MIXED SIGNAL MSC CIRCUIT DESIGN

Technische Universität Berlin

A Divider-less Automatic Frequency Calibration for Millimeter-Wave Sub-Sampling Phase-Locked Loop

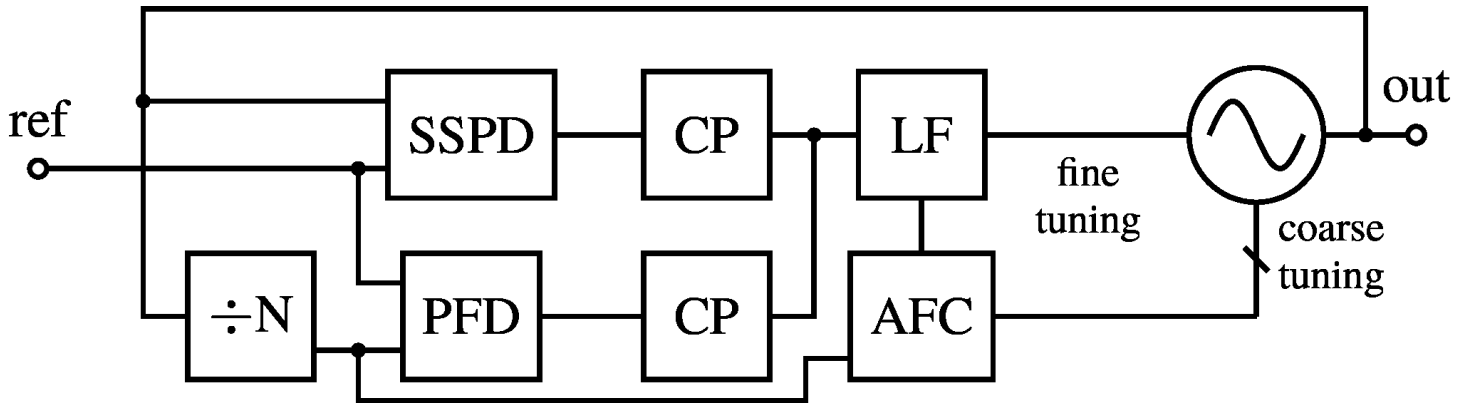
Patrick Kurth, Urs Hecht, Enne Wittenhagen, Friedel Gerfers – Technische Universität Berlin

System Introduction and Problem Statement

- Sub-sampling phase-locked Loop for best jitter performance

  No multiplication of the loop noise by N^2 due to lack of frequency divider

- Sub-sampling phase detector has limited lock-in range and can not distinguish between integer multiples of the reference frequency
- SSPLLs employ secondary charge pump PLL to ensure proper locking

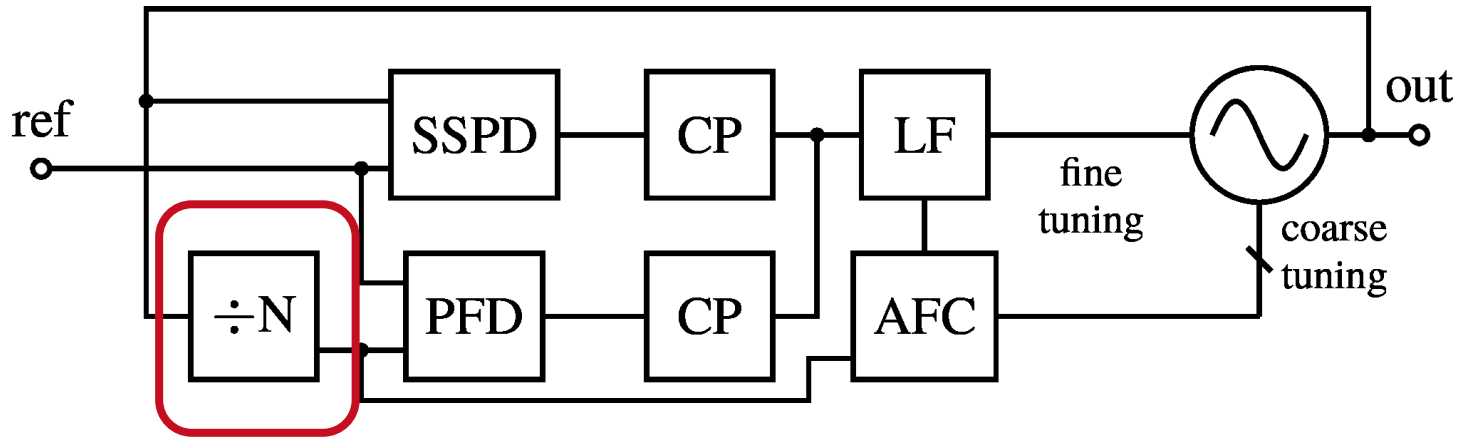


System Introduction and Problem Statement

- Sub-sampling phase-locked Loop for best jitter performance

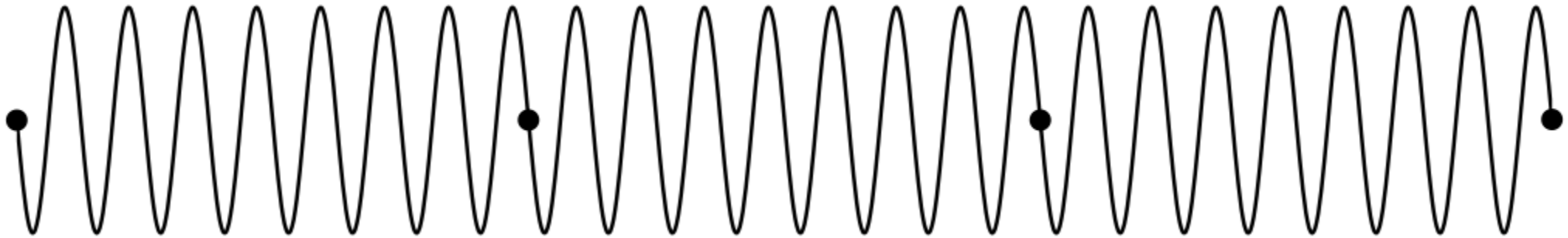
  No multiplication of the loop noise by N^2 due to lack of frequency divider

- Sub-sampling phase detector has limited lock-in range and can not distinguish between integer multiples of the reference frequency
- SSPLLs employ secondary charge pump PLL to ensure proper locking
- Divider for mmw frequencies (usually CML or IL) is power hungry and large in area
- Divider-less approach desirable for high-performance mmw SSPLLs



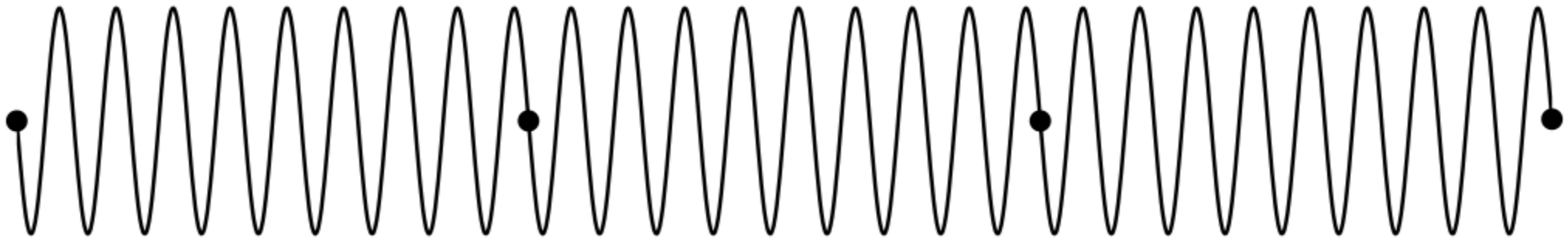
Frequency Information by Delayed Sampling

- SSPLL in (true or false) lock: Phase detector samples at zero crossings
- Example for N = 8 ($f_{\mathrm{out}}/f_{\mathrm{ref}}$)



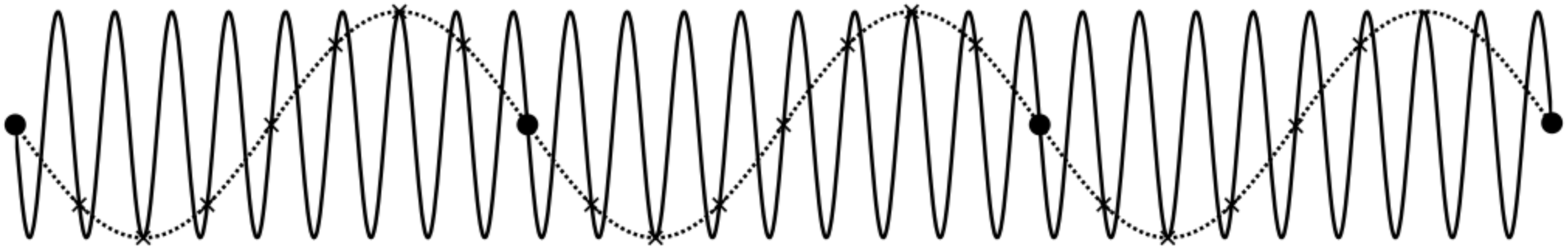
Frequency Information by Delayed Sampling

- SSPLL in (true or false) lock: Phase detector samples at zero crossings
- Example for N = 8 ($f_{\mathrm{out}}/f_{\mathrm{ref}}$)



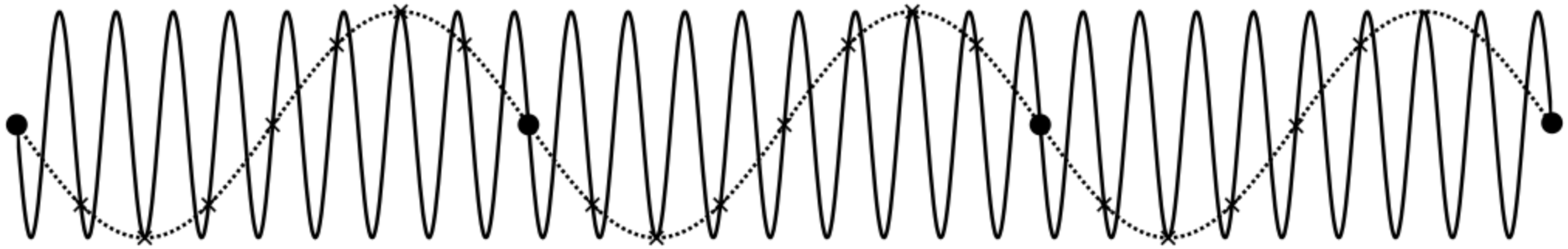
Frequency Information by Delayed Sampling

- SSPLL in (true or false) lock: Phase detector samples at zero crossings
- Example for N = 8 ($f_{\mathrm{out}}/f_{\mathrm{ref}}$)
- More sample points for frequency detection ($N_{\mathrm{aux}} = 7$)



Frequency Information by Delayed Sampling

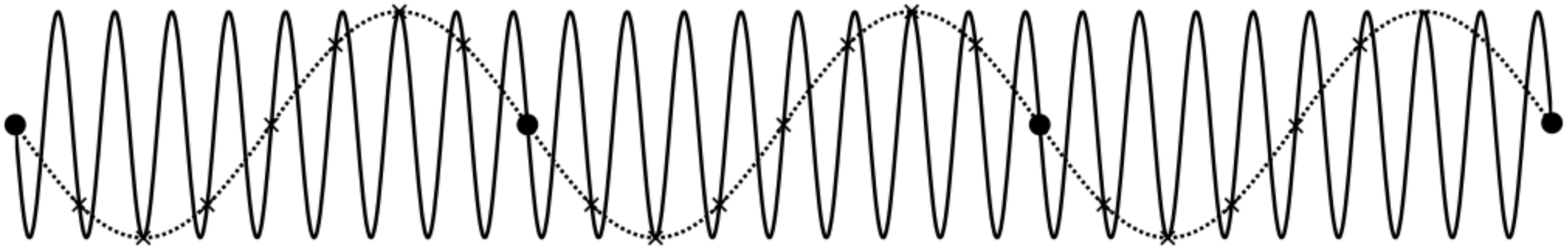
- SSPLL in (true or false) lock: Phase detector samples at zero crossings
- Example for N = 8 ($f_{\mathrm{out}}/f_{\mathrm{ref}}$)
- More sample points for frequency detection ($N_{\mathrm{aux}} = 7$)
- How many additional samplers are needed? -> Condition for detection of false lock:



Frequency Information by Delayed Sampling

- SSPLL in (true or false) lock: Phase detector samples at zero crossings
- Example for N = 8 ($f_{\mathrm{out}}/f_{\mathrm{ref}}$)
- More sample points for frequency detection ($N_{\mathrm{aux}} = 7$)
- How many additional samplers are needed? -> Condition for detection of false lock:

$$\frac{1}{f_{\mathrm{ref}} \cdot (N_{\mathrm{aux}} + 1)} \begin{cases} \neq k \cdot 1/(2 \cdot f_{\mathrm{lock}}) \\ = k \cdot 1/(2 \cdot f_{\mathrm{ref}}) \end{cases} \quad \text{with any } k \text{ and} \quad f_{\mathrm{lock}} = (N \pm i) \cdot f_{\mathrm{ref}}, i \in \left\{1, \ldots, \left\lceil \frac{\Delta f}{f_{\mathrm{ref}}} \right\rceil \right\}$$

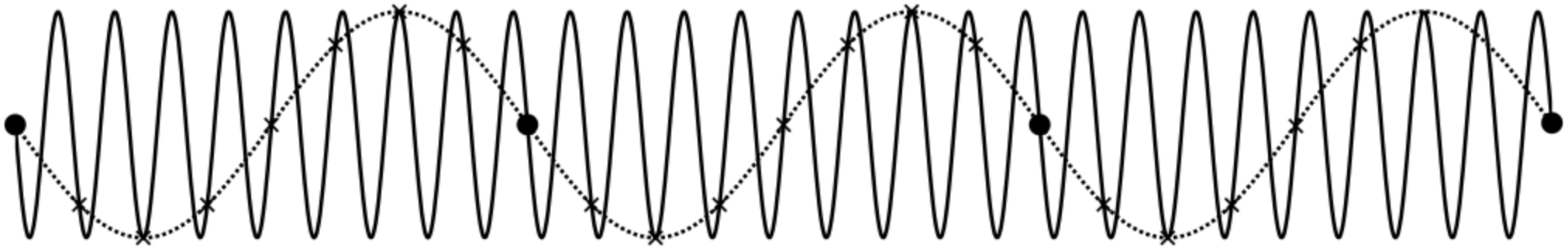


Frequency Information by Delayed Sampling

- SSPLL in (true or false) lock: Phase detector samples at zero crossings
- Example for N = 8 ($f_{\mathrm{out}}/f_{\mathrm{ref}}$)
- More sample points for frequency detection ($N_{\mathrm{aux}} = 7$)
- How many additional samplers are needed? -> Condition for detection of false lock:

$$\frac{1}{f_{\mathrm{ref}} \cdot (N_{\mathrm{aux}} + 1)} \begin{cases} \neq k \cdot 1/(2 \cdot f_{\mathrm{lock}}) \\ = k \cdot 1/(2 \cdot f_{\mathrm{ref}}) \end{cases} \quad \text{with any } k \text{ and} \quad f_{\mathrm{lock}} = (N \pm i) \cdot f_{\mathrm{ref}}, i \in \left\{1, \dots, \left\lceil \frac{\Delta f}{f_{\mathrm{ref}}} \right\rceil \right\}$$

$$\Leftrightarrow \frac{2(N \pm i)}{N_{aux} + 1} \begin{cases} \neq k \ \text{ for } \ i \neq 0 \\ = k \ \text{ for } \ i = 0 \end{cases}$$



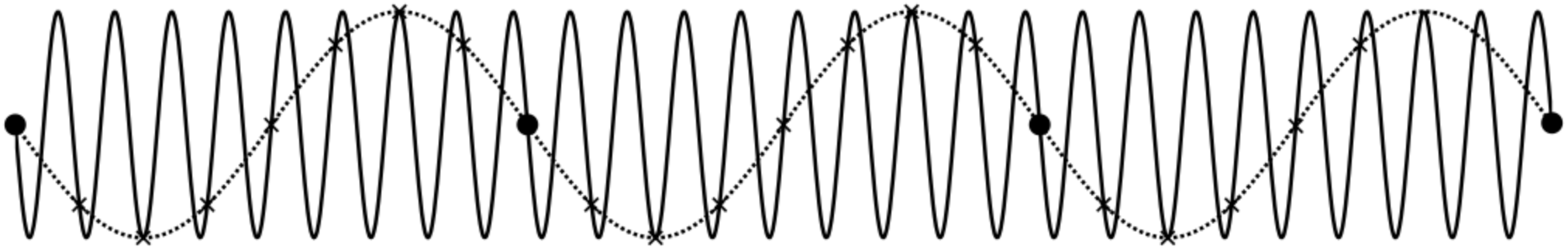
Frequency Information by Delayed Sampling

- SSPLL in (true or false) lock: Phase detector samples at zero crossings
- Example for N = 8 ($f_{\mathrm{out}}/f_{\mathrm{ref}}$)
- More sample points for frequency detection ($N_{\mathrm{aux}} = 7$)
- How many additional samplers are needed? -> Condition for detection of false lock:

$$\frac{1}{f_{\mathrm{ref}} \cdot (N_{\mathrm{aux}} + 1)} \begin{cases} \neq k \cdot 1/(2 \cdot f_{\mathrm{lock}}) \\ = k \cdot 1/(2 \cdot f_{\mathrm{ref}}) \end{cases} \quad \text{with any } k \text{ and} \quad f_{\mathrm{lock}} = (N \pm i) \cdot f_{\mathrm{ref}}, i \in \left\{1, \ldots, \left\lceil \frac{\Delta f}{f_{\mathrm{ref}}} \right\rceil \right\}$$

$$\Leftrightarrow \frac{2(N \pm i)}{N_{aux} + 1} \begin{cases} \neq k \ \text{ for } \ i \neq 0 \\ = k \ \text{ for } \ i = 0 \end{cases}$$

-> solve numerically for N_{aux}



Detection Serialization

- Number of required auxiliary sampler N_{aux} needs to be calculated from the condition for false lock
- Depends on the ratio of the output and input frequency of the PLL and the tuning range of the oscillator

Detection Serialization

- Number of required auxiliary sampler N_{aux} needs to be calculated from the condition for false lock
- Depends on the ratio of the output and input frequency of the PLL and the tuning range of the oscillator
- For $N = 64$ and a tuning range of around 20 % $N_{\mathrm{aux}} = 15$!
- This is not feasible in terms of capacitive load in mm-wave SSPLLs

Detection Serialization

- Number of required auxiliary sampler N_{aux} needs to be calculated from the condition for false lock
- Depends on the ratio of the output and input frequency of the PLL and the tuning range of the oscillator
- For $N = 64$ and a tuning range of around 20 % $N_{\mathrm{aux}} = 15$!
- This is not feasible in terms of capacitive load in mm-wave SSPLLs
- Luckily, parallel detection is not needed → Samples can be saved and digitized sequentially and stored in a shift register
- This greatly simplifies the analog interface (only one sampler and comparator)

Detection Serialization

- Number of required auxiliary sampler N_{aux} needs to be calculated from the condition for false lock
- Depends on the ratio of the output and input frequency of the PLL and the tuning range of the oscillator
- For $N = 64$ and a tuning range of around 20 % $N_{\mathrm{aux}} = 15$!
- This is not feasible in terms of capacitive load in mm-wave SSPLLs
- Luckily, parallel detection is not needed → Samples can be saved and digitized sequentially and stored in a shift register
- This greatly simplifies the analog interface (only one sampler and comparator)
- Slightly higher detection time, but this is negligible

Clock Generation

- Delay PLL reference with delay-locked loop
- Needed minimum resolution: $T_{\mathrm{ref}}/(N_{\mathrm{aux}}+1) \rightarrow$ DLL with at least $N_{\mathrm{aux}}+1$ delay elements
- Combine $N_{\mathrm{aux}}+1$ delayed clocks into one auxiliary clock signal by sequentially cycling through all delayed clocks $\rightarrow$ slightly higher frequency than reference

Clock Generation

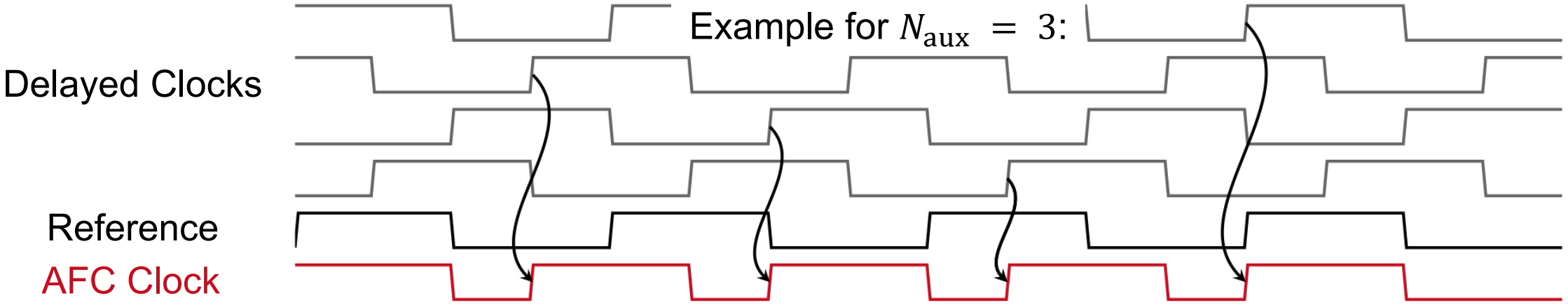
- Delay PLL reference with delay-locked loop
- Needed minimum resolution: $T_{\mathrm{ref}}/(N_{\mathrm{aux}}+1) \rightarrow$ DLL with at least $N_{\mathrm{aux}}+1$ delay elements
- Combine $N_{\mathrm{aux}}+1$ delayed clocks into one auxiliary clock signal by sequentially cycling through all delayed clocks $\rightarrow$ slightly higher frequency than reference
- Tricky timing issue: update edge selector at rising or falling edge? (risk of glitches)

Clock Generation

- Delay PLL reference with delay-locked loop
- Needed minimum resolution: $T_{\mathrm{ref}}/(N_{\mathrm{aux}}+1) \rightarrow$ DLL with at least $N_{\mathrm{aux}}+1$ delay elements
- Combine $N_{\mathrm{aux}}+1$ delayed clocks into one auxiliary clock signal by sequentially cycling through all delayed clocks $\rightarrow$ slightly higher frequency than reference
- Tricky timing issue: update edge selector at rising or falling edge? (risk of glitches)
- Solve glitch problem by cycling backwards through all clock phases
- This reorders the samples, which can easily be fixed in the frequency decoder

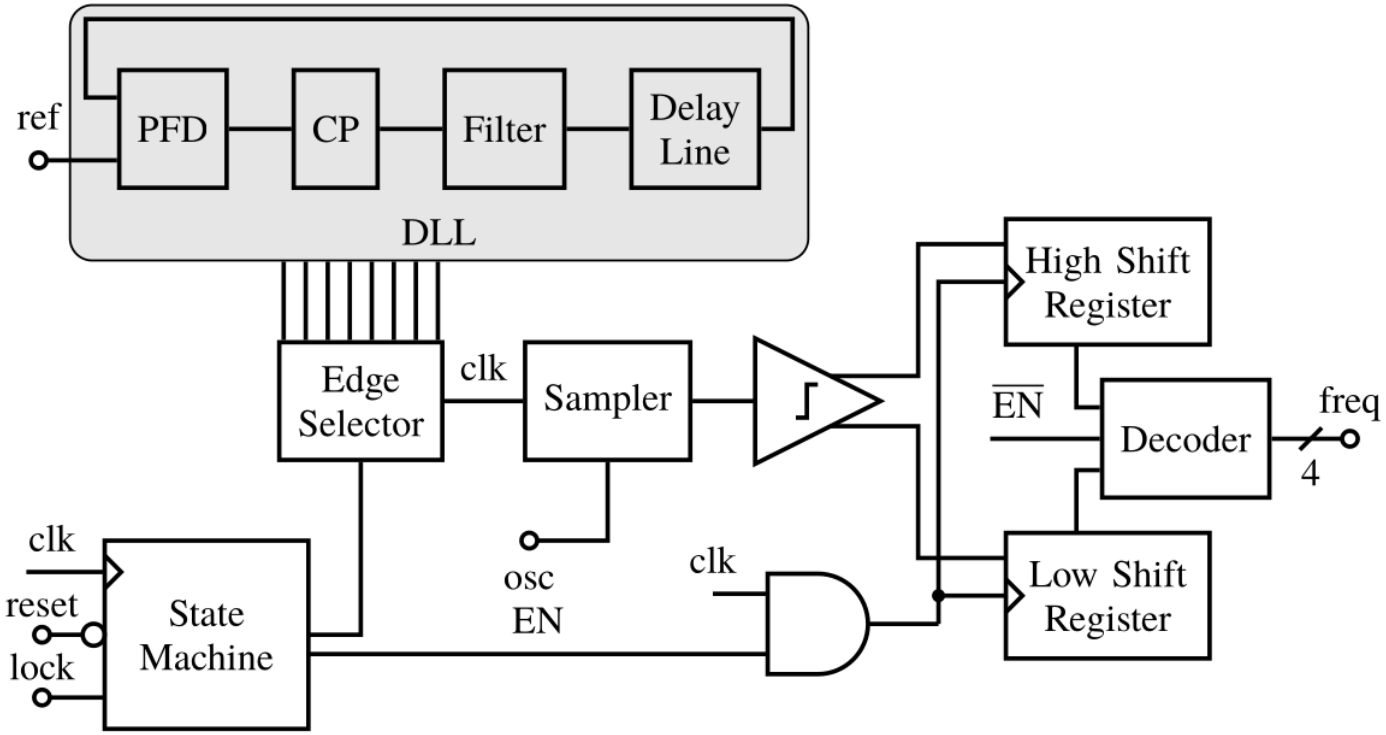
Clock Generation

- Delay PLL reference with delay-locked loop
- Needed minimum resolution: $T_{\mathrm{ref}}/(N_{\mathrm{aux}}+1) \rightarrow$ DLL with at least $N_{\mathrm{aux}}+1$ delay elements
- Combine $N_{\mathrm{aux}}+1$ delayed clocks into one auxiliary clock signal by sequentially cycling through all delayed clocks $\rightarrow$ slightly higher frequency than reference
- Tricky timing issue: update edge selector at rising or falling edge? (risk of glitches)
- Solve glitch problem by cycling backwards through all clock phases
- This reorders the samples, which can easily be fixed in the frequency decoder



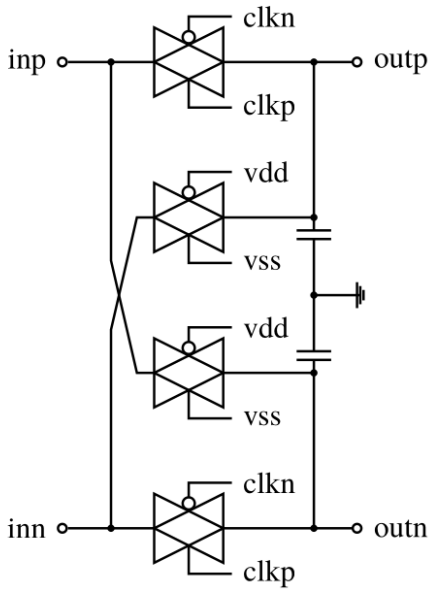
System Overview

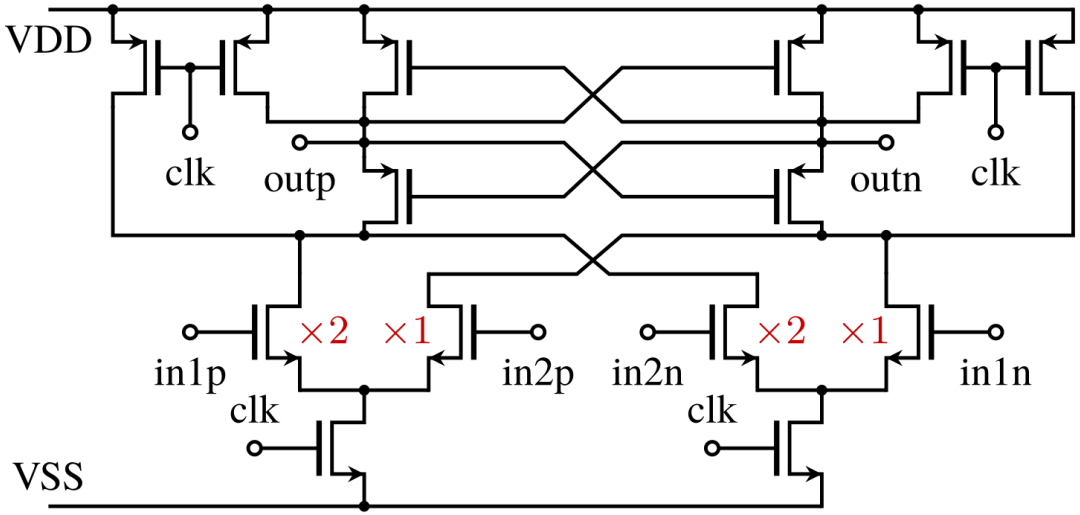
- Directly sample VCO without divider to determine frequency state
- Convert to digital representation (high, low, zero -> HHH0LLL)
- Decode frequency state and update coarse oscillator tuning
- Delayed clocks generated by a DLL and an edge selector
- High-speed sampler for VCO
- “2-Bit-ADC”: Window comparator
- Shift register and frequency state decoder
- Finite state machine controls AFC



Sampler and Comparator

- Fully-differential sampler: transmission gate as switch
- Cross-coupling between positive and negative nodes to cancel signal feedthrough during the hold phase
- Window comparator with two fully-differential comparators (strongARM latches)
- Symmetric design cancels charge injection, voltage droop etc.
- Deliberate offset (approximately 150 mV) by transistor sizing for improved noise margin and timing errors





Frequency Decoder

- Decode frequency state: map quantized samples from shift register to lock state of PLL
- Update coarse tuning of the oscillator
- Simple algorithm is sufficient: increase coarse frequency setting when lock frequency is too low, otherwise decrease

Frequency Decoder

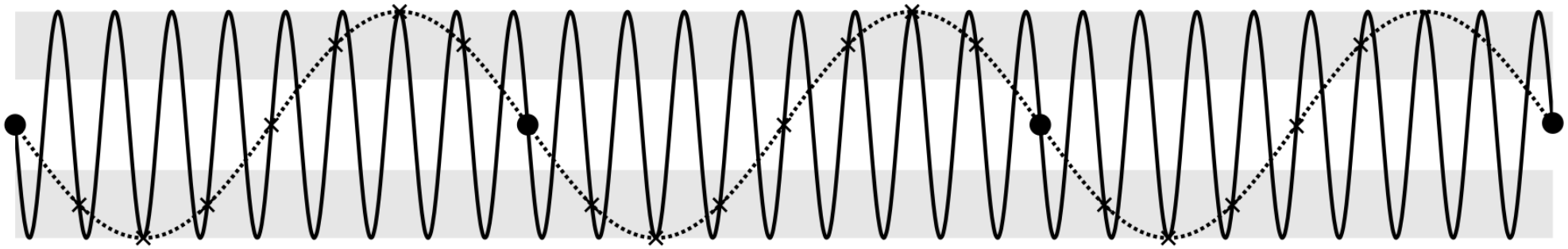
- Decode frequency state: map quantized samples from shift register to lock state of PLL
- Update coarse tuning of the oscillator
- Simple algorithm is sufficient: increase coarse frequency setting when lock frequency is too low, otherwise decrease
- Charge pump polarity (negative, positive) of SSPLL is important
  - Negative feedback in both cases
  - Frequency decoder mapping is inverse for negative CP polarity
- Decoder is implemented as lookup table (simple combinatorial logic)

Frequency Decoder

- Decode frequency state: map quantized samples from shift register to lock state of PLL
- Update coarse tuning of the oscillator
- Simple algorithm is sufficient: increase coarse frequency setting when lock frequency is too low, otherwise decrease
- Charge pump polarity (negative, positive) of SSPLL is important

  Negative feedback in both cases

  Frequency decoder mapping is inverse for negative CP polarity
- Decoder is implemented as lookup table (simple combinatorial logic)
- Map patterns such as 0LLL0HHH0LLL0HHH to frequency state (e.g. -1)



Lock Detector

- The AFC can only run when PLL is in lock -
  > Divider-less lock detector needed

Lock Detector

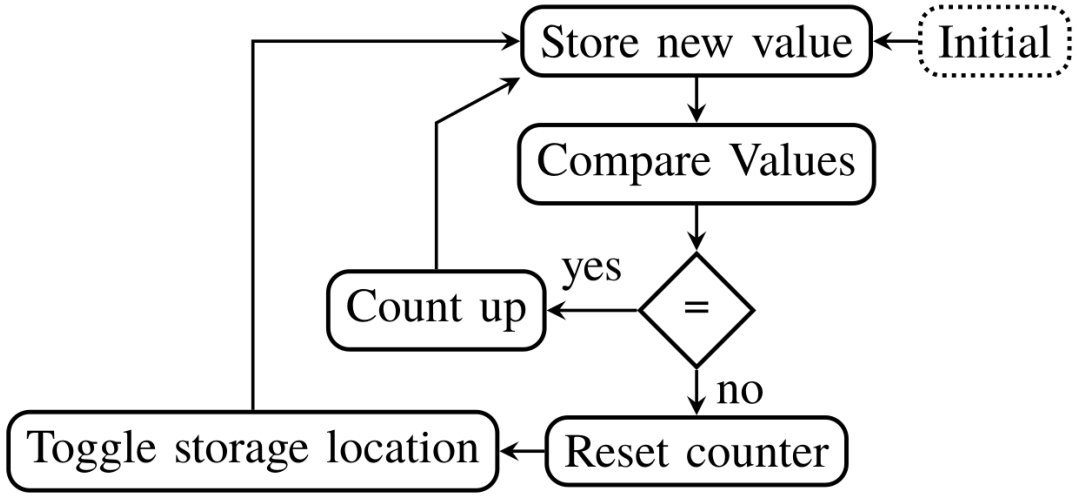
- The AFC can only run when PLL is in lock -> Divider-less lock detector needed
- Compare consecutive samples after the main SSPD (two storage locations)

Lock Detector

- The AFC can only run when PLL is in lock -> Divider-less lock detector needed
- Compare consecutive samples after the main SSPD (two storage locations)
- SSPLL is in lock when many samples are equal (internal counter saturates)

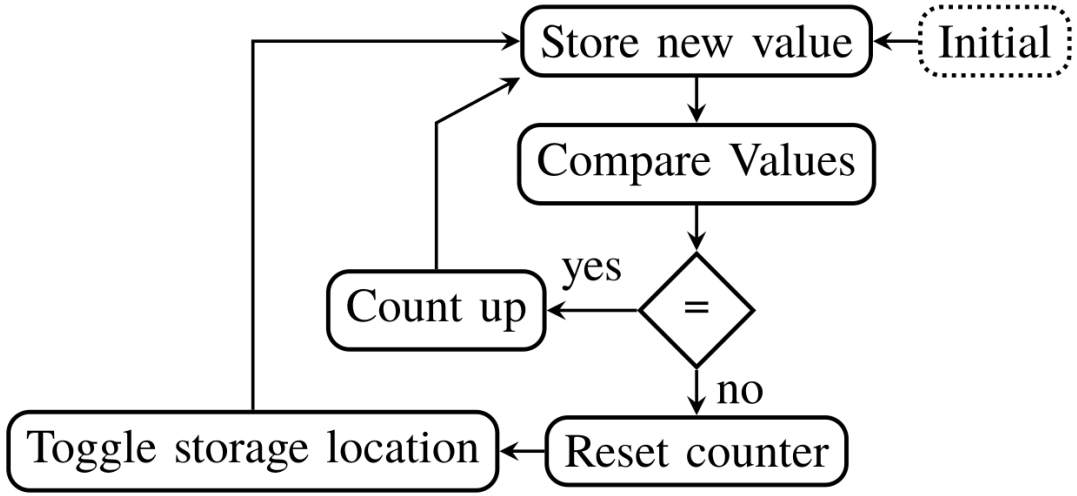
Lock Detector

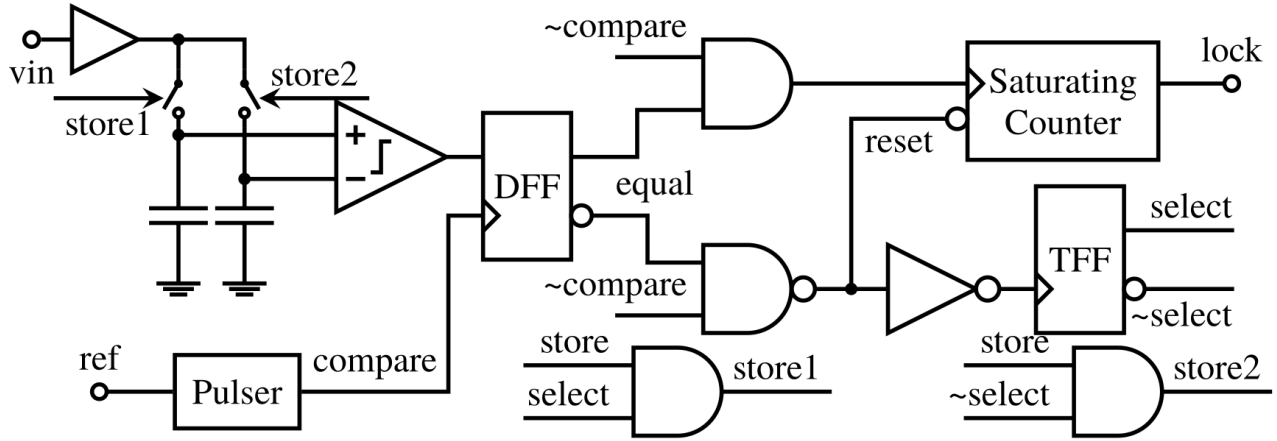
- The AFC can only run when PLL is in lock -> Divider-less lock detector needed
- Compare consecutive samples after the main SSPD (two storage locations)
- SSPLL is in lock when many samples are equal (internal counter saturates)



Lock Detector

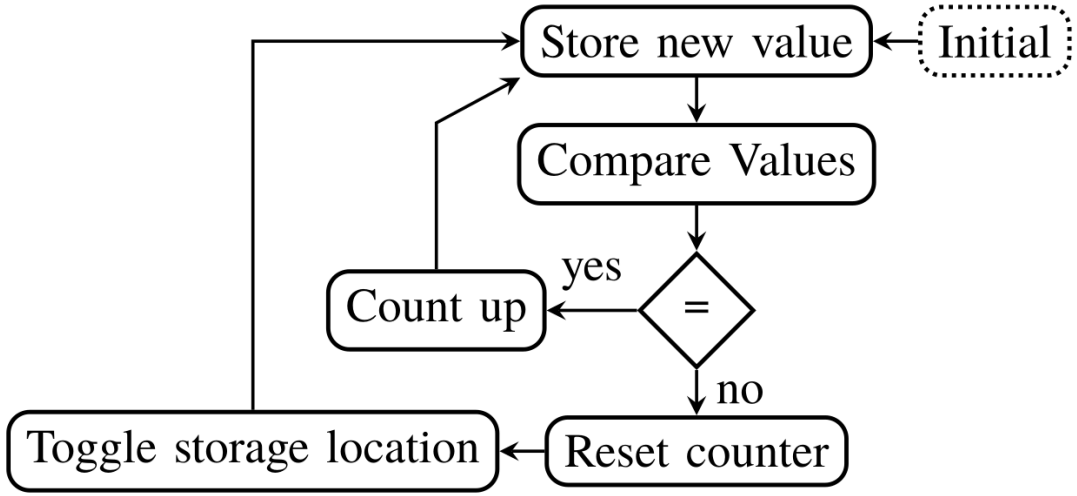
- The AFC can only run when PLL is in lock -> Divider-less lock detector needed
- Compare consecutive samples after the main SSPD (two storage locations)
- SSPLL is in lock when many samples are equal

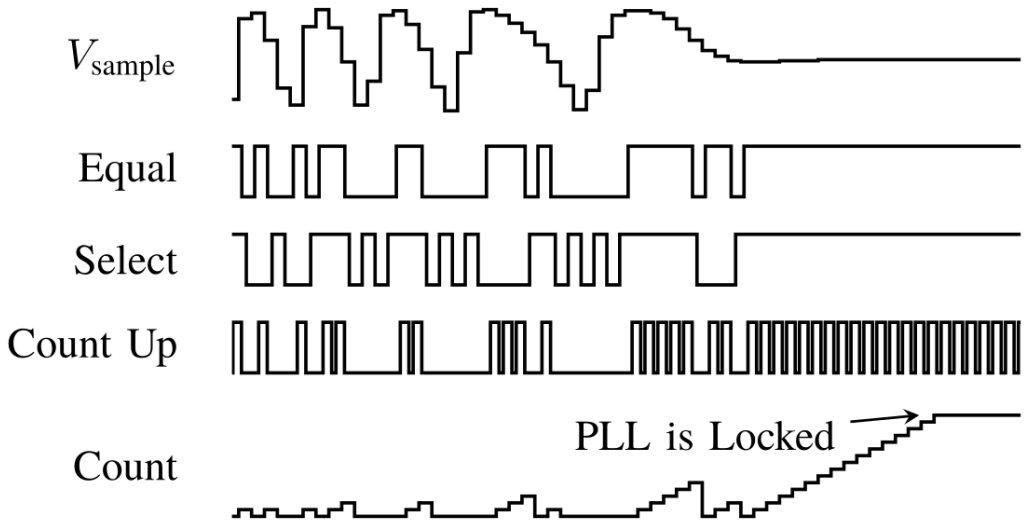


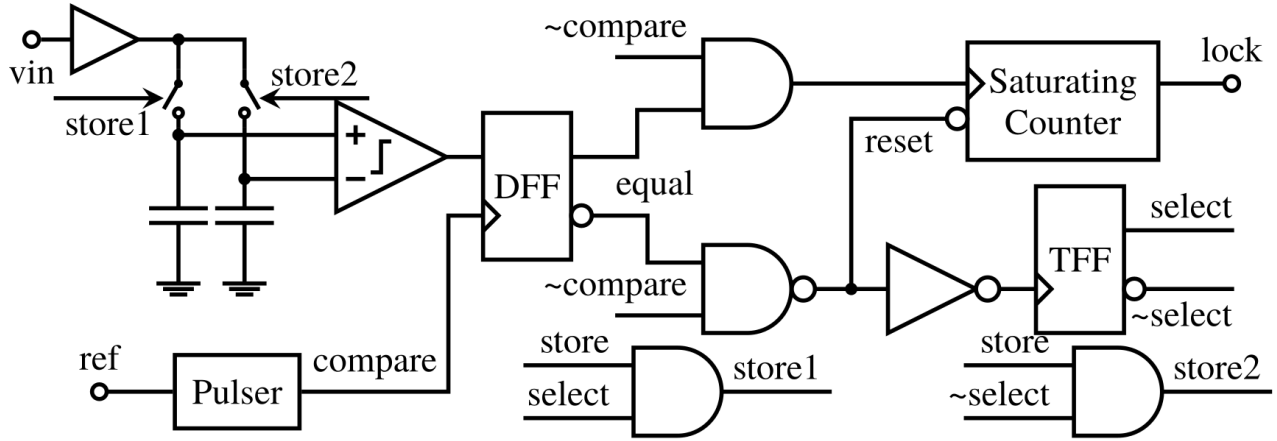


Lock Detector

- The AFC can only run when PLL is in lock -> Divider-less lock detector needed
- Compare consecutive samples after the main SSPD (two storage locations)
- SSPLL is in lock when many samples are equal

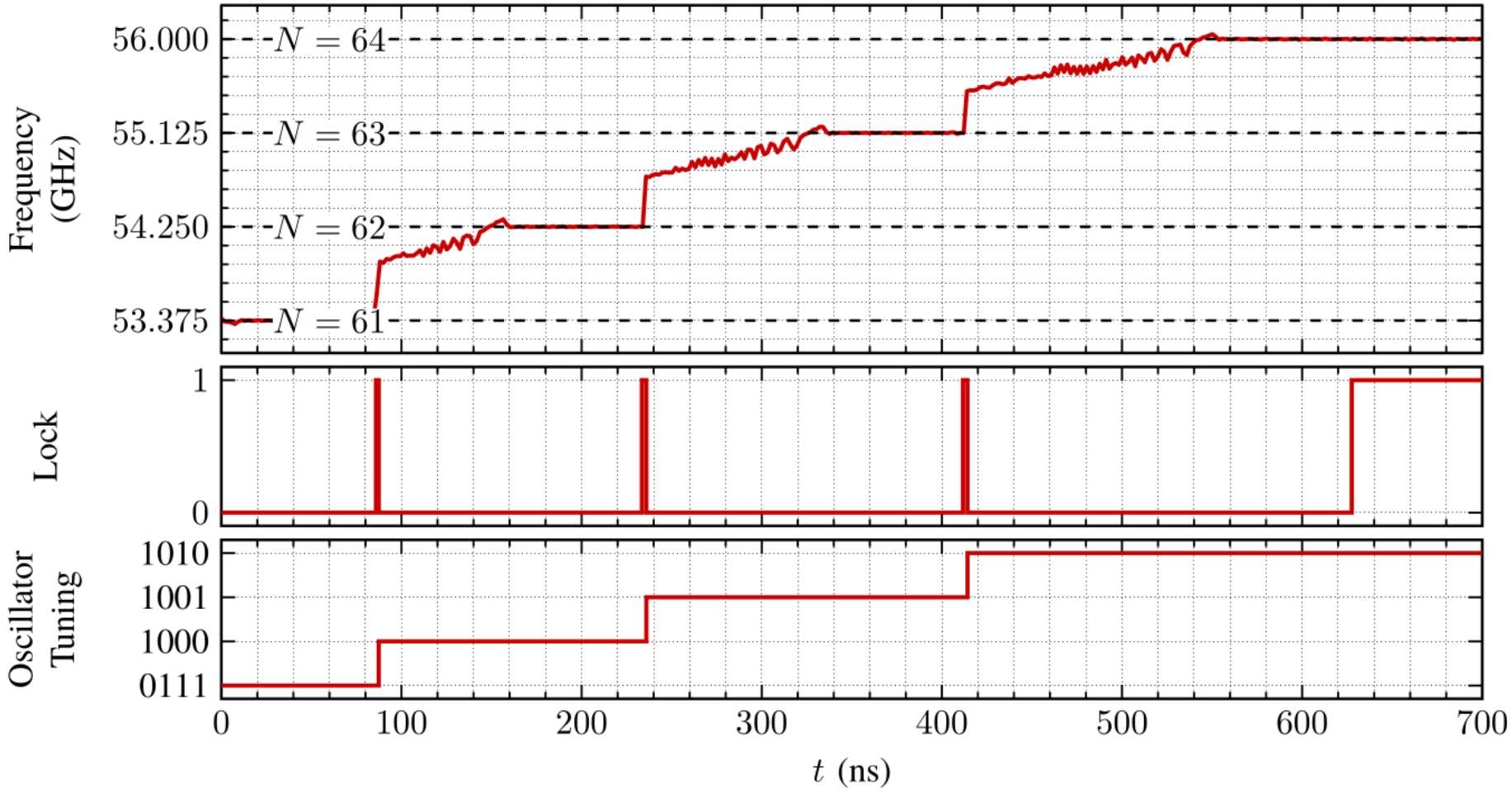






Automatic Frequency Calibration – Simulation Results

- SSPLL with $f_{\mathrm{ref}} = 875\ \mathrm{MHz}$, $f_{\mathrm{out}} = 56\ \mathrm{GHz}$ $(N = 64)$



Comparison with State-of-the-Art

Work	[1]	[7]	[8]	This work
Technology (nm)	40	180	65	22 (FDSOI)
Frequency (GHz)	14	2.3	40.5	56
Reference Frequency (GHz)	200	48	100	875
Methodology	FLL	FLL + Counter	Secondary PLL	Sub-sampling with DLL
FLL/AFC DC Power (mW)	1.5	46.7$^{\$}$	4.59	1.1
Divider	$\div$ 70	$\div$ 2	none	none
AFC/FLL Area (μm^2)	8600*	35 600*	33 600*	2000$^{\#}$

$^{\$}$ Total PLL power consumption * Estimated from die photograph $^{\#}$ Pre-layout estimation

[1] Z. Zhang *et al.*, "A 0.65-V 12-16-GHz Sub-Sampling PLL with 56.4-fsrms Integrated Jitter and -256.4-dB FoM", *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1665—1683, 2020.

[7] W. Chang *et al.*, "A Fractional-N Divider-Less Phase-Locked Loop With a Subsampling Phase Detector," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 12, pp. 2964–2975, 2014.

[8] Wang, Hao *et al.*, "Low-Power and Low-Noise Millimeter-Wave SSPLL With Subsampling Lock Detector for Automatic Dividerless Frequency Acquisition," *IEEE Transactions on Microwave Theory and Techniques*, vol. 69, no. 1, pp. 469–481, 2021.

Conclusion

- Divider-less automatic frequency calibration for sub-sampling phase locked loops
- Suitable for highest frequencies, enables truly divider-less millimeter-wave SSPLLs
- Mainly digital implementation -> synthesizable and scalable with technology advances
- System can be extremely low power and area
- Analog front-end: Sampler, Comparator and DLL; Only sampler needs to have high speed/bandwidth
- Serialized sampling greatly reduces complexity of analog front-end
- Implementation independent of internal PLL architecture (analog, digital, discrete-time) as well as VCO type (LC, ring, etc.)
- AFC can run continuously without significantly impairing power consumption

Thank you for your Attention.
Questions?