

---

# Comparing Tightly and Loosely Coupled Mesochronous Synchronizers in a NoC Switch Architecture

**Daniele Ludovici**

*Computer Engineering lab – TUDelft - NL*

*MPSoC research group – UNIFE – Italy*

daniele@ce.et.tudelft.nl



University of Bologna



# OUTLINE

---

- GALS Network-on-Chip design paradigm
- Need for mesochronous synchronization
- Cost-effective mesochronous synchronization
  - ✓ A tightly coupled mesochronous synchronizer with the switch architecture
- Results
  - ✓ area overhead
  - ✓ power consumption
  - ✓ skew tolerance
- Conclusions

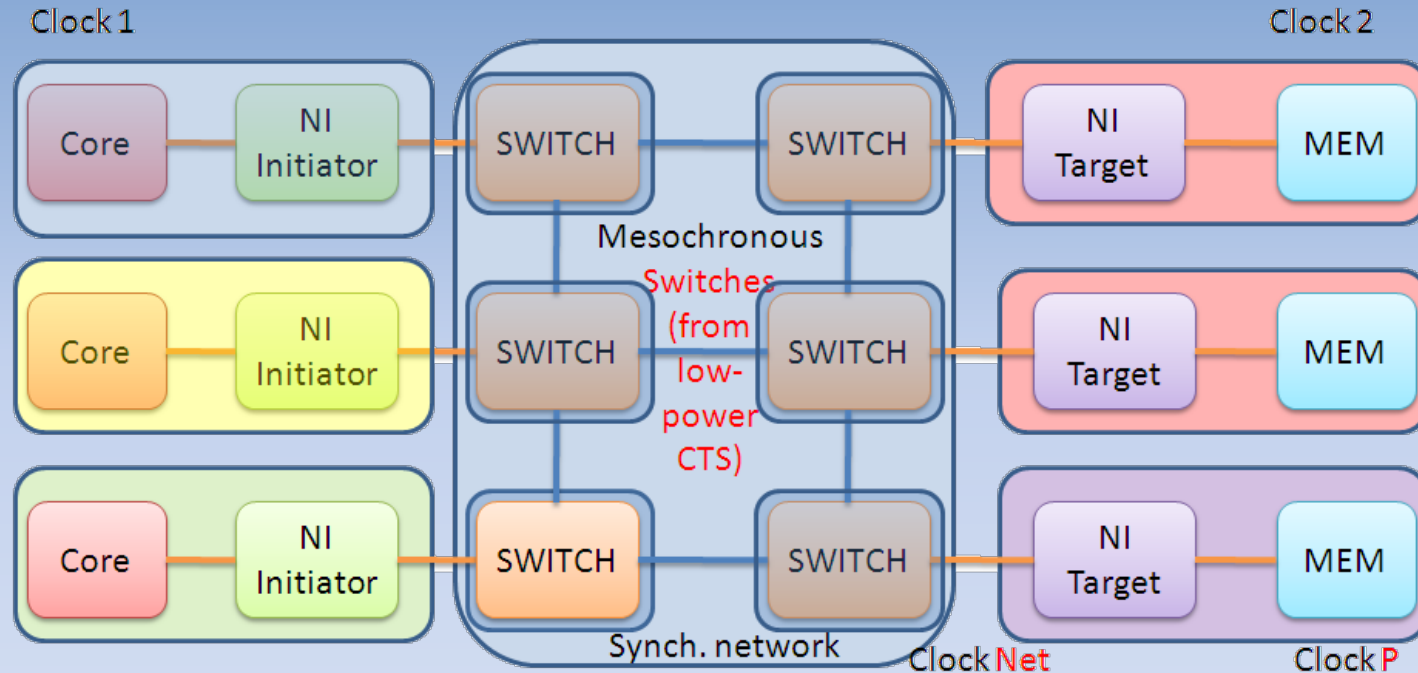
# MOTIVATION

---

- ❑ There is today little doubt on the fact that a high-performance and cost-effective NoC can be designed in 45nm and beyond under a relaxed synchronization assumption
  - ✓ Chip size, clock rate, global wires length increment
  
- ❑ A possible solution: GALS NoC
  - ✓ Processing blocks are separated and clocked independently
  - ✓ No global clock distribution => simplified timing closure
  - ✓ No rigid timing constraints between local clock domains

# GALS implementation

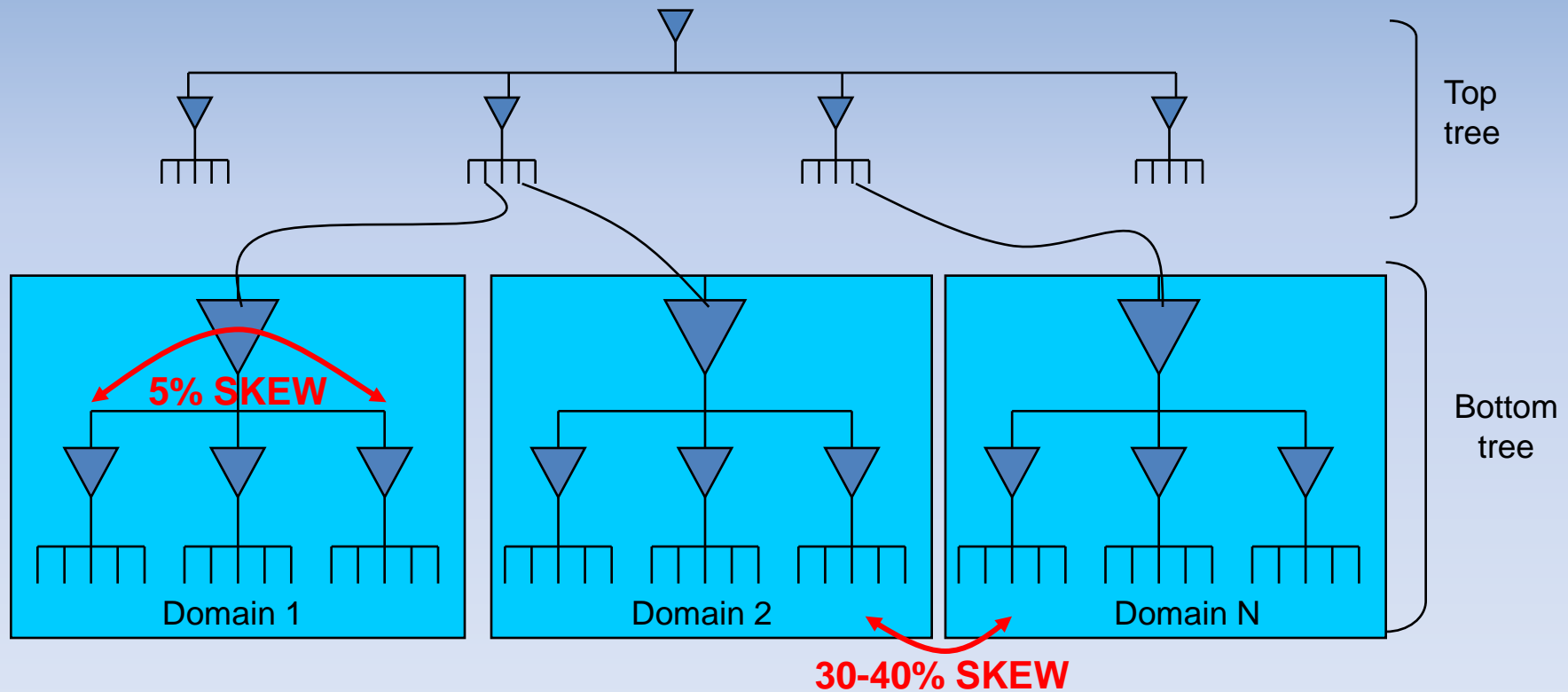
We chose one GALS implementation variant:



- **Affordable to** current mainstream **EDA design toolflows** with incremental effort
- Conscious use of area/power expensive **dual-clock FIFOs** (*used only at the network boundary*)
- **More compact mesochronous synchronizers** are used in the network
- **Hierarchical Clock Tree Synthesis by relieving clock phase offset constraints**

# Mesochronous Synchronization

Hierarchical clock tree with relaxed skew constraints might significantly decrease clock tree power



**Challenge: implementing cost-effective mesochronous synchronization**

# State-of-The-Art

<i>WORK</i>	<i>ADVANTAGES</i>	<i>DISADVANTAGES</i>
Edman et al Dally et al.	Standard-cells implementation Good timing margins	<b>High overhead</b> for a single link (8 flops + 1 mux for each bit line)
Semiat et al.	Advocates for brute force-synch	Tested for very <b>low speed</b> (25 MHz) <b>full custom</b> components
Kim et al. Mesgarzadeh et al.	Well suited for high-speed systems	<b>full custom</b> components <b>area</b> occupancy not assessed

Recently, 2 schemes target *technology-aware*, *low-overhead* and *standard cell* design focusing on NoCs (from STM and Intel)

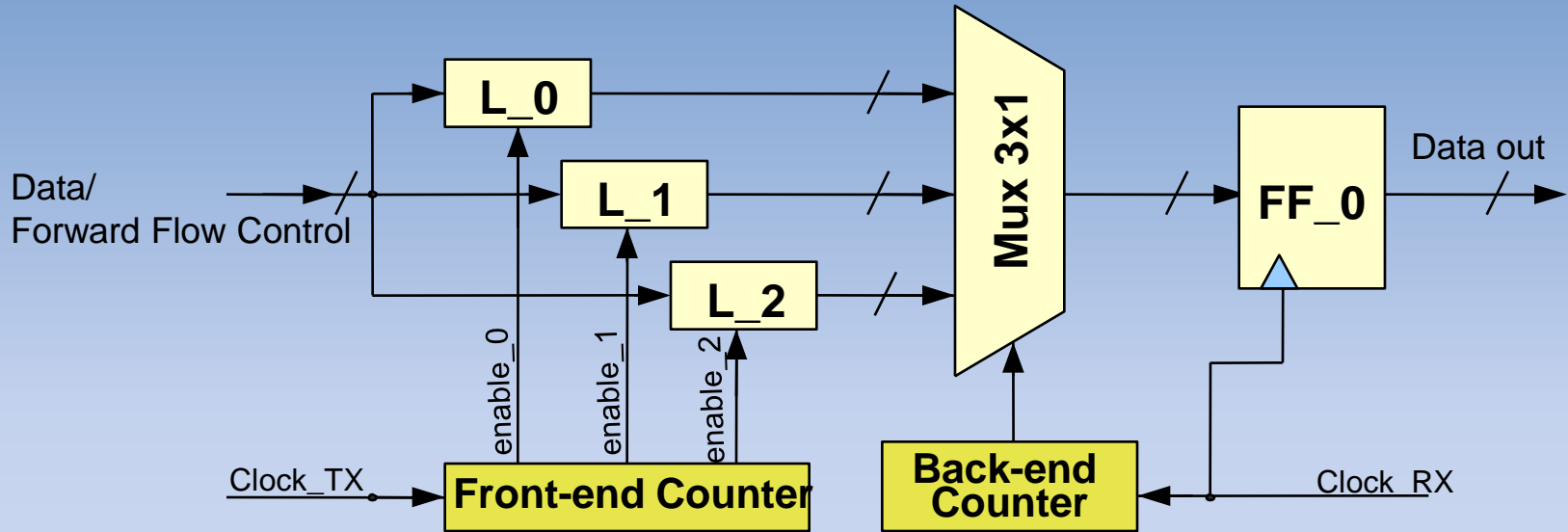
- source synchronous* design style

- leveraging *ping-pong buffering* schemes to counter timing and metastability concerns

- mesochronous synchronizers are still considered yet another module of the NoC architecture, thus impacting modularity

- Overhead for adding mesochronous synchronization support to a NoC link (with respect to fully synchronous communication) not well understood yet

# Proposed synchronizer

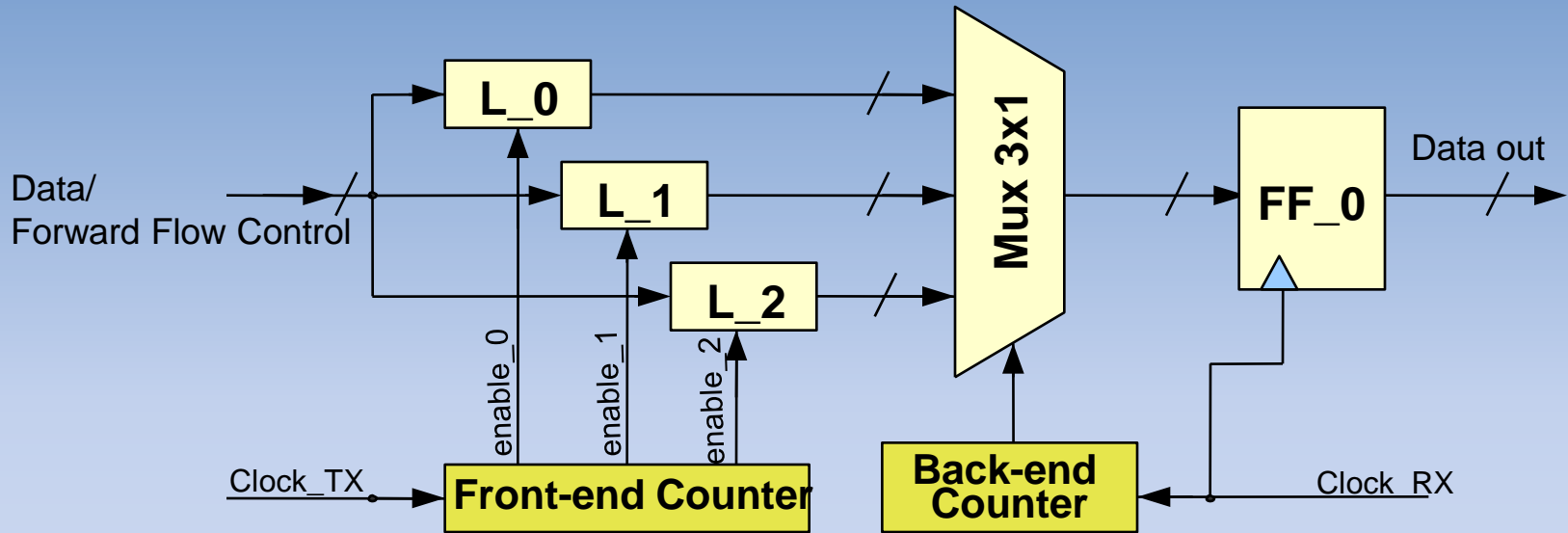


***Underlying principle: Information can safely settle in the front-end latches before being sampled by the target domain clock***

## ***Front-end:***

- Clock\_TX used as a strobe signal for data and flow control wires, thus avoiding timing problems associated with phase offset of clock signals
- Sampling through a number of latches used in a rotating fashion based on a counter

# Proposed synchronizer

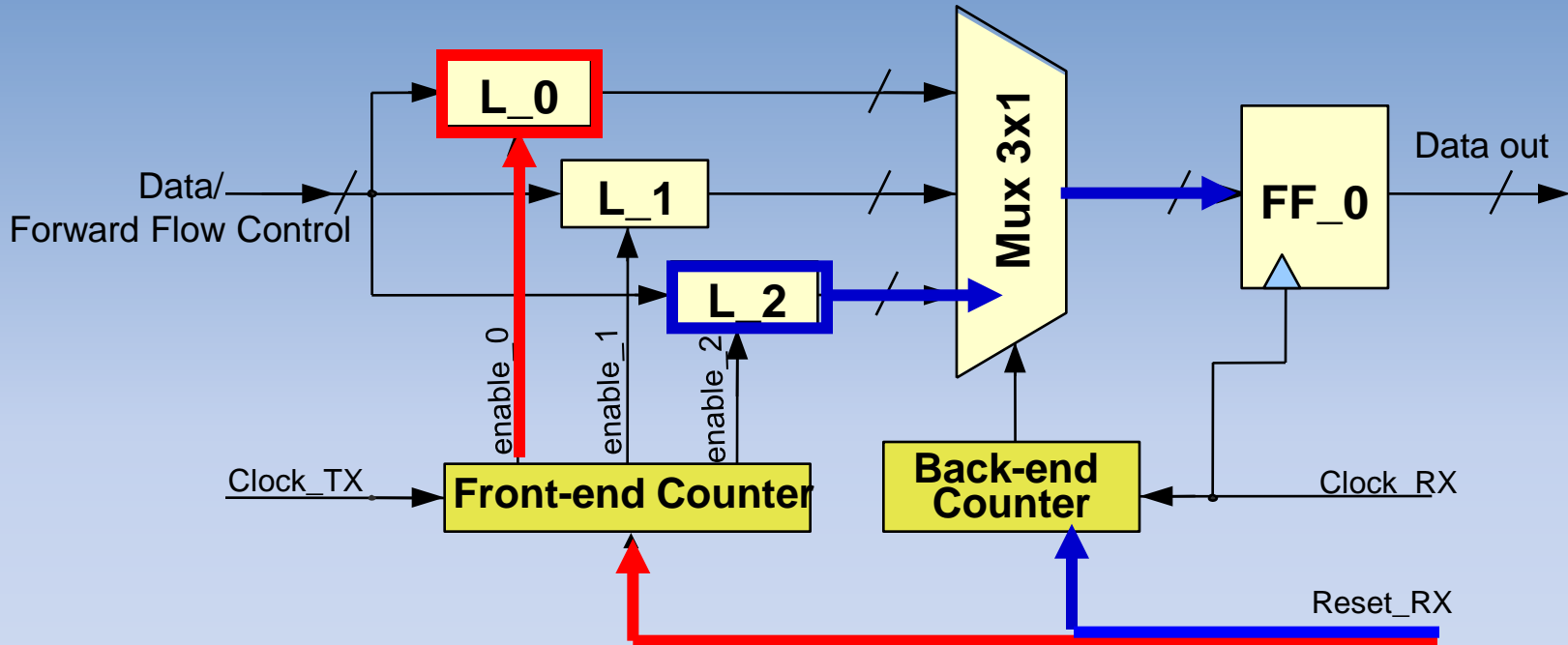


***Underlying principle: Information can safely settle in the front-end latches before being sampled by the target domain clock***

## ***Back-end:***

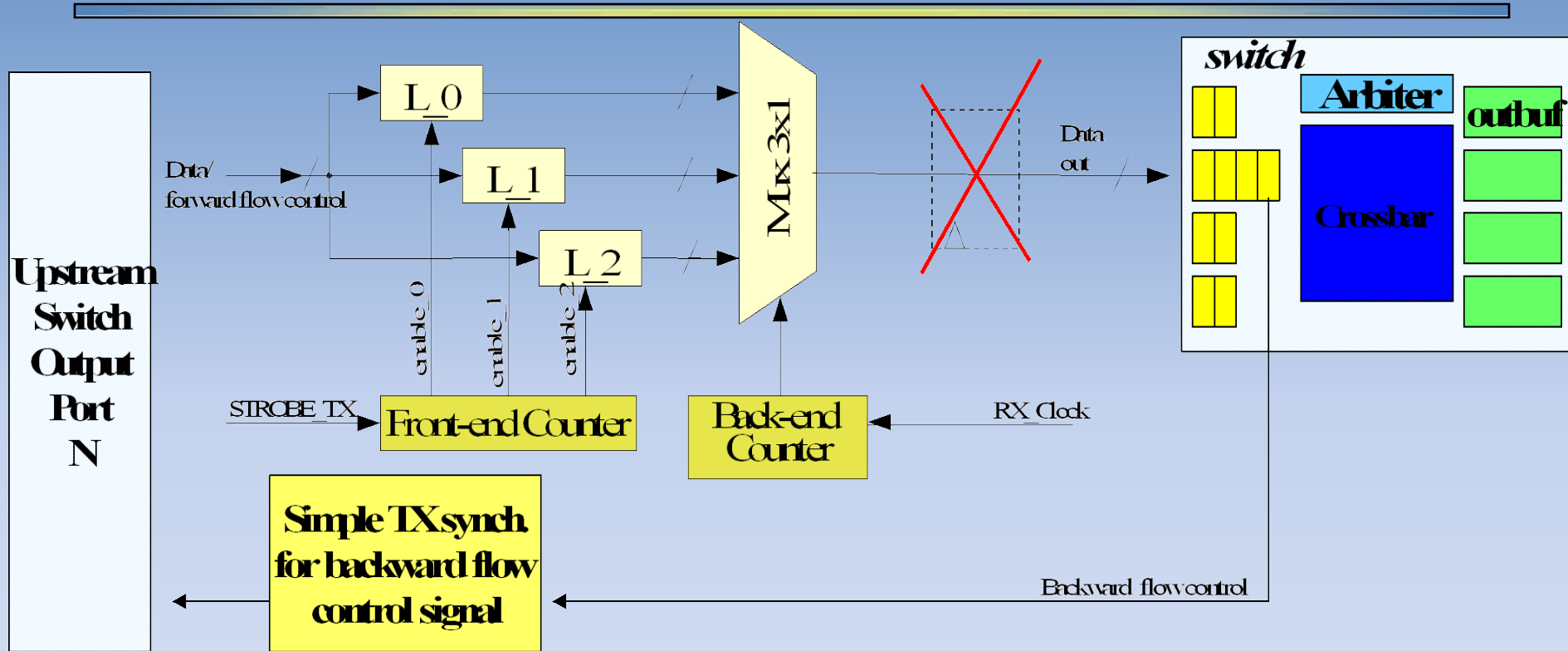
- Leverages local clock of the RX domain
- Samples data from one of the latches in the front-end thanks to multiplexing logic based on a counter

# Proposed synchronizer



- 3 input latch banks ensure timing constraints are safely met
  - ✓ data stability window at latch outputs is enough to tolerate wide range of clock phase offset
  - ✓ phase detector can be avoided
    - ✓ A unique bootstrap configuration can deal with all phase skew scenario
- Main challenge:
  - ✓ enforce timing margins for the NoC domain
  - ✓ study implications of synchronizer integration into a NoC

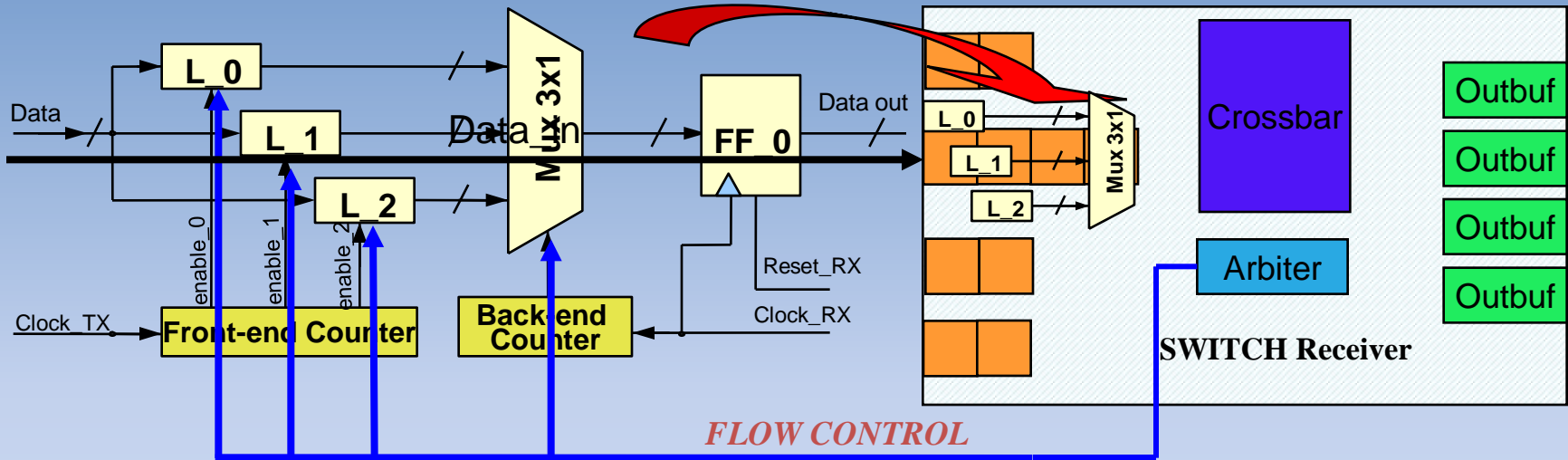
# Flow control



- Flow control implications considered

- ✓ xpipes comes with stall/go flow control; 2-stage buffer at each switch input
- ✓ Optimization: the back-end flip-flop IS the switch input buffer
- ✓ At least a 4 slot buffer is needed to keep using stall/go
- ✓ A small single-bit synchronizer needed to synchronize backward flow control signal

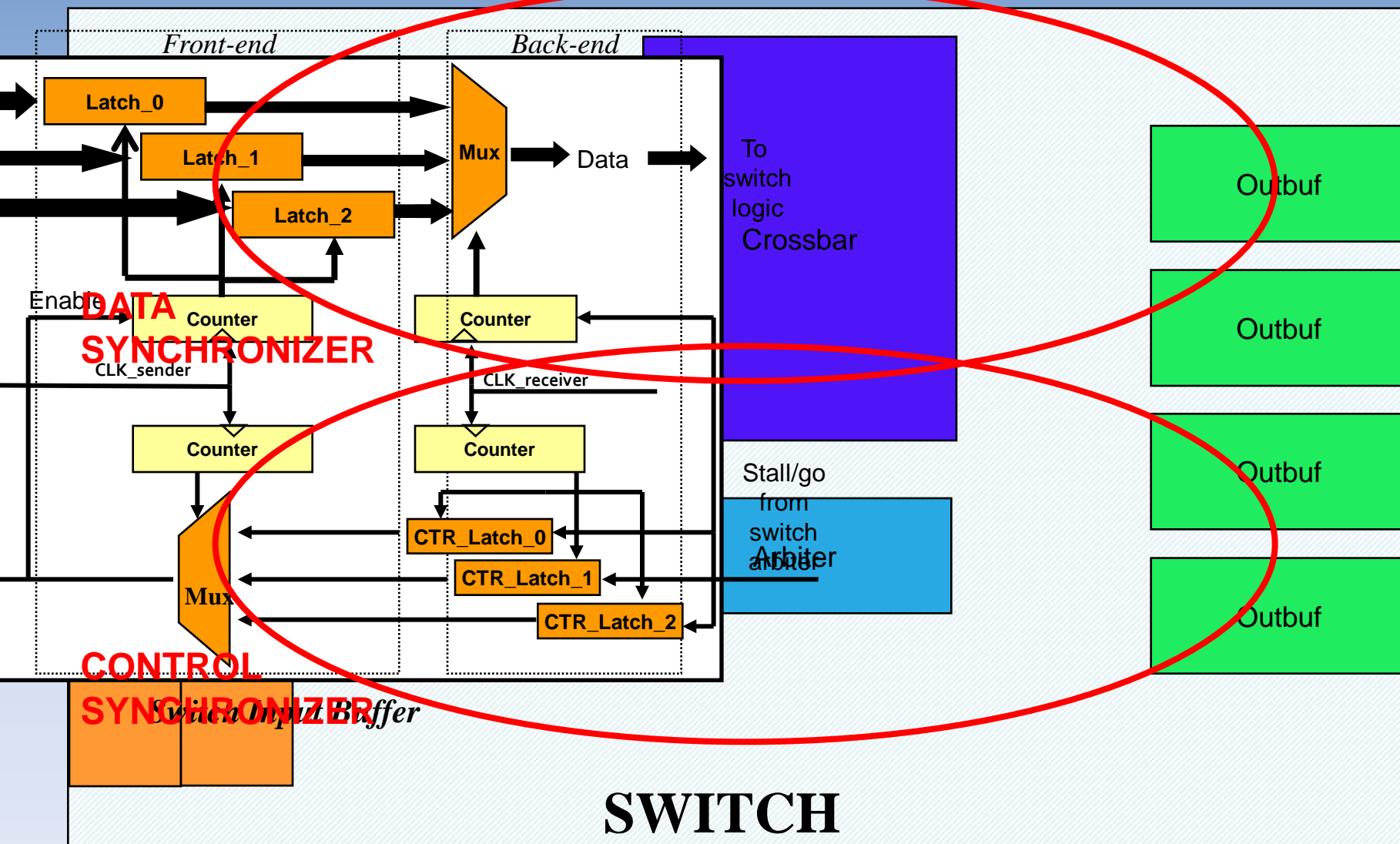
# Optimization



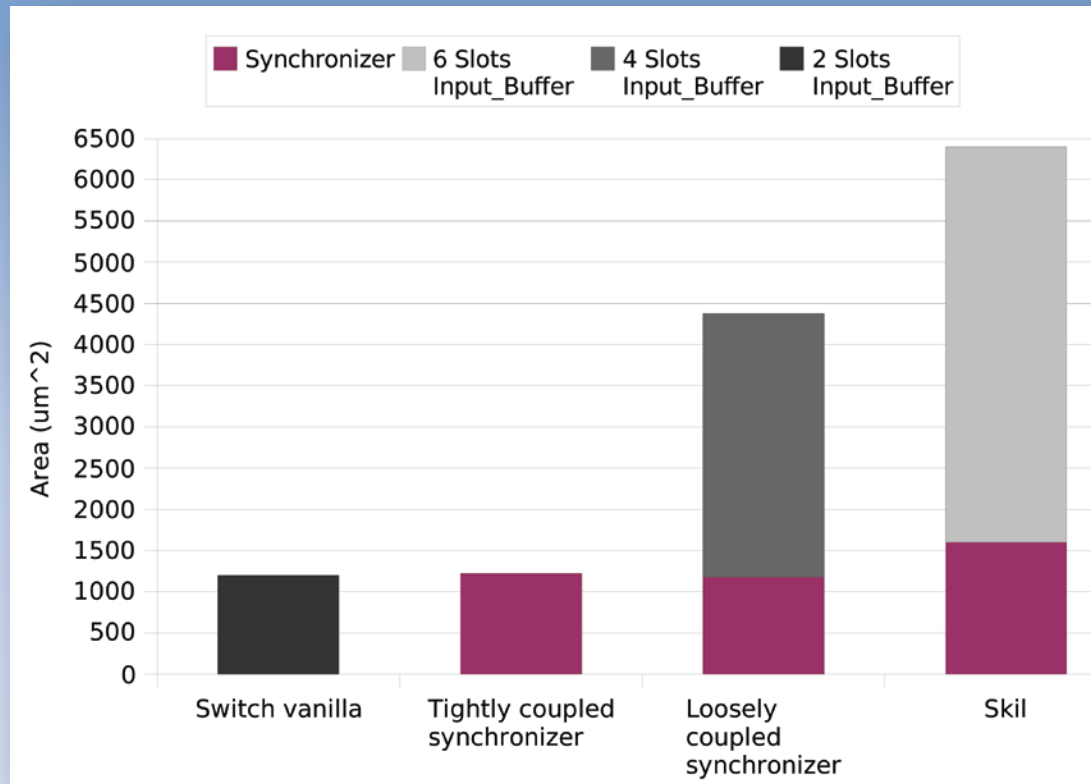
- Why not bringing flow control to the synchronizer latches as well?
- So that data can be stalled there, without need for extra buffer in the switch.
- Why not using the synchronizer IN PLACE OF the switch input buffer at all?

**A multi-purpose switch input buffer (buffering, synchronization and flow control) might lead to large area/power savings, lower latency and would preserve modularity**

# Tightly-coupled synchronizer (in the switch architecture)



# Synchronization Area Overhead



*65nm UMC technology library, target frequency 1 GHz*

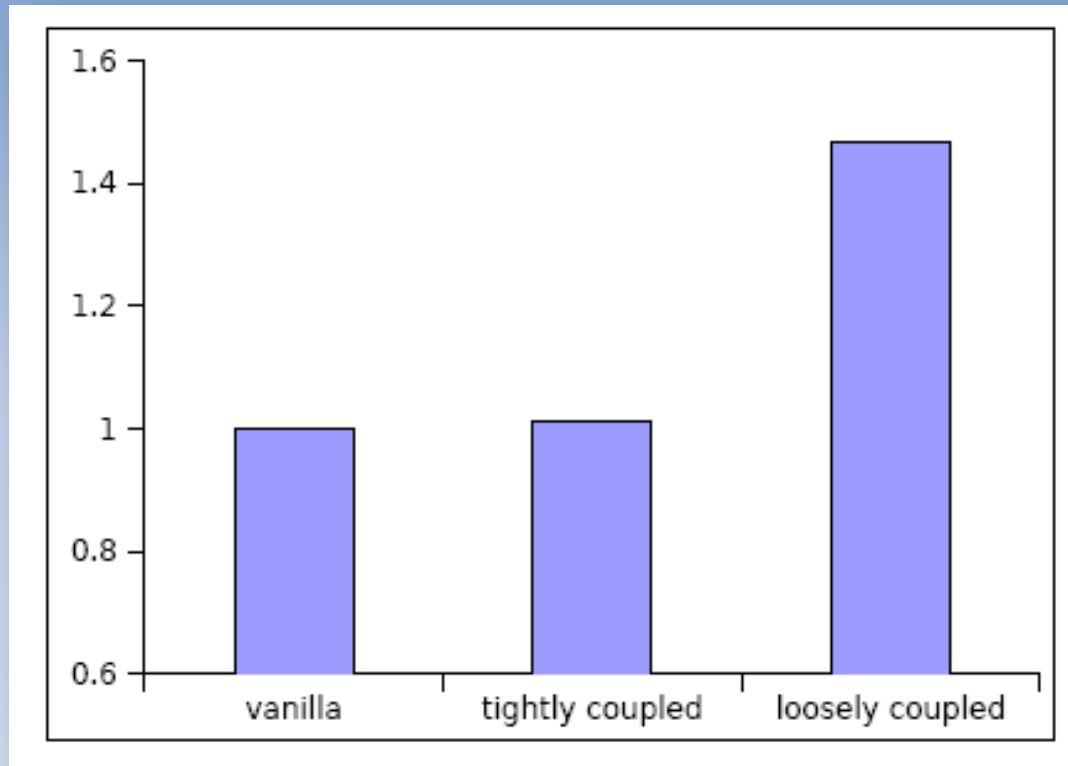
*Synchronization area = synchronizer area + input buffer area*

Tightly coupled architecture has a comparable synchronization area with vanilla switch

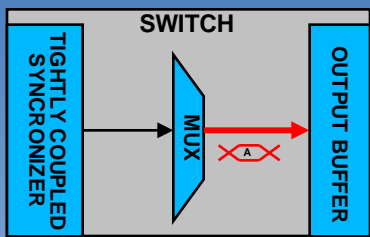
**A loosely coupled mesochronous synchronizer  
requires up to 5x the sychronization area of a vanilla switch!**

# Power analysis

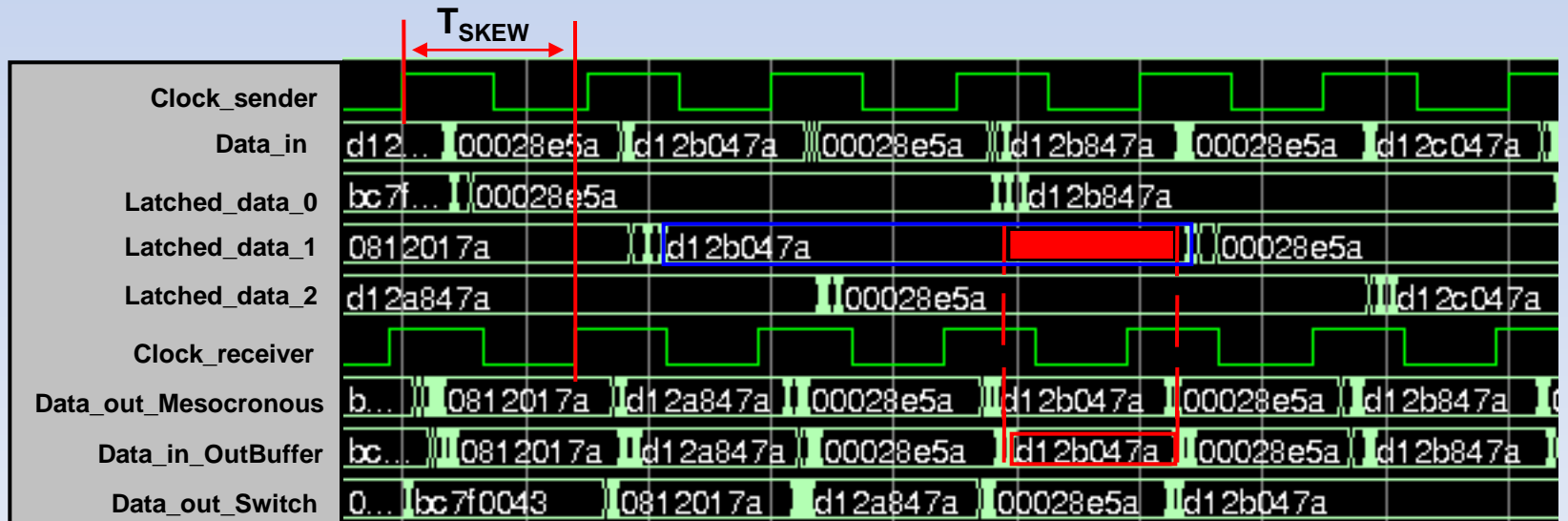
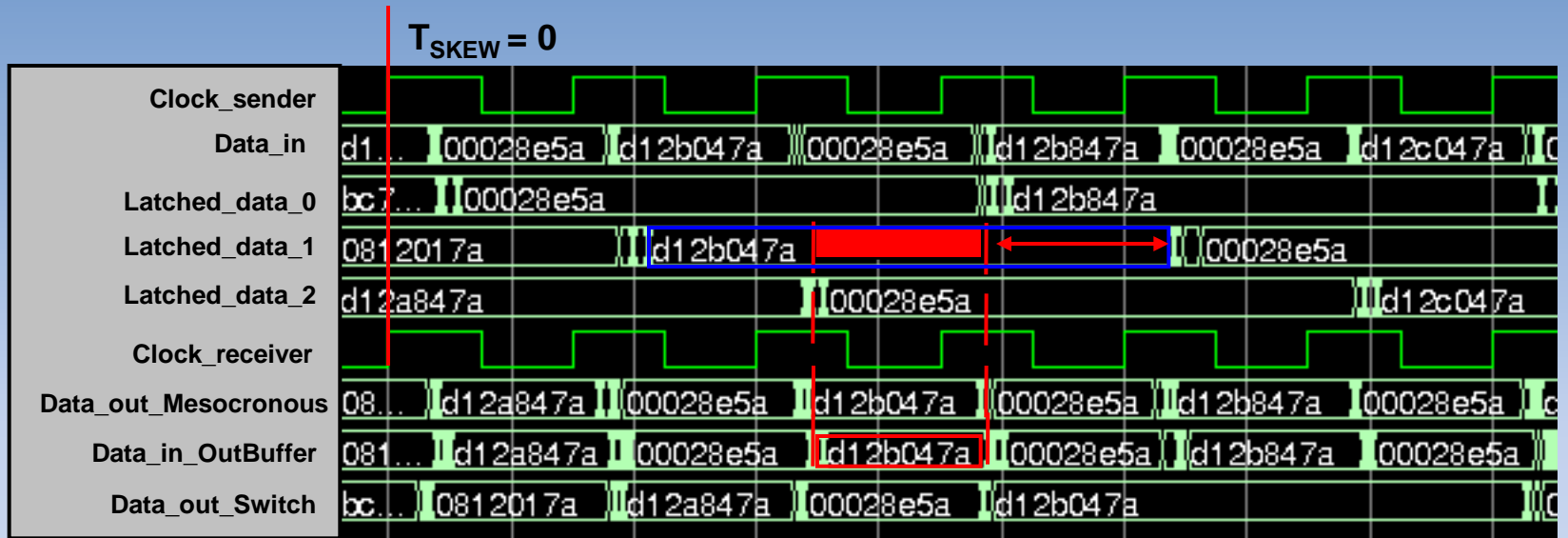
---



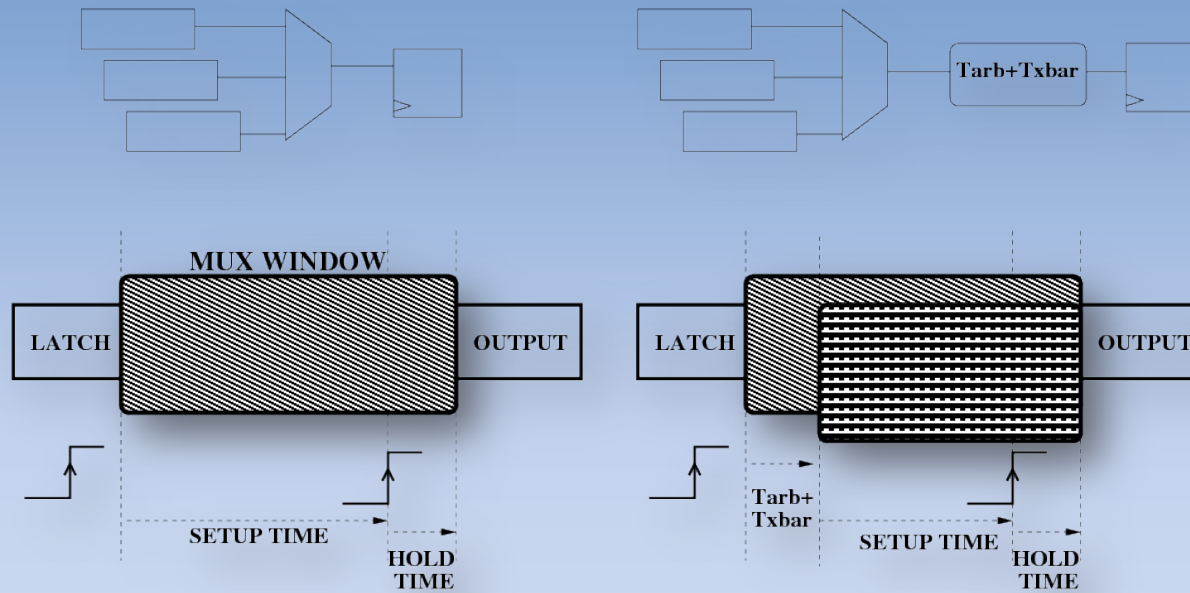
Full bandwidth parallel communication flows  
Same power of the fully synchronous switch!



# TIGHTLY COUPLED SYNCHRONIZER

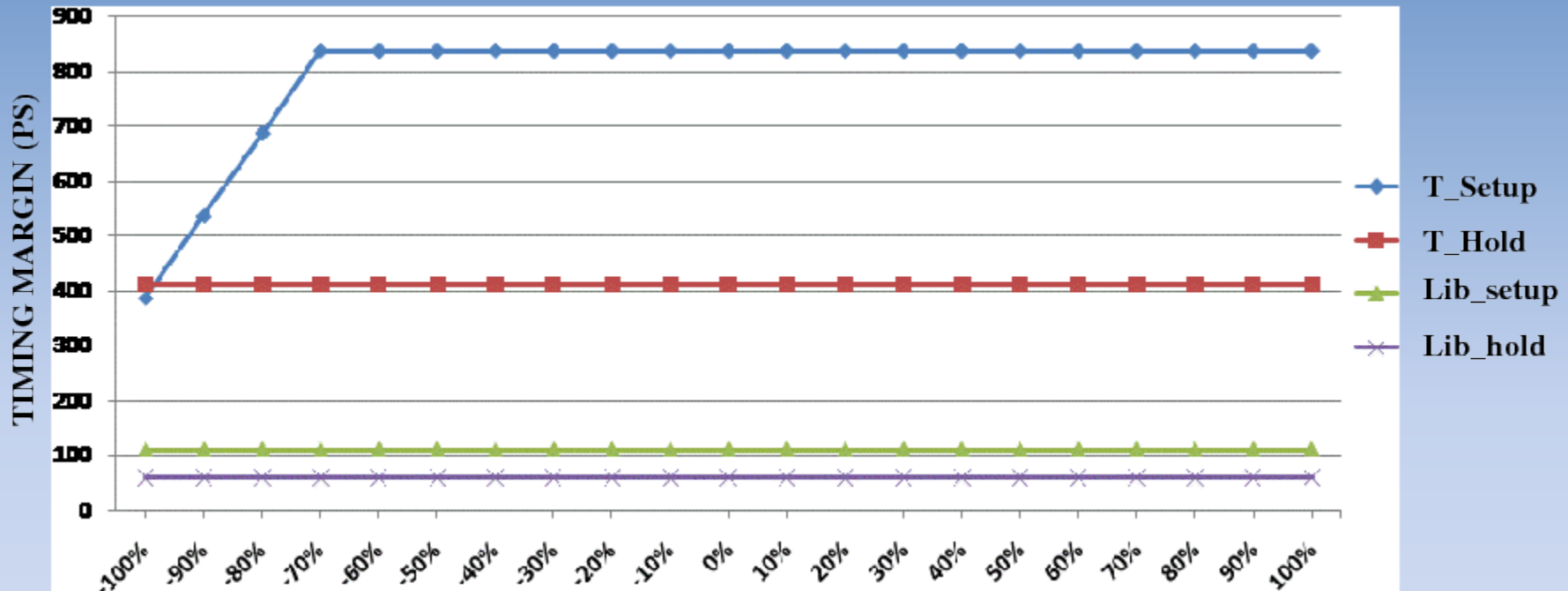


# SKEW TOLERANCE



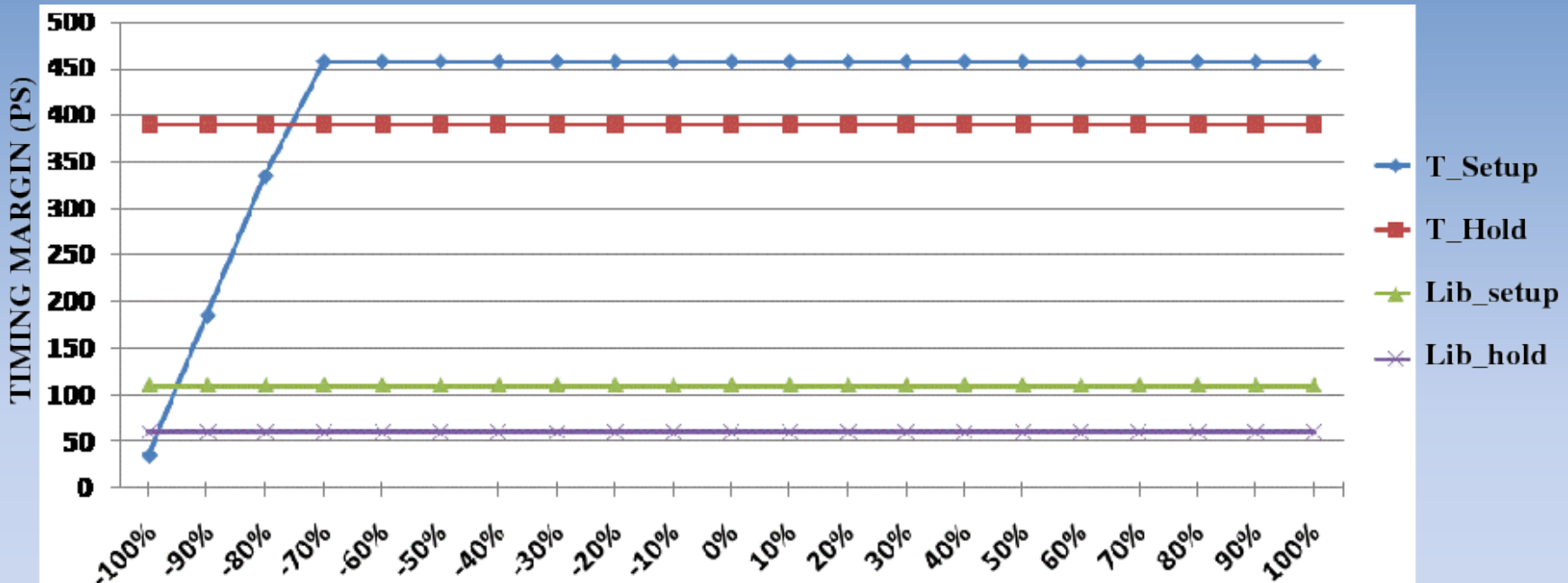
- ❑ **Setup Time:** from the beginning of mux window to the rising edge of the sampling element.
- ❑ **Hold Time:** from the rising edge of the sampling element to the end of the mux window.
- ❑ For the tightly coupled these metrics are taken at the output buffer.  $T_{arb} + T_{xbar}$  reduces “setup time” for the tightly coupled synchronizer.

# Loosely Coupled Skew Tolerance



- Pos. and Neg. skew are expressed as % of the clock period.
- Setup and Hold time compared with those of a FF in 65nm lib.
- **Hold Time** is stable and it has a solid margin.
- **Setup Time** decreases when latch outputs end switching inside the mux window BUT there is still a safe margin!

# Tightly Coupled Skew Tolerance



- ❑ **Hold Time** is stable and it has a solid margin
- ❑  $T_{arb} + T_{xbar}$  lower the Setup Time curve starting point
- ❑ **Setup Time** becomes even more critical for high negative skew
- ❑ Tightly coupled synch cannot work beyond -95% skew!

# Summing up

---

- ❑ A loosely coupled mesochronous synchronizer placed in front of the switch fabric implies large buffering in the switch input buffer.
- ❑ We advocate for a tightly coupled synchronizer with the switch architecture.
  - ✓ **multi-purpose input buffer in charge of synchronization, buffering and flow control.**
  - ✓ synchronous and mesochronous ports allowed in the same switch.
- ❑ **No area/power/timing overhead** with respect to a fully synchronous switch.
- ❑ Trade-off with skew tolerance.
- ❑ Maximum achievable skew is typically less than 100%.

---

**Thank you!**