Revisiting Accelerator-Rich CMPs: Challenges and Solutions

Nasibeh Teimouri, Hamed Tabkhi and Gunar Schirner

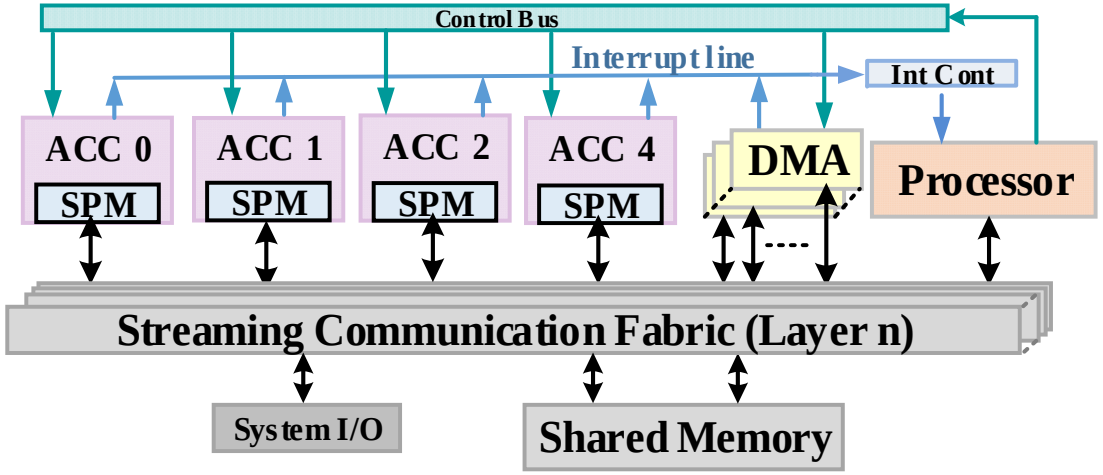
Embedded System Lab. (ESL)
Department of Electrical and Computer Engineering
Northeastern University, Boston (MA), USA

Northeastern

Embedded Systems Laboratory

ACC-based CMPs

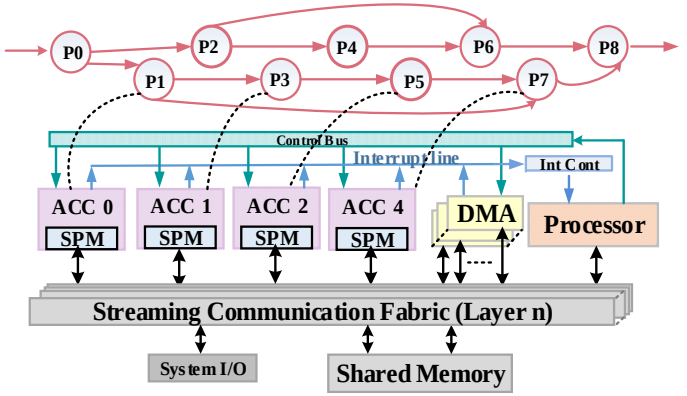
- Many ACCs coupled with processor core(s) on a single chip
  - To deliver power / performance efficiency
    - **ACC(s)** for compute-intense kernels
    - **Core(s)** for remaining portion of programs + control / synchronization



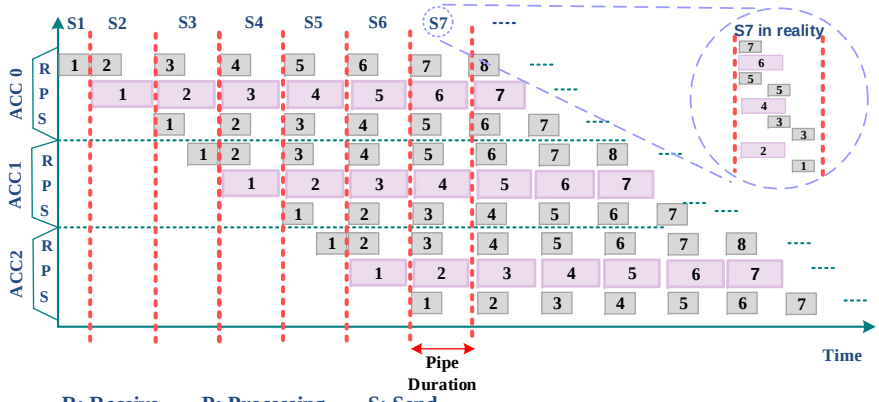
- **Architecture supports**
  - Scratch Pad Memory (SPM) per ACC
  - Shared memory
  - Direct Memory Transfer (DMA)
  - Multi-channel streaming communication fabric
    - E.g. NoC, AXI, ML-AHB
  - Control bus
  - Interrupt lines

Scalability Limitation of ACC-Rich CMPs

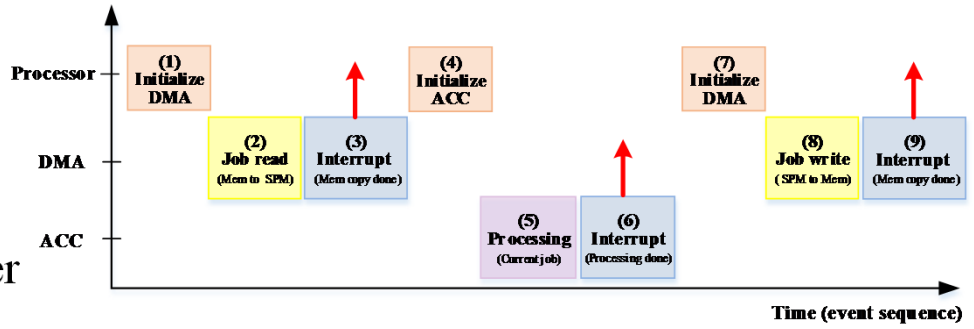
- Streaming applications suitable for ACC-BASED CMPs
  - E.g. vision, multimedia and etc.
  - Captured in data flow models



- **Ideal:** Perfect pipeline
  - Producer/Consumer kernels working in parallel based on availability of data
- **Challenge:** Serialization effect
  - Increasing # of ACCs and competition over the shared resources



- **Shared resources**
  1- Processor for synchronization/control
  2- On-chip memory for local SPMs
  3- Communication fabric for data transfer



Contributions

1. **Holistic view** and **formulation** of **scalability limitations** of ACC-based CMPs
   - To bring insights about the source of inefficiency of ACC-based CMPs
   - To open a path toward efficient architecture solutions

2. An **architecture template** to mitigate the resource bottlenecks
   - To allow an efficient realization of streaming applications with ACCs

3. An **experimental evaluation** through automatically generated Virtual Platforms (VPs)
   - To validate the proposed analytical/formulated model of ACC-based CMP
   - To demonstrate the benefits of proposed architecture template.

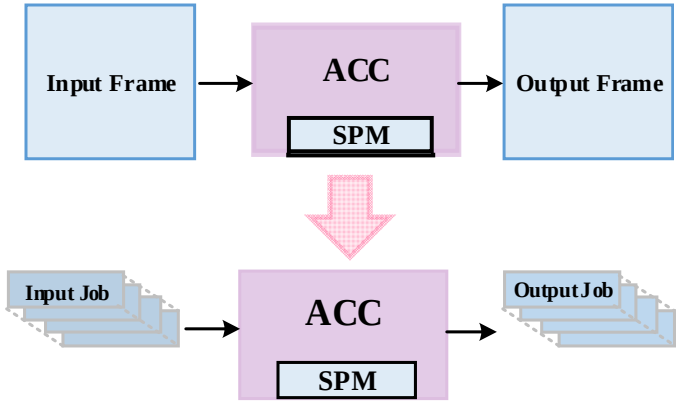
Related Work

- Few works hint to the scalability issues of ACC-based CMPs
  - Studying only one resource bottleneck
    - Memory [ISLPED_12,CAL_14]
      - Run-time SPM optimization per ACC regarding run-time needs of ACCs
    - Host Processor [DAC_14]
      - Accelerator Block Composer~(ABC)
    - Communication Fabric [CISP_09, DATE_13]
      - Hierarchical interconnection to localize inter-ACC traffic

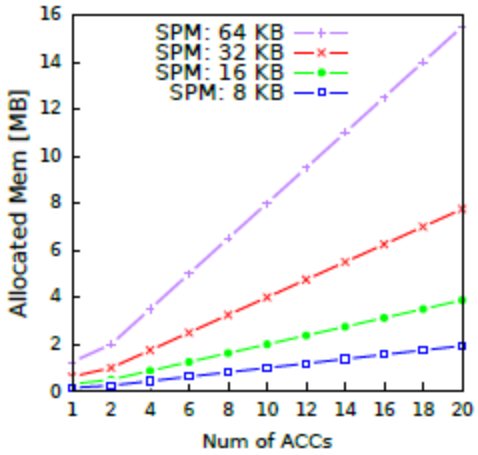
→ Lack of insights and holistic view of the limitations of ACC-based CMPs

Holistic View of Scalability Limitations (1)

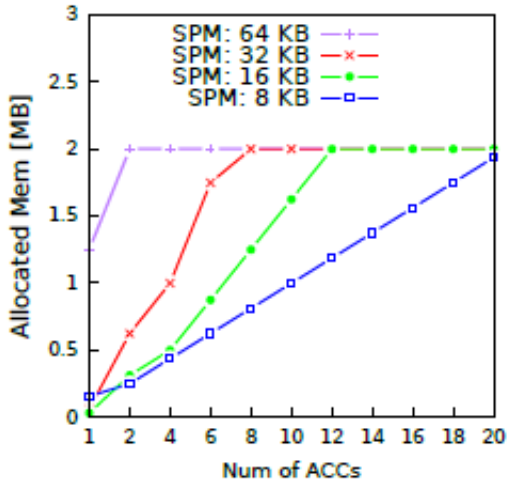
- SPM contains data under processing
  - Impossible to allocate a large SPM per ACCs
    - Due to limited on-chip memory budget
  - Data split into smaller chunks (Job)
  - Job size depends on size of SPM



- Increasing #of ACCs
  - Linear increase of memory demand
    - Assumption: fixed job size / SPM size



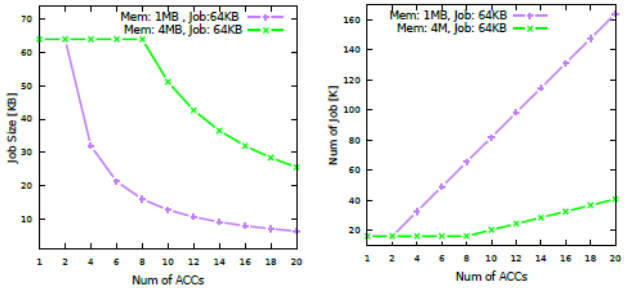
  - On-chip memory limitation



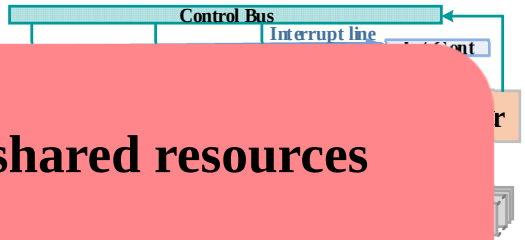
→ **Need to reduce job size with increasing # of ACCs**

Holistic View of Scalability Limitations (2)

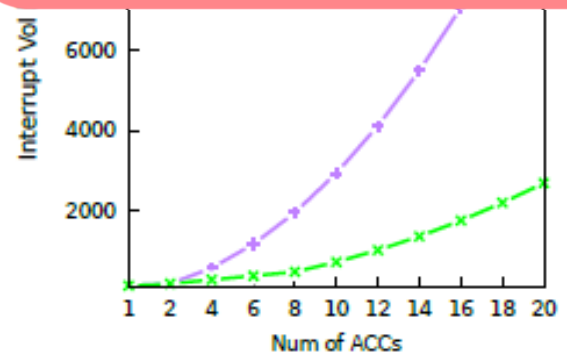
- Increasing #of ACCs with fixed memory
  - Decreasing job size
  - Increasing # of jobs



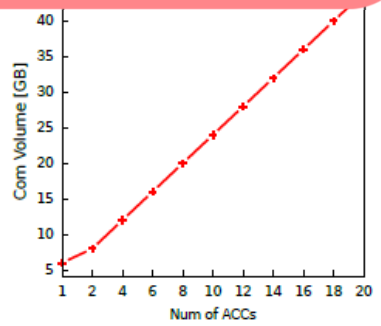
- Increasing # of jobs
  - More synchronization load on host processor



- **Increasing # of ACCs → Excessive demand on the shared resources**
  - **1- Memory**
  - **2- Host processor**
  - **3- Communication fabric**



- Linear increase in communication volume
  - To transfer ACC-to-ACC streaming data



Mathematic Analysis of the Scalability Concerns

- Extracting the mathematic analysis
  - Taking competition over the shared resources and serialization /arbitration effects in run-time execution
    - #of Memory ports and memory controller frequency
    - Processor's frequency
    - #of parallel access to communication fabric and BW

Assumptions	
Processor	- 1 core /1GH ($Freq_{Proc}$) - Light OS : 20000 cycles as ISR latency (~$Latency_{ISR}$)
Communication Fabric	- 4 and 8 parallel layers, each one 32-bit width with dedicated DMA
Memory	- 1MB and 16MB
ACCs	- Double-buffered SPMs - Computating 1 byte per cycle at 200MHz ($Freq_{ACC}$)

$$Time_{Exec} = (Num_{Job} + Num_{ACC} - 1) * Latency_{Pipe}$$

$$Latency_{Pipe} = (Computation_{ACC} + Communication_{ACC})$$

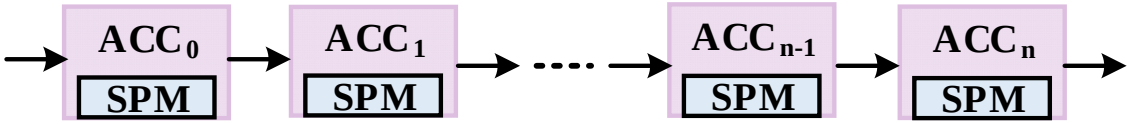
$$Latency_{Pipe} = Max\{Latency_{ACC\_Operation}, Latency_{ACC\_Traffic}\} + Latency_{Synch} + Latency_{Arb}$$

$$Latency_{ACC\_Operation} = (\frac{Size_{Job}}{ACC_{Freq}})$$

$$Latency_{ACC\_Traffic} = Size_{Job} * (\frac{Bus_{layers}}{Bus_{Freq}} + \frac{Mem_{ports}}{Mem_{Freq}})$$

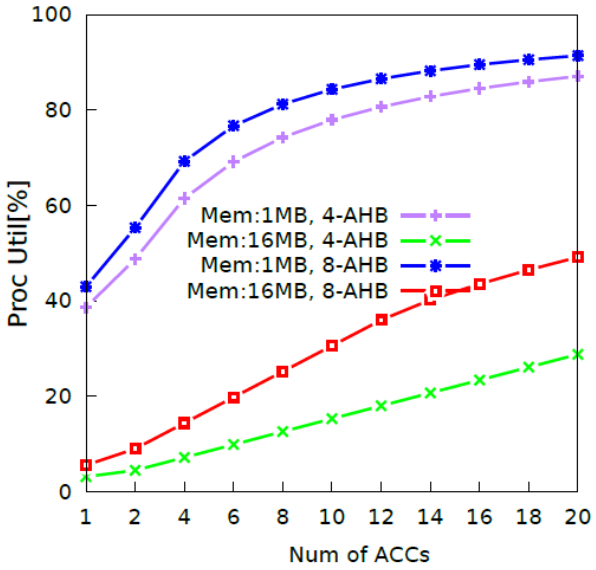
$$Latency_{Synch} = Num_{ACC} * 3 * Freq_{Proc} + Latency_{ISR}$$

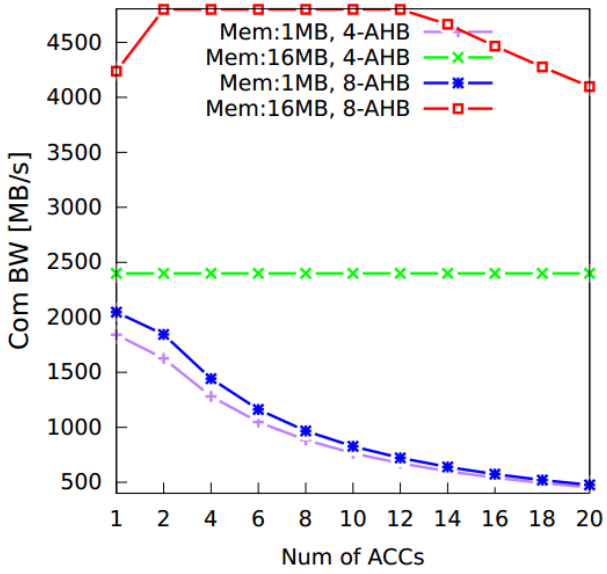
- Benchmark: Chain of kernels in producer/consumer fashion, all on individual ACC



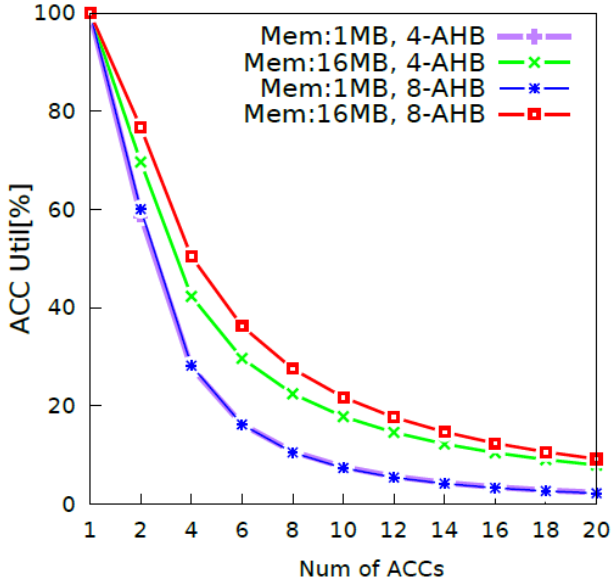
Analysis Results

	#of AHB layers	Mem Size (MB)
Scenario 1	4	1
Scenario 2	4	16
Scenario 3	8	1
Scenario 4	8	16





- **Increasing #of ACC with fixed size of memory**
  1. Smaller size of job → More Synch requests
     → Higher processor utilization
  2. Heavier volume of traffic to/from memory
     → Busier communication fabric and memory



➢ **ACCs dependency on the shared resources**

Busier shared resources → **Less ACCs utilization**

Solution for the Scalability Concerns

- **Architecture demand toward reducing the load on the shared resources**

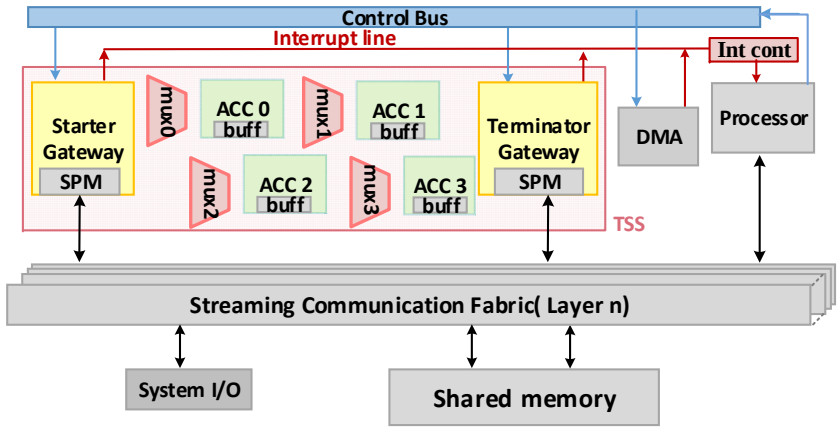
1. Traffic over the **Communication fabric**

1. The synchronization load on the **Processor**

3. The pressure on the **Shared memory**

- **The proposed Architecture template (TSS)**

1. Point-to-point connections among ACCs
   - MUX-based interconnect across ACCs
     - -Direct data transfer
     - -Reducing the traffic on the **Communication fabric**



2. Self-Synchronization per ACC
   - Gateway
     - -Splitting the data into smaller chunks
     - -ACCs' self synchronization/operation of all the chunks
     - -Eliminating the synchronization request to the **Processor**

→3. Possible smaller SPMs per ACC
   - Reducing the pressure on the **Memory**

Experimental Platform

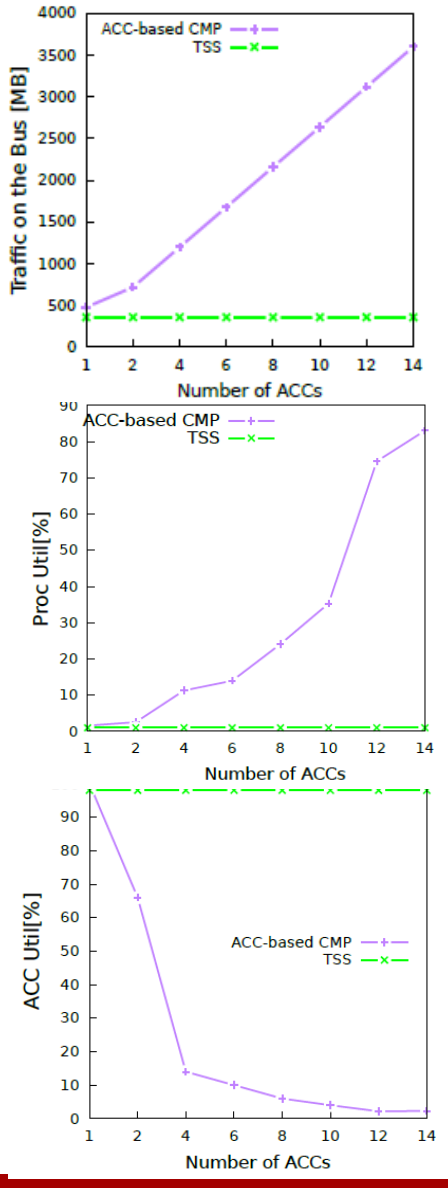
- Automatically generated Virtual Platforms for both ACC-based CMP and TSS
  1. To validate the proposed analytical model and demonstrate the scalability issues in real platform
  2. To show how the proposed TSS outperforms ACC-based CMP

- **VP setup**
  - Cycle approximate model of ARM9
  - ML-AHB captured in cycle accurate model (BFM)
- **Processing data**
  - 60MB
- **Application**
  - Chain of producer/consumer kernels
  - Real app: Object Tracking Vision Flow

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

TSS vs. ACC-based CMP

- To validate the proposed analytical model
  - Increasing #of ACCs
    - Growing volume of traffic over the communication fabric
    - Increasing processor utilization
    - Decreasing ACC utilization
- To compare TSS vs. ACC-based CMP
  - Increasing #of ACCs
    - Masking traffic to < 14%
    - Reliving the load on the processor to < 2%
    - Improving ACC utilization to > 90%



Summary

- **Systematic analysis of scalability issues in ACC-based CMP**
  - Holistic view of shared resources bottleneck emerging in ACC-based CMP with growing #of ACCs
    - ACC under-utilization no matter how much resources are exploited
  - Mathematic formulation of the scalability issues

- **An architecture template to relieve the bottlenecks on the shared resources**
  - Hiding ACCs computation/communication from the shared resources

- **Experimental evaluation through automatically generated Virtual Platforms (VPs) for both ACC-based CMP and TSS**
    - Verifying the proposed mathematic formulation
    - Demonstrating how the proposed template architecture overcome ACCs scalability

Question Please!

Thank you…