Sine/Cosine using CORDIC Algorithm

Prof. Kris Gaj

Gaurav Doshi, Hiren Shah

Outlines

- Introduction
- Basic Idea
- CORDIC Principles
- Hardware Implementation
- FPGA & ASIC Results
- Conclusion

Introduction

- CORDIC (COordinate Rotation DIgital Computer)
- Introduced in 1959 by Jack E. Volder
- Efficient to compute sin, cos, tan, sinh, cosh, tanh
- Its an Hardware Efficient Algorithm
- Iterative Algorithm for Circular Rotation
- No Multiplication
- Delay/Hardware cost comparable to division or square rooting.

Why CORDIC ?

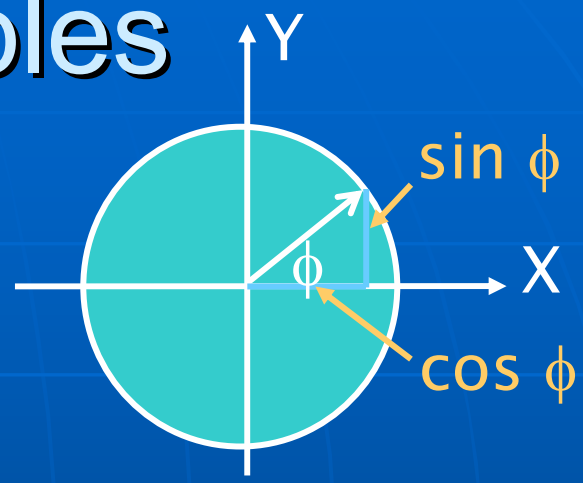
- How to evaluate trigonometric functions?
  - Table lookup
  - Polynomial approximations
  - CORDIC
- Compared to other approaches, CORDIC is a clear winner when :
  - Hardware Multiplier is unavailable (eg. microcontroller)
  - You want to save the gates required to implement (eg. FPGA)

Basic Ideas

- Embedding of elementary function evaluation as a generalized rotation operation.
- Decompose rotation operation into successive basic rotations.
- Each basic rotation can be realized with shift-and-add arithmetic operations.

CORDIC Principles

- Basic idea
  - Rotate (1,0) by ϕ degrees to get (x,y): x=cos(ϕ), y=sin(ϕ)



- Rotation of any (x,y) vector:

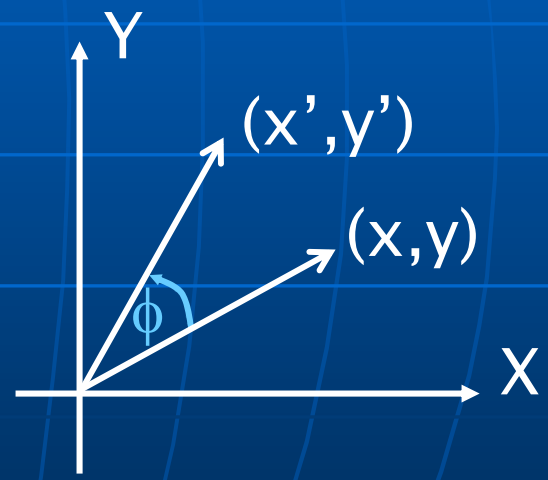
$$x' = x.\cos(\phi) - y.\sin(\phi)$$

$$y' = y.\cos(\phi) + x.\sin(\phi)$$

- Rearrange as:

$$x' = \cos(\phi).[x - y.\tan(\phi)]$$

$$y' = \cos(\phi).[y + x.\tan(\phi)]$$



Note: $\frac{\sin(\phi)}{\cos(\phi)} = \tan(\phi)$

Key Idea

$$\begin{aligned} x' &= \cos(\phi)(x - y\tan(\phi)) \\ y' &= \cos(\phi)(y + x\tan(\phi)). \end{aligned}$$

Can compute rotation φ in steps where each step is of size

$$\tan(\phi) = \pm 2^{-i}.$$

Expansion Vector K

- K = $\Pi \cos\varphi$ depends on rotation angle $\varphi_1, \varphi_2, \varphi_3 \ldots\ldots \varphi_n$
- Since same angles are rotated always K is a constant that can be pre-computed
- K = 1.646760258121

Iterative rotations

$$x_{i+1} = K_i(x_i - (y_i d_i 2^{-i}))$$
$$y_{i+1} = K_i(y_i + (x_i d_i 2^{-i})).$$

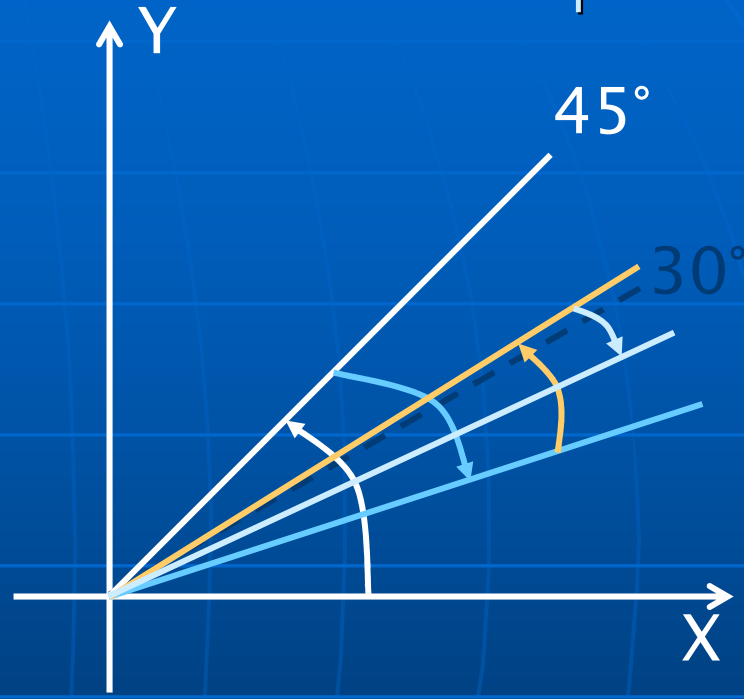
- **di decision (rotation mode)**
- Zi is introduced to keep track of the angle that has been rotated (z0 = φ)

$$z_{i+1} = z_i - d_i \tan^{-1}(2^{-i})$$

- **di = -1 if $z_i < 0$**
  = 1 otherwise

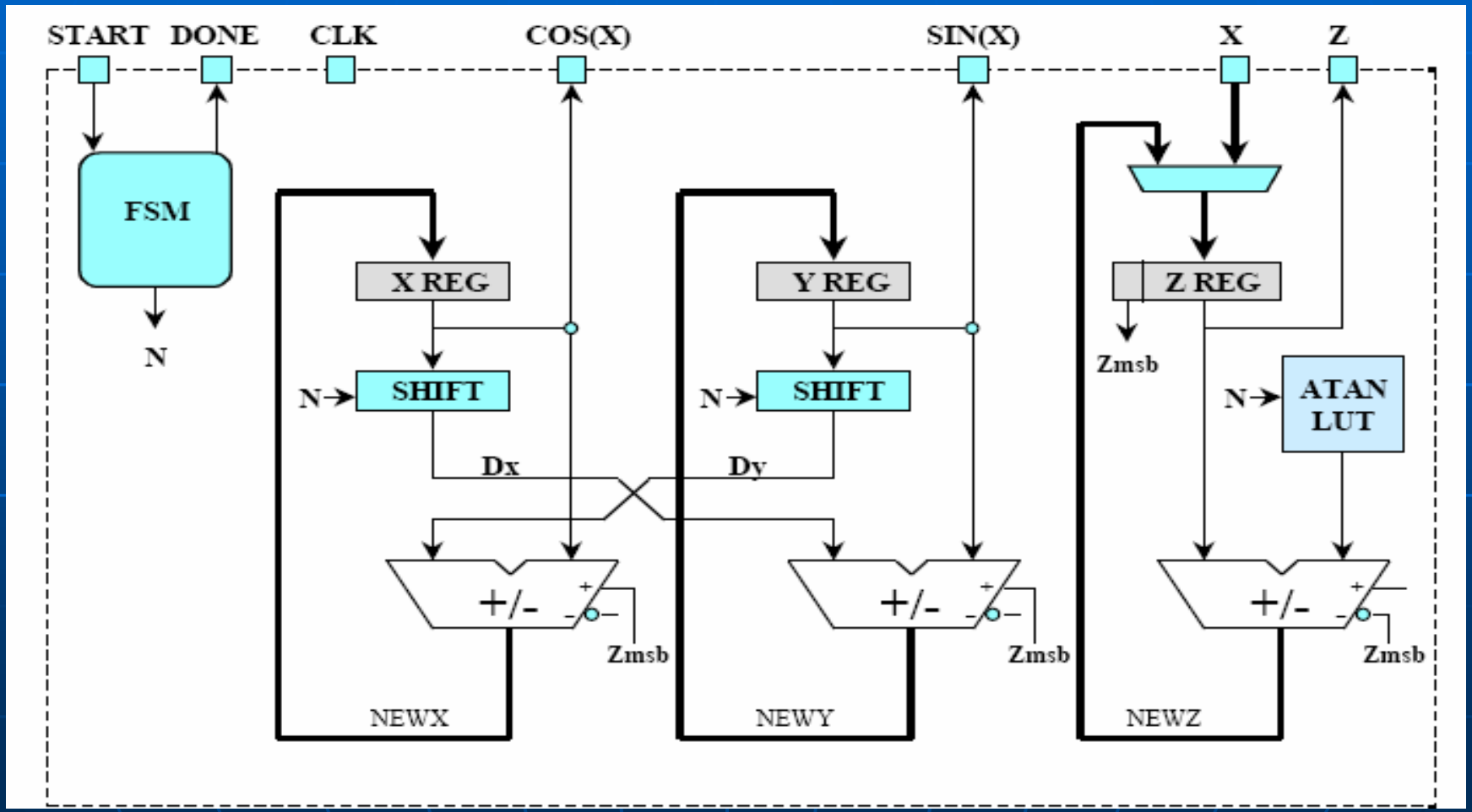
Example: Rewriting Angles in Terms of α_i

- Find α_i such that $\tan(\alpha_i)=2^{-i}$: (or, $\alpha_i=\tan^{-1}(2^{-i})$)
- Example: $\phi=30.0°$
  - Start with α_0 = 45.0 (> 30.0)
  - 45.0 – 26.6 = 18.4 (< 30.0)
  - 18.4 + 14.0 = 32.4 (> 30.0)
  - 32.4 – 7.1 = 25.3 (< 30.0)
  - 25.3 + 3.6 = 28.9 (< 30.0)
  - 28.9 + 1.8 = 30.7 (> 30.0)
  - . . .



- ϕ = 30.0
  $\approx$ 45.0 – 26.6 + 14.0 – 7.1 + 3.6
  + 1.8 – 0.9 + 0.4 – 0.2 + 0.1
  = 30.1

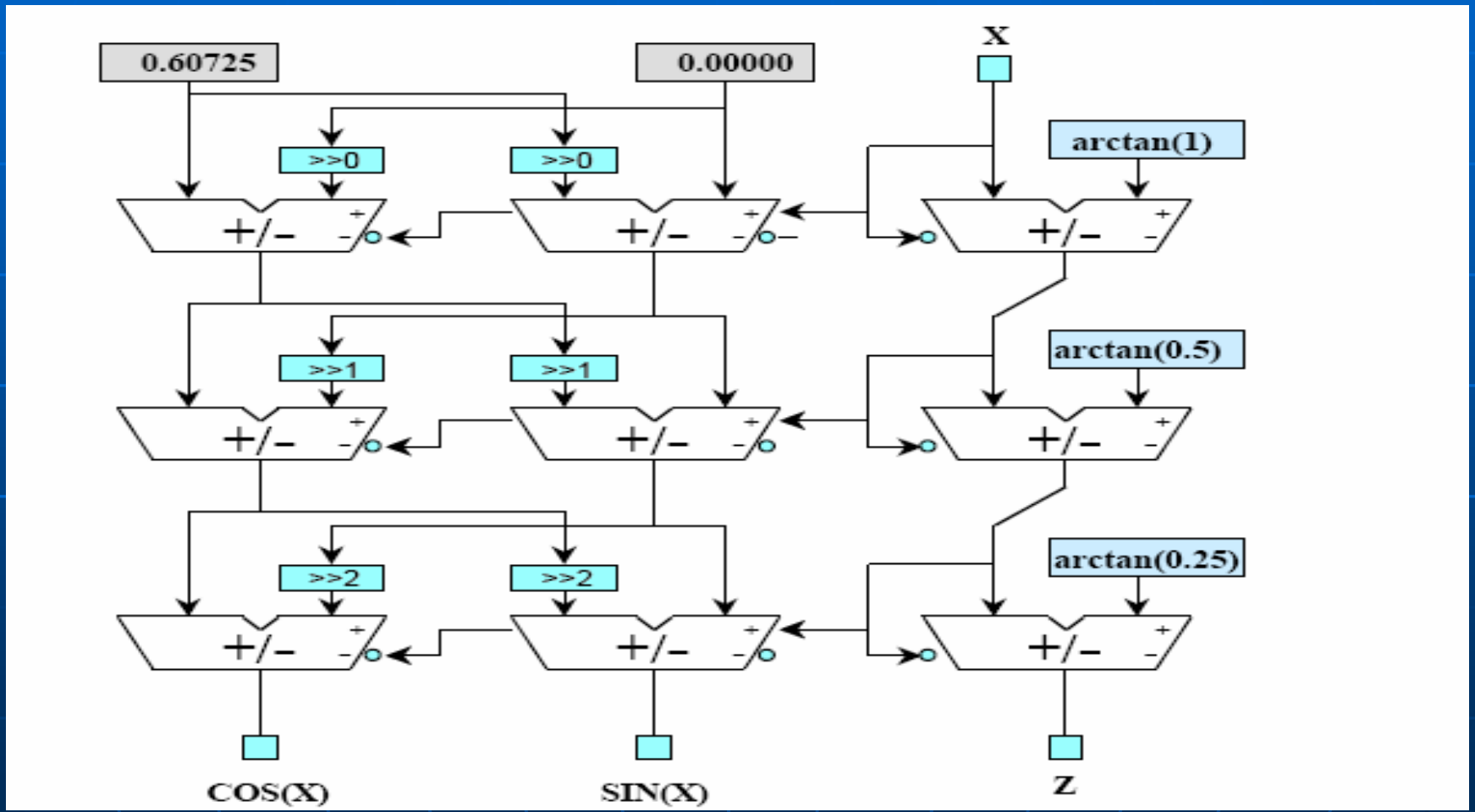
Sequential/Iterative CORDIC



Cont..

- Maximum number of Clock Cycles to calculate output
- Minimum Clock Period per itration
- Variable Shifters do not map well on certain FPGA’s due to high Fan-in

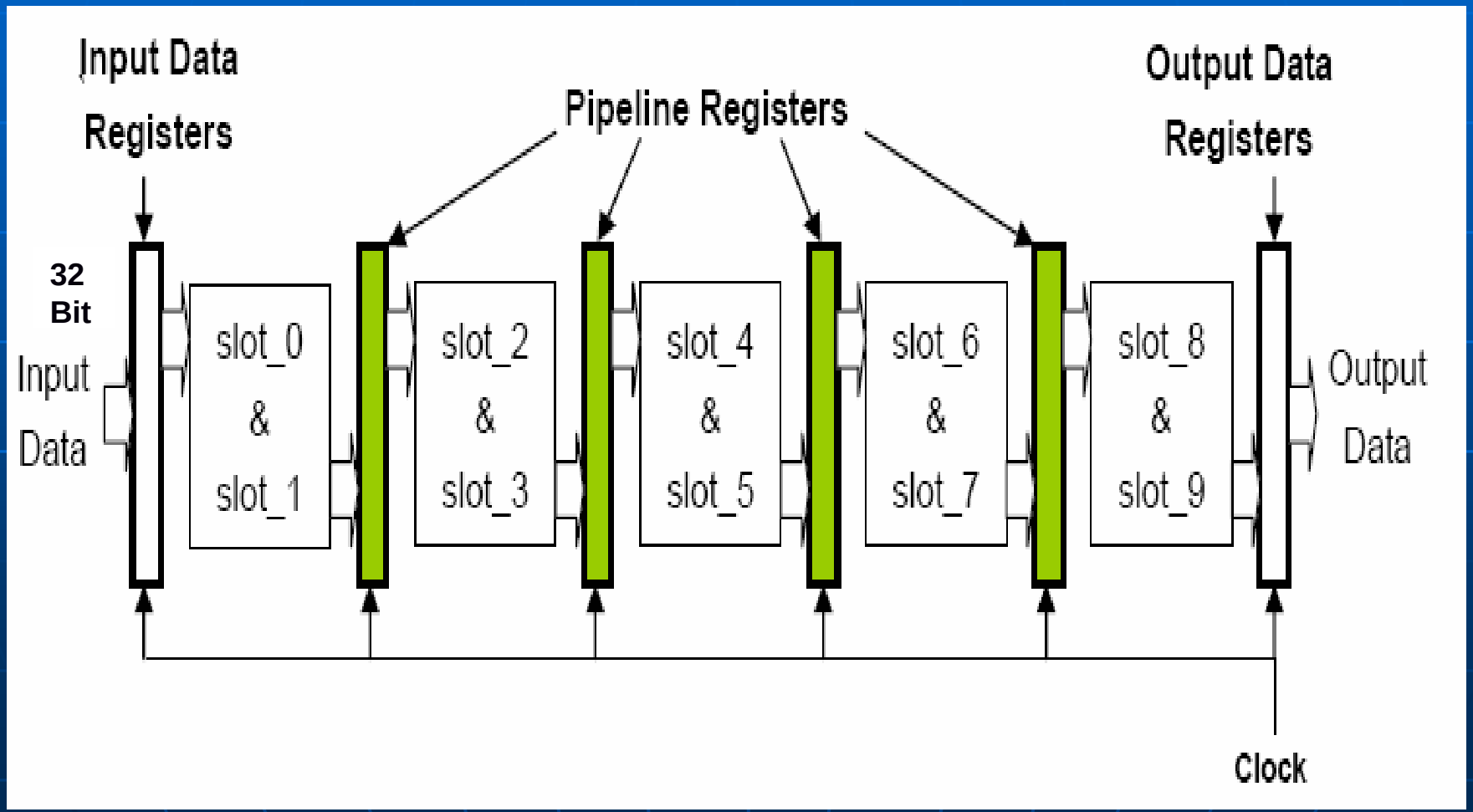
Parallel/Cascaded CORDIC



Parallel CORDIC Cont..

- Combinational circuit
- More Delay, but processing time is reduced as compared to iterative circuit.
- Shifters are of fixed shift, so they can be implemented in the wiring.
- Constants can be hardwired instead of requiring storage space.

Pipeline Architecture



Parallel Pipelined CORDIC

- Parallel CORDIC can be pipelined by inserting registers between the adders stages.
- In most FPGA architectures there are already registers present in each logic cell, so pipeline registers has no hardware cost.
- Number of stages after which pipeline register is inserted can be modeled, considering clock frequency of system.
- When operating at greater clock period power consumption in later stages reduces due to lesser switching activity in each clock period.

Redundant Addition

- Main delay in critical path of the CORDIC iteration is that of the adder.
- To reduce this delay we can use redundant adders.
- In signed digit number system addition becomes carry free.

Example

- r = 10 , digit set [0,9]

	5	7	8	2	4	9	
+	6	2	9	3	8	9	
	11	9	17	5	12	18	[0,18]
	11	9	16	5	12	16	[0,16]
0	0	1	0	0	2		[0,2]
	11	10	16	5	14	16	[0,16]

Example

- r = 2, digit set [-1,0,1]

	1	0	0	-1	7 in decimal
	1	0	-1	0	6 in decimal
1	0	-1	0		C_i
0	0	0	1	-1	U_i
1	0	-1	1	-1	13 in decimal

Language, Platform, Tools

- Language – Verilog HDL
- Platform
  - Xilinx FPGA
  - ASIC – TSMC LibraryAldec Active-HDL
- Tools
  - Aldec Active-HDL, Synplify Pro, Xilinx ISE (Windows Platform)
  - Cadence – Verilog-XL & Simvision
  - Synopsys Design Analyzer (Unix Platform)

FPGA(3s200ft256) Results

	Sequential	Parallel	Pipeline
LUT	597	864	867
Gate Count	5833	10371	14536
Path Delay	9.7		9.7

ASIC (TSMC) Results

	Sequential	Parallel	Pipeline	Parallel – SD Adder
Area	9138	39019	62937	356015
Power	554µW	22.9mW	3mW	27.4mW
Arrival Time	9.7	28.3	9.7	7.82

Clock Frequency

- FPGA – 85 MHz
- ASIC – 150Mhz

Application

- CORDIC sine/cosine generators in satellite data processing systems – attitude determination

  Cordic modules has been proposed for calculation of Legendre Polynomials. Increase in speed 40% compared to s/w running on 333Mhz Pentium

- Direct Digital Synthesis – generates a new frequency based upon an original reference frequency.

Verification

- C code was used to generate test vectors

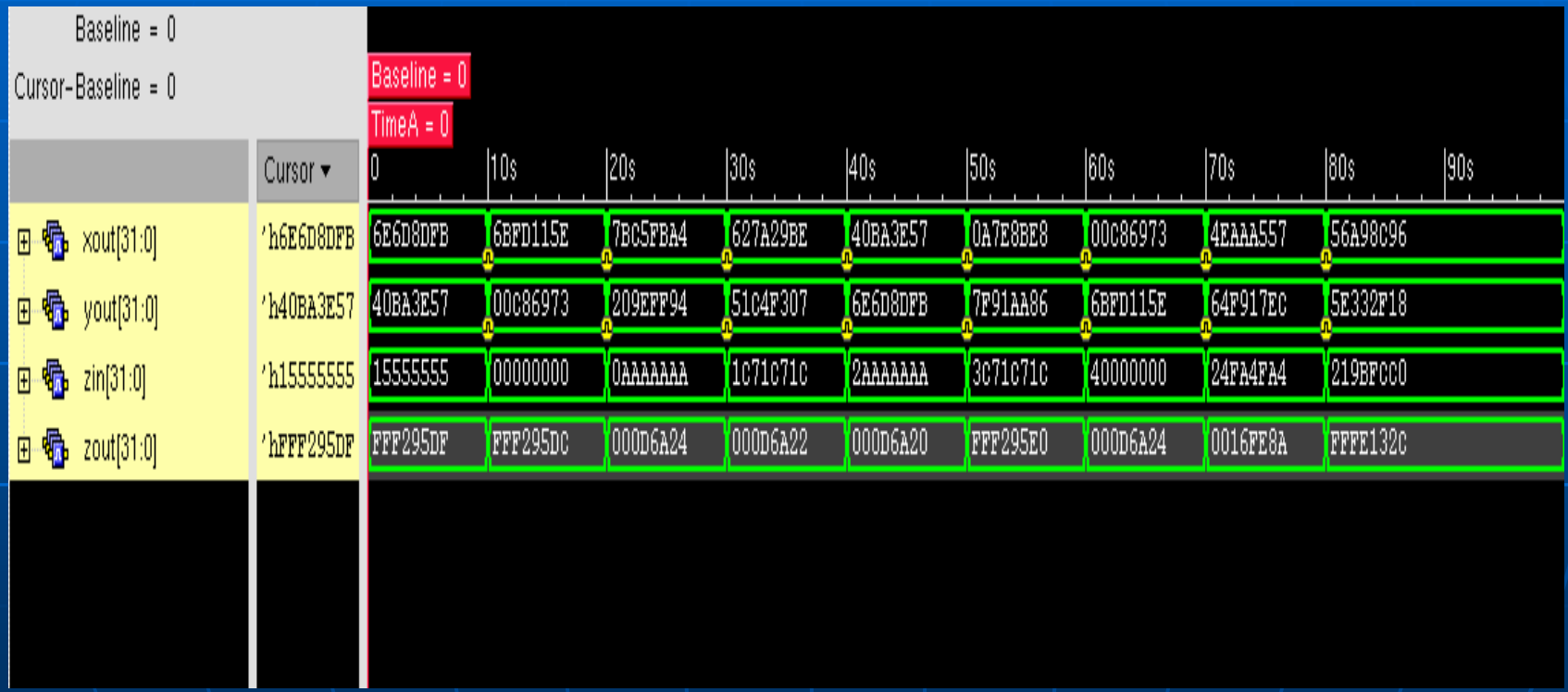
$360^\circ = 2^{32}$

$1^\circ = \frac{232}{360^\circ} = (0B60B60)_{16}$

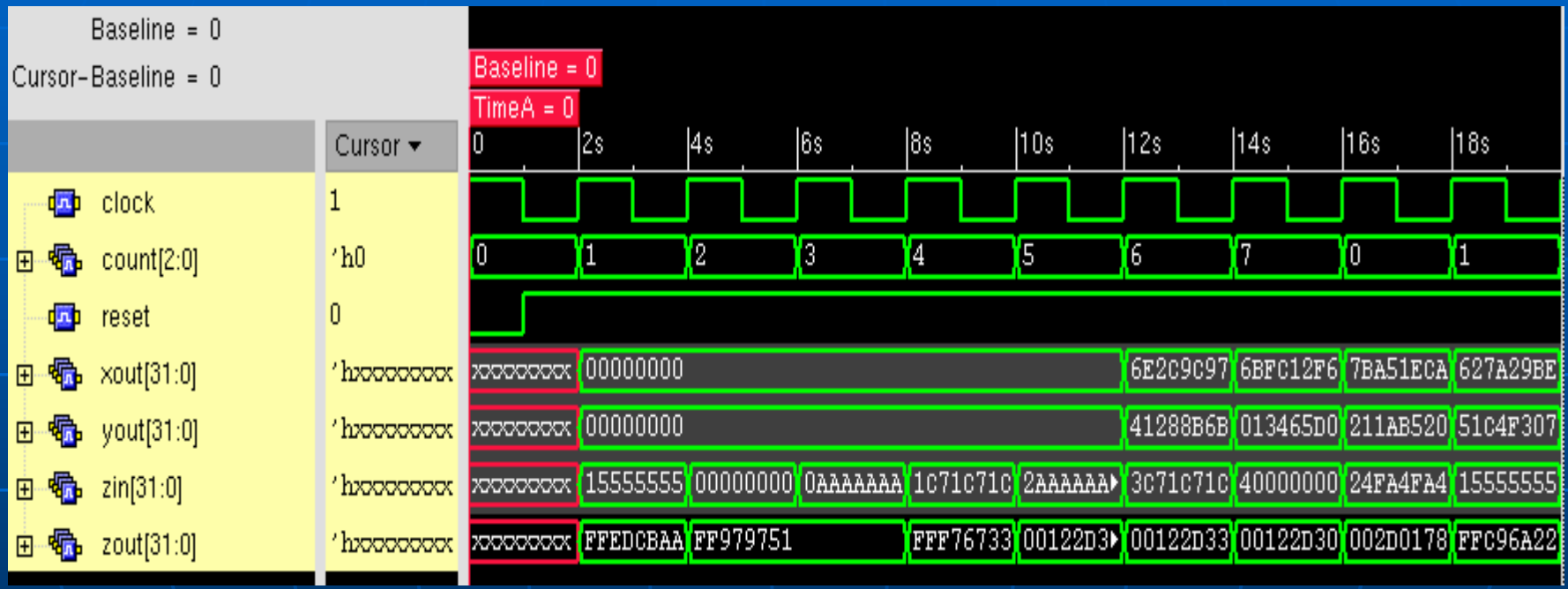
i.e $30^\circ = \frac{2^{32}}{360} \times 30 = (15555555)_{16}$

- Verification of Results done by simulator given by Isreal Koren

Output Waveform - Parallel



Output Waveform - Pipeline



Status of Design

- Written – 100 %
- Functional Simulation – 100%
- Timing Simulation – 50%
- Analyzed for Speed and Area – 80%

Further Optimization

- Area Optimized – If adders are shared.
- Speed & Area Optimized – Modified CORDIC algorithm

Problem

- Difficulty in implementation of Redundant Adders.

Conclusion

- A trade- off speed/area will determine the right structural approach to CORDIC FPGA implementation for an application

References

- Computer Arithmetic – B. Parhami
- Digital Arithmetic – Milos.E
- Survey of CORDIC algorithms for FPGA based computers – R.Andraka
- FPGA Implementation of Sine and Cosine Generators using the CORDIC algorithm.
- Computer Arithmetic – I.Koren (SD Adder)

Thank You !