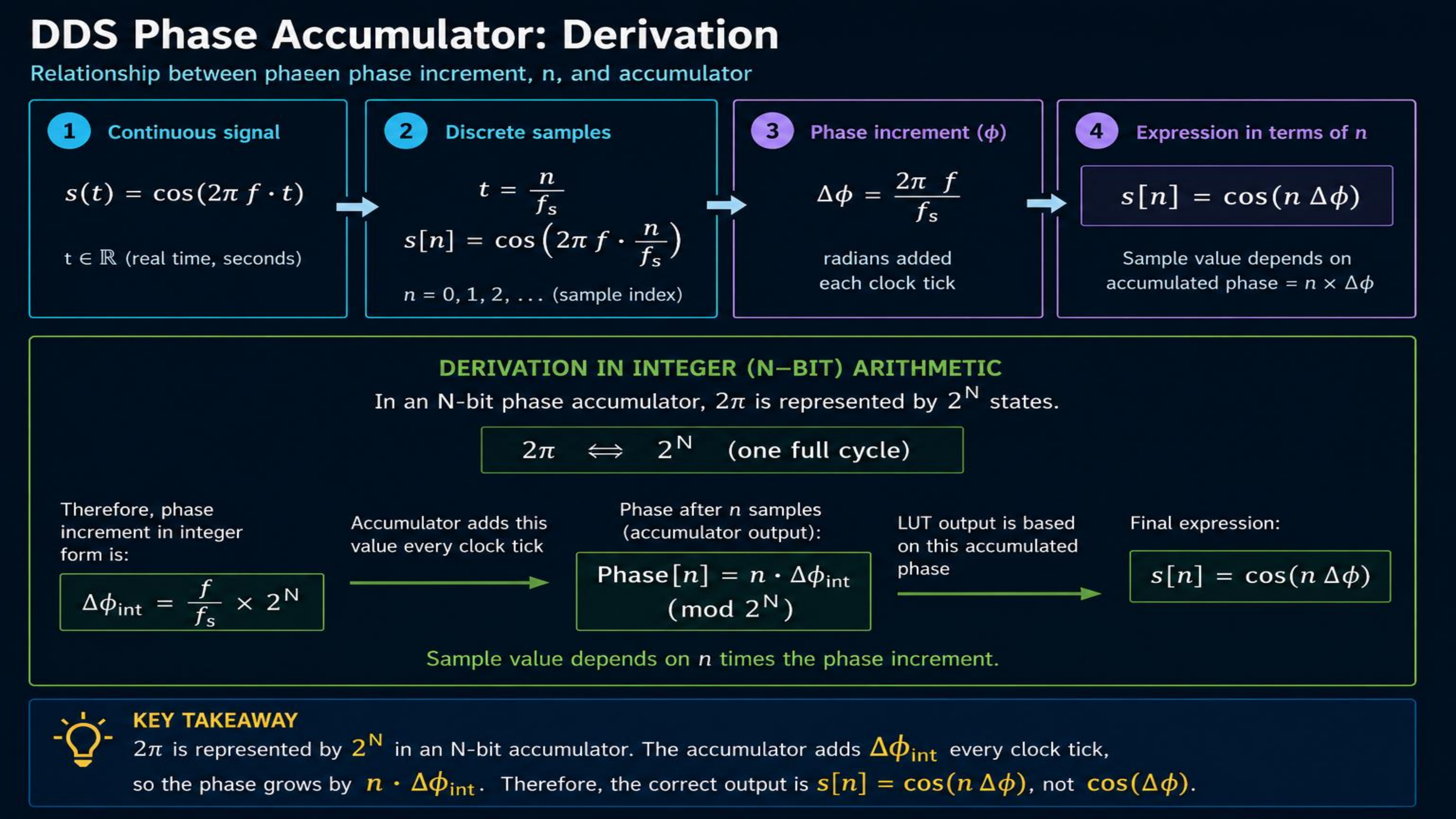
# Embedded
# DSP Design

*From signal theory to silicon — explained clearly.*



**Farid. M, Png**


DDS Phase Accumulator: Derivation
Relationship between phaeen phase increment, n, and accumulator
1 Continuous signal
$s(t) = \cos(2\pi f \cdot t)$
$t \in \mathbb{R}$ (real time, seconds)
2 Discrete samples
$t = \frac{n}{f_s}$
$s[n] = \cos\left(2\pi f \cdot \frac{n}{f_s}\right)$
$n = 0, 1, 2, \ldots$ (sample index)
3 Phase increment ($\phi$)
$\Delta\phi = \frac{2\pi f}{f_s}$
radians added each clock tick
4 Expression in terms of $n$
$s[n] = \cos(n\,\Delta\phi)$
Sample value depends on accumulated phase = $n \times \Delta\phi$
DERIVATION IN INTEGER (N–BIT) ARITHMETIC
In an N-bit phase accumulator, $2\pi$ is represented by $2^N$ states.
$2\pi \iff 2^N$ (one full cycle)
Therefore, phase increment in integer form is:
$\Delta\phi_{int} = \frac{f}{f_s} \times 2^N$
Accumulator adds this value every clock tick
Phase after $n$ samples (accumulator output):
$\text{Phase}[n] = n \cdot \Delta\phi_{int} \pmod{2^N}$
LUT output is based on this accumulated phase
Final expression:
$s[n] = \cos(n\,\Delta\phi)$
Sample value depends on $n$ times the phase increment.
KEY TAKEAWAY
$2\pi$ is represented by $2^N$ in an N-bit accumulator. The accumulator adds $\Delta\phi_{int}$ every clock tick,
so the phase grows by $n \cdot \Delta\phi_{int}$. Therefore, the correct output is $s[n] = \cos(n\,\Delta\phi)$, not $\cos(\Delta\phi)$.

# DDS Phase Accumulator: Derivation

From continuous signal → integer hardware register

### 1 Continuous signal

$s(t) = \cos(2\pi \cdot f \cdot t)$

$t \in \mathbb{R}$ (real time, seconds)

### 2 Discrete samples

$t = n / f_s$

$s[n] = \cos(2\pi \cdot f \cdot n / f_s)$

### 3 Phase increment (ϕ)

$\Delta\phi = 2\pi \cdot f / f_s$

radians added each clock tick

### 4 Integer mapping

Map $2\pi \rightarrow 2^N$ states

$\Delta\phi^{int} = (f / f_s) \times 2^N$

**Result**

$$\Delta\phi\_int = ( f\_desired / f\_system ) \times 2^N$$

← add this integer to accumulator every clock tick

**N-bit accumulator**

Wraps at $2^N \equiv 2\pi$.
Overflow = one full cycle.

**Bigger Δϕ_int**

Faster table traversal.
Higher output frequency.

**Frequency resolution**

Finest step = $f\_system / 2^N$.
More bits → finer tuning.

$f_s$ = system clock rate | N = accumulator bit-width | Δϕ_int is an integer — no floating point needed in hardware

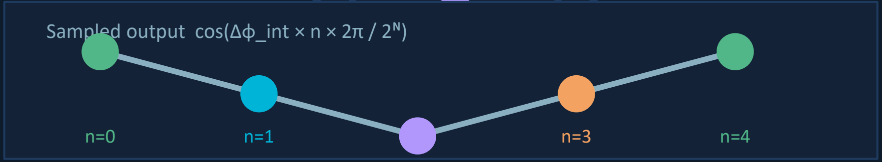
# Worked Example

$f_s = 4$ Hz • $f^{desired} = 1$ Hz • N = 4 bits ($2^4 = 16$ accumulator states)

$$\Delta\phi\_int = (f\_d/f\_s)\times 2^N = (1/4)\times 2^4 = 0.25\times 16 = \mathbf{4}$$ ← add 4 to accumulator every clock tick


| Tick (n) | Acc value | Phase | Output cos |
|---|---|---|---|
| 0 | 0 | 0° | cos(0°) = 1.00 |
| 1 | 4 | 90° | cos(90°) = 0.00 |
| 2 | 8 | 180° | cos(180°) = −1.00 |
| 3 | 12 | 270° | cos(270°) = 0.00 |
| 4 | 0 (wrap) | 360°=0° | cos(0°) = 1.00 ✓ |

*After 4 ticks: accumulator wraps back to 0 → exactly 1 full cycle → output = 1 Hz ✓*



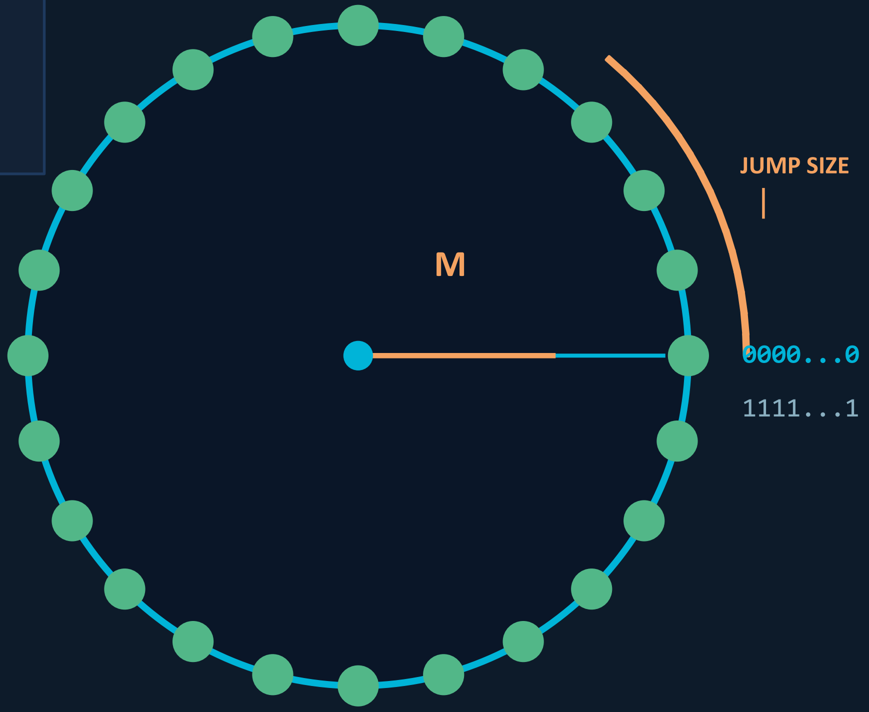

**Key insight:** The wheel has 16 stops. We jump 4 stops per tick.
4 jumps × 4 stops = 16 stops = 1 full revolution = 1 Hz

N=4 bits → 16 accumulator states | Δϕ_int = 4 | Every 4 clock ticks the accumulator completes one full 16-step revolution

# Digital Phase Wheel

DDS accumulator state-space — N-bit phase circle with M-step jumps

$$f_o = \frac{M \times f_s}{2^N}$$




| *n* | NUMBER OF POINTS |
|---|---|
| 8 | 256 |
| 12 | 4,096 |
| 16 | 65,535 |
| 20 | 1,048,576 |
| 24 | 16,777,216 |
| 28 | 268,435,456 |
| 32 | 4,294,967,296 |
| 48 | 281,474,976,710,656 |

**N** = accumulator width (bits) **fs** = system clock

**M = phase increment (jump size)**

Each clock tick the accumulator advances M steps around the $2^N$-point wheel. One full revolution = one output cycle.

# References

- https://embeddeddesign.org/frequency-sweehttps://embeddeddesign.org/
- https://github.com/faridfma/FIR_25KHz_BW_DDS_In/tree/master