BusyMap, an Efficient Data Structure to Observe Interconnect Contention in SystemC TLM-2.0

Emad Arasteh[1], Vivek Govindasamy[2], and Rainer Dömer[2]
arasteh@chapman.edu

1 Fowler School of Engineering, Chapman University, Orange, CA, USA
2 Henry Samueli School of Engineering, University of California, Irvine, USA

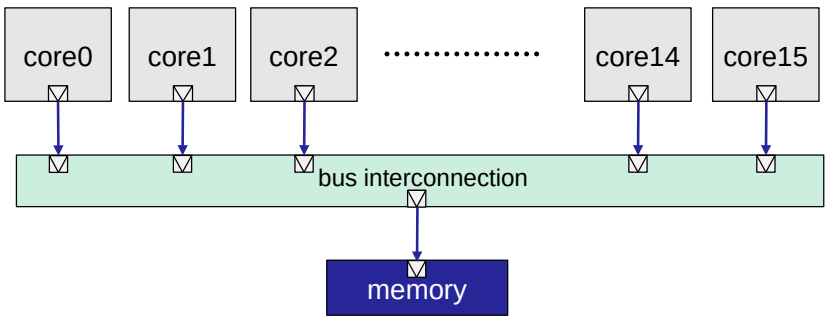
UCIRVINE
UNIVERSITY of CALIFORNIA · IRVINE

System Platform Exploration Lab (SPEL)
Center for Embedded and Cyber-physical Systems (CECS)

Problem Definition

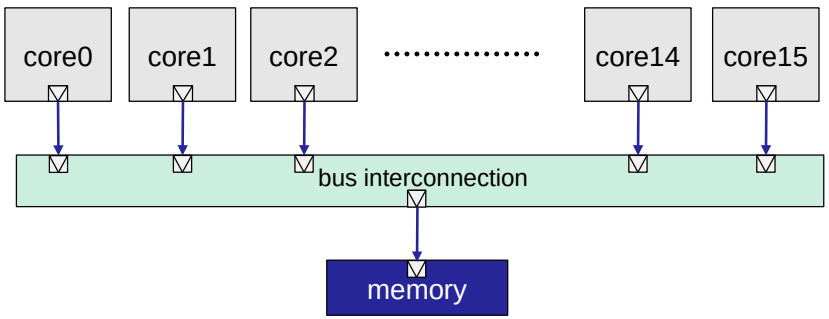
Problem Definition

- Bus contention is a critical aspect in modeling modern multiprocessor system on a chip (MPSoC)



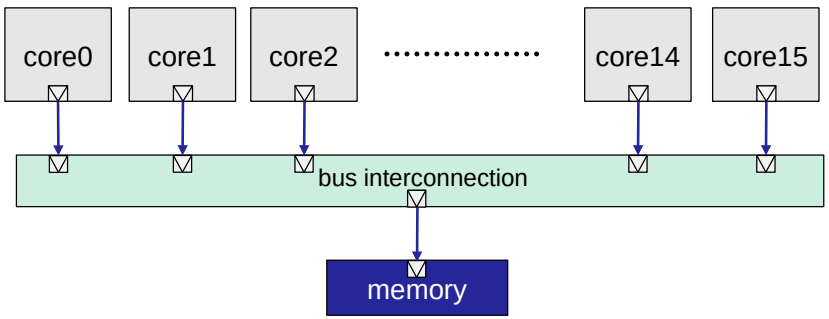
Problem Definition

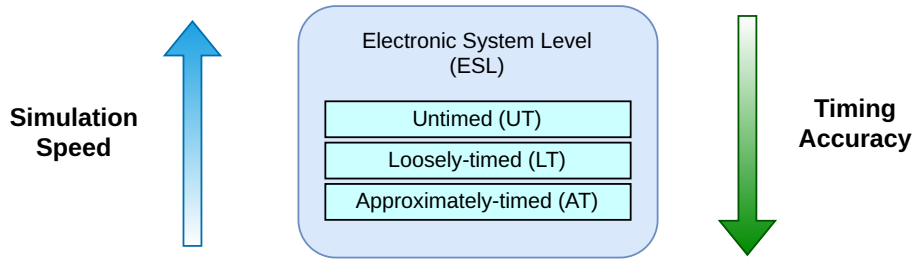
- Bus contention is a critical aspect in modeling modern multiprocessor system on a chip (MPSoC)
- SystemC TLM-2.0 aids system designers with performance estimation



Problem Definition

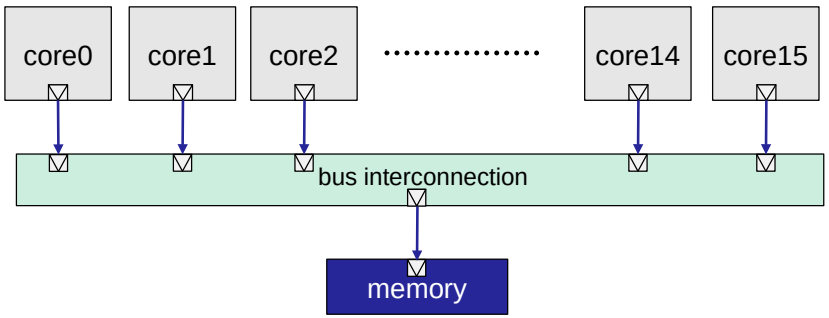
- Bus contention is a critical aspect in modeling modern multiprocessor system on a chip (MPSoC)
- SystemC TLM-2.0 aids system designers with performance estimation
  - Loosely-timed (LT)
    - adequate timing, fast simulation, no notion of contention

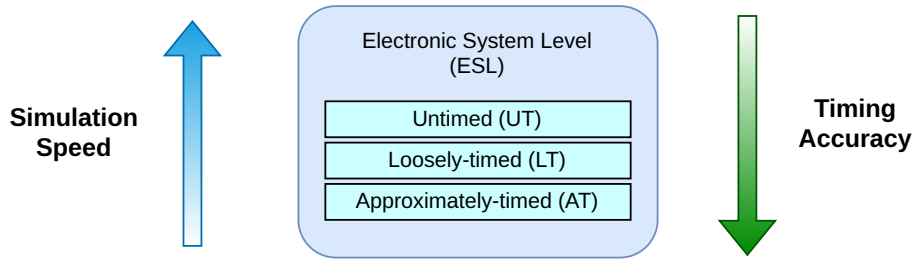




Problem Definition

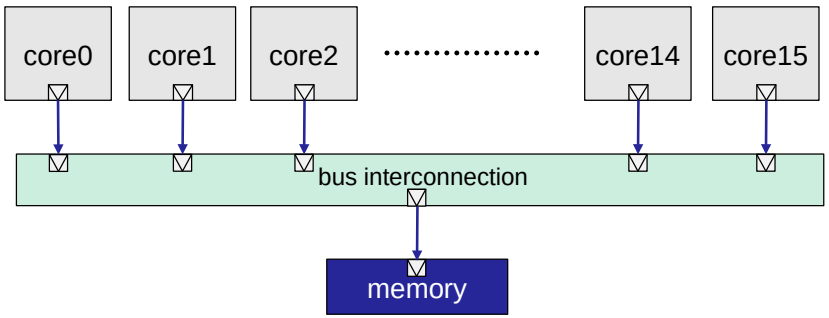
- Bus contention is a critical aspect in modeling modern multiprocessor system on a chip (MPSoC)
- SystemC TLM-2.0 aids system designers with performance estimation
  - Loosely-timed (LT)
    - adequate timing, fast simulation, no notion of contention
  - Approximately-timed (AT)
    - accurate timing, slow simulation, complex coding, model contention

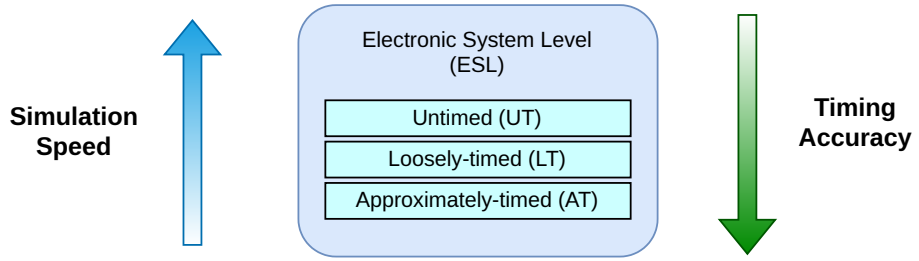




Problem Definition

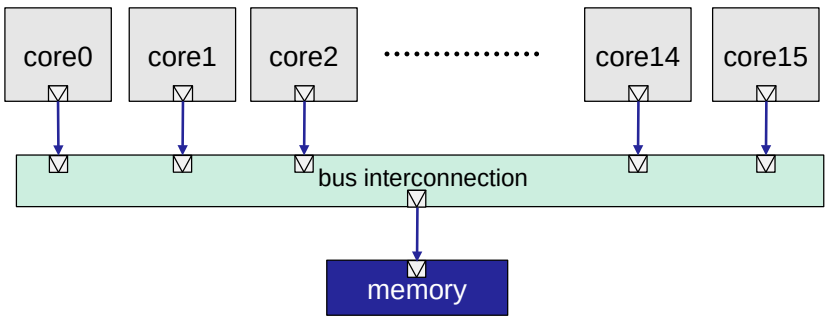
- Bus contention is a critical aspect in modeling modern multiprocessor system on a chip (MPSoC)
- SystemC TLM-2.0 aids system designers with performance estimation
  - Loosely-timed (LT)
    - adequate timing, fast simulation, no notion of contention
  - Approximately-timed (AT)
    - accurate timing, slow simulation, complex coding, model contention
- Can we model contention fast but accurately for early system design?

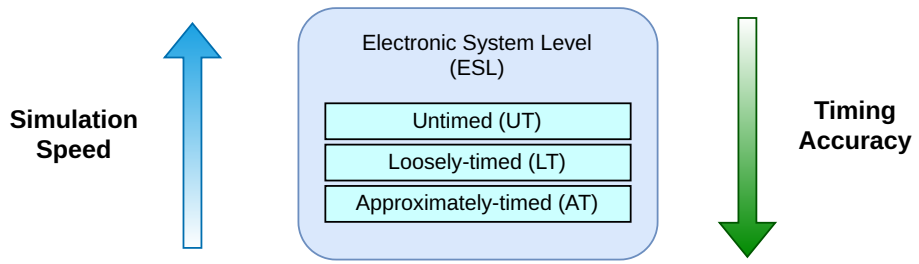




Problem Definition

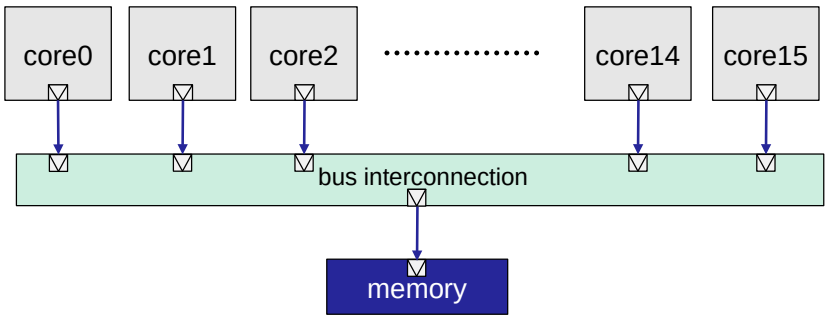
- Bus contention is a critical aspect in modeling modern multiprocessor system on a chip (MPSoC)
- SystemC TLM-2.0 aids system designers with performance estimation
  - Loosely-timed (LT)
    - adequate timing, fast simulation, no notion of contention
  - Approximately-timed (AT)
    - accurate timing, slow simulation, complex coding, model contention
- Can we model contention fast but accurately for early system design?
  - Should support temporal decoupling and multi-level interconnects

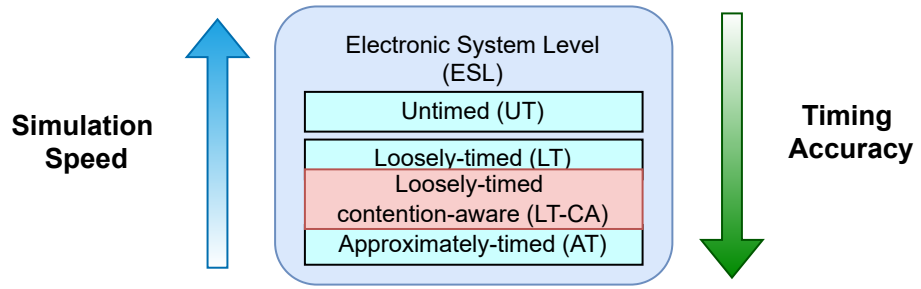




Problem Definition

- Bus contention is a critical aspect in modeling modern multiprocessor system on a chip (MPSoC)
- SystemC TLM-2.0 aids system designers with performance estimation
  - Loosely-timed (LT)
    - adequate timing, fast simulation, no notion of contention
  - Approximately-timed (AT)
    - accurate timing, slow simulation, complex coding, model contention
- Can we model contention fast but accurately for early system design?
  - Should support temporal decoupling and multi-level interconnects
- We introduce Loosely-Timed Contention-Aware (LT-CA) modeling to model contention fast, accurate, and early in the design flow

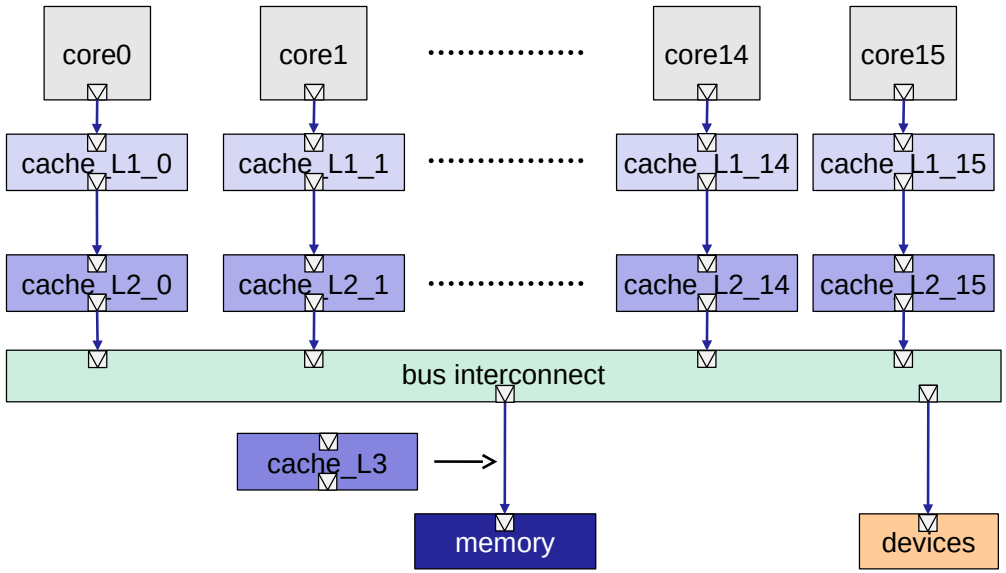




Loosely-Timed Contention-Aware (LT-CA)

Loosely-Timed Contention-Aware (LT-CA)

- Our key contributions
  - TLM-2.0 loosely-timed resource contention modeling supporting **temporal decoupling** with **high accuracy** at **high-speed** simulation
  - Support multiple-level hierarchical interconnects, including **multi-level caches** or **multiple levels of buses**



LT-CA BusyUntil Contention

LT-CA BusyUntil Contention

- In our prior work, we propose **BusyUntil**, which uses a state variable in the interconnect to keep track of contention

LT-CA BusyUntil Contention

- In our prior work, we propose **BusyUntil**, which uses a state variable in the interconnect to keep track of contention

Algorithm 1: Modeling bus contention using a time stamp `busy_until` (adapted from [14])

```
Module Bus_BusyUntil begin
    target_socket S_in[NUM_TARGETS];
    initiator_socket S_OUT[NUM_INITIATORS];
    time bus_delay;
    time contention := 0;
    time busy_until := 0;
    Procedure ForwardRequest (trans, delay) begin
        // perform address translation
        socket := decode_and_translate(trans.address);
        // forward the transaction
        d1 := delay;
        socket→b_transport(transaction, delay);
        d2 := delay;
        memory_delay := d2 - d1;
        // maintain bus contention
        busy_span := bus_delay + memory_delay;
        busy_until := max(busy_until, global_time);
        busy_delay := busy_until - global_time;
        busy_until += busy_span;
        contention += busy_delay;
        delay += bus_delay + busy_delay;
    end
end
```

LT-CA BusyUntil Contention

- In our prior work, we propose **BusyUntil**, which uses a state variable in the interconnect to keep track of contention
- Simple and effective approach but requires improvements for
  - temporal decoupling
  - multi-level interconnects

Algorithm 1: Modeling bus contention using a time stamp `busy_until` (adapted from [14])

```
Module Bus_BusyUntil begin
    target_socket S_in[NUM_TARGETS];
    initiator_socket S_OUT[NUM_INITIATORS];
    time bus_delay;
    time contention := 0;
    time busy_until := 0;
    Procedure ForwardRequest(trans, delay) begin
        // perform address translation
        socket := decode_and_translate(trans.address);
        // forward the transaction
        d1 := delay;
        socket→b_transport(transaction, delay);
        d2 := delay;
        memory_delay := d2 - d1;
        // maintain bus contention
        busy_span := bus_delay + memory_delay;
        busy_until := max(busy_until, global_time);
        busy_delay := busy_until - global_time;
        busy_until += busy_span;
        contention += busy_delay;
        delay += bus_delay + busy_delay;
    end
end
```

LT-CA BusyMap Contention (1)

LT-CA BusyMap Contention (1)

- We introduce a new data structure, **BusyMap**, to replace the state variable in **BusyUntil**

LT-CA BusyMap Contention (1)

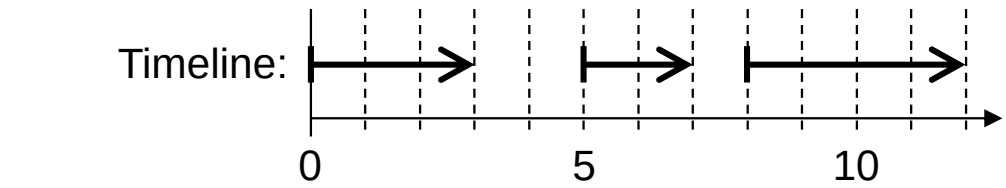
- We introduce a new data structure, **BusyMap**, to replace the state variable in **BusyUntil**
- **BusyMap** allows temporally decoupled initiator modules that use *out-of-order* transactions with different `delay` offsets from the simulator `global_time`

LT-CA BusyMap Contention (1)

- We introduce a new data structure, **BusyMap**, to replace the state variable in **BusyUntil**
- **BusyMap** allows temporally decoupled initiator modules that use *out-of-order* transactions with different `delay` offsets from the simulator `global_time`
- Ordered map of key-value (k, v) of sc_time
  - key k specifies the start time when the resource becomes busy
  - value v specifies the duration of how long the resource is used

LT-CA BusyMap Contention (1)

- We introduce a new data structure, **BusyMap**, to replace the state variable in **BusyUntil**
- **BusyMap** allows temporally decoupled initiator modules that use *out-of-order* transactions with different `delay` offsets from the simulator `global_time`
- Ordered map of key-value (k, v) of *sc_time*
  - key k specifies the start time when the resource becomes busy
  - value v specifies the duration of how long the resource is used



Busy Periods:	[0, 3)	[5, 7)	[8, 12)
Map Elements:	0→3	5→2	8→4
Type:	`std::map<sc_time,sc_time>`		

LT-CA BusyMap Contention (2)

LT-CA BusyMap Contention (2)

- Bus contention model contains the ordered **busy_map**
- Details of algorithm can be found in the paper

Algorithm 2: Modeling bus contention using BusyMap

```
Module Bus_BusyMap begin
    target_socket S_in[NUM_TARGETS];
    initiator_socket S_OUT[NUM_INITIATORS];
    time bus_delay;
    time contention := 0;
    ordered_map<time,time> busy_map;
    Procedure ForwardRequest (trans, delay) begin
        // perform address translation
        socket := decode_and_translate(trans.address);
        // forward the transaction
        d1 := global_time + delay;
        socket→b_transport(transaction, delay);
        d2 := global_time + delay;
        memory_delay := d2 - d1;
        // maintain bus contention
        busy_span := bus_delay + memory_delay;
        AdvanceTime();
        available_slot := FindFreeSlot(d1, busy_span);
        busy_delay := availabe_slot - d1;
        SetBusy(available_slot, busy_span);
        contention += busy_delay;
        delay += bus_delay + busy_delay;
    end
    Procedure AdvanceTime begin
        if busy_map.empty() then
            return;
        end
        keep := busy_map.upper_bound(global_time);
        if keep = busy_map.begin() then
            return;
        end
        cut := prev(keep);
        if cut→start = global_time then
            busy_map.erase(busy_map.begin(), cut);
            return;
        end
        if (cut→start + cut→duration) > global_time then
            cut_duration := cut→start + cut→duration -
              global_time;
            busy_map.erase(busy_map.begin(), keep);
            busy_map[global_time] := cut_duration;
        else
            if keep = busy_map.end() then
                busy_map.clear();
            else
                busy_map.erase(busy_map.begin(), keep);
            end
        end
    end
end
```

Algorithm 3: Modeling bus contention using BusyMap (continued)

```
Function FindFreeSlot (earliest,span) begin
    if busy_map.empty() then
        return earliest;
    end
    iter := busy_map.upper_bound(earliest);
    if iter = busy_map.begin() then
        gap_at := 0;
    else
        gap_at := prev(iter)→start + prev(iter)→duration;
    end
    if gap_at<earliest then
        gap_at := earliest;
    end
    while iter ≠ busy_map.end() do
        gap_duration := iter→start - gap_at;
        if span ≤ gap_duration then
            return gap_at;
        end
        gap_at := iter→start + iter→duration;
        iter++;
    end
    return gap_at;
end
Procedure SetBusy (slot,span) begin
    if busy_map.empty() then
        busy_map[slot] := span;
        return;
    end
    r := busy_map.upper_bound(slot);
    if r = busy_map.begin() then
        if r→start = slot + span then
            busy_map[slot] := span + r→duration;
            busy_map.erase(r);
        else
            // no adjacency, insert a new element
            busy_map[slot] := span;
        end
        return;
    end
    l := prev(r);
    if l→start + l→duration = slot then
        if (r ≠ busy_map.end()) and (r→start = slot + span)
          then
            // merge in between adjacent elements
            l→duration += span + r→duration;
            busy_map.erase(r);
        else
            // merge with adjacent element on the left
            l→duration += span;
        end
    else
        if (r ≠ busy_map.end()) and (r→start = slot + span)
          then
            // merge with adjacent element on the right
            busy_map[slot] := span + r→duration;
            busy_map.erase(r);
        else
            // no adjacency, insert a new element
            busy_map[slot] := span;
        end
    end
end
```

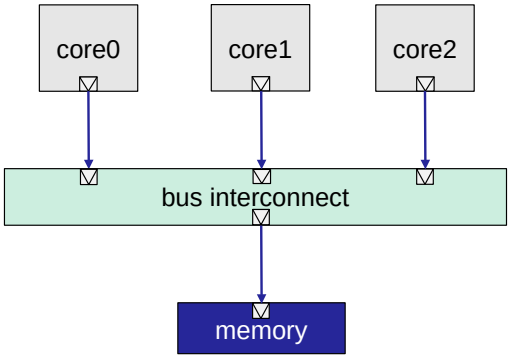
Experimental Measurements and Results (1)

Experimental Measurements and Results (1)

- **BusyUntil** and **BusyMap** on synthetic and real-world SystemC models

Experimental Measurements and Results (1)

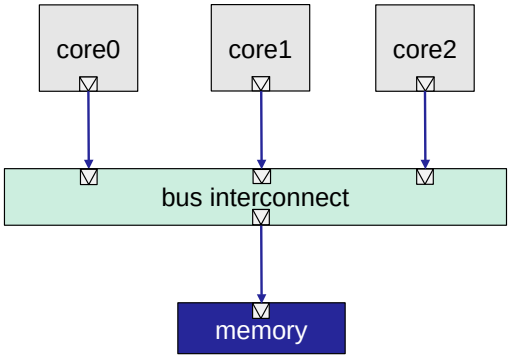
- **BusyUntil** and **BusyMap** on synthetic and real-world SystemC models
- **Bus3Init**



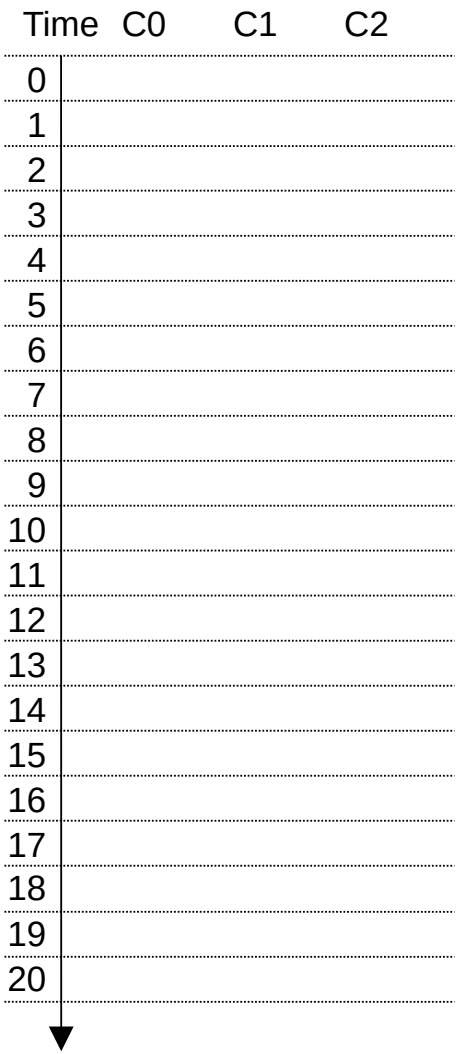
- Simulation trace for **BusyUntil** bus with global quantum value of zero

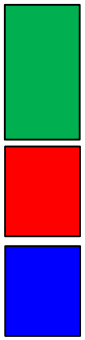
Experimental Measurements and Results (1)

- **BusyUntil** and **BusyMap** on synthetic and real-world SystemC models
- **Bus3Init**



- Simulation trace for **BusyUntil** bus with global quantum value of zero





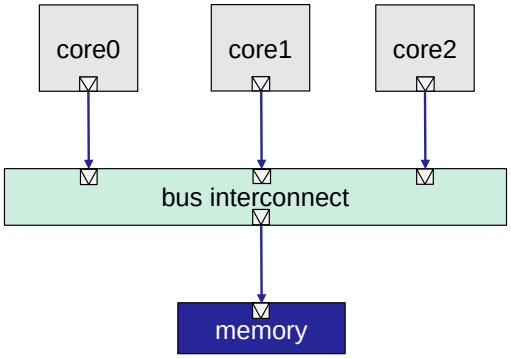
Computation

Bus contention

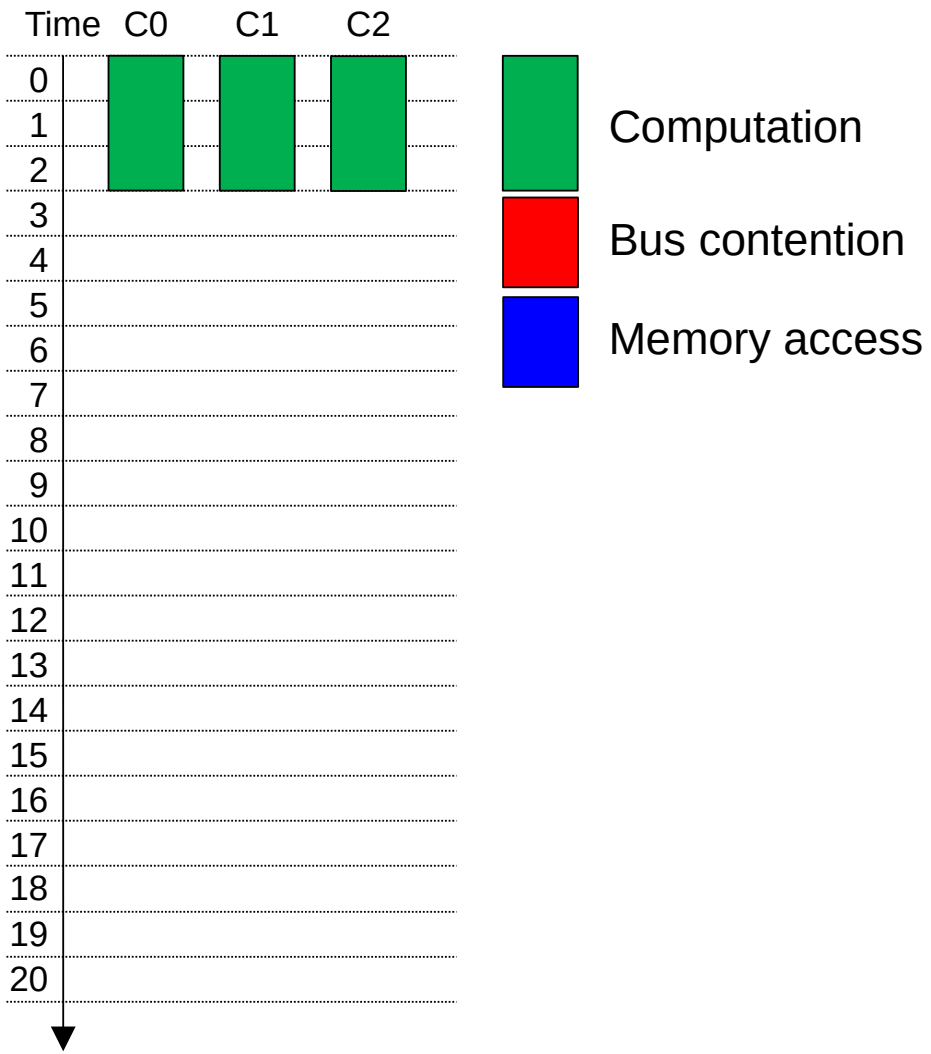
Memory access

Experimental Measurements and Results (1)

- **BusyUntil** and **BusyMap** on synthetic and real-world SystemC models
- **Bus3Init**

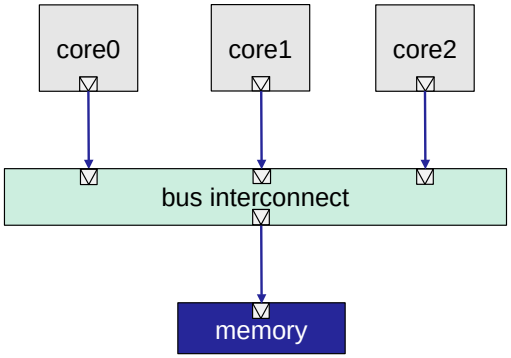


- Simulation trace for **BusyUntil** bus with global quantum value of zero

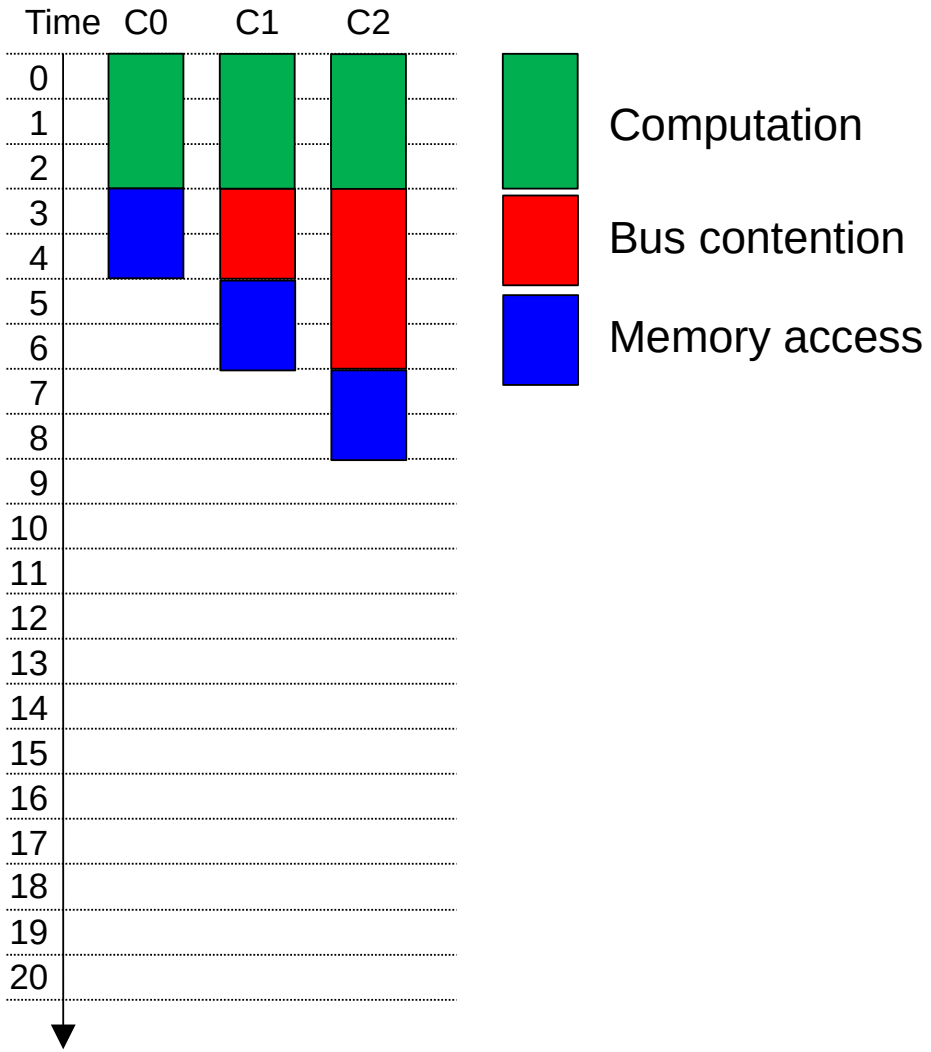


Experimental Measurements and Results (1)

- **BusyUntil** and **BusyMap** on synthetic and real-world SystemC models
- **Bus3Init**

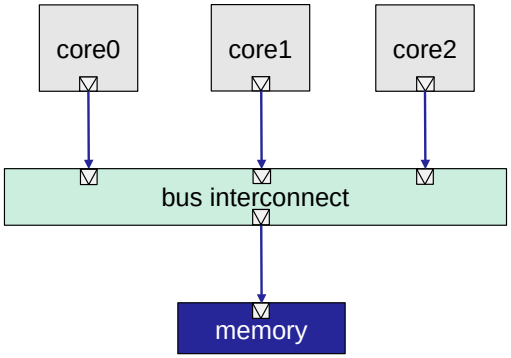


- Simulation trace for **BusyUntil** bus with global quantum value of zero

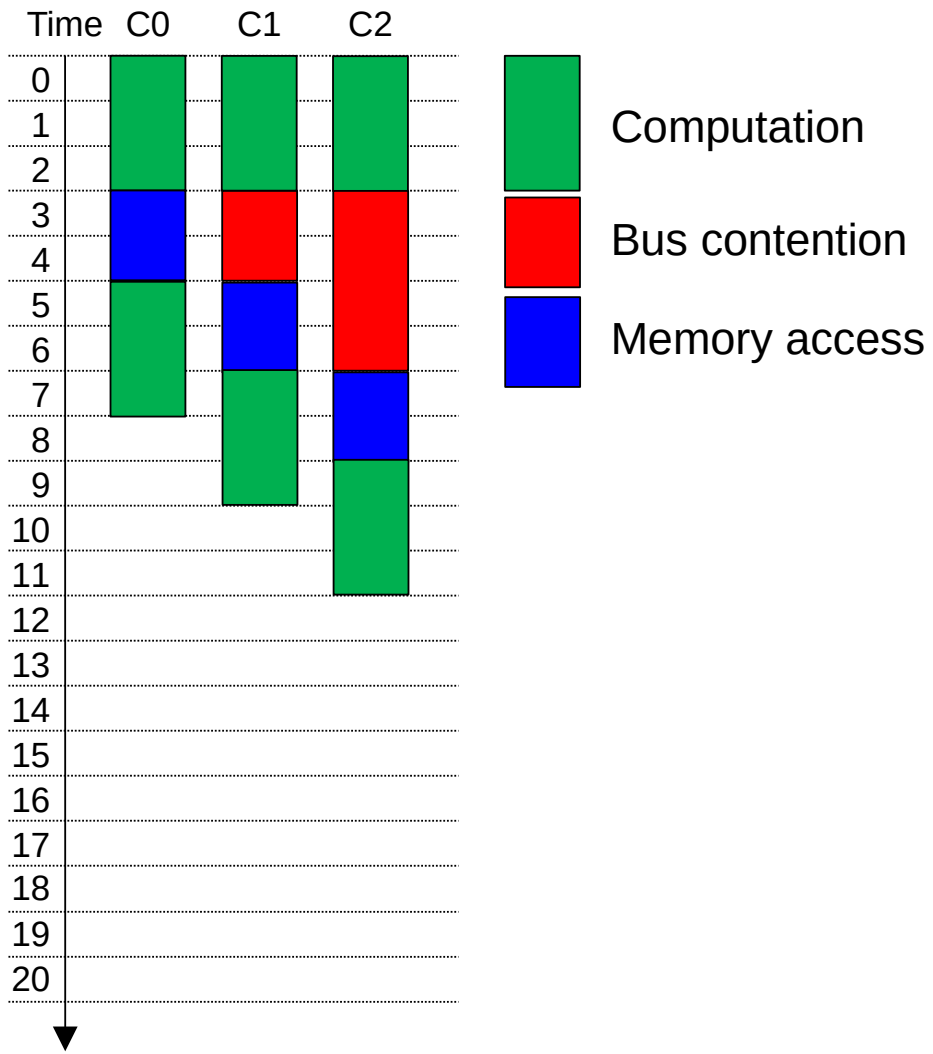


Experimental Measurements and Results (1)

- **BusyUntil** and **BusyMap** on synthetic and real-world SystemC models
- **Bus3Init**

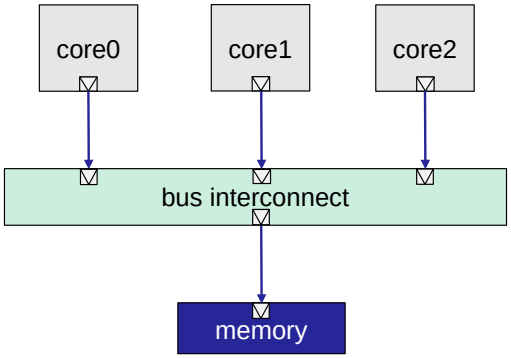


- Simulation trace for **BusyUntil** bus with global quantum value of zero

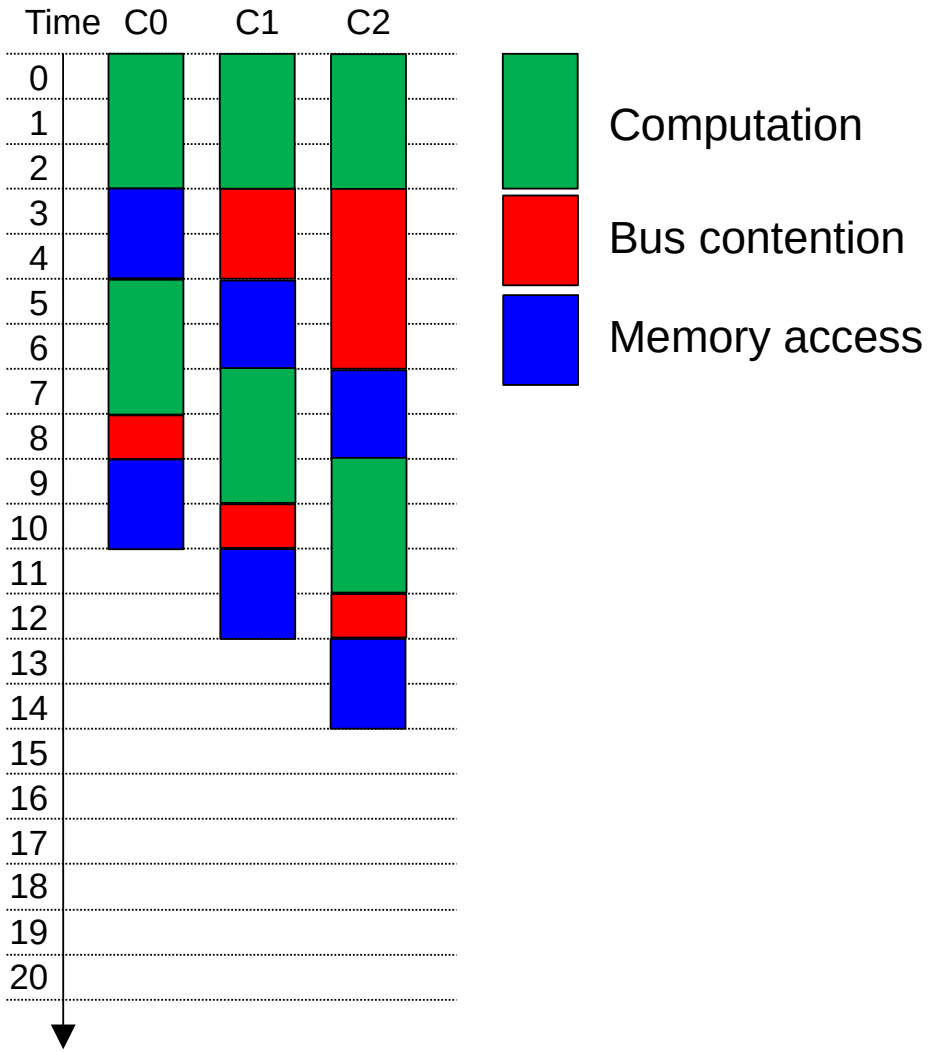


Experimental Measurements and Results (1)

- **BusyUntil** and **BusyMap** on synthetic and real-world SystemC models
- **Bus3Init**

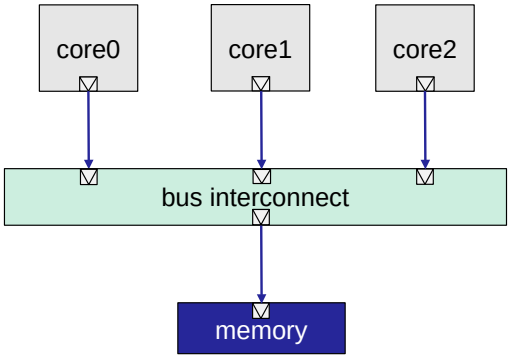


- Simulation trace for **BusyUntil** bus with global quantum value of zero

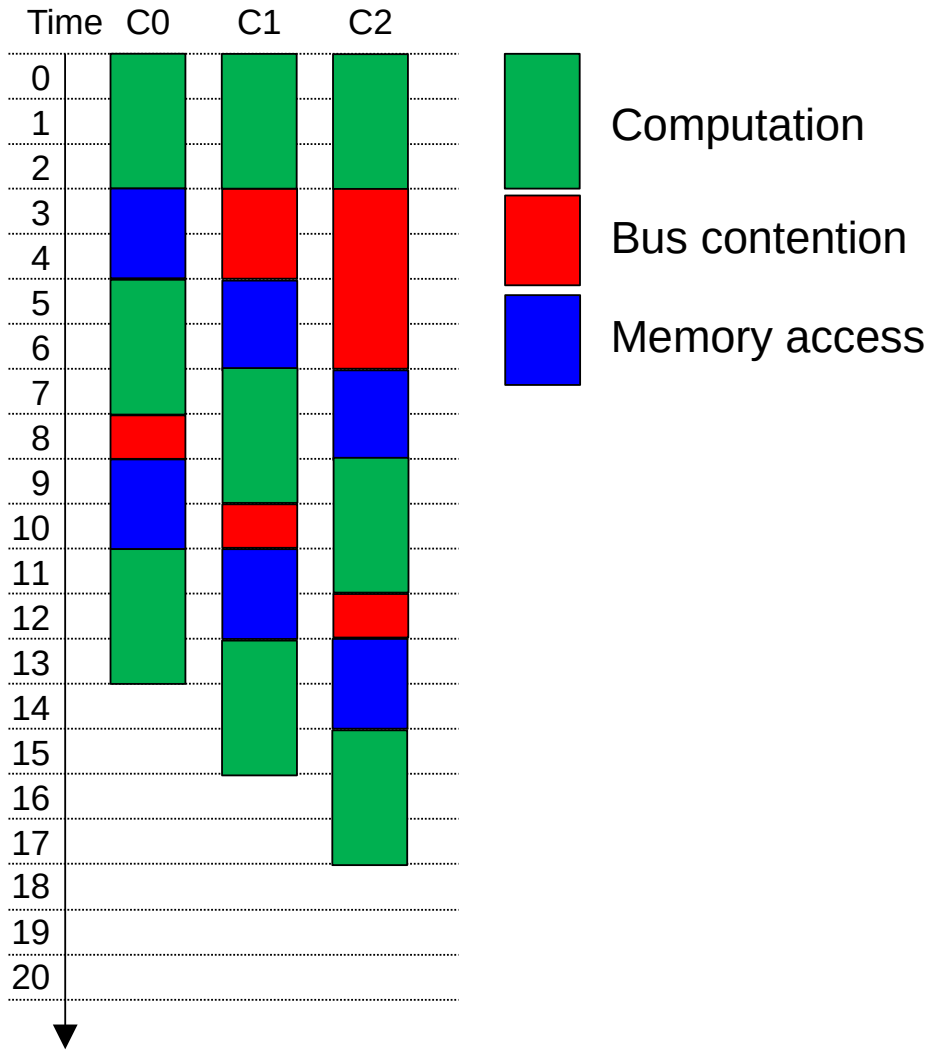


Experimental Measurements and Results (1)

- **BusyUntil** and **BusyMap** on synthetic and real-world SystemC models
- **Bus3Init**

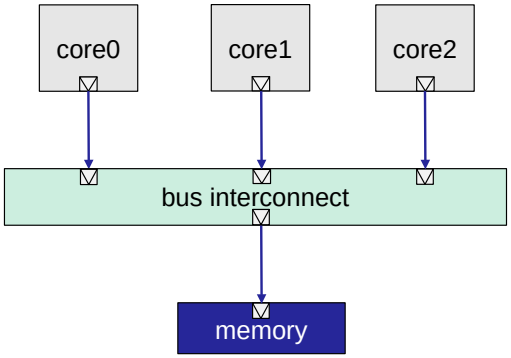


- Simulation trace for **BusyUntil** bus with global quantum value of zero

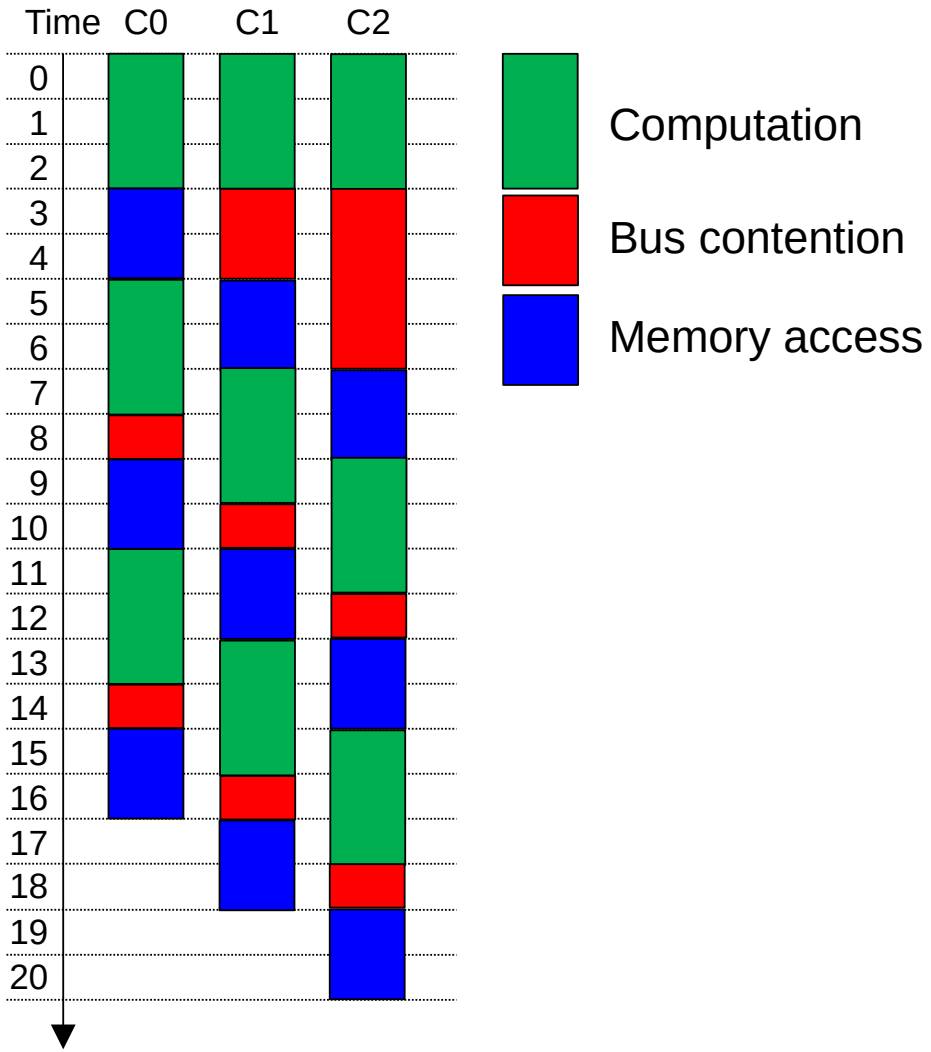


Experimental Measurements and Results (1)

- **BusyUntil** and **BusyMap** on synthetic and real-world SystemC models
- **Bus3Init**



- Simulation trace for **BusyUntil** bus with global quantum value of zero



Experimental Measurements and Results (2)

Experimental Measurements and Results (2)

- To validate BusyMap's temporal decoupling accuracy, we sweep global quantum values from 0 to 13 units

	Simulated time		Bus contention	
Global quantum	BusyUntil	BusyMap	BusyUntil	BusyMap
0	21	21	12	12
1	21	21	12	12
2	21	21	12	12
3	21	21	12	12
4	21	21	12	12
5	21	21	12	12
6	22	20	15	14

	Simulated time		Bus contention	
Global quantum	BusyUntil	BusyMap	BusyUntil	BusyMap
7	22	23	15	14
8	21	25	19	14
9	21	20	19	14
10	27	20	24	14
11	27	22	24	16
12	27	27	24	16
13	29	29	32	16

Experimental Measurements and Results (2)

- To validate BusyMap's temporal decoupling accuracy, we sweep global quantum values from 0 to 13 units

	Simulated time		Bus contention	
Global quantum	BusyUntil	BusyMap	BusyUntil	BusyMap
0	21	21	12	12
1	21	21	12	12
2	21	21	12	12
3	21	21	12	12
4	21	21	12	12
5	21	21	12	12
6	22	20	15	14

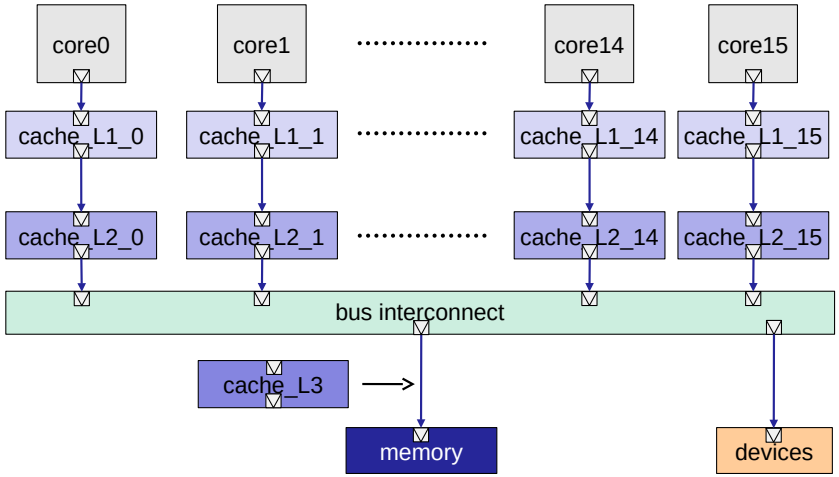
	Simulated time		Bus contention	
Global quantum	BusyUntil	BusyMap	BusyUntil	BusyMap
7	22	23	15	14
8	21	25	19	14
9	21	20	19	14
10	27	20	24	14
11	27	22	24	16
12	27	27	24	16
13	29	29	32	16

- The simulated time and bus contention values show the high accuracy of BusyMap

Experimental Measurements and Results (3)

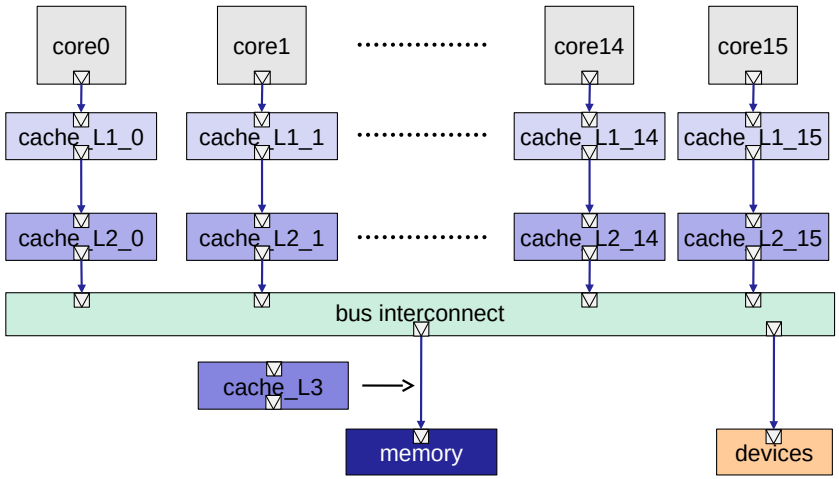
Experimental Measurements and Results (3)

- Parallel JPEG simulation results running on RISC-V SMP VP



Experimental Measurements and Results (3)

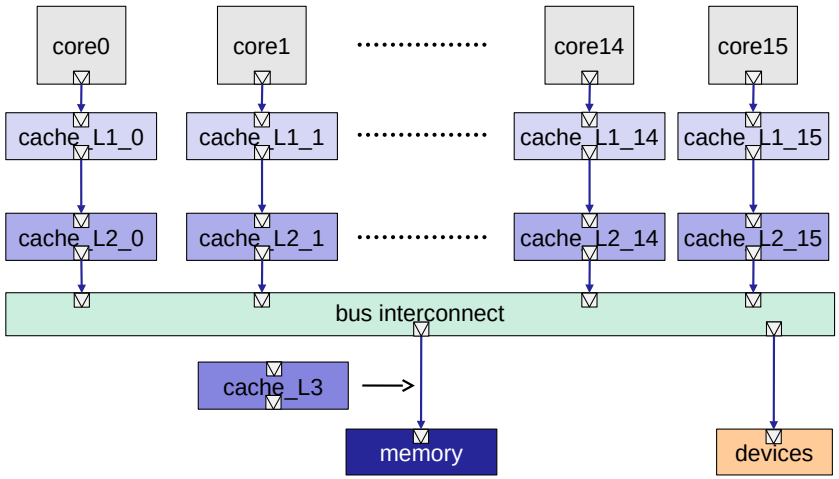
- Parallel JPEG simulation results running on RISC-V SMP VP



	Simulated time		Bus Contention		Simulator run-time		# wait statements (cores)		# wait statements (caches)	
Global quantum	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap
0ns	2.33s	2.33s	3.94s	4.00s	30m49s	18m50s	63574400	63628015	21359780	0
10ns	N/A	2.33s	N/A	4.00s	N/A	18m48s	N/A	63628015	N/A	0
100ns	N/A	2.33s	N/A	3.90s	N/A	13m33s	N/A	22157149	N/A	0
1000ns	N/A	2.44s	N/A	4.71s	N/A	11m38s	N/A	6478006	N/A	0
10000ns	N/A	2.65s	N/A	5.95s	N/A	10m20s	N/A	896267	N/A	0

Experimental Measurements and Results (3)

- Parallel JPEG simulation results running on RISC-V SMP VP

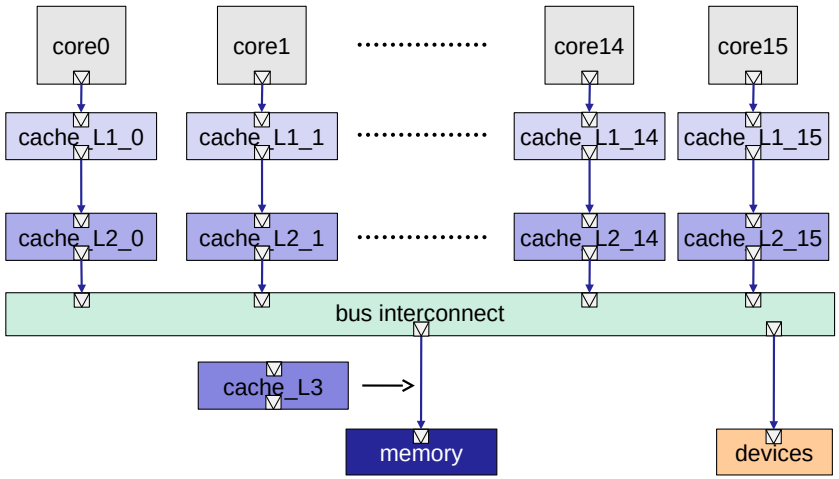


	Simulated time		Bus Contention		Simulator run-time		# wait statements (cores)		# wait statements (caches)	
Global quantum	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap
0ns	2.33s	2.33s	3.94s	4.00s	30m49s	18m50s	63574400	63628015	21359780	0
10ns	N/A	2.33s	N/A	4.00s	N/A	18m48s	N/A	63628015	N/A	0
100ns	N/A	2.33s	N/A	3.90s	N/A	13m33s	N/A	22157149	N/A	0
1000ns	N/A	2.44s	N/A	4.71s	N/A	11m38s	N/A	6478006	N/A	0
10000ns	N/A	2.65s	N/A	5.95s	N/A	10m20s	N/A	896267	N/A	0

- BusyMap significantly improves simulator run-time (3x)

Experimental Measurements and Results (3)

- Parallel JPEG simulation results running on RISC-V SMP VP

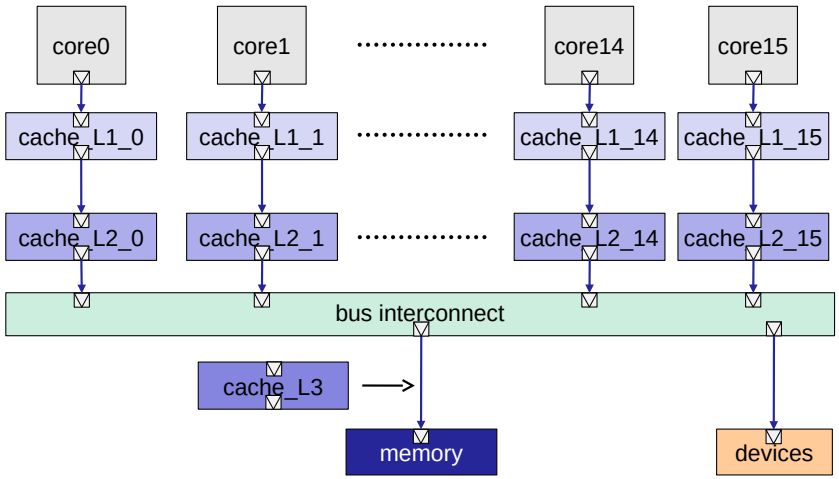


	Simulated time		Bus Contention		Simulator run-time		# wait statements (cores)		# wait statements (caches)	
Global quantum	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap
0ns	2.33s	2.33s	3.94s	4.00s	30m49s	18m50s	63574400	63628015	21359780	0
10ns	N/A	2.33s	N/A	4.00s	N/A	18m48s	N/A	63628015	N/A	0
100ns	N/A	2.33s	N/A	3.90s	N/A	13m33s	N/A	22157149	N/A	0
1000ns	N/A	2.44s	N/A	4.71s	N/A	11m38s	N/A	6478006	N/A	0
10000ns	N/A	2.65s	N/A	5.95s	N/A	10m20s	N/A	896267	N/A	0

- BusyMap significantly improves simulator run-time (3x)
- BusyMap supports temporal decoupling

Experimental Measurements and Results (3)

- Parallel JPEG simulation results running on RISC-V SMP VP



	Simulated time		Bus Contention		Simulator run-time		# wait statements (cores)		# wait statements (caches)	
Global quantum	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap	BusyUntil	BusyMap
0ns	2.33s	2.33s	3.94s	4.00s	30m49s	18m50s	63574400	63628015	21359780	0
10ns	N/A	2.33s	N/A	4.00s	N/A	18m48s	N/A	63628015	N/A	0
100ns	N/A	2.33s	N/A	3.90s	N/A	13m33s	N/A	22157149	N/A	0
1000ns	N/A	2.44s	N/A	4.71s	N/A	11m38s	N/A	6478006	N/A	0
10000ns	N/A	2.65s	N/A	5.95s	N/A	10m20s	N/A	896267	N/A	0

- BusyMap significantly improves simulator run-time (3x)
- BusyMap supports temporal decoupling
- BusyMap has high accuracy in simulated time and contention

Conclusion & Future Work

Conclusion & Future Work

- This work improves high-level modeling and simulation of interconnect contention by navigating trade-offs between simulation speed and timing accuracy

Conclusion & Future Work

- This work improves high-level modeling and simulation of interconnect contention by navigating trade-offs between simulation speed and timing accuracy
- Specifically, we proposed BusyMap for modeling bus contention in SystemC TLM-2.0 LT-CA models
  - Supports temporal decoupled with *out-of-order* transactions
  - Can effectively model multi-level interconnects and caches

Conclusion & Future Work

- This work improves high-level modeling and simulation of interconnect contention by navigating trade-offs between simulation speed and timing accuracy
- Specifically, we proposed BusyMap for modeling bus contention in SystemC TLM-2.0 LT-CA models
  - Supports temporal decoupled with *out-of-order* transactions
  - Can effectively model multi-level interconnects and caches
- We achieve a speedup of up to 3x on a 16-core host with only 10% accuracy loss in simulated time and bus contention

Conclusion & Future Work

- This work improves high-level modeling and simulation of interconnect contention by navigating trade-offs between simulation speed and timing accuracy
- Specifically, we proposed BusyMap for modeling bus contention in SystemC TLM-2.0 LT-CA models
  - Supports temporal decoupled with *out-of-order* transactions
  - Can effectively model multi-level interconnects and caches
- We achieve a speedup of up to 3x on a 16-core host with only 10% accuracy loss in simulated time and bus contention
- Our future work includes the evaluation of BusyMap on more industry-strength applications

References (1)

- [DVCon'23b] C. Raccomandato, E. Arasteh, R. Dömer, *"Grid-based Mapping and Analysis of a GoogLeNet CNN using MapGL Editor"*, Proceedings (engineering track) of the Design and Verification Conference in Europe, Munich, Germany, November 2023.
- [DVCon'23a] C. Raccomandato, E. Arasteh, R. Dömer, *"MapGL: Interactive Application Mapping and Profiling on a Grid of Processing Cells"*, Proceedings (research track) of the Design and Verification Conference in Europe, Munich, Germany, November 2023.
- [TECS'23] E. Arasteh, R. Dömer: *"Fast Loosely-Timed System Models with Accurate Memory Contention"*, Journal of ACM Transactions on Embedded Computing Systems, July 2023.
- [FSE'23] N. Farzan, E. Arasteh, *"Visualizing Transaction-Level Modeling Simulations of Deep Neural Networks"*, Fowler School of Engineering, Technical Report 23-01, August 2023.
- [IESS'22] V. Govindasamy, E. Arasteh, R. Dömer: *"Minimizing Memory Contention in an APNG Encoder using a Grid of Processing Cells"*, Proceedings of the International Embedded Systems Symposium, "Designing Modern Embedded Systems: Software, Hardware, and Applications" Springer, Lippstadt, Germany, November 2022.
- [FDL'21] E. Arasteh, R. Dömer: *"Improving Parallelism in System Level Models by Assessing PDES Performance"*, Proceedings of Forum on Specification and Design Languages, Antibes, France, Sep. 2021.
- [CECS'21] E. Arasteh, R. Dömer: *"Systematic Evaluation of Six Models of GoogLeNet using PDES"*, Center for Embedded and Cyber-Physical Systems, Technical Report 21-03, 2021, Sep. 2021.

References (2)

- [Springer'20] R. Dömer, Z. Cheng, D. Mendoza, E. Arasteh: *"Pushing the Limits of Parallel Discrete Event Simulation for SystemC"*, in "A Journey of Embedded and Cyber-Physical Systems" by Jian-Jia Chen, Springer Nature, Switzerland, August 2020.
- [DATE'20] D. Mendoza, Z. Cheng, E. Arasteh, R. Dömer: *"Lazy Event Prediction using Defining Trees and Schedule Bypass for Out-of-Order PDES"*, Design, Automation and Test in Europe Conference, Grenoble, France, March 2020.
- [ASPDAC'20] Z. Cheng, E. Arasteh, R. Dömer: *"Event Delivery using Prediction for Faster Parallel SystemC Simulation"*, Asia and South Pacific Design Automation Conference, Beijing, China, Jan. 2020.
- [IESS'19] E. Arasteh, R. Dömer: *"An Untimed SystemC Model of GoogLeNet"*, Proceedings of the International Embedded Systems Symposium, Friedrichshafen, Germany, Sep. 2019.