

針對內建人工智慧之車用晶片且以
【零故障】為目標之智慧型測試方法

Process Resilient Fault-Tolerant Delay-Locked Loop Using TMR with Dynamic Timing Correction

楊竣宇 (Darren J.-Y. Yang)

03/26

Outline

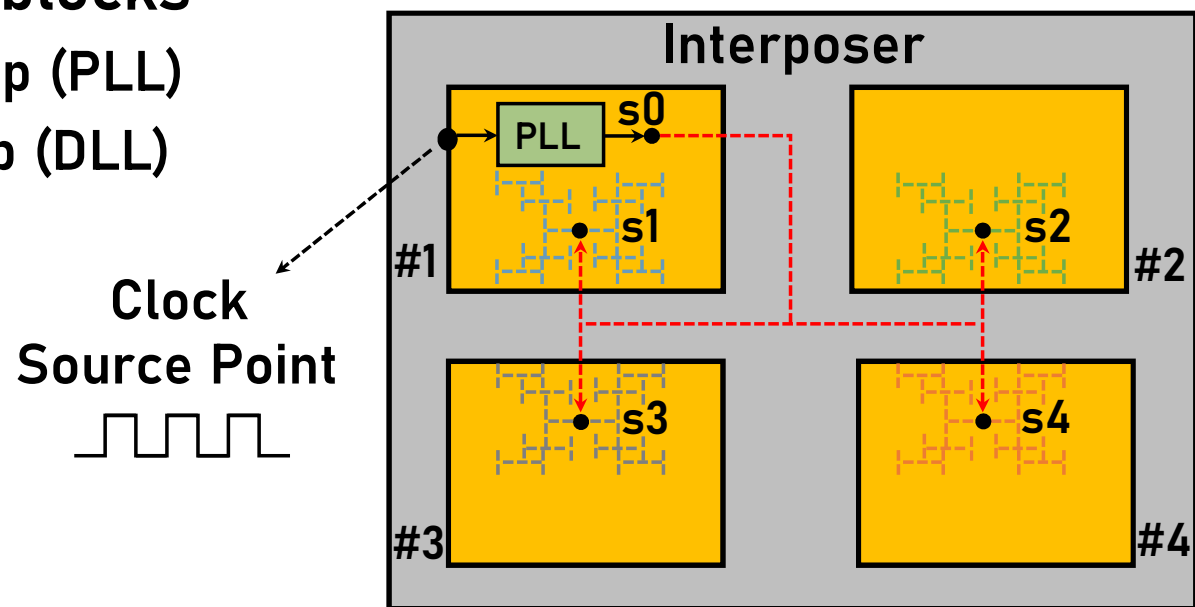
- Introduction
- Evolution of Fault and Error Tolerant (FET) DLL
- Experimental Results
- Conclusion

Outline

- **Introduction**
 - Background
 - Problem
 - Objective
- Evolution of Fault and Error Tolerant (FET) DLL
- Experimental Results
- Conclusion

Clock Distribution Problem

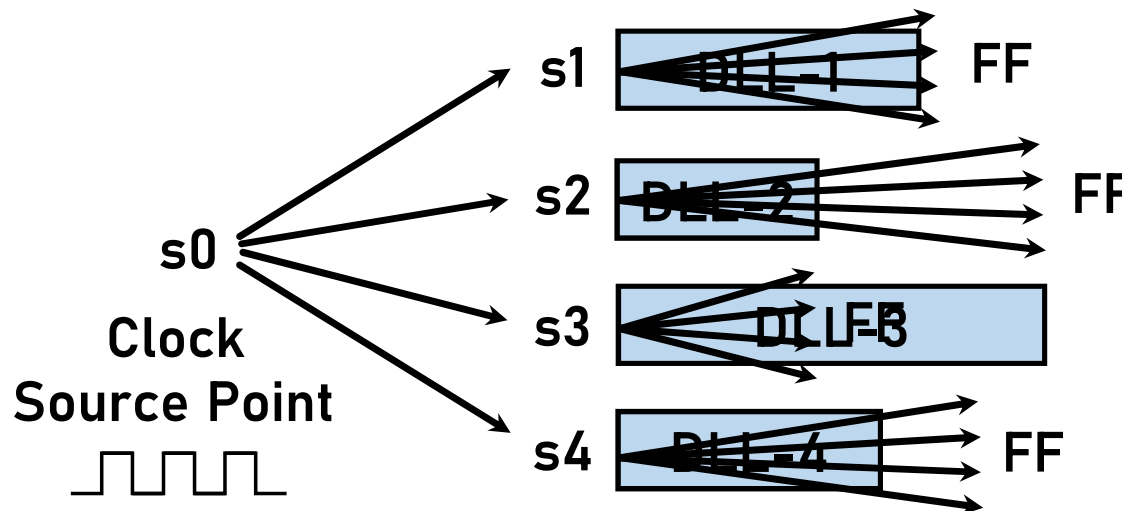
- In a heterogeneously integrated Multi-Die IC
 - Functional dies are designed and fabricated with different process
- Important building blocks
 - Phase-Locked Loop (PLL)
 - Delay-Locked Loop (DLL)



C.-Y. Cheng, S.-Y. Huang, D.-M. Kwai, and Y.-F. Chou, "DLL-Assisted Clock Synchronization Method for Multi-Die ICs", Proc. of IEEE Int'l Conf. on Computer Design, pp. 473-476, Nov. 2017.

DLL-Assisted Clock Synchronization

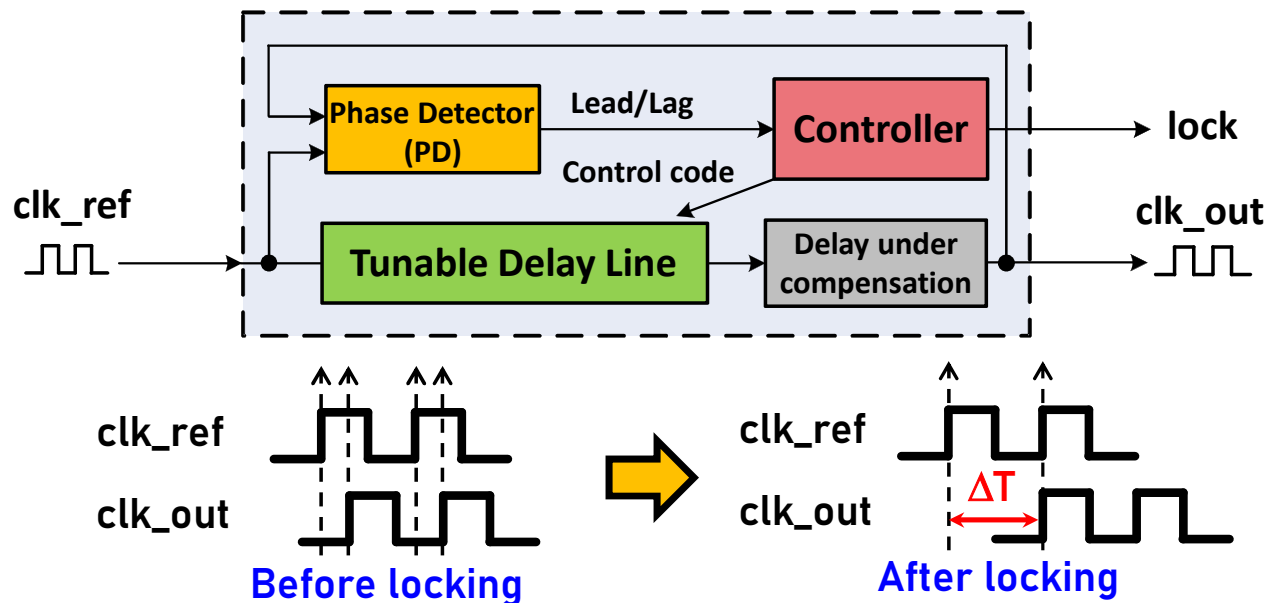
- Objective: All Flip-Flop in each die receive clock at the same time
- Different dies → Different Within-Die Clock Latencies
→ Flip-Flops (FFs) receive the clock signal with skews



- The inter-die clock skew is to be minimized by inserted DLLs
- The width of a DLL box denoted as its input-to-output delay

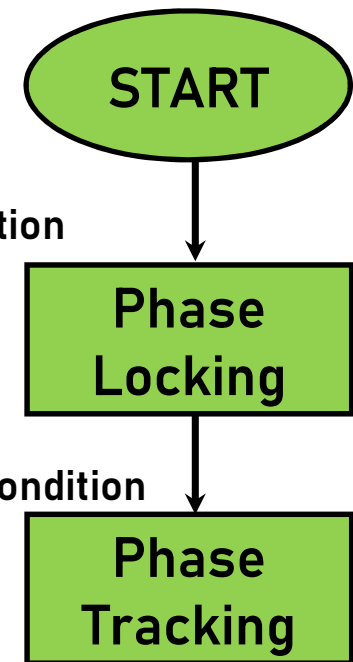
Architecture of a Basic DLL

- The basic DLL consists of 3 components:
 - (1) Phase Detector (2) Controller (3) Tunable Delay Line (TDL)
- DLL will experience two stages:
 - (1) Phase Locking (2) Phase Tracking



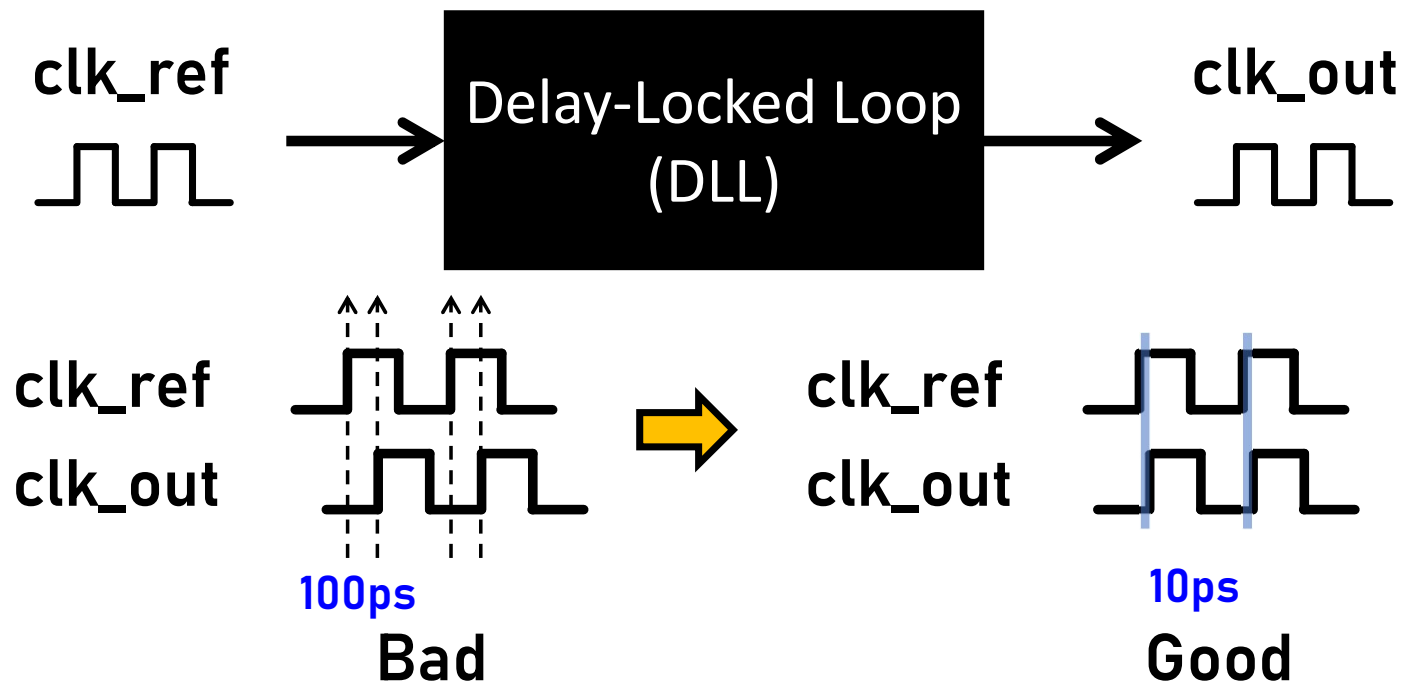
Derive algorithm
find locking condition

Derive algorithm
maintain locking condition



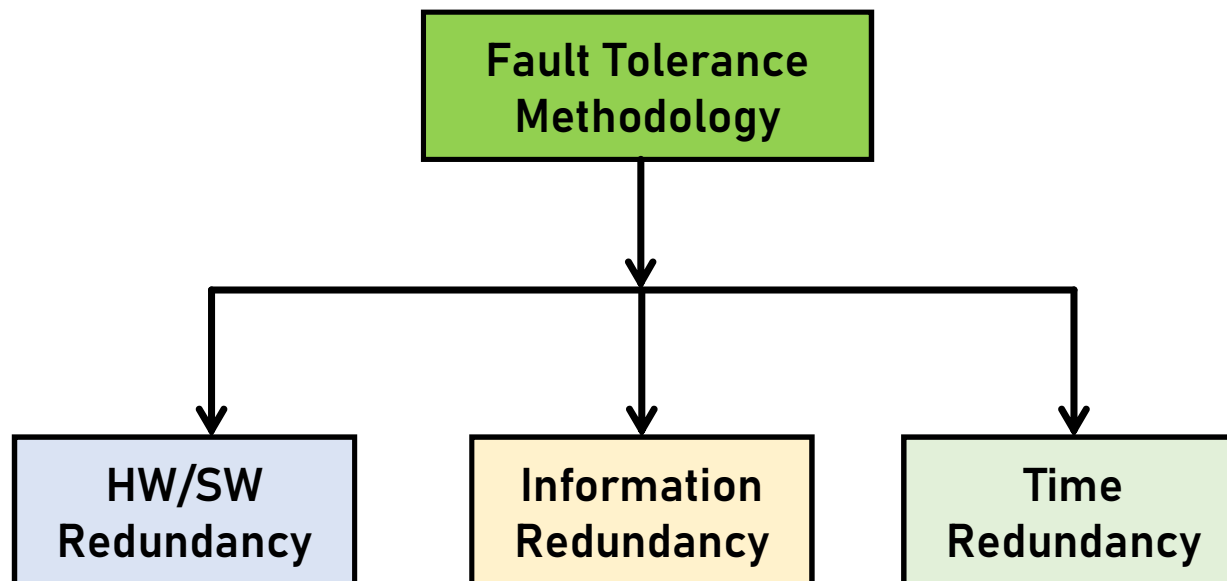
Performance We Care – Max. Phase Error

- Definition of maximum phase error
 - The worst-case phase error amount between `clk_ref` and `clk_out` over a time frame (e.g., 1000 cycles) after the DLL is locked!



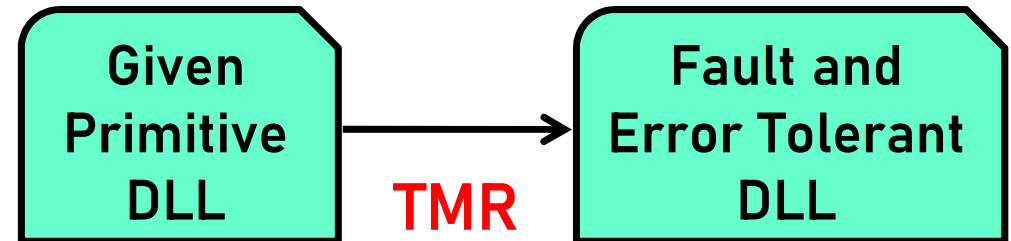
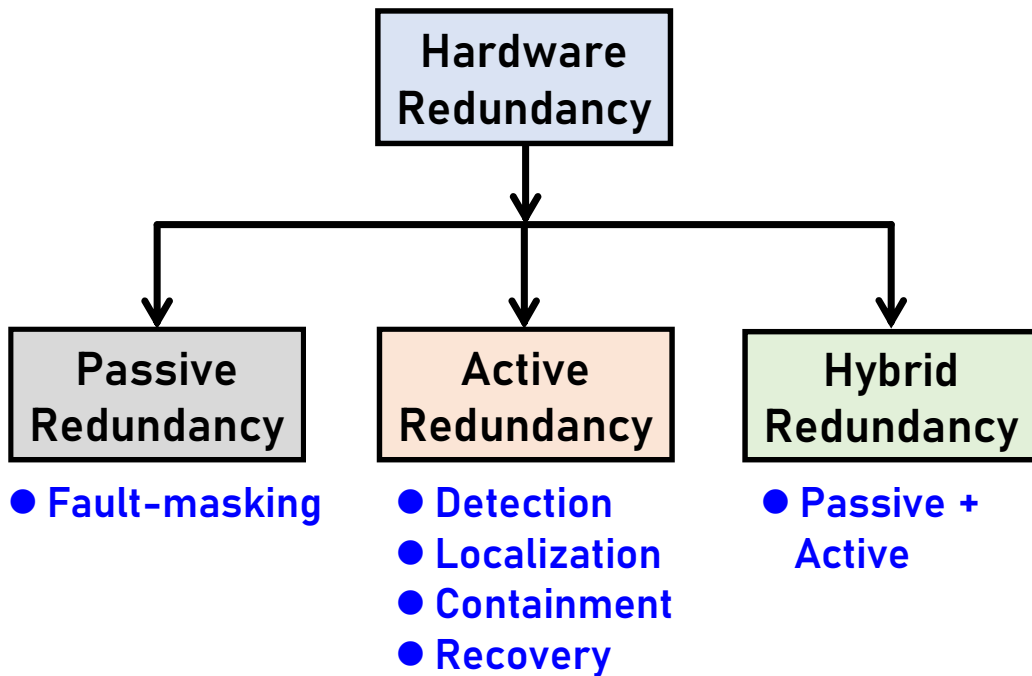
Objective of This Work

- Build a Fault and Error Tolerant (FET) scheme for clock subsystem
 - Especially for Delay-Locked Loop (DLL)



Objective of This Work

- Hardware redundancy is suitable for clock subsystem
- Active redundancy will need other circuit to monitor our DLL
- We take passive redundancy – Triple Module Redundancy (TMR)

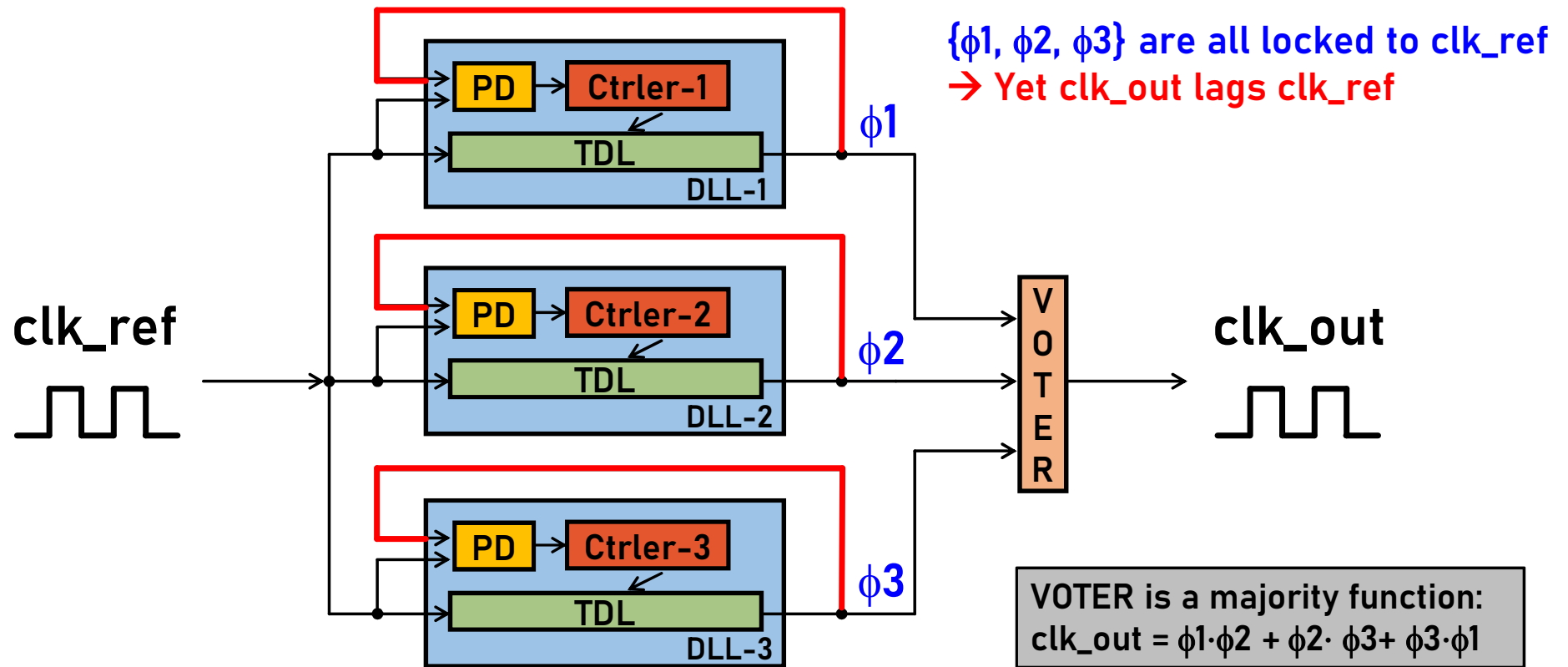


Outline

- Introduction
- **Evolution of Fault and Error Tolerant (FET) DLL**
 - Naïve FET-DLL Architecture
 - FET-DLL with static timing correction
 - Process-Resilient FET-DLL with dynamic timing correction
- Experimental Results
- Conclusion

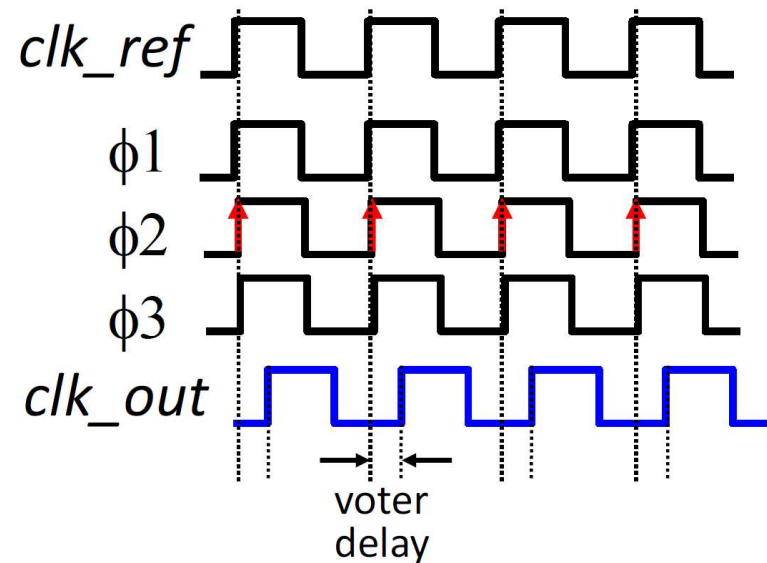
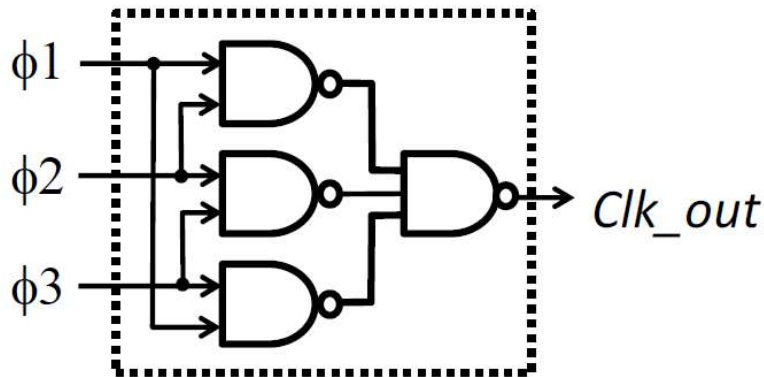
Naïve FET-DLL Architecture

- Three primitive DLL decide the output through VOTER
- Each DLL performs their phase-locking simultaneously



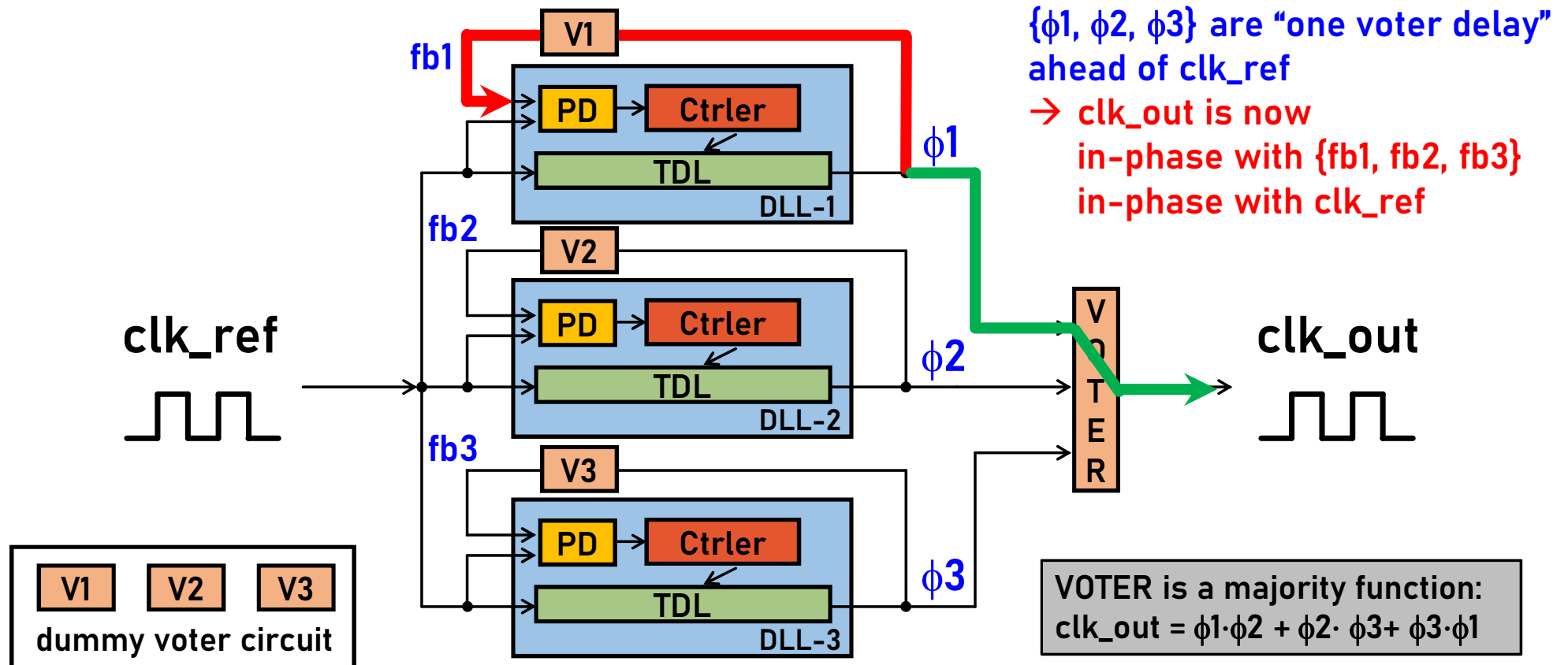
Output Lagging Problem

- The VOTER circuit takes time to calculate
 - $\delta_{\text{voter1}} = \text{Delay } (\phi1 \rightarrow \text{clk_out})$
 - $\delta_{\text{voter2}} = \text{Delay } (\phi2 \rightarrow \text{clk_out})$
 - $\delta_{\text{voter3}} = \text{Delay } (\phi3 \rightarrow \text{clk_out})$
 - $\text{Max}(\delta_{\text{voter1}}, \delta_{\text{voter2}}, \delta_{\text{voter3}})$ greater than 100ps



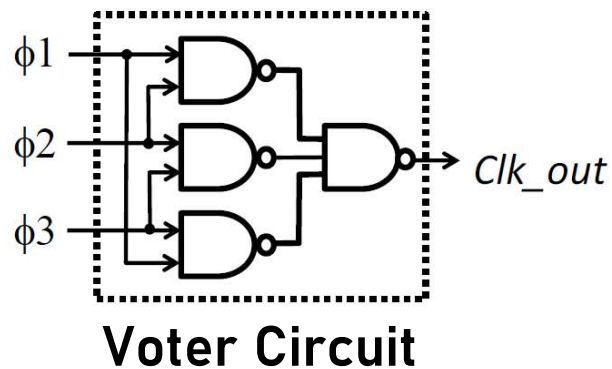
FET-DLL with static timing correction

- Fix the output-lagging problem
 - Add dummy voter circuit on its feedback path

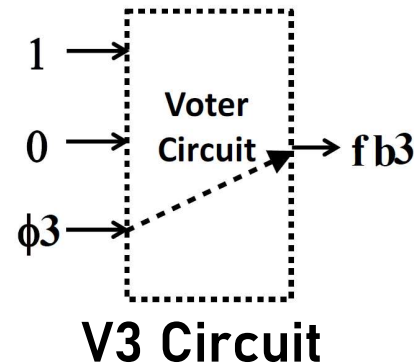
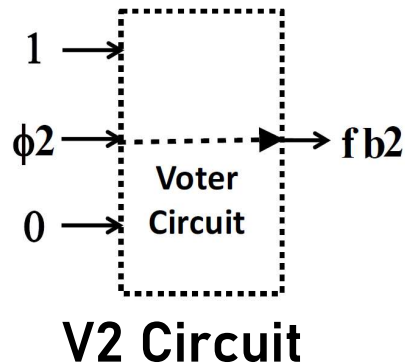
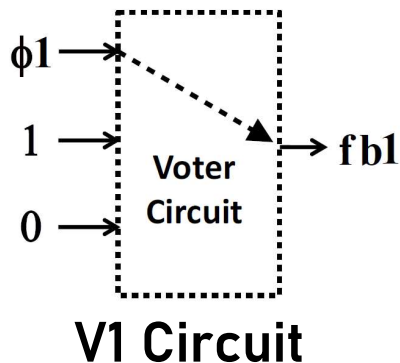


Detailed Voter circuit and Its dummy

- Feedback signal dominate dummy output by input assignment



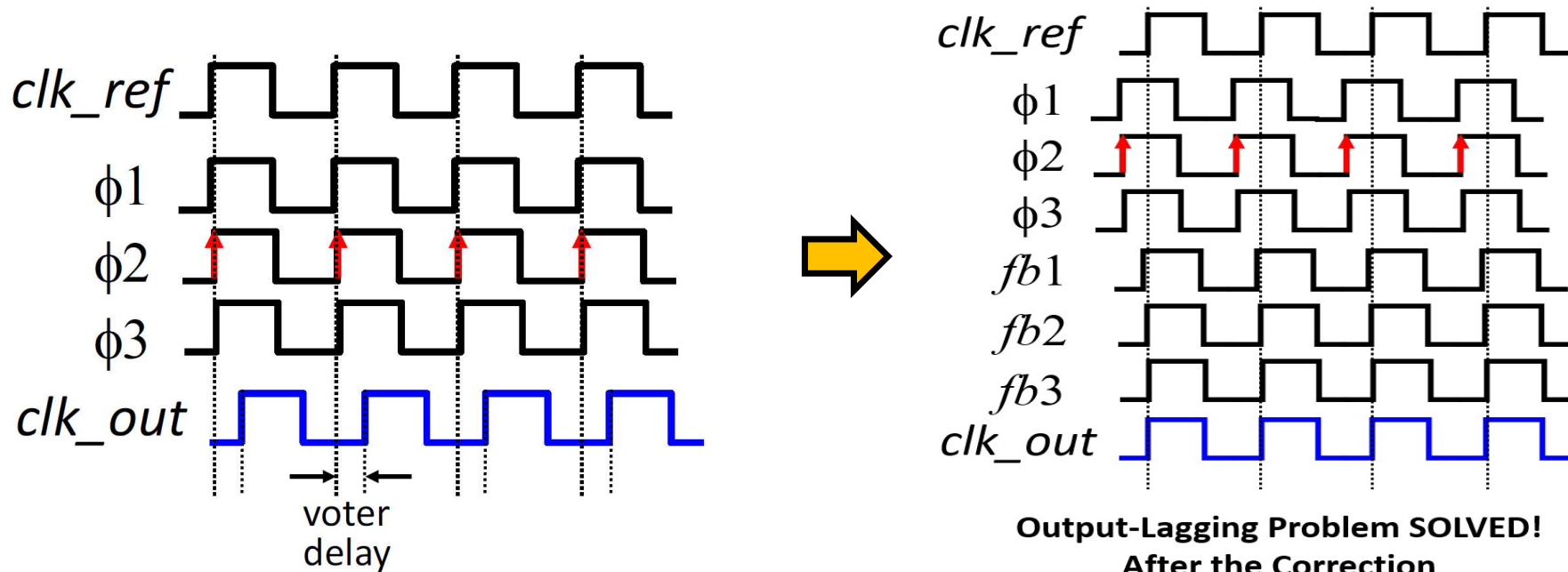
- $\delta_{voter1} = \text{Delay } (\phi 1 \rightarrow clk_out)$
- $\delta_{voter2} = \text{Delay } (\phi 2 \rightarrow clk_out)$
- $\delta_{voter3} = \text{Delay } (\phi 3 \rightarrow clk_out)$



- $\delta_{v1} = \text{Delay } (\phi 1 \rightarrow fb1)$
- $\delta_{v2} = \text{Delay } (\phi 2 \rightarrow fb2)$
- $\delta_{v3} = \text{Delay } (\phi 3 \rightarrow fb3)$

Timing Relationships after timing correction

- Clk_out and {fb1, fb2, fb3} have similar phase
- Since {fb1, fb2, fb3} in-phase with clk_ref
- clk_out is roughly in-phase with clk_ref



Process Variation Issue

Ideally, we have wished that

Delay of {V1, V2, V3} = Delay of the “VOTER”

But in reality,

there could be mismatch due to process variation

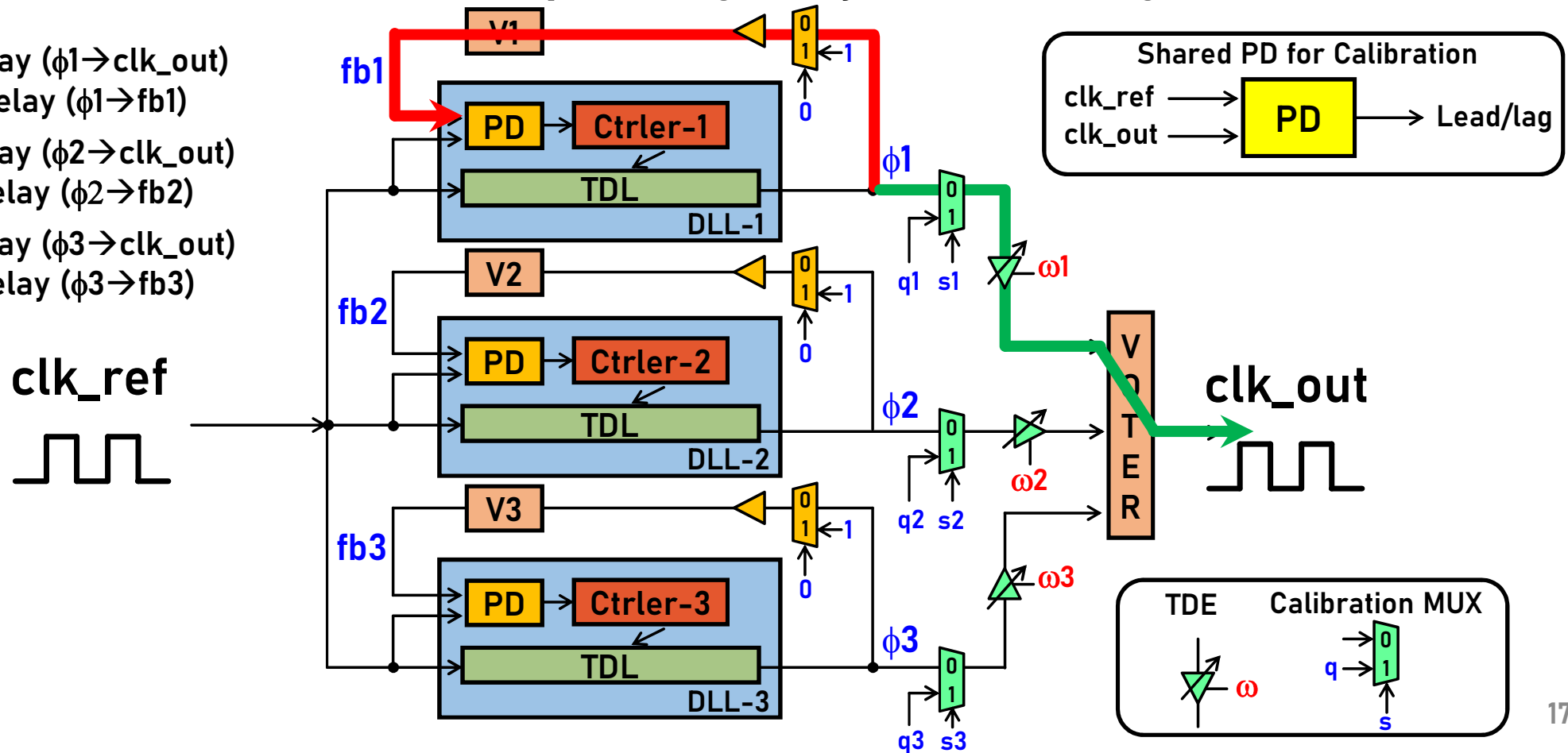
Outline

- Introduction
- **Evolution of Fault and Error Tolerant (FET) DLL**
 - Naïve FET-DLL Architecture
 - FET-DLL with static timing correction
 - **Process-Resilient FET-DLL with dynamic timing correction**
- Experimental Results
- Conclusion

FET-DLL with Dynamic Calibration

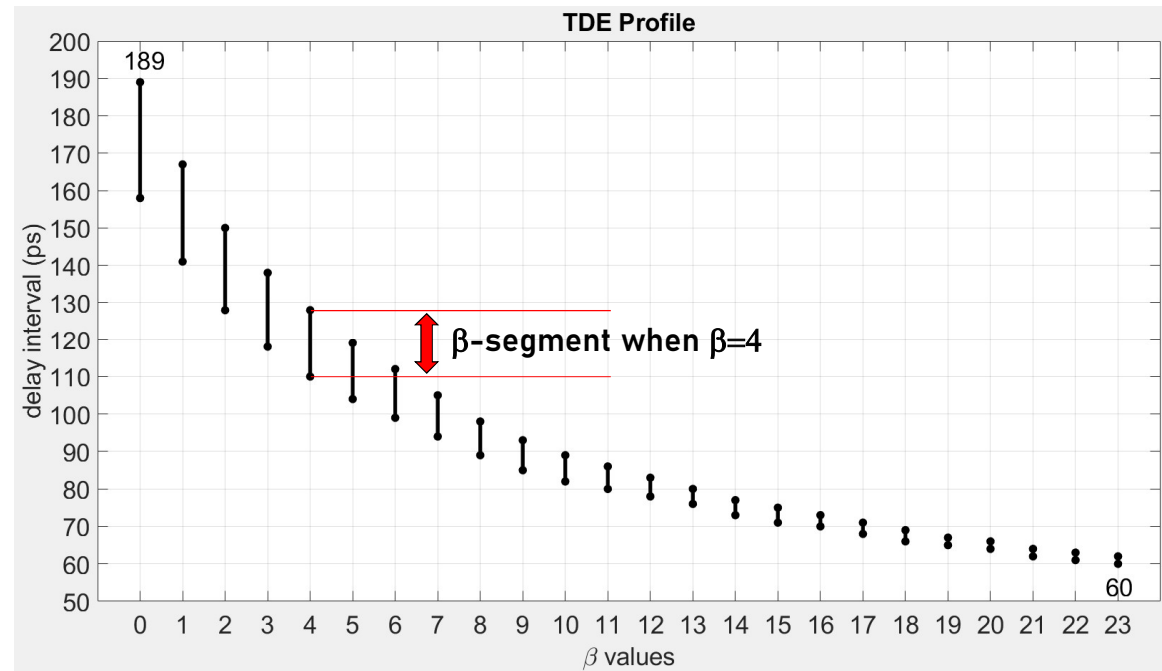
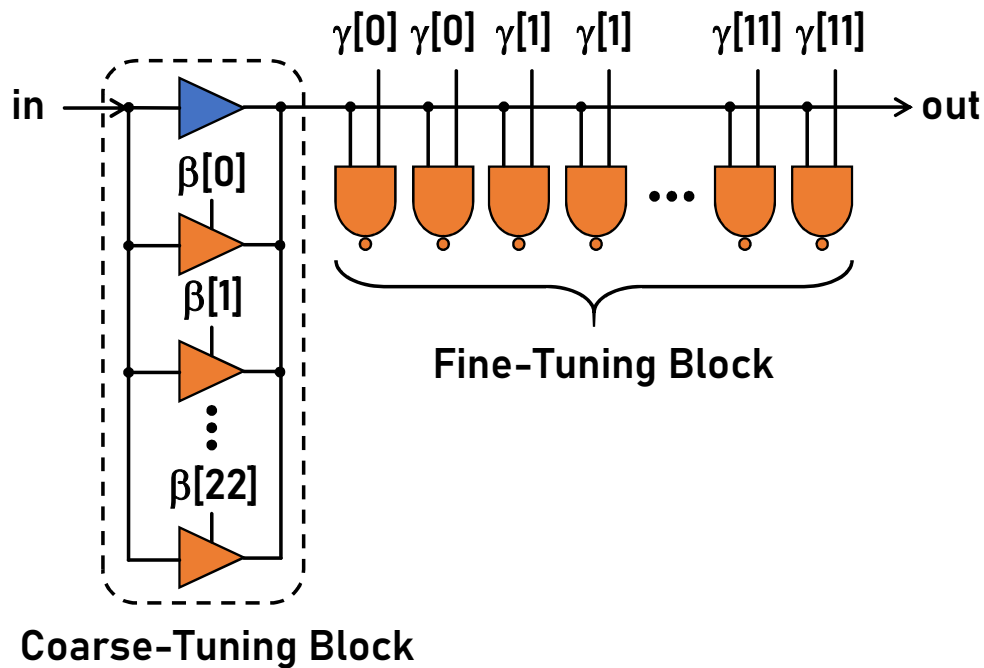
- Enhance FET-DLL incorporating a “dynamic timing correction”

- Delay ($\phi_1 \rightarrow \text{clk_out}$)
= Delay ($\phi_1 \rightarrow \text{fb1}$)
- Delay ($\phi_2 \rightarrow \text{clk_out}$)
= Delay ($\phi_2 \rightarrow \text{fb2}$)
- Delay ($\phi_3 \rightarrow \text{clk_out}$)
= Delay ($\phi_3 \rightarrow \text{fb3}$)

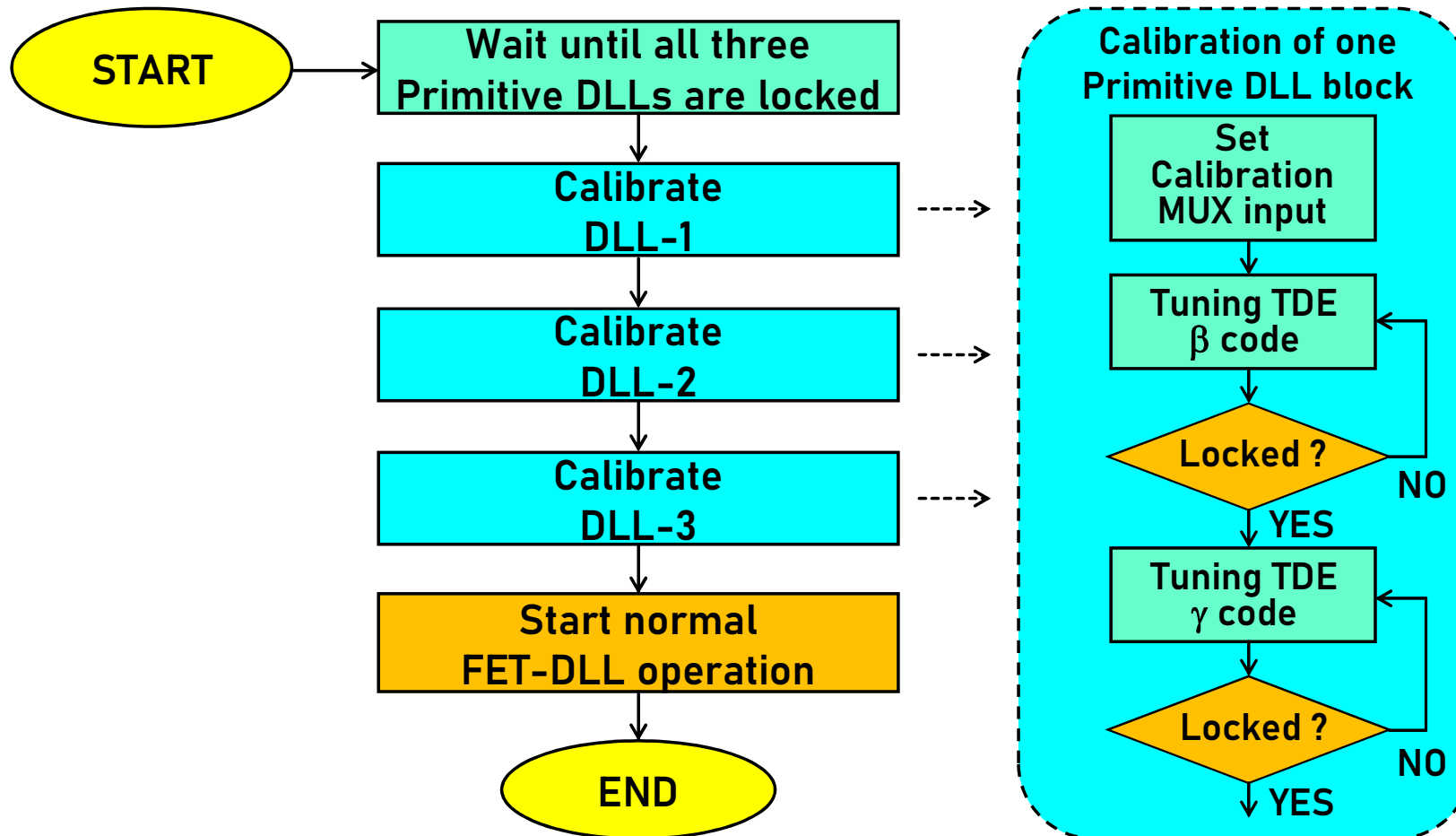


Tunable Delay Element (TDE)

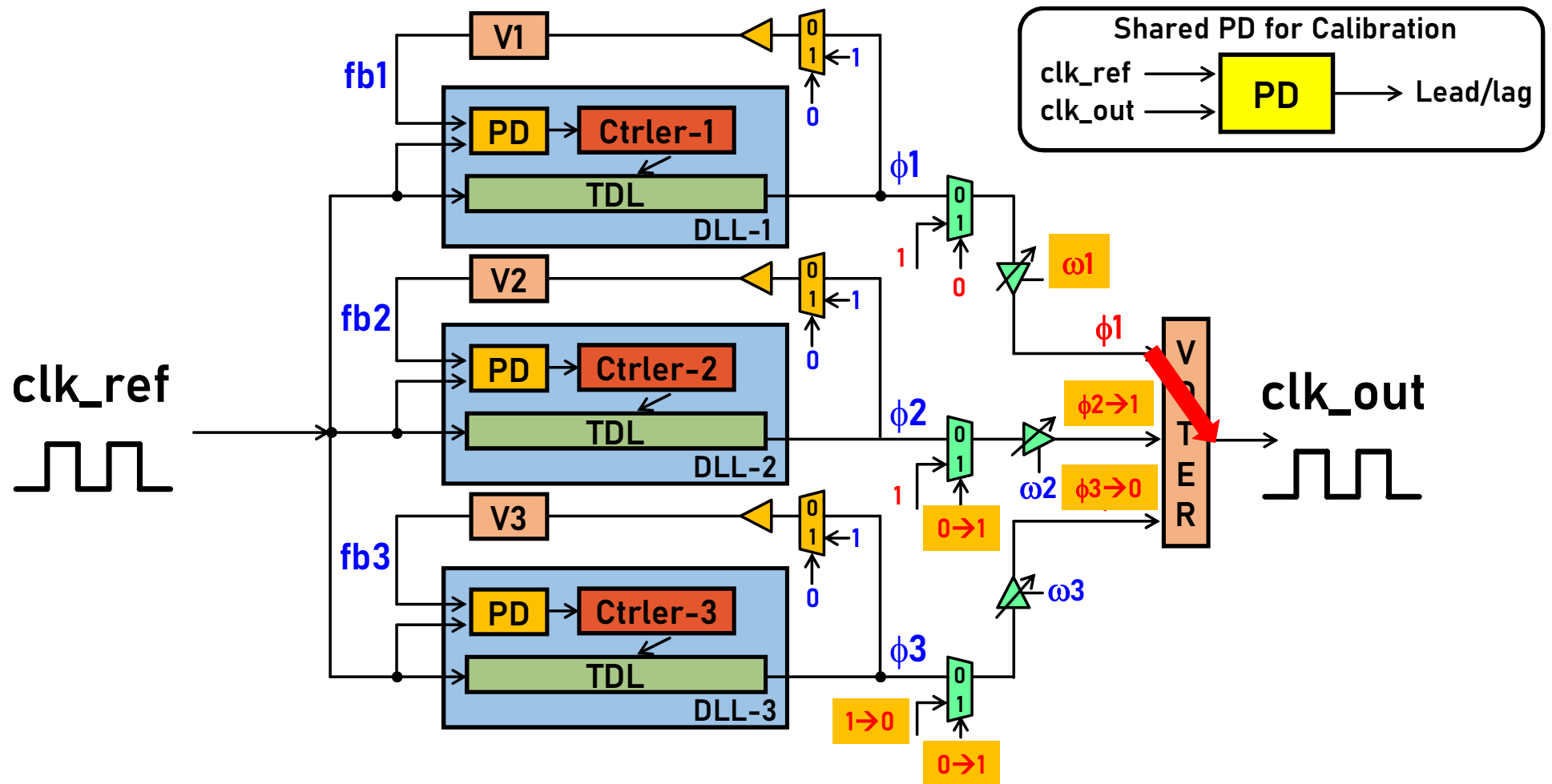
- Delay amount from input to output can be tuned by two level
 - Tunable driving strength – controlled by β -code
 - Tunable output capacitance – controlled by γ -code



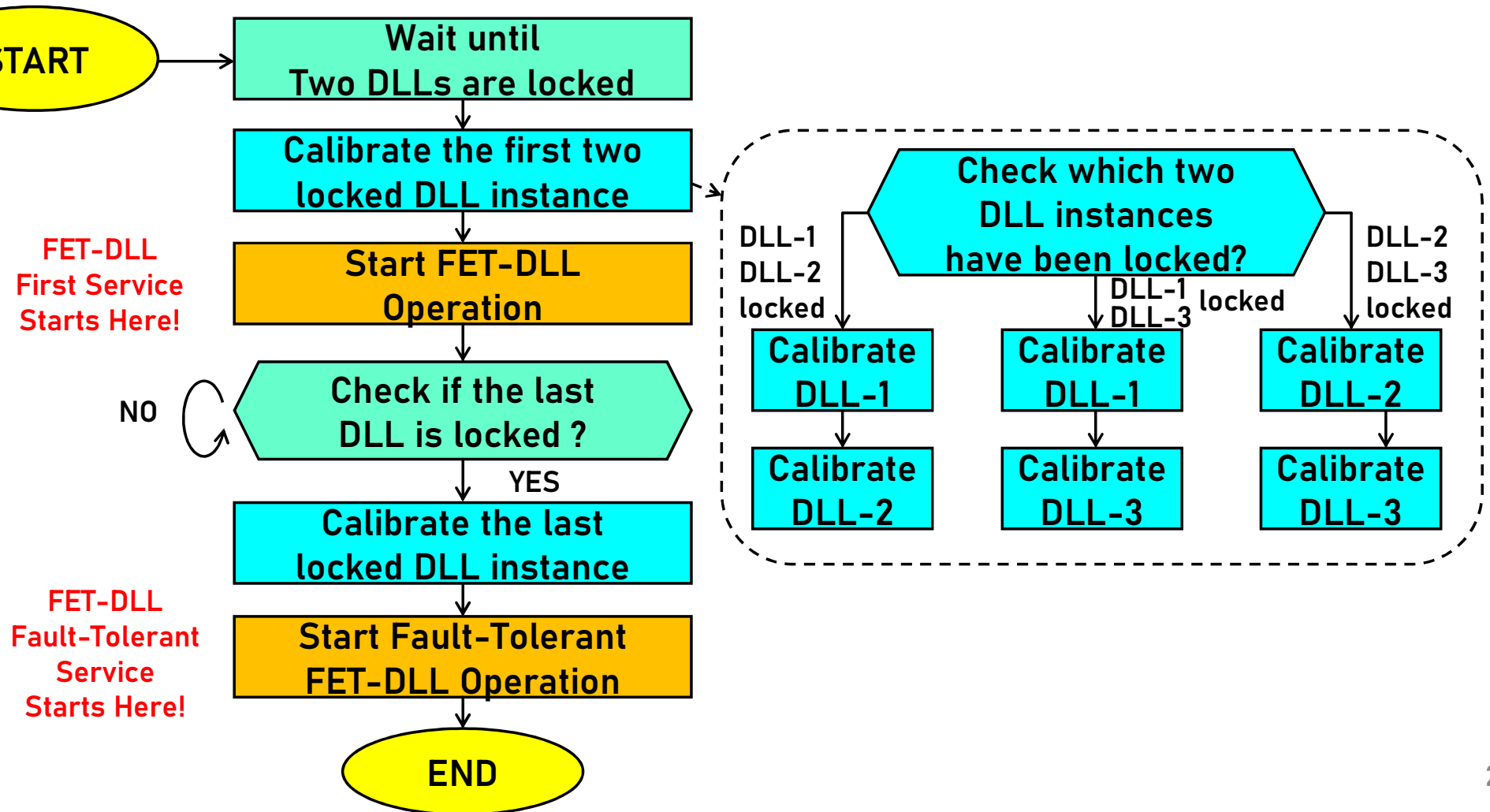
Overall Online Calibration Procedure



Calibration for one DLL At a time



Once one DLL-instance is faulty ...

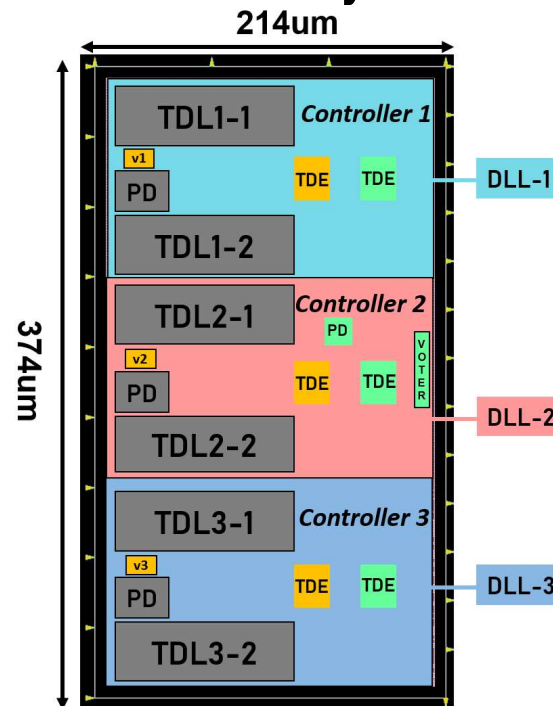


Outline

- Introduction
- Evolution of Fault and Error Tolerant (FET) DLL
- **Experimental Results**
- Conclusion

Layout of FET-DLL with Dynamic Calibration

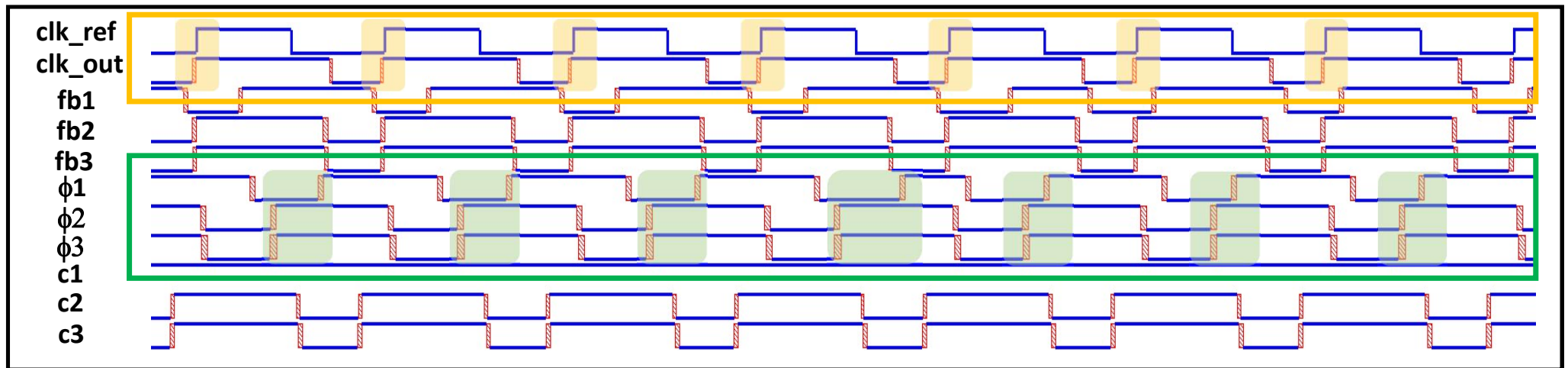
- The FET-DLL design using a 90nm CMOS process
- The primitive DLL instance is a synthesizable one in reference



Z.-H. Zhang, W. Chu, and S.-Y. Huang, "A Ping-Pong Methodology for Boosting the Resilience of Cell-Based Delay-Locked Loop", IEEE Access, Vol. 7, pp. 97928-97937, Aug. 2019.

1st Post-Layout Simulation Scenario

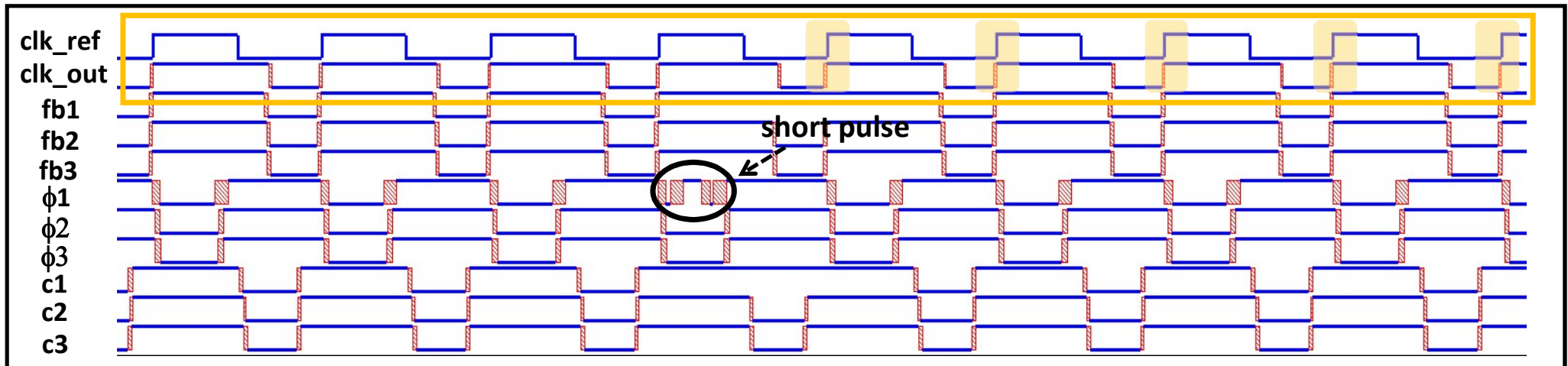
- When there is a random timing drift at $\phi 1$



Output clk_out is not affected !

2nd Post-Layout Simulation Scenario

- When there is a short-pulse error at $\phi 1$




Output clk_out is not affected !

Max. Phase Error Comparison

- The performance of 4 versions of DLL design (5 Corners)

DLL Version	Max. Phase Error (ps)
Primitive DLL (not fault/Error tolerant)	10
FET-DLL with Naïve TMR	130
FET-DLL with Static Timing Correction	20
FET-DLL with Dynamic Timing Calibration	11

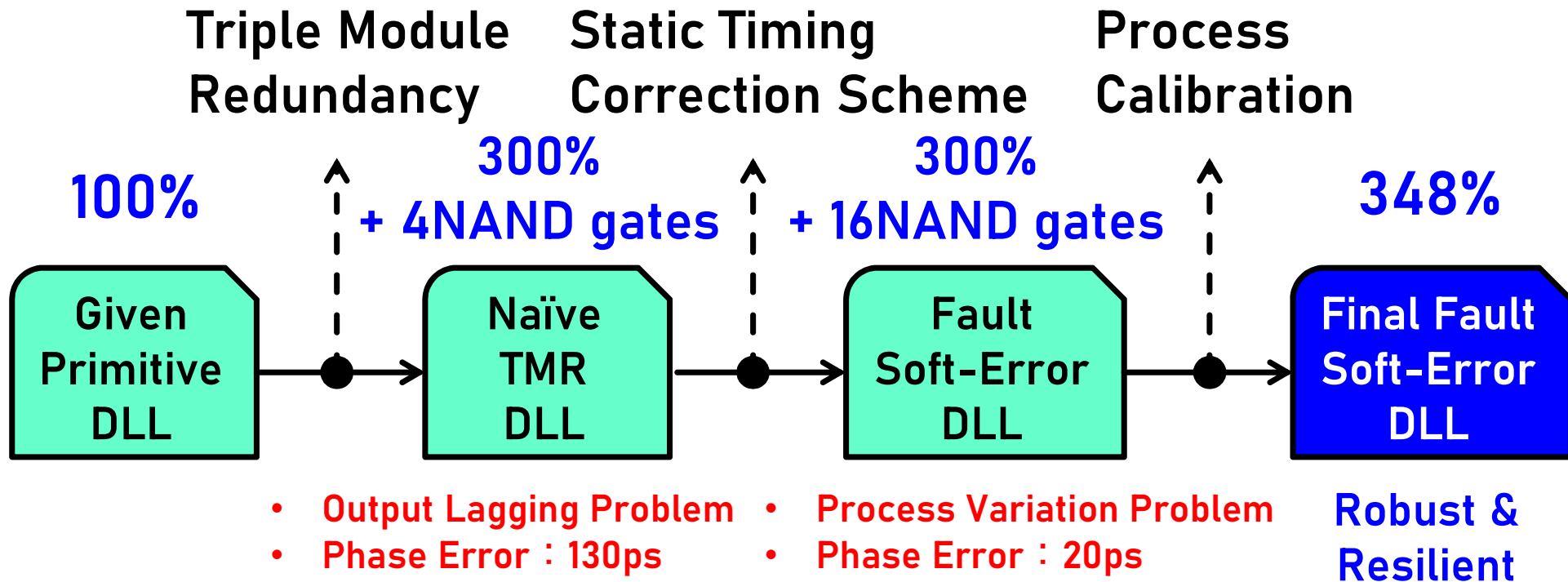


45%

Outline

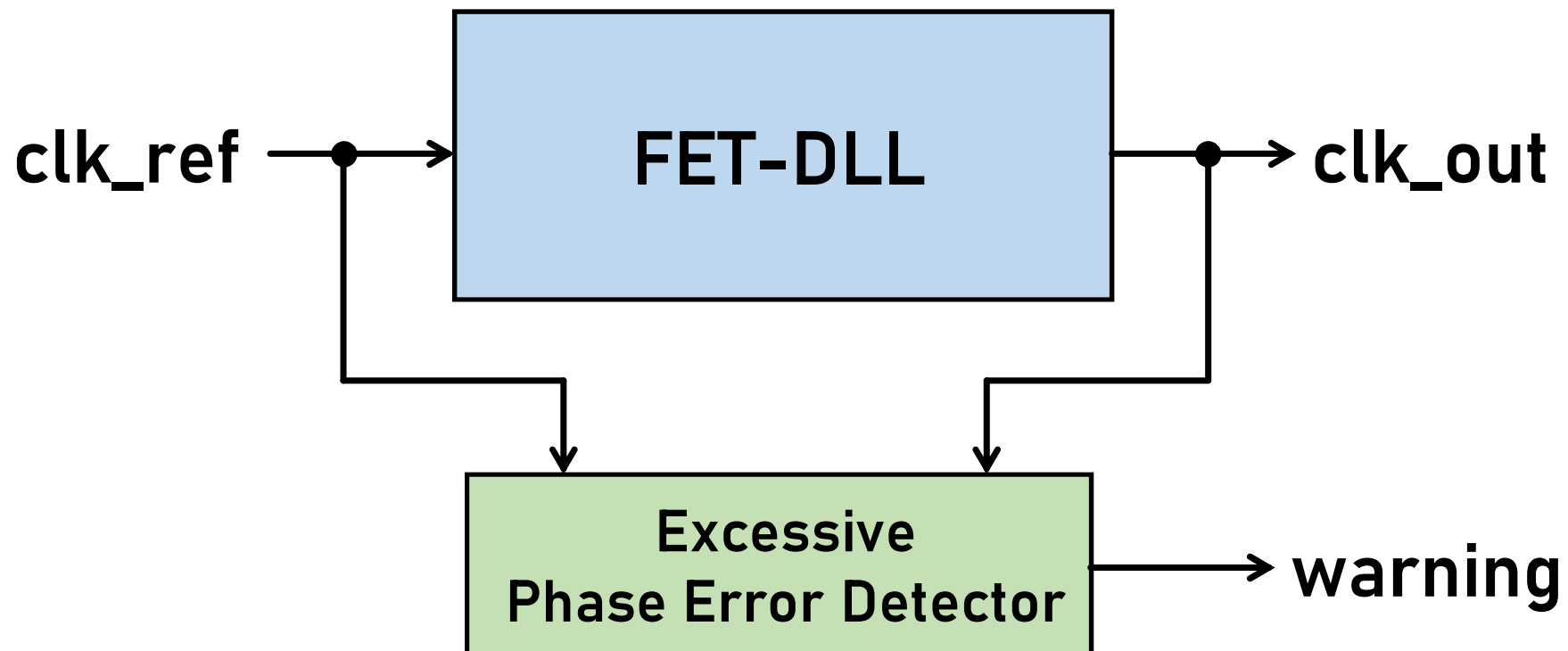
- Introduction
- Evolution of Fault and Error Tolerant (FET) DLL
- Experimental Results
- **Conclusion**

Conclusion

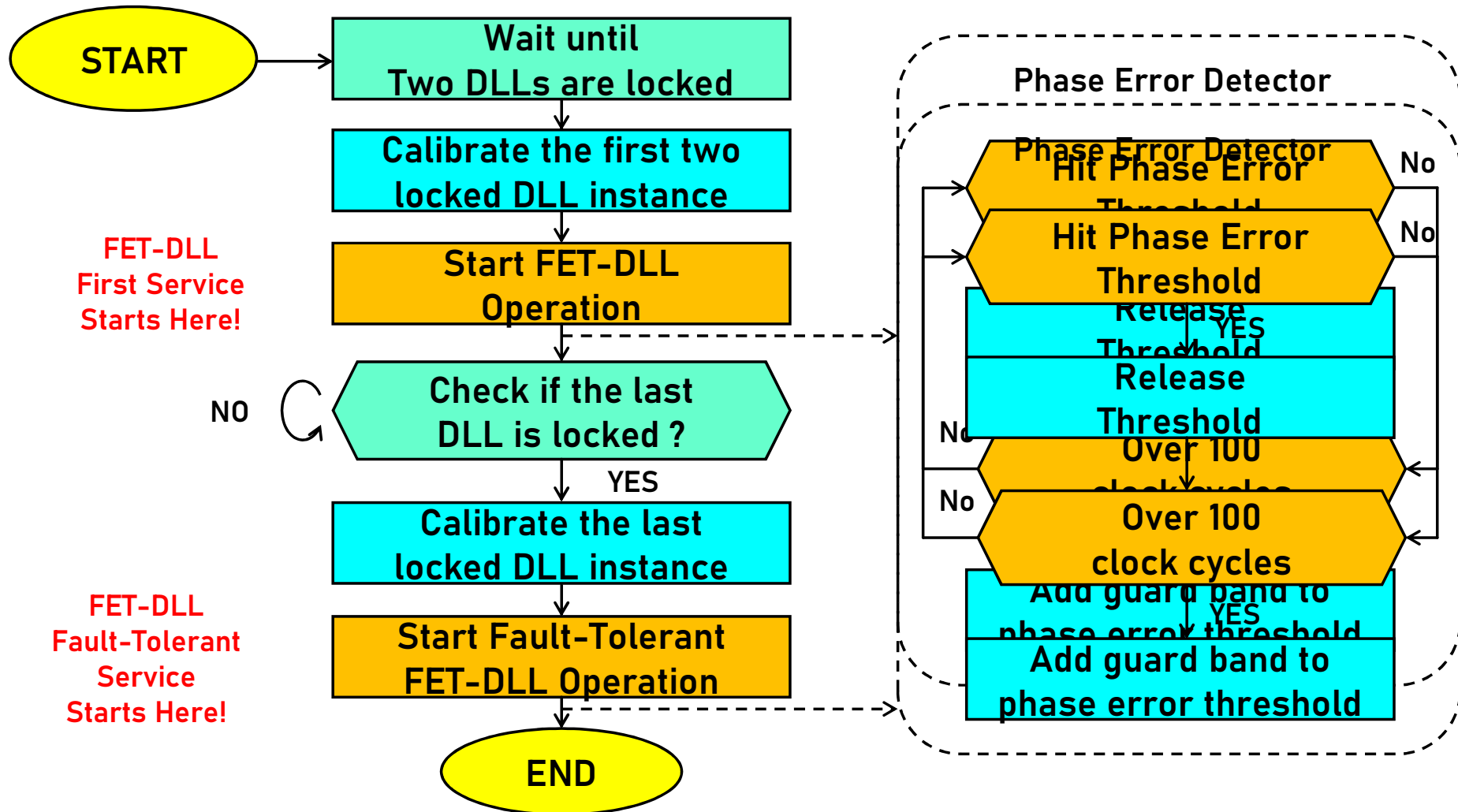


Future Work

- FET-DLL with graceful degradation via a low-cost Excessive Phase Error Detector



Future Work



Thank you !