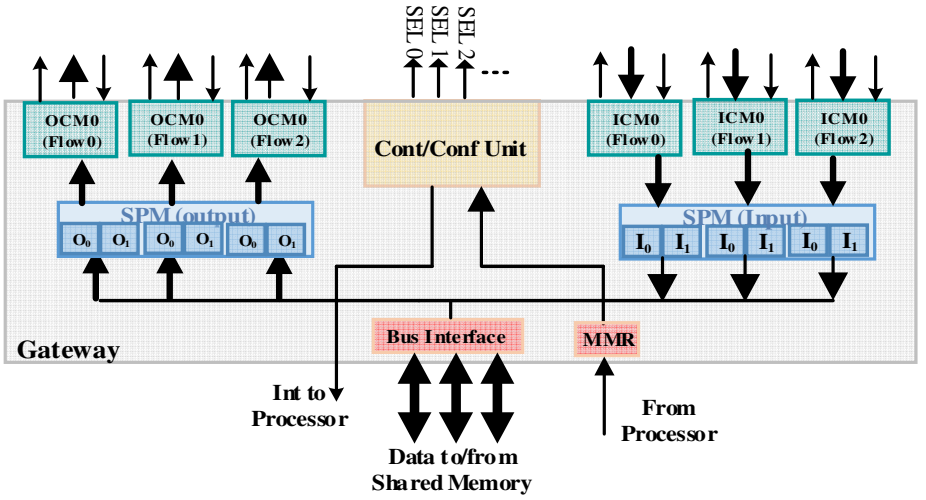


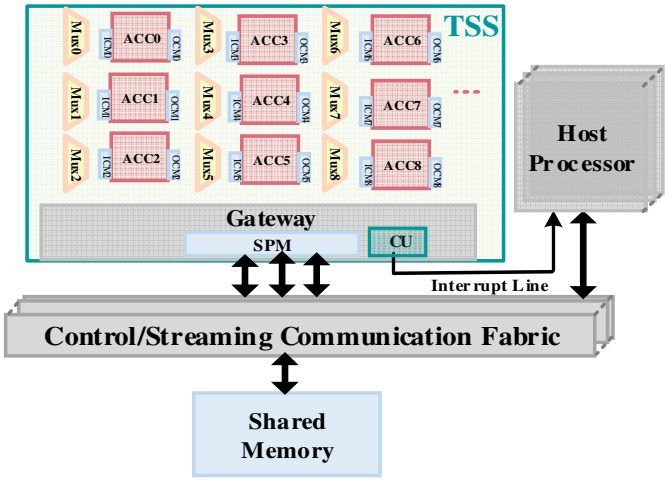
- **Current choice: MUX based interconnect**
  - Simplicity
  - Parallelism



TSS: System Integration and Benefits

- **Gateway**
  - Interface to system for each flow/stream (ACC chain)
  - Configuration & control
  - Granularity adjustment
- **Benefits**
  - Each ACC chain appears as one ACC to processor
    - Hides all internals
  - Much smaller internal granularity
    - Minimal as per ACC's algorithm
    - **Reduces on-chip memory**





Outline

- Trend: Increasing ACC Coverage and Density
  - Motivation and challenges
- Problem Definition
- ACC-to-ACC Communication Semantics
- Transparent Self-Synchronizing (TSS) Architecture Template
- Experimental Results
- Conclusions

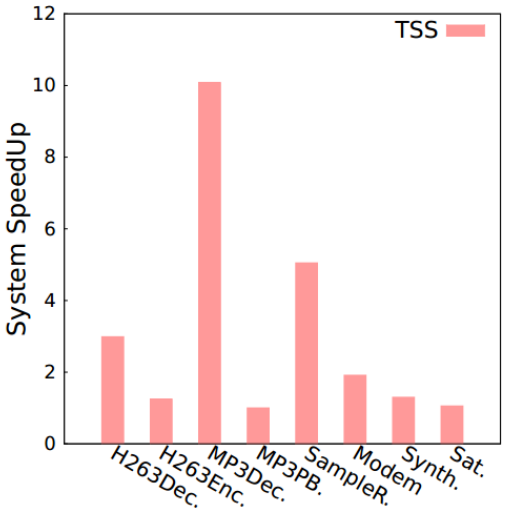
Experimental Setup

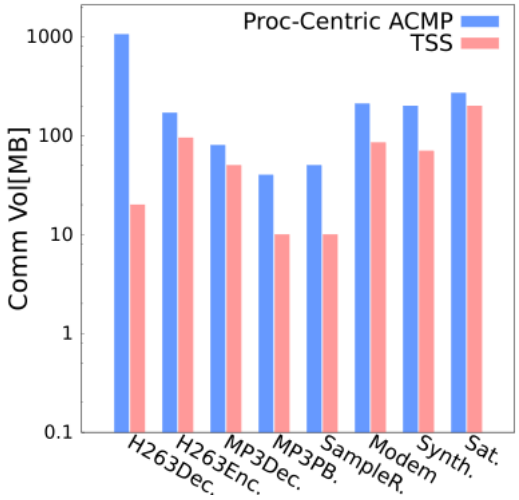
- **Compare: Processor-centric ACMP, TSS**
  - Same HW / SW Mapping
  - Impact of architecture on performance?
- **8 streaming applications (SDF3)**
  - H263Dec, H263Enc, MP3dec, MP3PB, Sam.Rate, Modem, Synthetic, Satellite
- **ISS-based (OVP) Virtual platforms**
  - Automatically generated
  - 2MB total on-chip mem

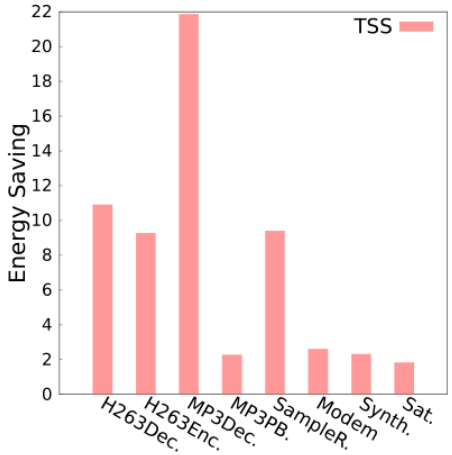
Virtual Platform Settings	
Processor	-ARM9 /500MH -OS : UCOS II
Communication Fabric	-Multi-layer AMA-AHB (32-bit) -Freq: 200MHz -Dedicated DMA per channel
Memory	- 2 MB
ACCs	-Double-buffered -Freq: 200MHz

TSS over ACMP: System Performance and Memory Saving

- Average speedup: 3 times
  - Minimize interaction with the processor
    - 1/7th of orchestration demand
      - Self-synchronization (OCM/ICM)
  - Reduces system load
    - 1/7th (avg) of on-chip memory
      - Smaller internal job size
    - 1/10th of traffic on system fabric
      - ACC-to-ACC comm. fabric.
- 1/8th energy consumption
  - Fewer off-chip access
  - Smaller on-chip mem.







Outline

- Trend: Increasing ACC Coverage and Density
  - Motivation and challenges
- Problem Definition
- ACC-to-ACC Communication Semantics
- Transparent Self-Synchronizing (TSS) Architecture Template
- Experimental Results
- Conclusions

Conclusions

- **Defined semantic aspects of ACC communication**
  - Synchronization
  - Data access model
  - Data granularity
  - Data representation / marshalling
- **Introduced architecture template Transparent Self-Synchronizing (TSS)**
  - Efficient realization of semantics
    - Separation computation / communication with ICM/OCM
    - Internal interconnect network
    - Adjustable internal granularity (through gateway)
  - Each ACC chain regardless of length appears as one ACC
- **Illustrated architecture benefits (processor-centric vs. TSS)**
  - 8 streaming apps (SDF3) mapped to ISS-based VPs
  - 3x speedup (at 1/8th energy consumption) with same HW/SW mapping

Thank you!