Cross Clock-Domain TDM Virtual Circuits for Networks on Chips

Zhonghai Lu

Dept. of Electronic Systems
School for Information and Communication Technology
KTH - Royal Institute of Technology, Stockholm

May 4, 2011

Outline: The PARC structure

1. Problem
2. Approach
3. Result
4. Conclusion

Outline: The PARC structure

1. Problem
2. Approach
3. Result
4. Conclusion

Outline: The PARC structure

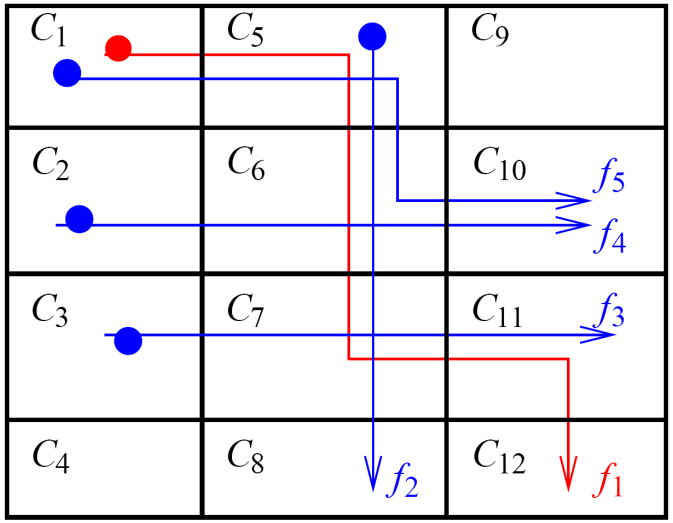
1. Problem
2. Approach
3. Result
4. Conclusion

Outline: The PARC structure

1. Problem
2. Approach
3. Result
4. Conclusion

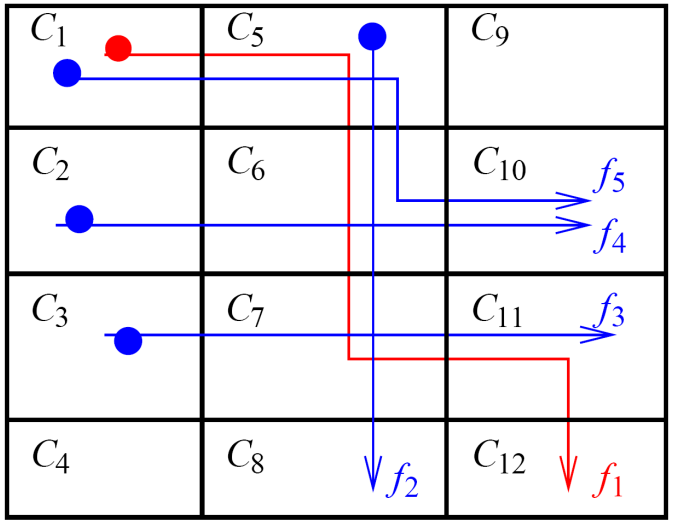
Quality-of-Service (QoS): A key challenge for NoCs

- QoS communication: Provide performance (delay and throughput) guarantees under worst-case conditions.
- Complications
  - Non-deterministic network contention for shared buffers, crossbars, and links.
  - An advanced SoC typically encompasses many clock domains to manage the clocking complexity and increase power efficiency.



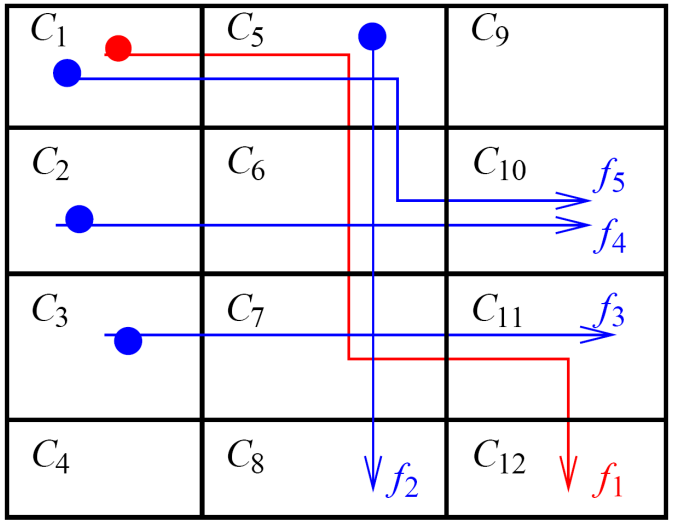
Quality-of-Service (QoS): A key challenge for NoCs

- QoS communication: Provide performance (delay and throughput) guarantees under worst-case conditions.
- Complications
  - Non-deterministic network contention for shared buffers, crossbars, and links.
  - An advanced SoC typically encompasses many clock domains to manage the clocking complexity and increase power efficiency.



Quality-of-Service (QoS): A key challenge for NoCs

- QoS communication: Provide performance (delay and throughput) guarantees under worst-case conditions.
- Complications
  - Non-deterministic network contention for shared buffers, crossbars, and links.
  - An advanced SoC typically encompasses many clock domains to manage the clocking complexity and increase power efficiency.



Related Work

- A. Hansson, M. Subburaman, and K. Goossens.
  Aelite: A flit-synchronous network on chip with composable and predictable services.
  In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'09)*, April 2009.
- Z. Lu and A. Jantsch.
  TDM virtual-circuit configuration for network-on-chip.
  IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 16(8):1021–1034, 2008.
- S. Evain, J.-P. Diguet, and D. Houzet.
  NoC design flow for TDMA and QoS management in a GALS context.
  EURASIP Journal on Embedded Systems, pages 1–12, 2006.

Divide-and-Conquer

- Divide: Partition the chip into multiple communicating synchronous regions.
  - GALS communication style
  - A general FIFO domain interface
- Conquer: Design reuse plus formal performance analysis
  - For each synchronous region, reuse (synchronous) time-division-multiplexing (TDM) virtual circuits (VC);
  - Address cross-domain issues to ensure correct operation;
  - Determine cross-domain delay and backlog bounds.

Divide-and-Conquer

- Divide: Partition the chip into multiple communicating synchronous regions.
  - GALS communication style
  - A general FIFO domain interface
- Conquer: Design reuse plus formal performance analysis
  - For each synchronous region, reuse (synchronous) time-division-multiplexing (TDM) virtual circuits (VC);
  - Address cross-domain issues to ensure correct operation;
  - Determine cross-domain delay and backlog bounds.

Divide-and-Conquer

- Divide: Partition the chip into multiple communicating synchronous regions.
  - GALS communication style
  - A general FIFO domain interface
- Conquer: Design reuse plus formal performance analysis
  - For each synchronous region, reuse (synchronous) time-division-multiplexing (TDM) virtual circuits (VC);
  - Address cross-domain issues to ensure correct operation;
  - Determine cross-domain delay and backlog bounds.

Divide-and-Conquer

- Divide: Partition the chip into multiple communicating synchronous regions.
  - GALS communication style
  - A general FIFO domain interface
- Conquer: Design reuse plus formal performance analysis
  - For each synchronous region, reuse (synchronous) time-division-multiplexing (TDM) virtual circuits (VC);
  - Address cross-domain issues to ensure correct operation;
  - Determine cross-domain delay and backlog bounds.

Divide-and-Conquer

- Divide: Partition the chip into multiple communicating synchronous regions.
  - GALS communication style
  - A general FIFO domain interface
- Conquer: Design reuse plus formal performance analysis
  - For each synchronous region, reuse (synchronous) time-division-multiplexing (TDM) virtual circuits (VC);
  - Address cross-domain issues to ensure correct operation;
  - Determine cross-domain delay and backlog bounds.

Divide-and-Conquer

- Divide: Partition the chip into multiple communicating synchronous regions.
  - GALS communication style
  - A general FIFO domain interface
- Conquer: Design reuse plus formal performance analysis
  - For each synchronous region, reuse (synchronous) time-division-multiplexing (TDM) virtual circuits (VC);
  - Address cross-domain issues to ensure correct operation;
  - Determine cross-domain delay and backlog bounds.

Divide-and-Conquer

- Divide: Partition the chip into multiple communicating synchronous regions.
  - GALS communication style
  - A general FIFO domain interface
- Conquer: Design reuse plus formal performance analysis
  - For each synchronous region, reuse (synchronous) time-division-multiplexing (TDM) virtual circuits (VC);
  - Address cross-domain issues to ensure correct operation;
  - Determine cross-domain delay and backlog bounds.

TDM VC

- A VC is an end-to-end connection which reserves *cyclically* S out of N time slots to share each passing link bandwidth.

$$D_{max} = D_{init} + D_{trans} = ((N - S) + H) \cdot T \quad (1)$$

$$\Theta = \frac{S}{N} F \quad (2)$$

H: hop count from source to destination; F, T: clock frequency, period.

- Based on the same notion of time, globally coordinating slot allocation to share link bandwidth in a *time-division* fashion.
- Flow on a VC enjoys pipelined, in-order, contention-free delivery, like passing an established *circuit*.

TDM VC

- A VC is an end-to-end connection which reserves *cyclically* S out of N time slots to share each passing link bandwidth.

$$D_{max} = D_{init} + D_{trans} = ((N - S) + H) \cdot T \quad (1)$$

$$\Theta = \frac{S}{N} F \quad (2)$$

H: hop count from source to destination; F, T: clock frequency, period.

- Based on the same notion of time, globally coordinating slot allocation to share link bandwidth in a *time-division* fashion.
- Flow on a VC enjoys pipelined, in-order, contention-free delivery, like passing an established *circuit*.

TDM VC

- A VC is an end-to-end connection which reserves *cyclically* S out of N time slots to share each passing link bandwidth.

$$D_{max} = D_{init} + D_{trans} = ((N - S) + H) \cdot T \quad (1)$$

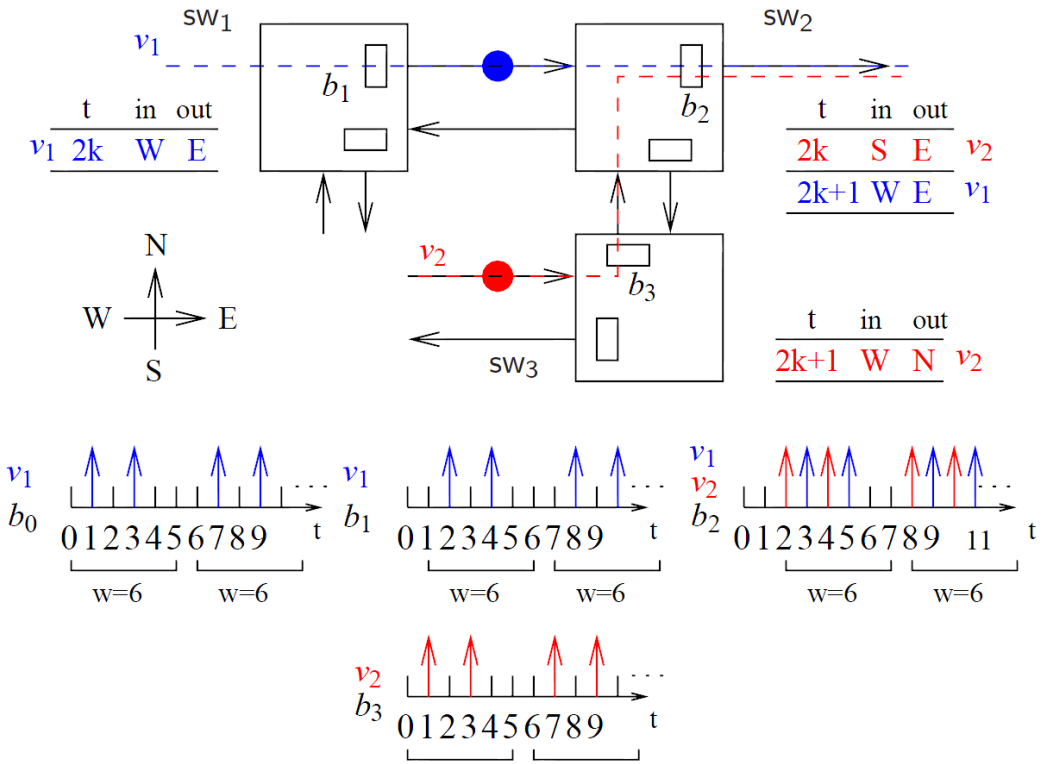
$$\Theta = \frac{S}{N} F \quad (2)$$

H: hop count from source to destination; F, T: clock frequency, period.

- Based on the same notion of time, globally coordinating slot allocation to share link bandwidth in a *time-division* fashion.
- Flow on a VC enjoys pipelined, in-order, contention-free delivery, like passing an established *circuit*.

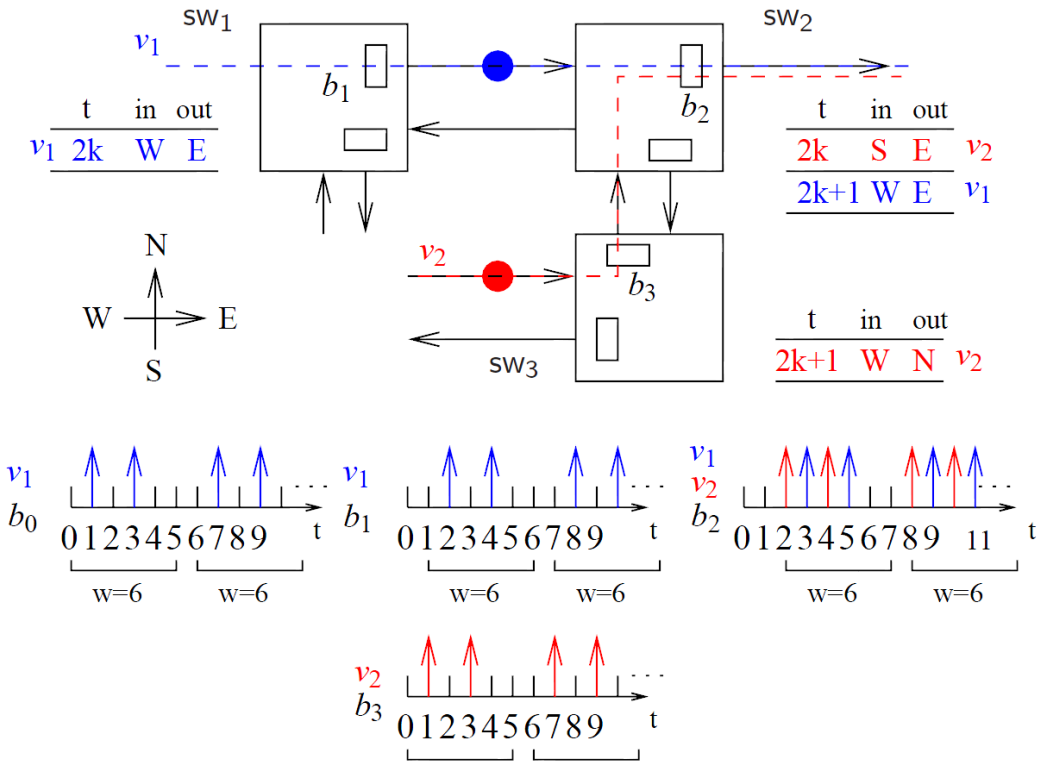
Property and Example

- TDM VC properties
  - Hard guarantees in delay and throughput
  - Low cost: one buffer per hop
  - Flow composition over shared links
- An example



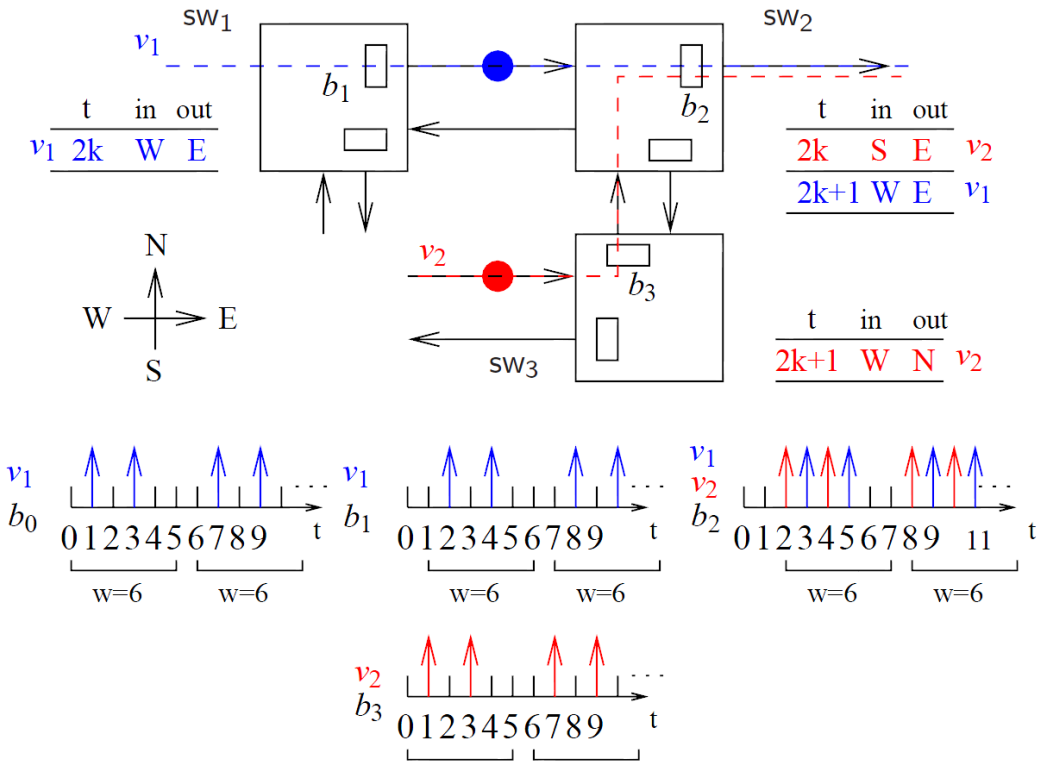
Property and Example

- TDM VC properties
  - Hard guarantees in delay and throughput
  - Low cost: one buffer per hop
  - Flow composition over shared links
- An example



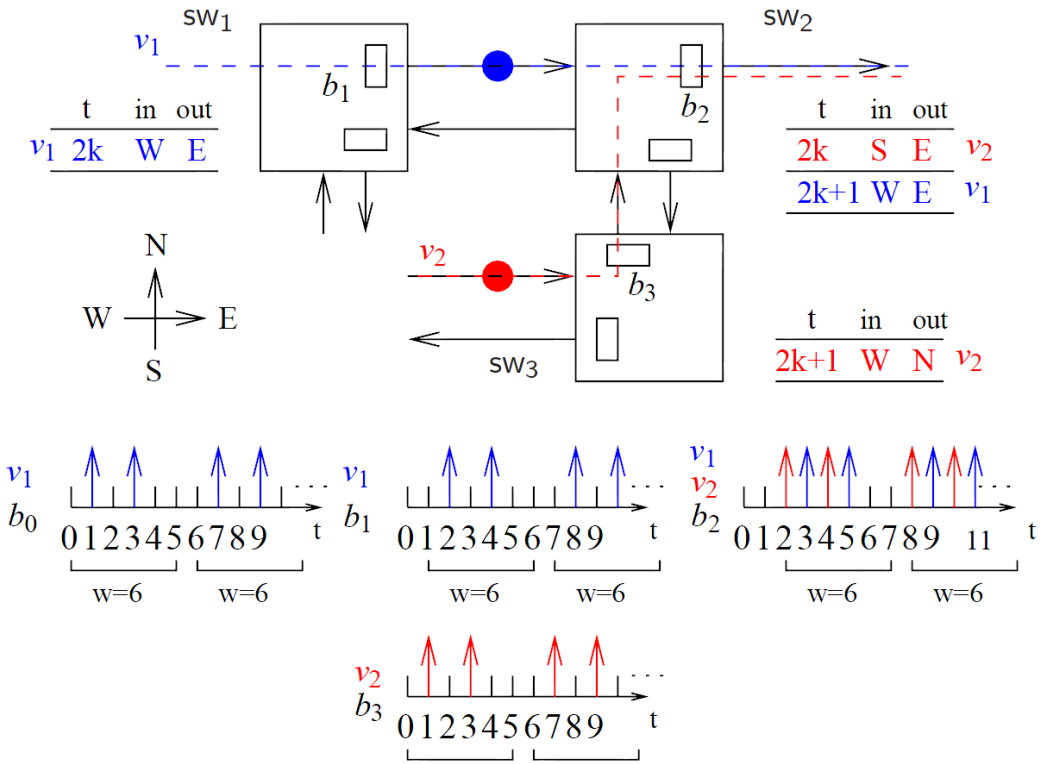
Property and Example

- TDM VC properties
  - Hard guarantees in delay and throughput
  - Low cost: one buffer per hop
  - Flow composition over shared links
- An example



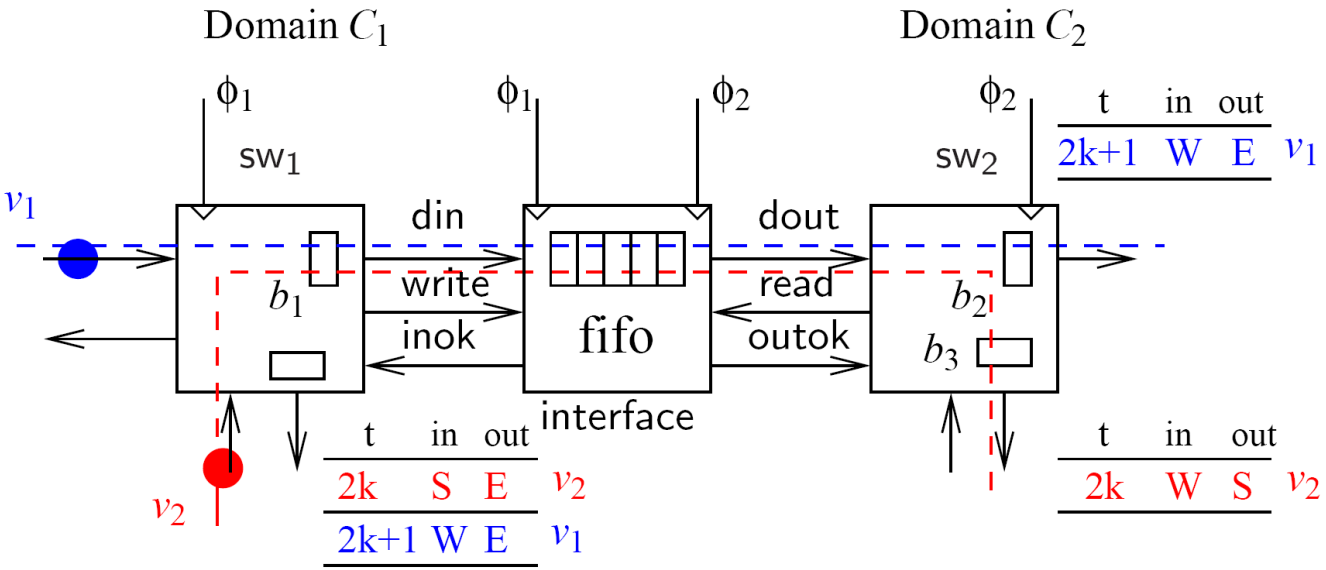
Property and Example

- TDM VC properties
  - Hard guarantees in delay and throughput
  - Low cost: one buffer per hop
  - Flow composition over shared links
- An example



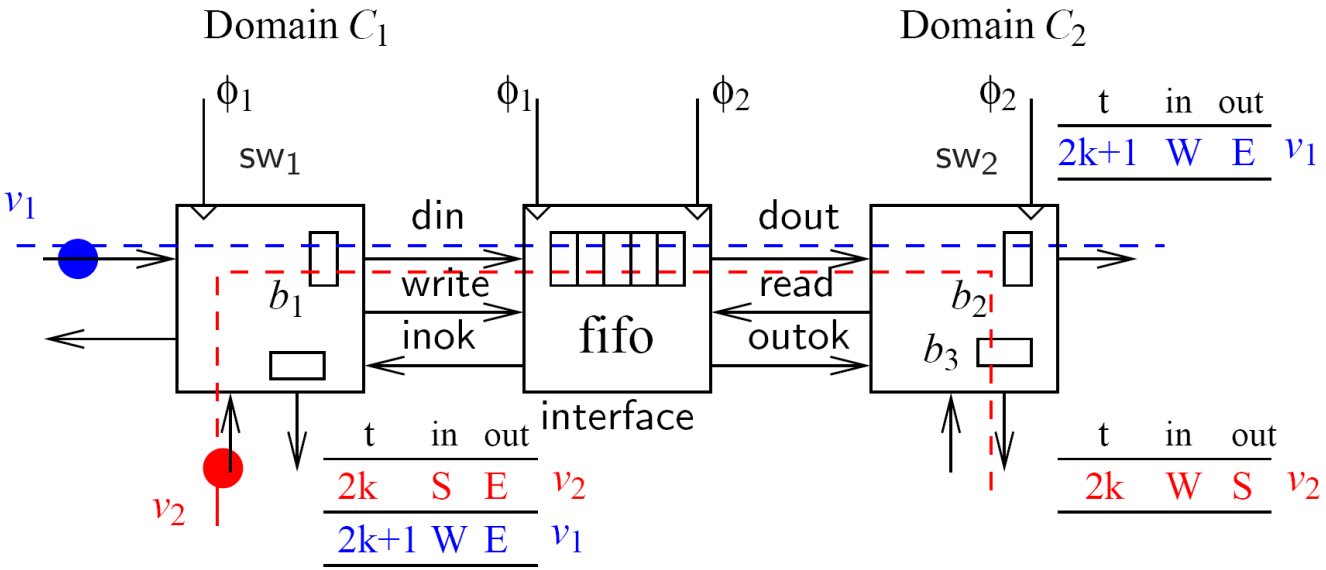
Clock domain interface

- One FIFO, two distinct clocks with fixed-relation frequency
  - One writer, one reader
  - Handshaking protocol.
- A composite flow may split after crossing the FIFO interface.



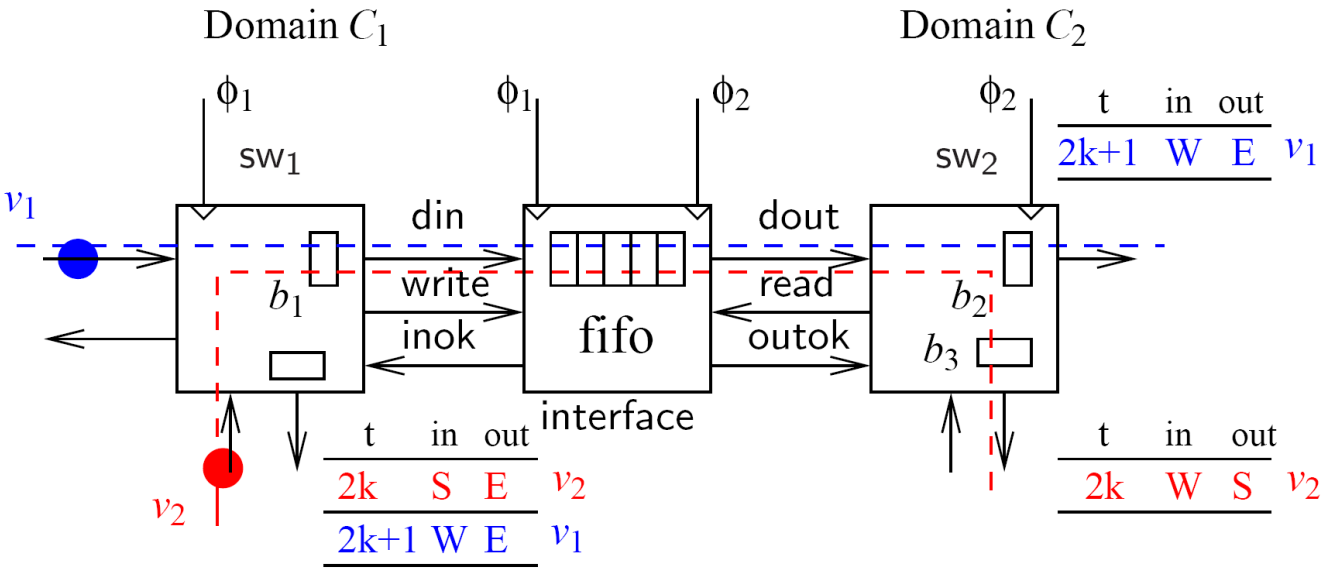
Clock domain interface

- One FIFO, two distinct clocks with fixed-relation frequency
  - One writer, one reader
  - Handshaking protocol.
- A composite flow may split after crossing the FIFO interface.



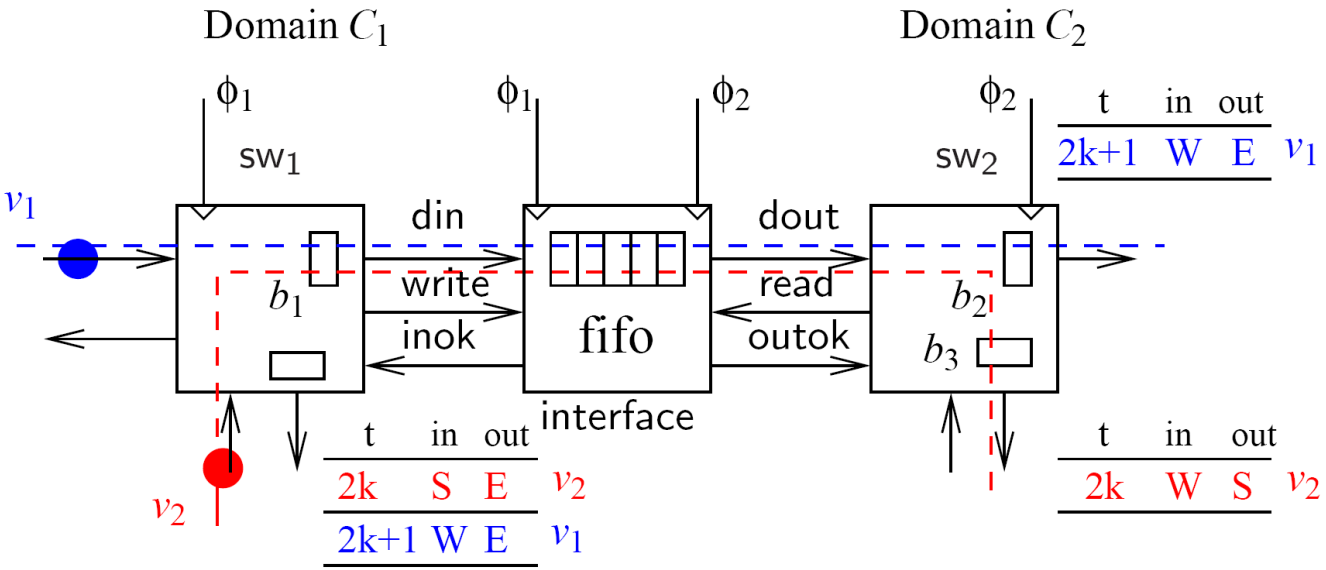
Clock domain interface

- One FIFO, two distinct clocks with fixed-relation frequency
  - One writer, one reader
  - Handshaking protocol.
- A composite flow may split after crossing the FIFO interface.



Clock domain interface

- One FIFO, two distinct clocks with fixed-relation frequency
  - One writer, one reader
  - Handshaking protocol.
- A composite flow may split after crossing the FIFO interface.



Cross domain issues

- Synchronous TDM VC assumes global synchrony, which does not exist any more with multiple clock domains.
- Correct operation
  - Sufficient bandwidth reservation
  - Read-write synchronization
  - Event preservation
- Performance analysis
  - Per-flow cross-domain delay bound
  - Cross-domain backlog bound (FIFO size)

Cross domain issues

- Synchronous TDM VC assumes global synchrony, which does not exist any more with multiple clock domains.
- Correct operation
  - Sufficient bandwidth reservation
  - Read-write synchronization
  - Event preservation
- Performance analysis
  - Per-flow cross-domain delay bound
  - Cross-domain backlog bound (FIFO size)

Cross domain issues

- Synchronous TDM VC assumes global synchrony, which does not exist any more with multiple clock domains.
- Correct operation
  - Sufficient bandwidth reservation
  - Read-write synchronization
  - Event preservation
- Performance analysis
  - Per-flow cross-domain delay bound
  - Cross-domain backlog bound (FIFO size)

Cross domain issues

- Synchronous TDM VC assumes global synchrony, which does not exist any more with multiple clock domains.
- Correct operation
  - Sufficient bandwidth reservation
  - Read-write synchronization
  - Event preservation
- Performance analysis
  - Per-flow cross-domain delay bound
  - Cross-domain backlog bound (FIFO size)

Cross domain issues

- Synchronous TDM VC assumes global synchrony, which does not exist any more with multiple clock domains.
- Correct operation
  - Sufficient bandwidth reservation
  - Read-write synchronization
  - Event preservation
- Performance analysis
  - Per-flow cross-domain delay bound
  - Cross-domain backlog bound (FIFO size)

Cross domain issues

- Synchronous TDM VC assumes global synchrony, which does not exist any more with multiple clock domains.
- Correct operation
  - Sufficient bandwidth reservation
  - Read-write synchronization
  - Event preservation
- Performance analysis
  - Per-flow cross-domain delay bound
  - Cross-domain backlog bound (FIFO size)

Cross domain issues

- Synchronous TDM VC assumes global synchrony, which does not exist any more with multiple clock domains.
- Correct operation
  - Sufficient bandwidth reservation
  - Read-write synchronization
  - Event preservation
- Performance analysis
  - Per-flow cross-domain delay bound
  - Cross-domain backlog bound (FIFO size)

Bandwidth reservation

- Non-decreasing bandwidth reservation

$$r \leq \frac{S_i}{N_i} F_i \leq \frac{S_j}{N_j} F_j \leq \cdots \leq \frac{S_k}{N_k} F_k \tag{3}$$

where r: flow rate; $\{S_i \leq N_i,\ S_j \leq N_j,\ \cdots S_k \leq N_k\}$.

- It will be *insufficient* if it is changed to $r \leq \frac{S_i}{N_i} F_i$, $r \leq \frac{S_j}{N_j} F_j$, $\cdots$ $r \leq \frac{S_k}{N_k} F_k$, as these inequalities can not ensure the preservation of all upstream events at the domain interface.

Bandwidth reservation

- Non-decreasing bandwidth reservation

$$r \leq \frac{S_i}{N_i} F_i \leq \frac{S_j}{N_j} F_j \leq \cdots \leq \frac{S_k}{N_k} F_k \tag{3}$$

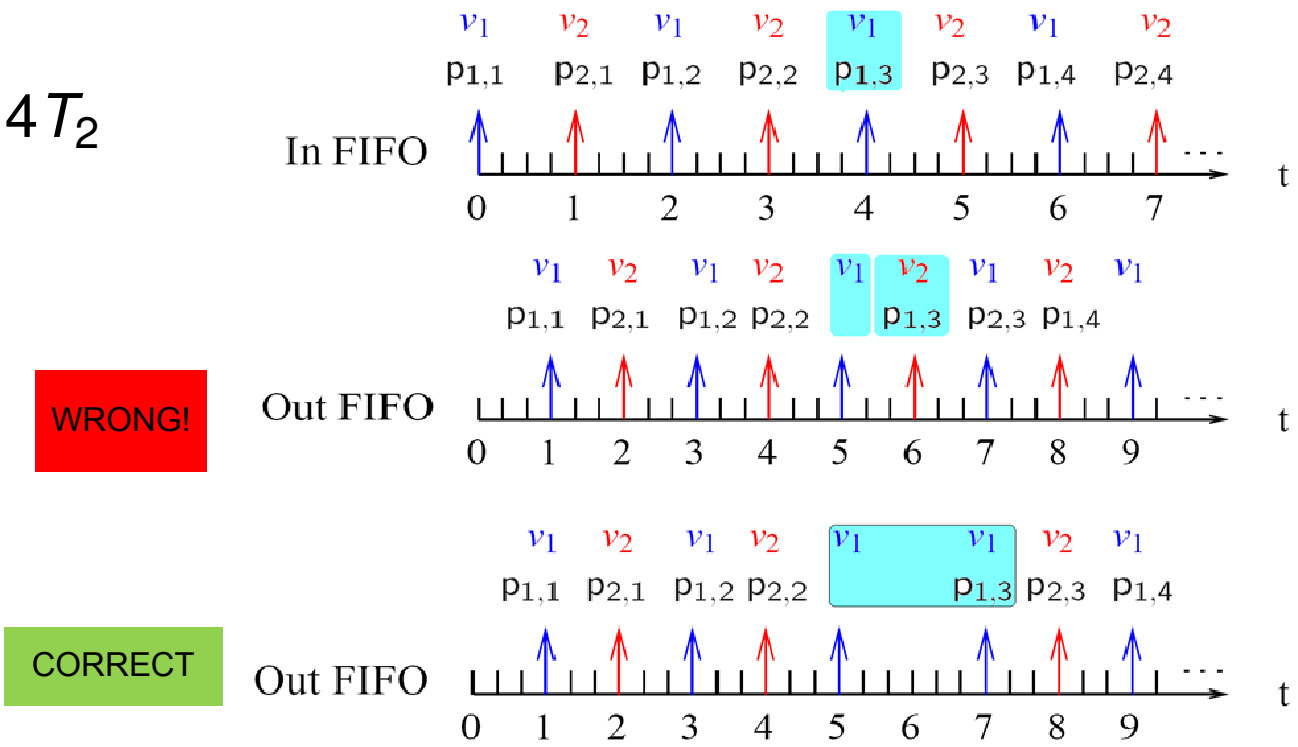
where r: flow rate; $\{S_i \leq N_i,\ S_j \leq N_j,\ \cdots S_k \leq N_k\}$.

- It will be *insufficient* if it is changed to $r \leq \frac{S_i}{N_i} F_i$, $r \leq \frac{S_j}{N_j} F_j$, $\cdots$ $r \leq \frac{S_k}{N_k} F_k$, as these inequalities can not ensure the preservation of all upstream events at the domain interface.

Read-write synchronization

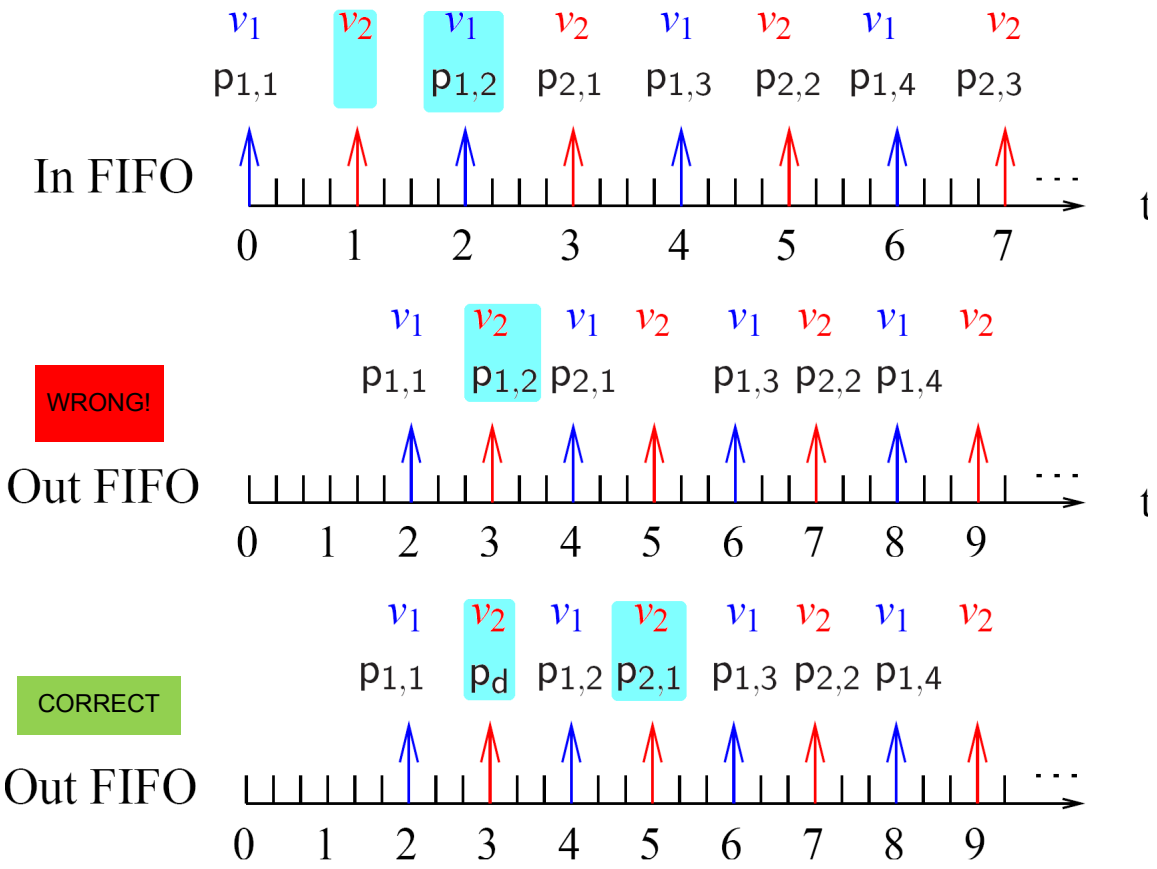
- Use blocking-read to align timing mismatch.

$$3T_1 = 4T_2$$



Event preservation

- Need to transmit all events to preserve timing information crossing the domain boundary.
- A "dummy" packet, P_d, may be used.



Network calculus

- Network Calculus is a mathematical framework to derive per-flow worst-case delay and backlog bounds in communication networks.
  - An *arrival curve* bounds the accumulated amount of traffic of a flow.
  - A *service curve* describes the minimal service level of a network element.
- References

R. L. Cruz.
A calculus for network delay, part I: Network elements in isolation and part II: Network analysis.
IEEE Transactions on Information Theory, 37(1), January 1991.

C.-S. Chang.
Performance Guarantees in Communication Networks.
Springer Verlag, 2000.

J. Y. L. Boudec and P. Thiran.
Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.
Number 2050 in LNCS, Springer Verlag, 2004.

Network calculus

- Network Calculus is a mathematical framework to derive per-flow worst-case delay and backlog bounds in communication networks.
  - An *arrival curve* bounds the accumulated amount of traffic of a flow.
  - A *service curve* describes the minimal service level of a network element.
- References

R. L. Cruz.
A calculus for network delay, part I: Network elements in isolation and part II: Network analysis.
IEEE Transactions on Information Theory, 37(1), January 1991.

C.-S. Chang.
Performance Guarantees in Communication Networks.
Springer Verlag, 2000.

J. Y. L. Boudec and P. Thiran.
Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.
Number 2050 in LNCS, Springer Verlag, 2004.

Network calculus

- Network Calculus is a mathematical framework to derive per-flow worst-case delay and backlog bounds in communication networks.
  - An *arrival curve* bounds the accumulated amount of traffic of a flow.
  - A *service curve* describes the minimal service level of a network element.
- References

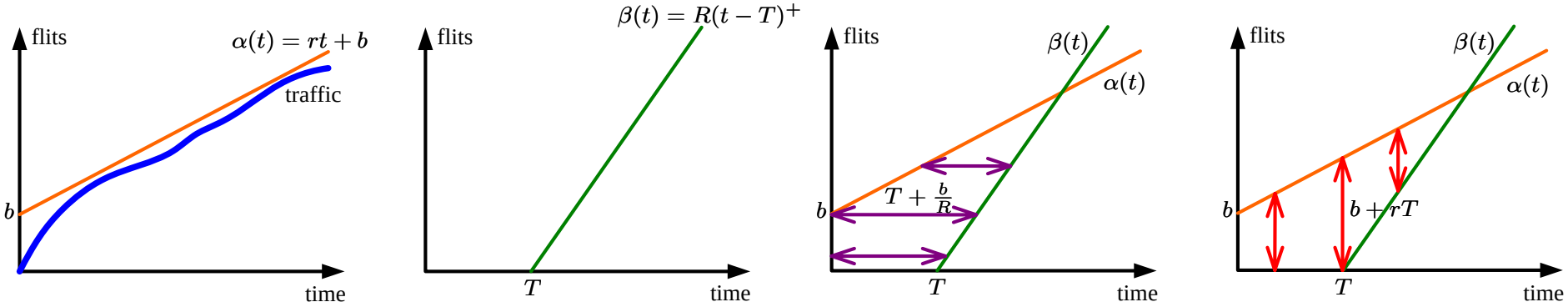
R. L. Cruz.
A calculus for network delay, part I: Network elements in isolation and part II: Network analysis.
IEEE Transactions on Information Theory, 37(1), January 1991.

C.-S. Chang.
Performance Guarantees in Communication Networks.
Springer Verlag, 2000.

J. Y. L. Boudec and P. Thiran.
Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.
Number 2050 in LNCS, Springer Verlag, 2004.

Linear arrival curve, Latency-rate service curve

- Linear arrival curve: $\alpha_{r,b}(t) = rt + b$, where r bounds the rate and b limits the burstiness of the flow.
- Latency-rate server: $\beta_{R,T}(t) = R(t - T)^+$, where R is the service rate and T the maximum processing delay.

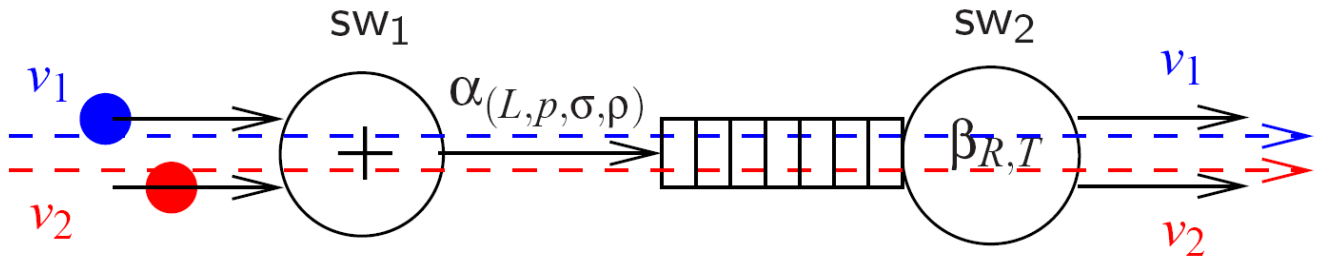


(a) Arrival Curve (b) Service Curve (c) Delay Bound (d) Backlog Bound

Figure 1: Performance bounds for linear arrival curve and latency-rate service curve.

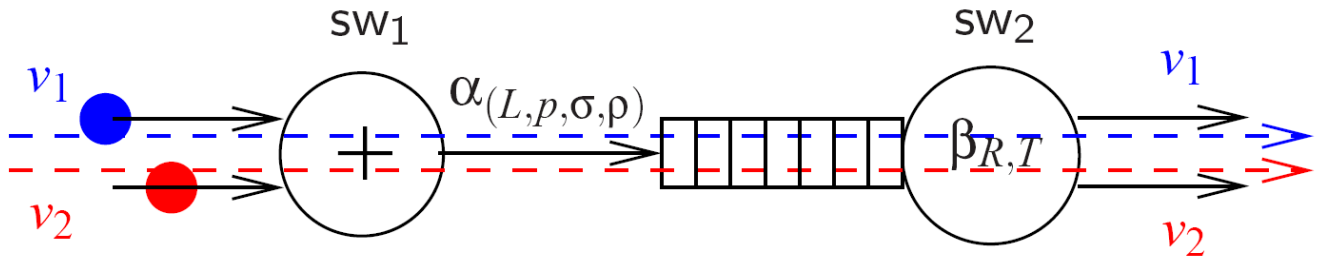
The analytical model

- Model the composite flow with a linear arrival curve, TSPEC (Traffic SPECification), $\alpha_{(L,p,\sigma,\rho)}(t) = min(L + pt, \sigma + \rho t)$, characterizing both peak and average traffic behavior.
- Model the service (reading data out) as a latency-rate server.
- Apply basic analytic results for delay and buffer bounds.



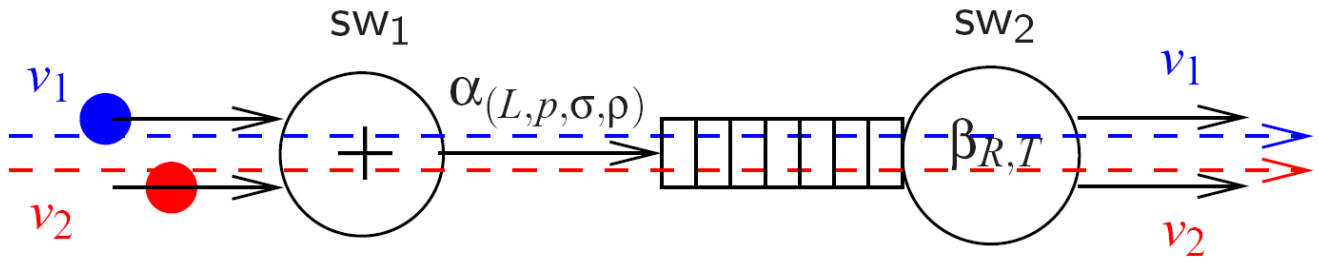
The analytical model

- Model the composite flow with a linear arrival curve, TSPEC (Traffic SPECification), $\alpha_{(L,p,\sigma,\rho)}(t) = min(L + pt, \sigma + \rho t)$, characterizing both peak and average traffic behavior.
- Model the service (reading data out) as a latency-rate server.
- Apply basic analytic results for delay and buffer bounds.



The analytical model

- Model the composite flow with a linear arrival curve, TSPEC (Traffic SPECification), $\alpha_{(L,p,\sigma,\rho)}(t) = min(L + pt, \sigma + \rho t)$, characterizing both peak and average traffic behavior.
- Model the service (reading data out) as a latency-rate server.
- Apply basic analytic results for delay and buffer bounds.



The arrival and service curves

- Determine parameters for TSPEC parameters, L, p, σ, ρ.

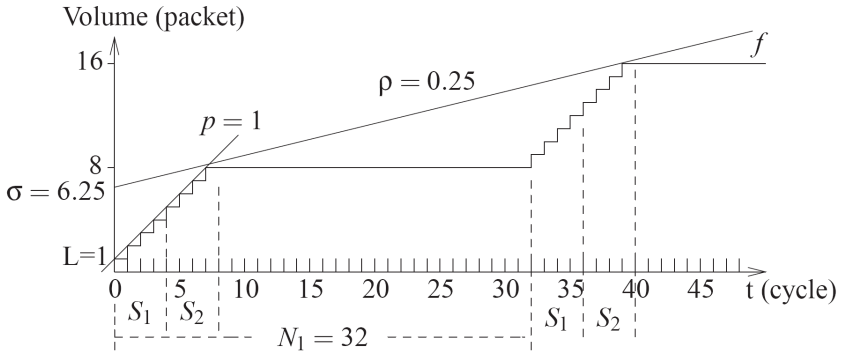
$$\sigma = (N_j - \sum_{i=1}^{n} S_i + 1) \sum_{i=1}^{n} S_i / N_j \tag{4}$$

$$\rho = F_j \sum_{i=1}^{n} S_i / N_j \tag{5}$$

- Determine parameters for R (rate), and T_{max}, maximum processing delay.

$$R = F_k \sum_{i=1}^{n} S_i / N_k \tag{6}$$

$$T_{max} = \tau_0 + \tau = \tau_0 + (N_k - \sum_{i=1}^{n} S_i) T_k \tag{7}$$



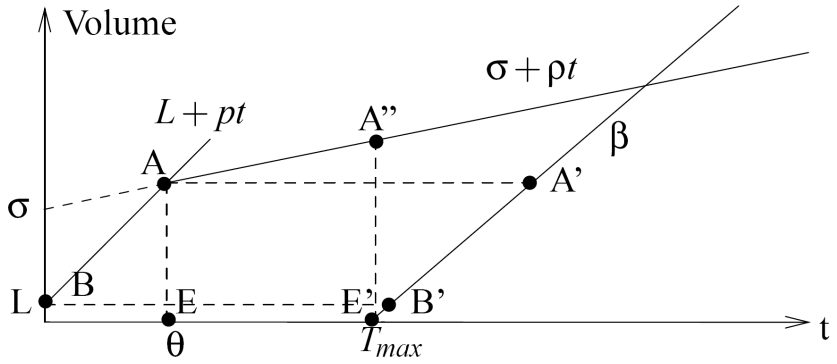
The delay and buffer bounds

- Delay bound

$$\bar{D} = \frac{L + \theta(p - R)^+}{R} + T_{max} \tag{8}$$

- Buffer bound

$$\bar{B} = \sigma + \rho T_{max} + (\theta - T_{max})^+[(p - R)^+ - p + \rho] \tag{9}$$



Cross-domain analysis steps

1. Configuration of slot tables
2. Derive cross-domain arrival and service curves
3. Calculate delay and backlog bounds

An example is given in the paper.

Conclusion

- With correct operation strategies, TDM VCs can smoothly cross clock domain boundaries, maintaining nice properties in hard delay and throughput guarantees.
- Network calculus can be applied for cross-domain delay and buffer analysis with ease.
- Extensions
  - Cover a boundary node with multiple clock domain interfaces
  - Support DVFS

Conclusion

- With correct operation strategies, TDM VCs can smoothly cross clock domain boundaries, maintaining nice properties in hard delay and throughput guarantees.
- Network calculus can be applied for cross-domain delay and buffer analysis with ease.
- Extensions
  - Cover a boundary node with multiple clock domain interfaces
  - Support DVFS

Conclusion

- With correct operation strategies, TDM VCs can smoothly cross clock domain boundaries, maintaining nice properties in hard delay and throughput guarantees.
- Network calculus can be applied for cross-domain delay and buffer analysis with ease.
- Extensions
  - Cover a boundary node with multiple clock domain interfaces
  - Support DVFS

Conclusion

- With correct operation strategies, TDM VCs can smoothly cross clock domain boundaries, maintaining nice properties in hard delay and throughput guarantees.
- Network calculus can be applied for cross-domain delay and buffer analysis with ease.
- Extensions
  - Cover a boundary node with multiple clock domain interfaces
  - Support DVFS

Conclusion

- With correct operation strategies, TDM VCs can smoothly cross clock domain boundaries, maintaining nice properties in hard delay and throughput guarantees.
- Network calculus can be applied for cross-domain delay and buffer analysis with ease.
- Extensions
  - Cover a boundary node with multiple clock domain interfaces
  - Support DVFS

Thank you!

Email: zhonghai@kth.se