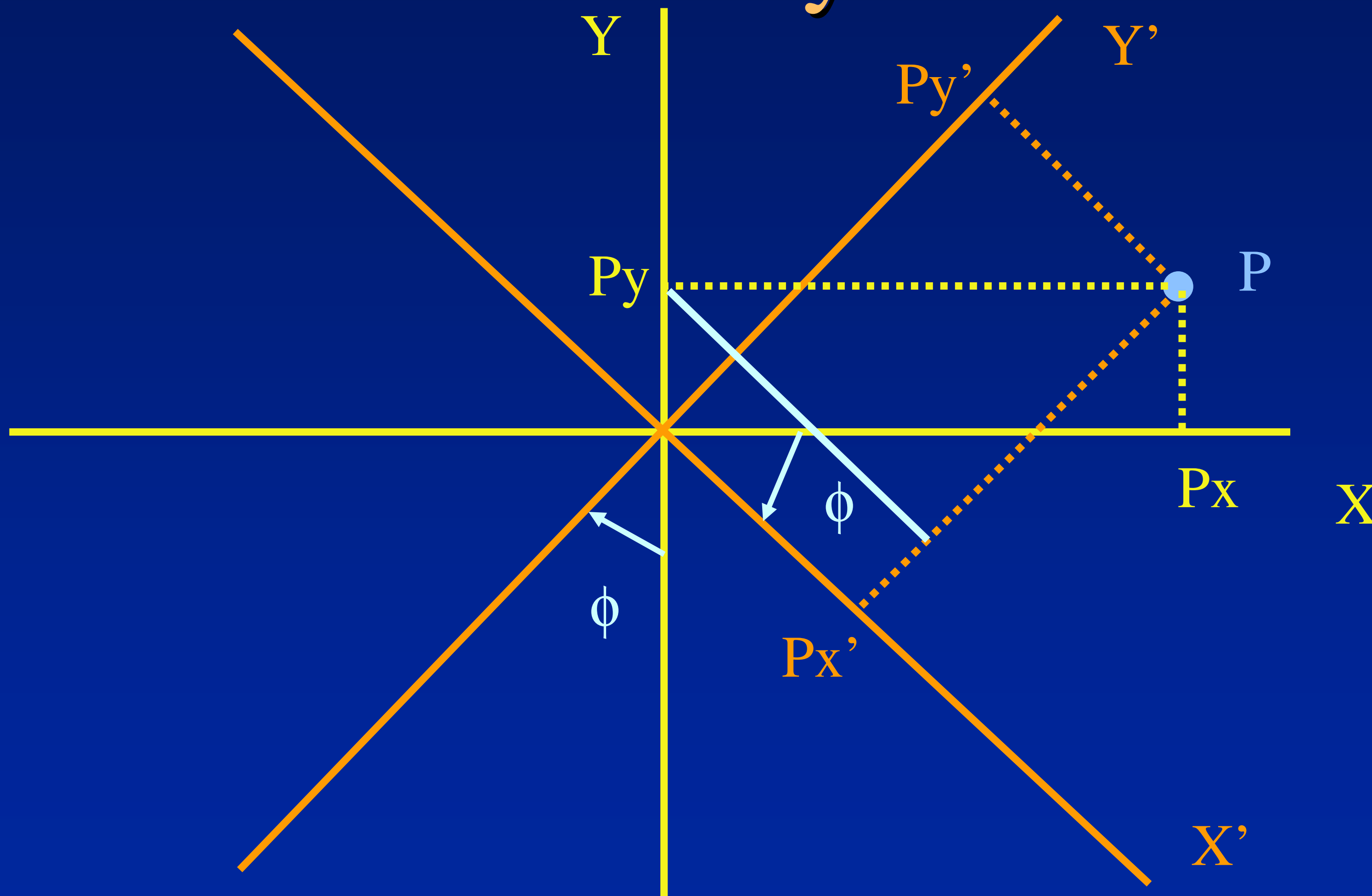
# Cordic Algorithms in FPGAs

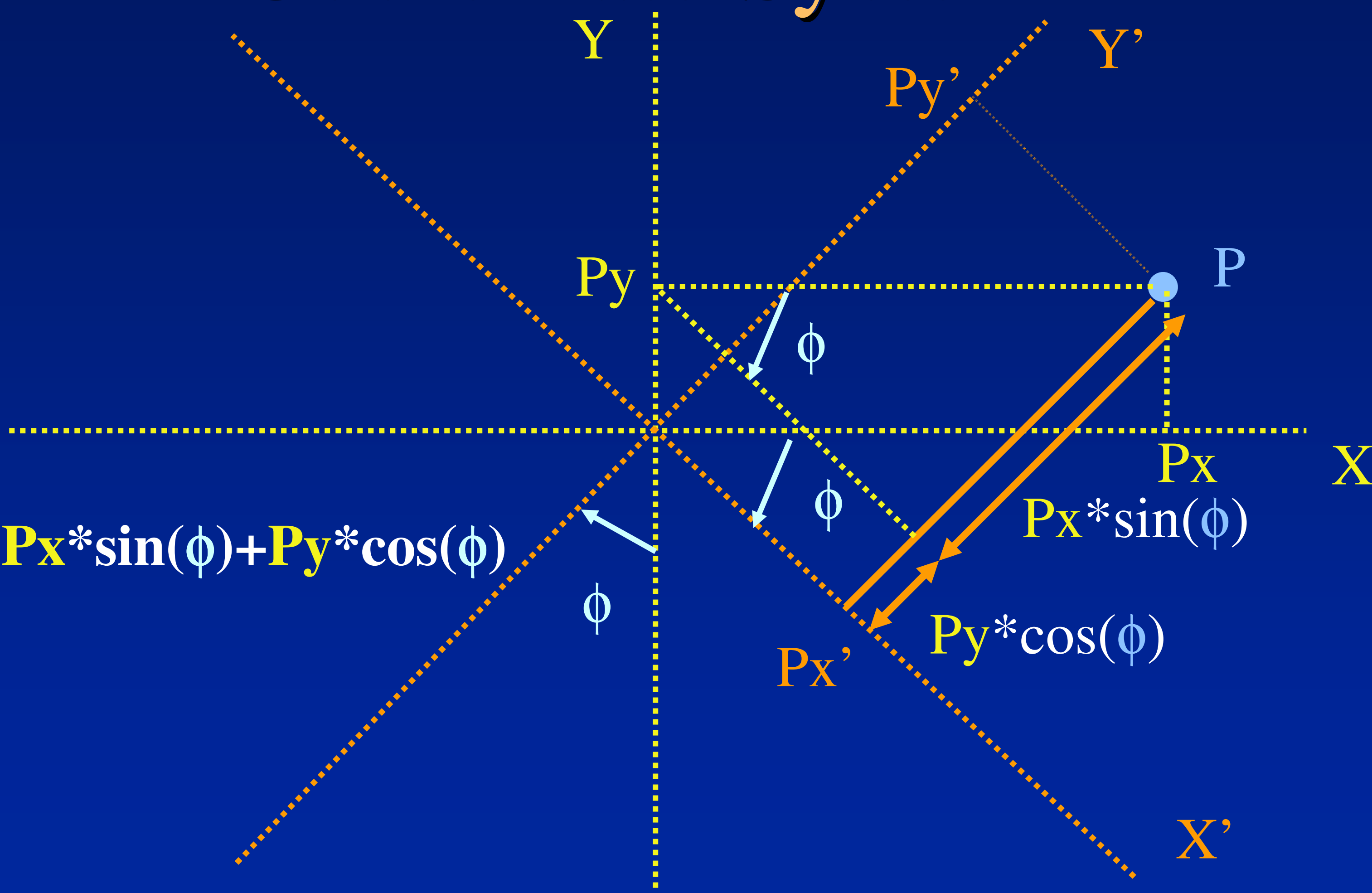
# Cordic Algorithm

- COrdinate Rotation Digital Computer
- An iterative method to perform Coordinate Rotations using only shifts and adds
- Hardware Efficient algorithms used for calculating trigonometric and transcendental functions

# Vector rotation in Cartesian Coordinate System

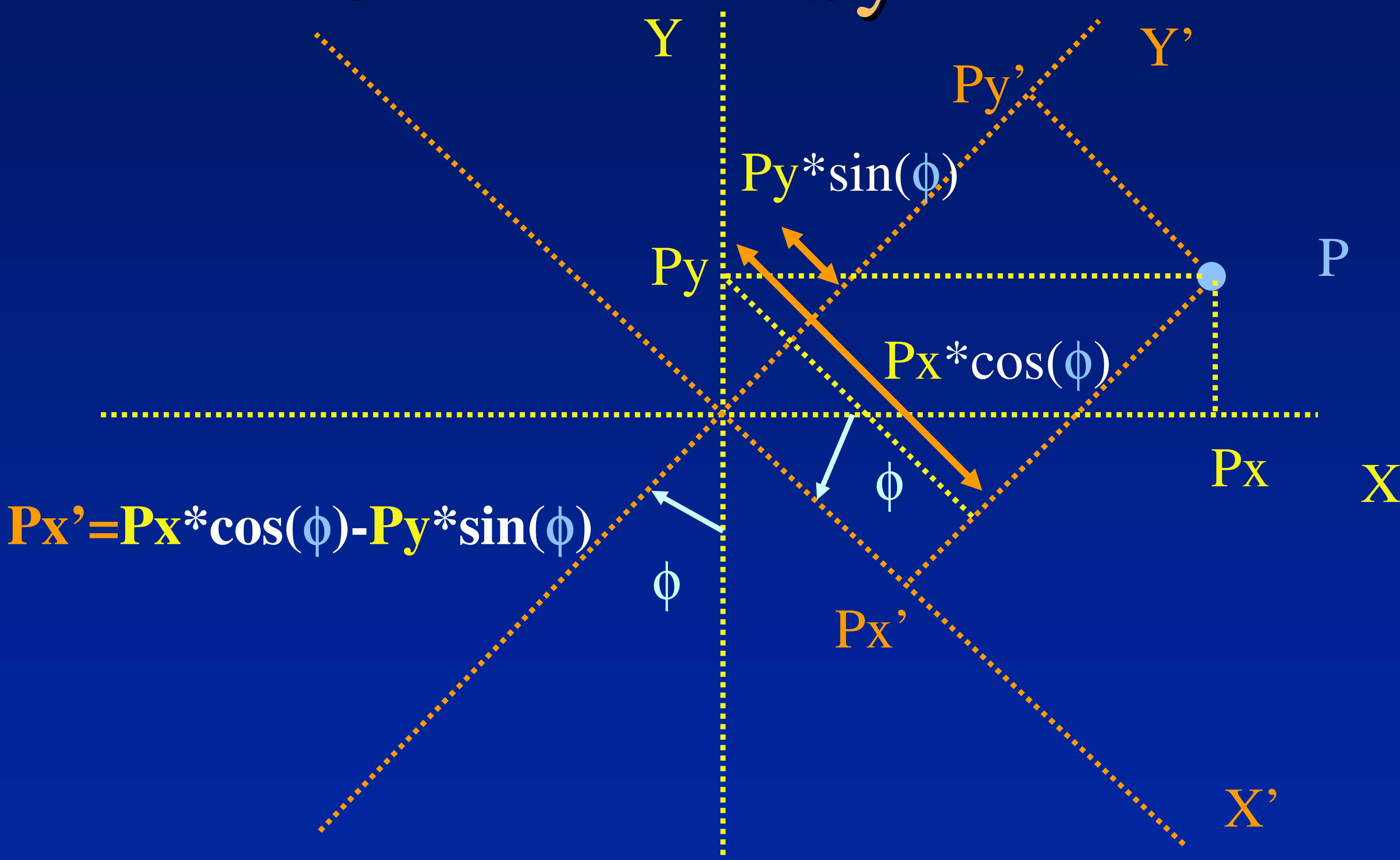


# Vector rotation in Cartesian Coordinate System




**Py'=Px*sin($\phi$)+Py*cos($\phi$)**

# Vector rotation in Cartesian Coordinate System



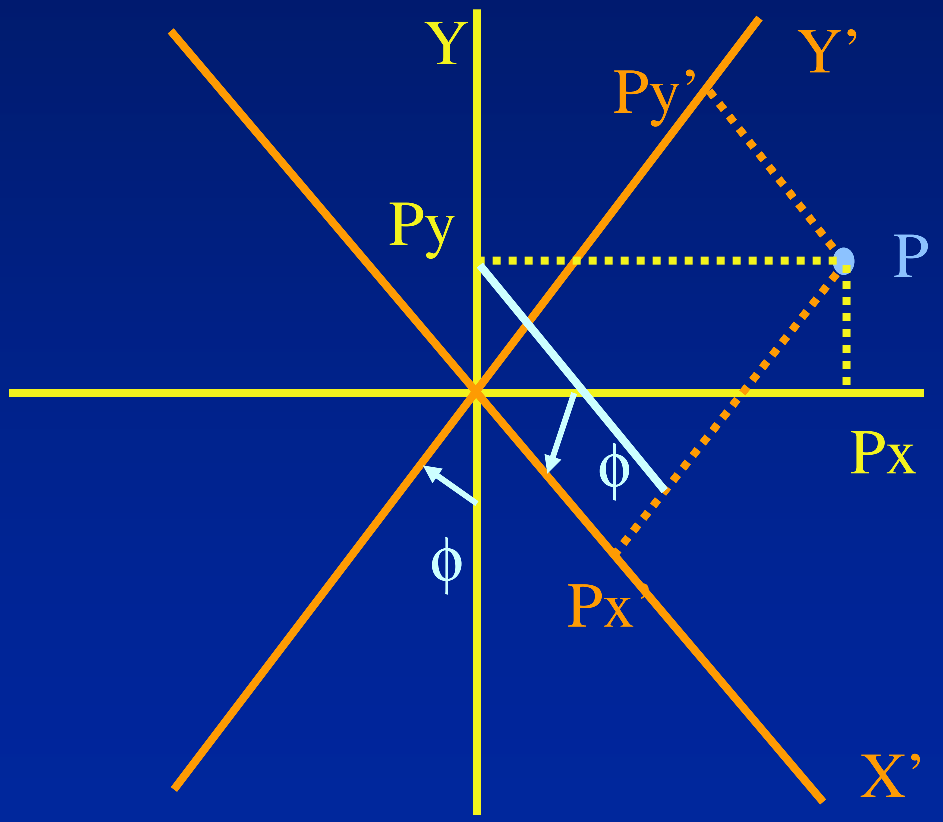
# Vector rotation in Cartesian Coordinate System

$Px' = Px * \cos(\phi) - Py * \sin(\phi)$

$Px' = \cos(\phi) * ( Px - Py * \tan(\phi))$

$Py' = Px * \sin(\phi) + Py * \cos(\phi)$

$Py' = \cos(\phi) * (Py + Px * \tan(\phi))$



# CORDIC Equations

$$x_{i+1} = \cos(\phi_i) * ( x_i - y_i * \tan(\phi_i) )$$
$$y_{i+1} = \cos(\phi_i) * ( y_i + x_I * \tan(\phi_i) )$$
$$z_{i+1} = z_i - \phi_i$$

Assume $\tan(\phi_i) = d_i * 2^{-i}$ where $d_i = +1$ or $-1$

$$\phi_i = d_i * \tan^{-1}(2^{-i})$$
$$K_i = \cos(\phi_i) = \cos(\tan^{-1}(2^{-I})) = 1/\sqrt{1+2^{-2i}}$$

# CORDIC Equations

- Factor Out the constant $K_i$ from each iteration

  $x_{i+1} = x_I - y_i * d_i * 2^{-i}$

  $y_{i+1} = y_I + x_i * d_i * 2^{-i}$

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

- Multiply by $A_n$ after n iterations to account for constants $K_i$

$$A_n = \prod_{i=1}^{n} \sqrt{1+2^{-2i}}$$

# CORDIC Equations

- Previous Equations are valid if initial angle $z_0 > -\pi/2$ and $z_0 < \pi/2$. For other angles, give an initial rotation of $\pi$.

- For initial rotation the equations are

  $x' = d * x$ where $d = -1$ if $x < 0$ else $d = 1$

  $y' = -d * y$

  $z' = z$ if $d$ is equal to $1$

  $= z - \pi$ if $d$ is equal to $-1$

- An alternate solution is to rotate by $\pi/2$ or $-\pi/2$ initially.

# Modes of Operation

- Hardware has three accumulators **x**, **y** and **z**

  $x_{i+1} = x_i - y_i * d_i * 2^{-i}$

  $y_{i+1} = y_i + x_i * d_i * 2^{-i}$

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

- Two modes of operation
  - ***Rotation Mode***
  - ***Vector Mode***

# Rotation Mode of Operation

- Initialize three accumulators by $x_0$, $y_0$ and $z_0$
- Rotate to make the angle accumulator $z_0$ equal to $\mathbf{0}$.
  - $d_i = -1$ if $z_i < 0$
  - $d_i = 1$ otherwise
- Final accumulator values are $x_n$, $y_n$ and $z_n$

$$x_{i+1} = x_i - y_i * d_i * 2^{-i}$$
$$y_{i+1} = y_i + x_i * d_i * 2^{-i}$$
$$z_{i+1} = z_i - d_i * \tan^{-1}(2^{-i})$$

$$x_n = A_n * (x_0 \cos(z_0) - y_0 \sin(z_0))$$
$$y_n = A_n * (y_0 \cos(z_0) + x_0 \sin(z_0))$$
$$z_n = 0$$

# Rotation Mode

$x_n = A_n*(x_0\cos(z_0) - y_0\sin(z_0))$

$y_n = A_n*(y_0\cos(z_0) + x_0\sin(z_0))$

$z_n = 0$

- General Vector Rotator :
  - Rotates a vector $(\mathbf{x_0,y_0})$ by an angle $\mathbf{z_0}$.
- Sine and Cosine Generator :
  - Initial Values $x_0=1/A_n$, $y_0=0$ and $z_0=\theta$
  - Final Values $x_n=\cos(\theta)$, $y_n=\sin(\theta)$ and $z_n=0$
- Polar to Rectangular Coordinate Conversion :
  - Initial Values $x_0=r$, $y_0=0$ and $z_0=\theta$
  - Final Values $x_n=A_n r\cos(\theta)$, $y_n= A_n r\sin(\theta)$ and $z_n=0$

# Vectoring Mode of Operation

- Initialize three accumulators by $\mathbf{x_0}$, $\mathbf{y_0}$ and $\mathbf{z_0}$
- Rotate to make the y accumulator $\mathbf{y_0}$ equal to **0**.
  - $\mathbf{d_i = -1}$ if $\mathbf{y_i > 0}$
  - $\mathbf{d_i = 1}$ otherwise
- Final accumulator values are $\mathbf{x_n}$, $\mathbf{y_n}$ and $\mathbf{z_n}$

$$x_{i+1} = x_i - y_i * d_i * 2^{-i}$$
$$y_{i+1} = y_i + x_i * d_i * 2^{-i}$$
$$z_{i+1} = z_i - d_i * \tan^{-1}(2^{-i})$$

$$x_n = A_n \sqrt{x_0^2 + y_0^2}$$
$$y_n = 0$$
$$z_n = z_0 + \tan^{-1}(y_0/x_0)$$

# Vectoring Mode

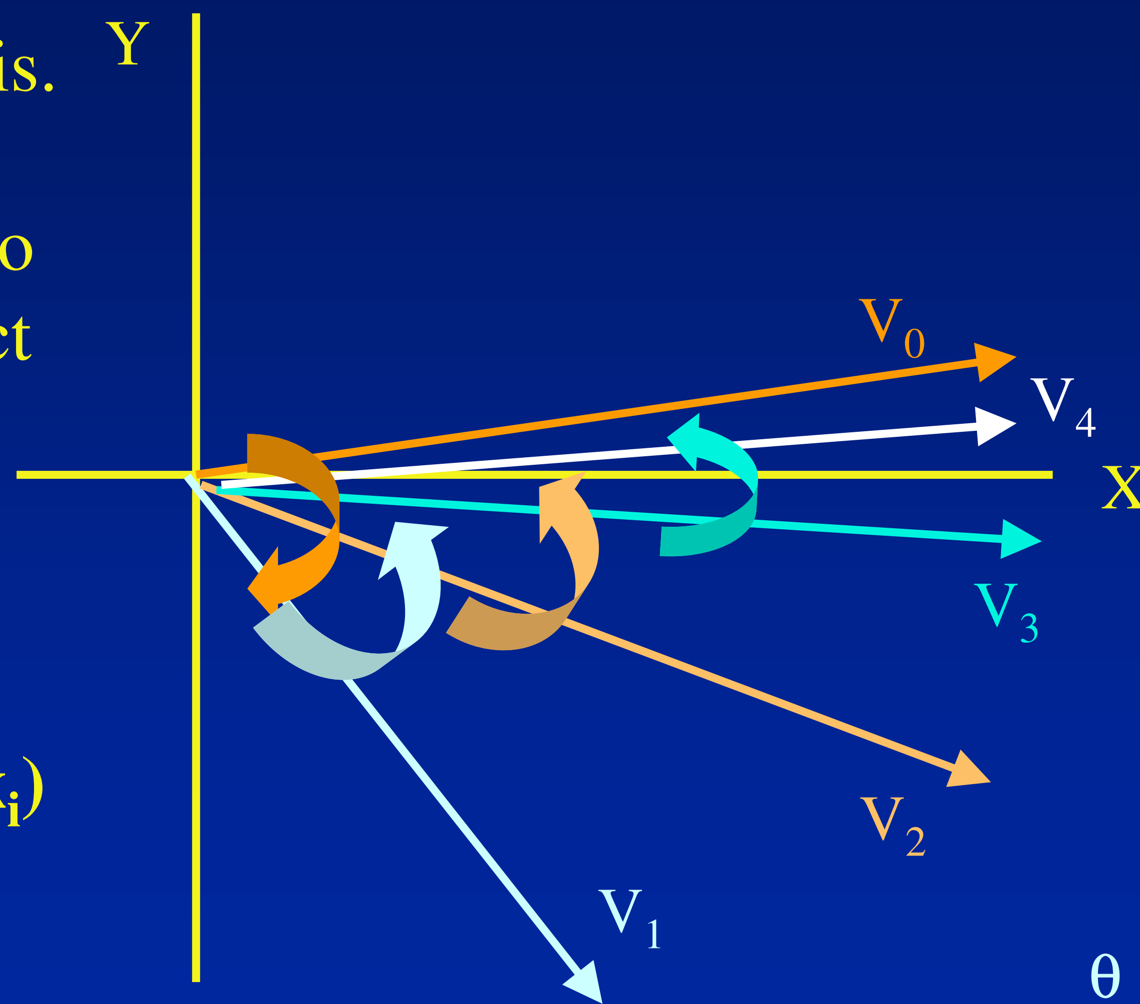
$$x_n = A_n \sqrt{x_0^2 + y_0^2}$$
$$y_n = 0$$
$$z_n = z_0 + \tan^{-1}(y_0/x_0)$$

- Rectangular to Polar Coordinate Conversion — $x_n$ is scaled magnitude and $z_n$ is the angle.
- $z_n$ is $\tan^{-1}(y)$ if $x_0=1$ and $z_0=0$

# Convergence

$V_0$ aligning with X axis.
$V_1$ is incorrect.
Subsequent rotations to $V_2$, $V_3$, … must correct the error

Condition
$$\tan^{-1}(y_{i+1}/x_{i+1}) > 0.5*\tan^{-1}(y_i/x_i)$$
ensures convergence



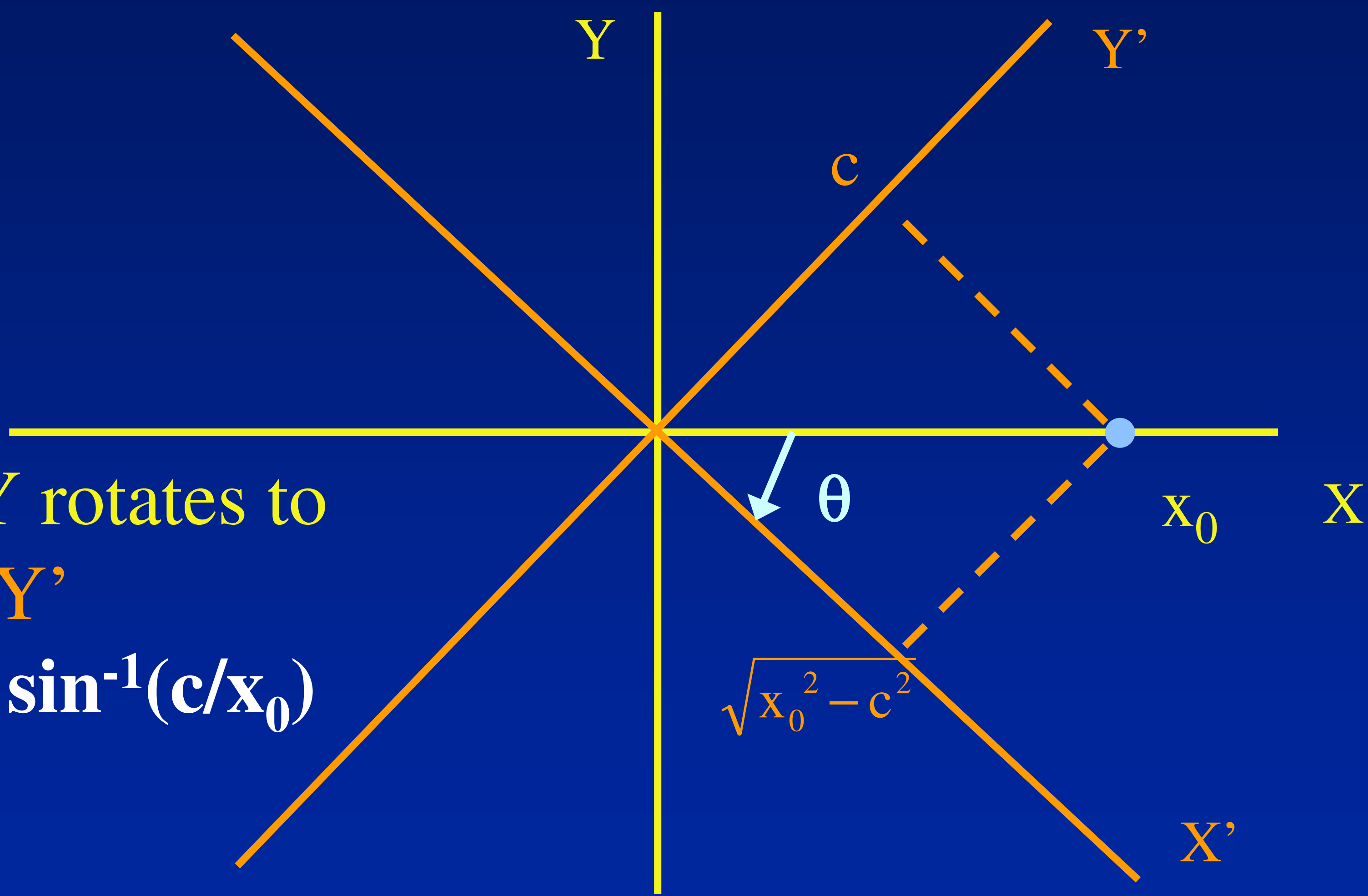
# Convergence

- Cordic Equations in the Rotation Mode and Vectoring Mode converge.
- Error reduces in each iteration.
- Each Rotation makes the accumulators more correct. Decide on the number of rotations from your application.

# Functions covered ….

- Vector Rotation.
- Rectangular to Polar Coordinate Conversion
- Polar to Rectangular Coordinate Conversion
- Sine, Cosine and Arctan Generators

# Arcsine using Inverse Cordic




X-Y rotates to

X'-Y'

$\theta = \sin^{-1}(c/x_0)$

# Arcsine

Rotate the vector so that $y_n$ is equal to **c**, a constant

$$x_{i+1} = x_I - y_i * d_i * 2^{-i}$$

$$y_{i+1} = y_i + x_i * d_i * 2^{-i}$$

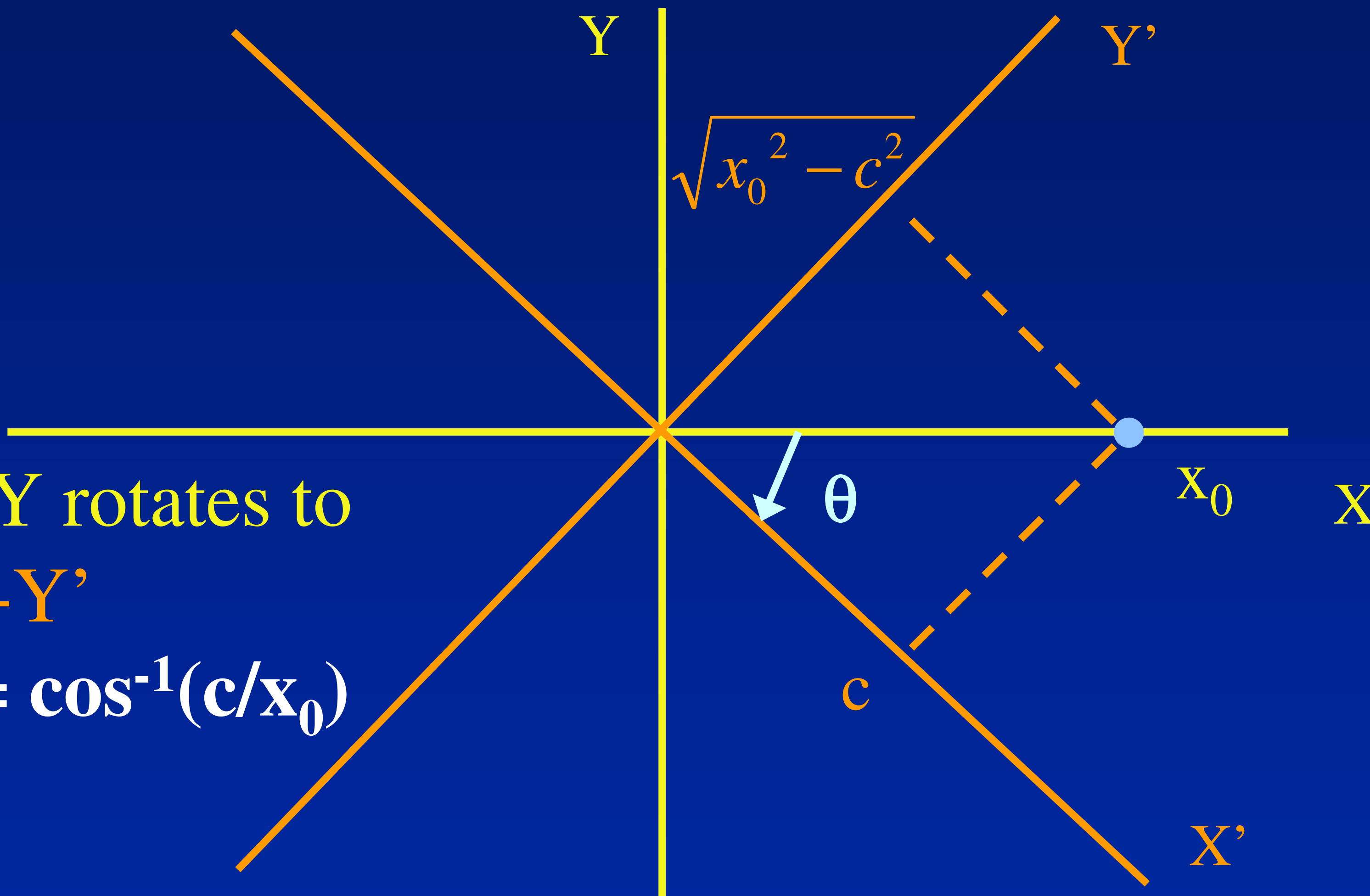
$$z_{i+1} = z_i - d_i * \tan^{-1}(2^{-i})$$

where

$d_i$ = +1 if $y_i < c$, -1 otherwise

- Initial Values : $x_0 = x_0$, $y_0 = 0$ and $z_0 = 0$
- Final Values : $x_n = \sqrt{(A_n x_0) - c^2}$ , $y_n = c$ and $z_n = \sin^{-1}(c/A_n x_0)$

# Arccosine using Inverse Cordic




X-Y rotates to

X'-Y'

$\theta = \cos^{-1}(c/x_0)$

# Arccosine

Rotate the vector so that $\mathbf{x_n}$ is equal to $\mathbf{c}$, a constant

$$x_{i+1} = x_i - y_i*d_i*2^{-i}$$

$$y_{i+1} = y_i + x_i*d_i*2^{-i}$$

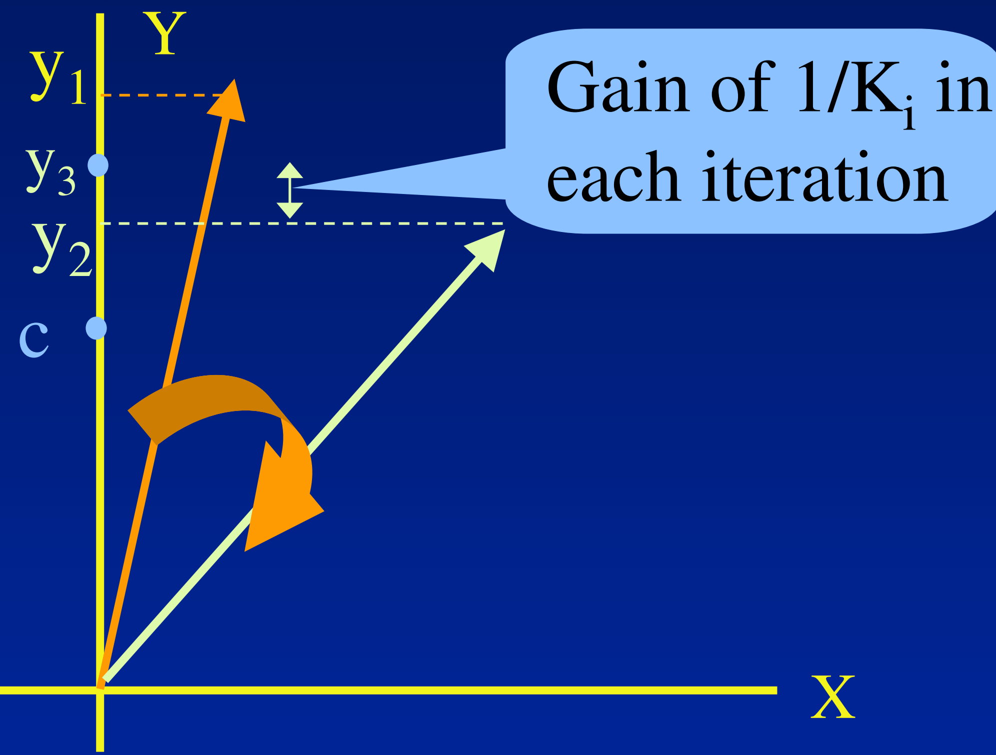
$$z_{i+1} = z_i - d_i*\tan^{-1}(2^{-i})$$

where

$d_i = -1$ if $x_i < c$, $+1$ otherwise

- Initial Values : $x_0 = x_0$, $y_0 = 0$ and $z_0 = 0$
- Final Values : $x_n = \sqrt{(A_n x_0)^2 - c^2}$ , $y_n = c$ and $z_n = \cos^{-1}(c/A_n x_0)$

# Convergence of Arcsine and Arccosine




$$\mathbf{1/K_i = \sqrt{1+2^{-2i}}}$$

- Due to gain $\mathbf{1/K_i}$ , $\mathbf{(y_1\text{-}y_3)/(y_1\text{-}c) < 1/2}$, which does not guarantee convergence

# Convergence of Arcsine and Arccosine

- Double Iteration Algorithm
  - For each iteration, rotate the vector twice instead of once.
- Double Iteration Algorithm makes the results of arcsine and arccosine more accurate.

# General Cordic Equation

- Circular Coordinate Transforms : What has been presented so far.
- Cordic Equations can be extended to calculate more functions
  - Linear Functions
  - Hyperbolic Functions

# Linear Cordic Equations

$$x_{i+1} = x_i - 0 * y_i * d_i * 2^{-i} = x_i$$
$$y_{i+1} = y_i + x_i * d_i * 2^{-i}$$
$$z_{i+1} = z_i - d_i * 2^{-i}$$

- *Rotation Mode* (**$d_i$ = -1** if **$z_i$ < 0**, **+1** otherwise)
  - A multiplier with shifts and adds
  - Final Values:

$$x_n = x_0$$
$$y_n = y_0 + x_0 z_0$$
$$z_n = 0$$

# Linear Cordic Equations

$x_{i+1} = x_i$
$y_{i+1} = y_i + x_i * d_i * 2^{-i}$
$z_{i+1} = z_i - d_i * 2^{-i}$

- Vectoring Mode (**$d_i$ = +1** if **$y_i$<0, -1** otherwise)
  - Calculates the ratio **$y_0/x_0$** in **z** accumulator
  - **$x_n = x_0$**
    **$y_n = 0$**
    **$z_n = z_0 - y_0/x_0$**

# Hyperbolic CORDIC Equations

$x_{i+1} = \cosh(\phi_i) * ( x_i + y_i * \tanh(\phi_i) )$

$y_{i+1} = \cosh(\phi_i) * ( y_i + x_i * \tanh(\phi_i) )$

$z_{i+1} = z_i - \phi_i$

Assume $\tan(\phi_i) = d_i * 2^{-i}$ where $d_i = +1$ or $-1$

$\phi_i = d_i * \tanh^{-1}(2^{-i})$

$K_i = \cosh(\phi_i) = \cosh(\tanh^{-1}(2^{-i})) = 1/\sqrt{1-2^{-2i}}$

# Hyperbolic CORDIC Equations

$$x_{i+1} = x_i + y_i * d_i * 2^{-i}$$
$$y_{i+1} = y_i + x_i * d_i * 2^{-i}$$
$$z_{i+1} = z_i - d_i * \tanh^{-1}(2^{-i})$$

Multiply by $A_n$ after n iterations to account for constants $K_i$

$$A_n = \prod_{i=1}^{n} \sqrt{1 - 2^{-2i}}$$

# Rotation Mode

•Converge Angle Accumulator **z** to **0**.

•$d_i$ = -1 if $z_i$ < 0, +1 otherwise

$$x_{i+1} = x_i + y_i * d_i * 2^{-i}$$
$$y_{i+1} = y_i + x_i * d_i * 2^{-i}$$
$$z_{i+1} = z_i - d_i * \tanh^{-1}(2^{-i})$$

- Initialize with $x_0$, $y_0$ and $z_0$.
- The Hyperbolic Cordic Equations converge to

$$x_n = A_n * ( x_0 \cosh(z_0) + y_0 \sinh(z_0) )$$
$$y_n = A_n * ( y_0 \cosh(z_0) + x_0 \sinh(z_0) )$$
$$z_n = 0$$

# Vectoring Mode

$x_{i+1} = x_i + y_i * d_i * 2^{-i}$
$y_{i+1} = y_i + x_i * d_i * 2^{-i}$
$z_{i+1} = z_i - d_i * \tanh^{-1}(2^{-i})$

•Converge Accumulator **y** to **0**.

•$d_i$ = +1 if $y_i < 0$, +1 otherwise

- Initialize with $x_0$, $y_0$ and $z_0$.
- The Hyperbolic Cordic Equations converge to

$$x_n = A_n \sqrt{x_0^2 - y_0^2}$$
$$y_n = 0$$
$$z_n = z_0 + \tanh^{-1}(y_0/x_0)$$

# Hyperbolic Cordic Equations

- Hyperbolic Equations require double iteration for convergence
- Hyperbolic functions similar to Trigonometric functions described earlier can be calculated
- For more information
  - Walther, J.S., “A Unified algorithm for elementary functions”, Spring Joint Computer Conf, 1971, proc, pp. 379-385.

# Unified Cordic Equation

## Unified Equations

$x_{i+1} = x_i - m*y_i*d_i*2^{-i}$

$y_{i+1} = y_i + x_i*d_i*2^{-i}$

$z_{i+1} = z_i - d_i*e_i$

## Linear Functions

*m = 0* and $e_i = 2^{-i}$

$x_{i+1} = x_i$

$y_{i+1} = y_i + x_i * d_i * 2^{-i}$

$z_{i+1} = z_i - d_i * 2^{-i}$

## Hyperbolic Functions

*m = 1* and $e_i = tanh^{-1}(2^{-i})$

$x_{i+1} = x_i + y_i * d_i * 2^{-i}$

$y_{i+1} = y_i + x_i * d_i * 2^{-i}$

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

## Trigonometric Functions

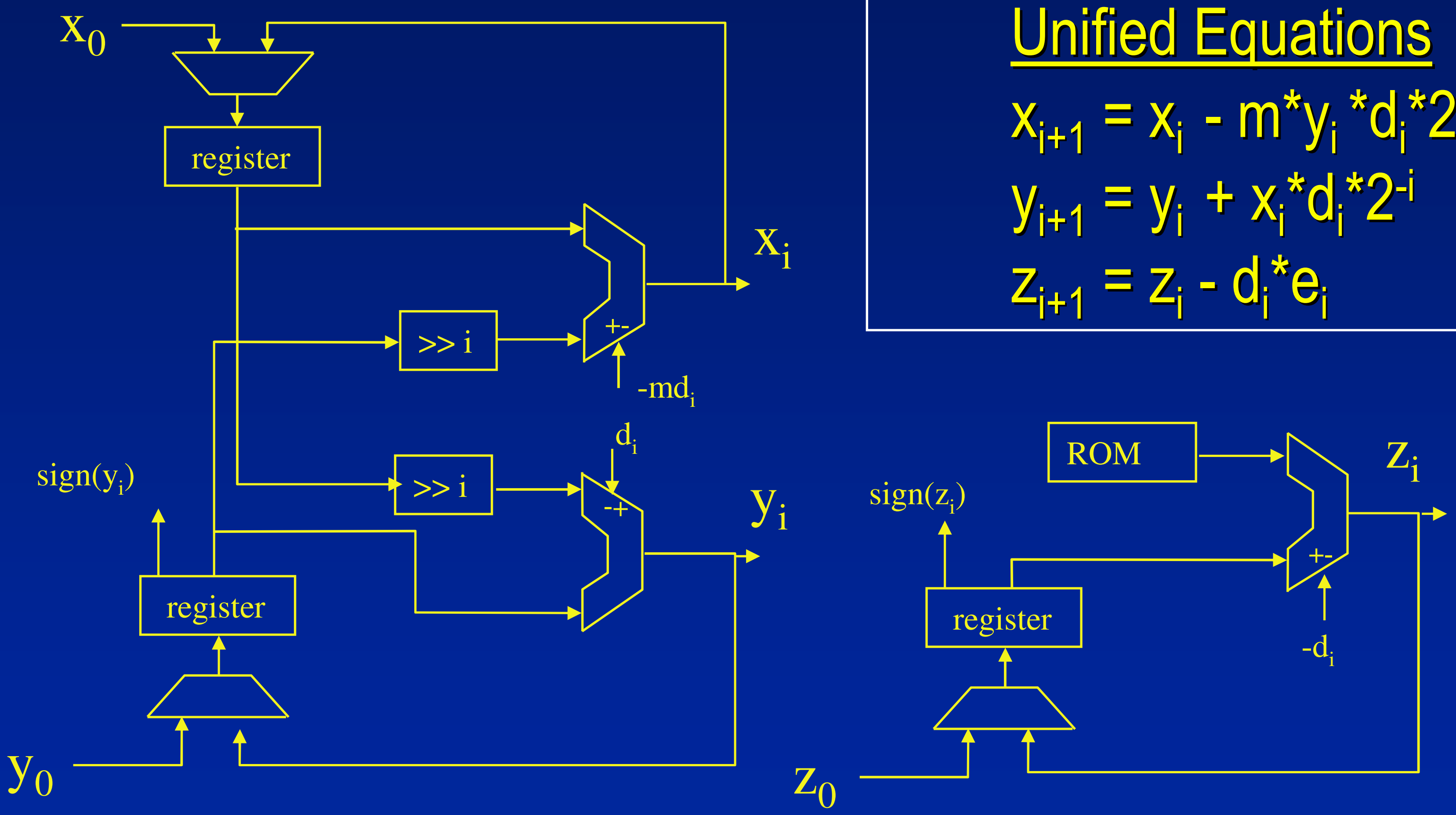
*m = -1* and $e_i = tan^{-1}(2^{-i})$

$x_{i+1} = x_i - y_i*d_i*2^{-i}$

$y_{i+1} = y_i + x_i*d_i*2^{-i}$

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

# Bit Parallel Implementation



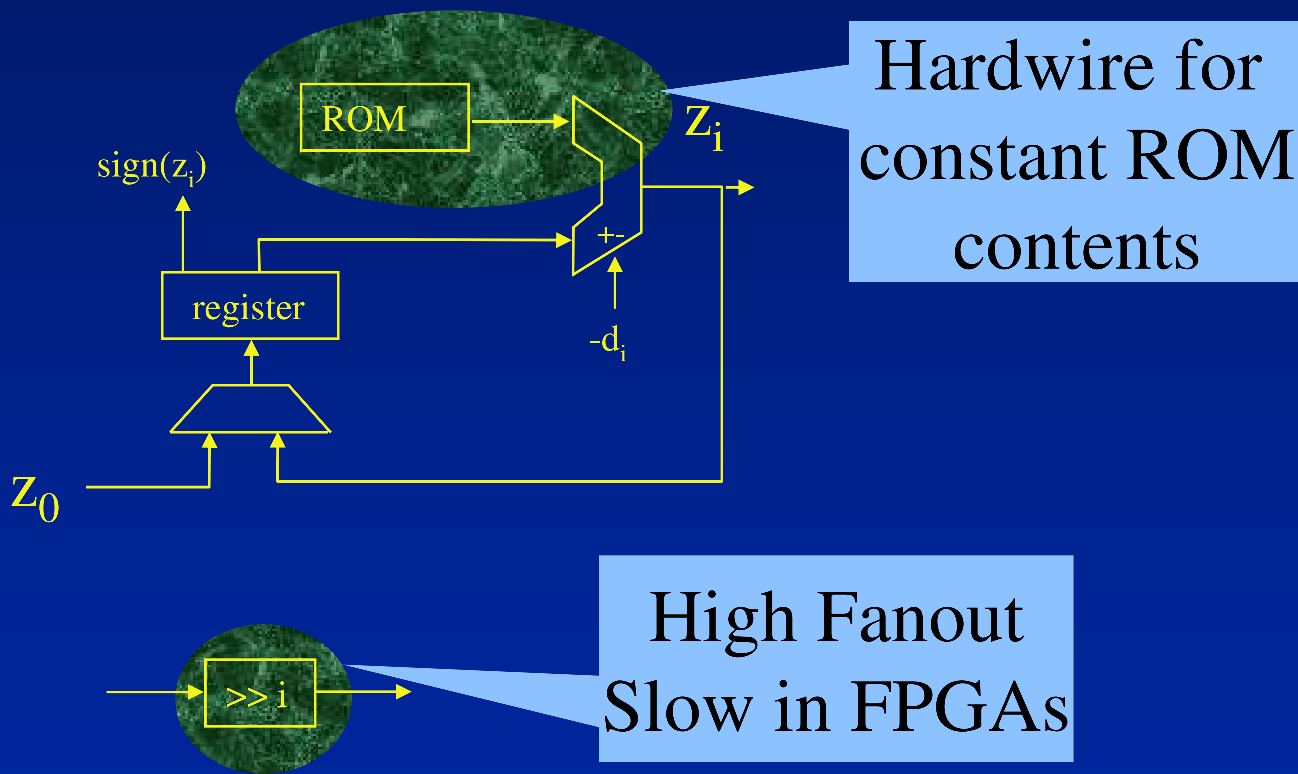

**Unified Equations**

$x_{i+1} = x_i - m*y_i*d_i*2^{-i}$

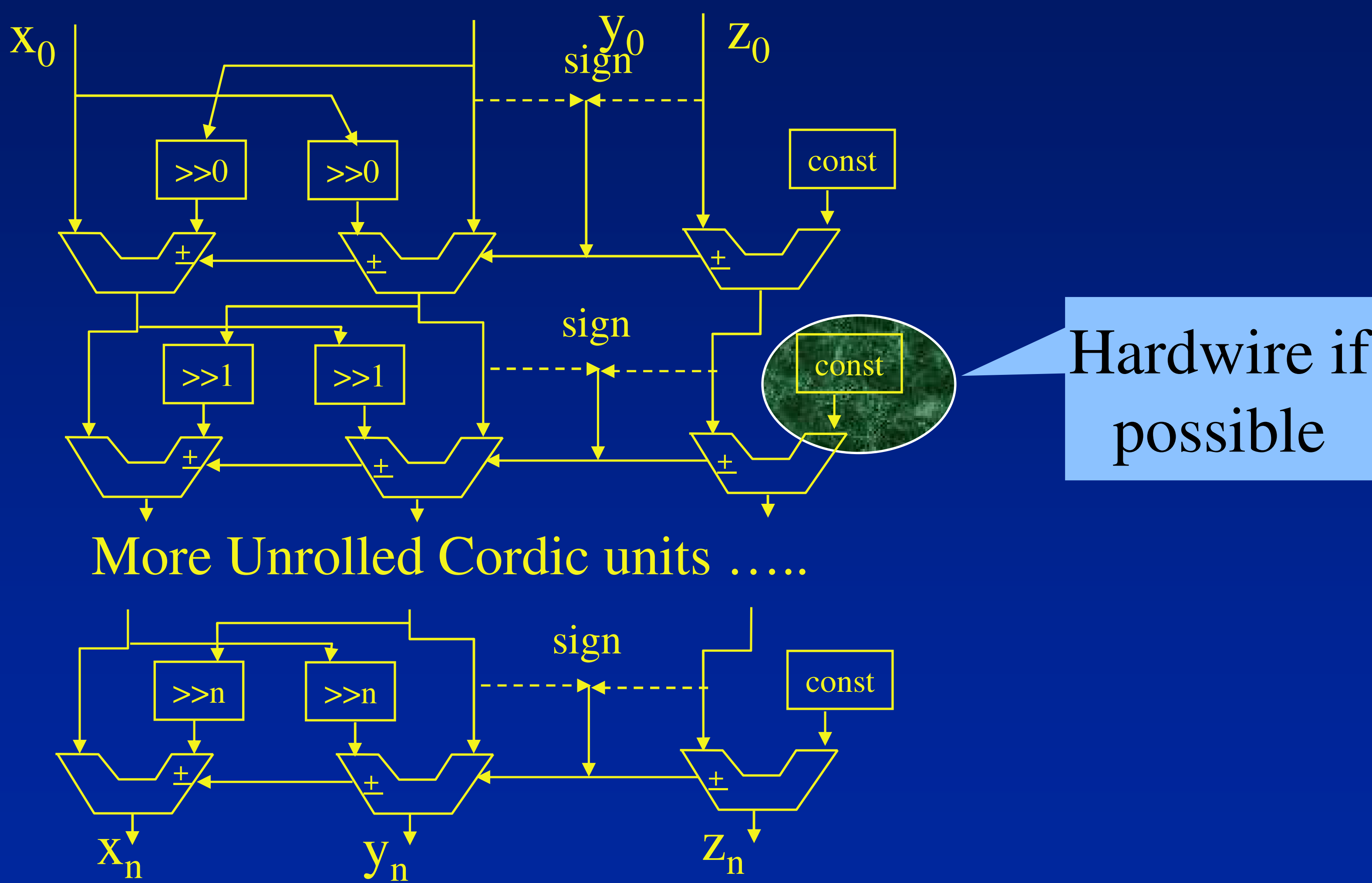
$y_{i+1} = y_i + x_i*d_i*2^{-i}$

$z_{i+1} = z_i - d_i*e_i$

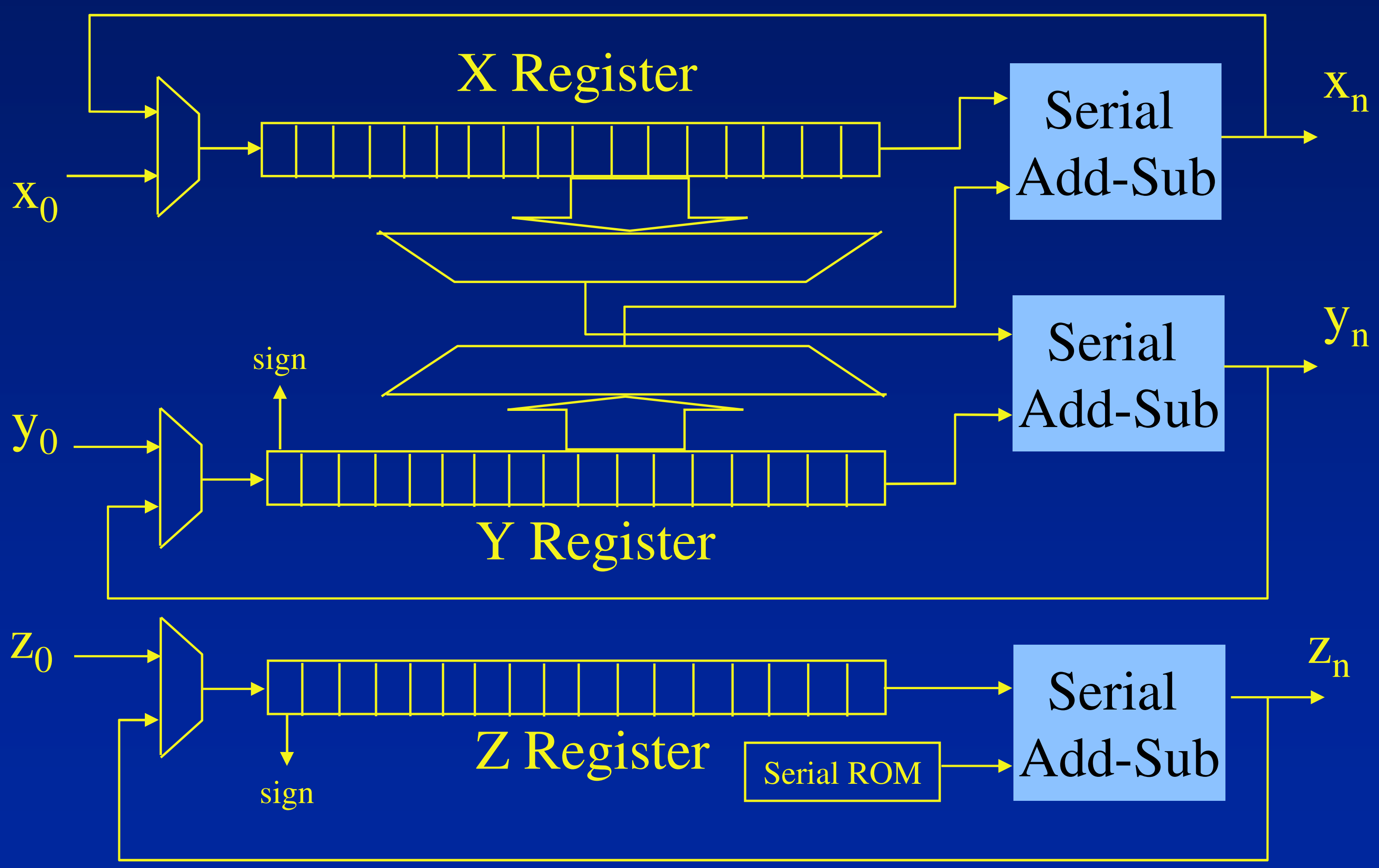
# Bit Parallel Implementation



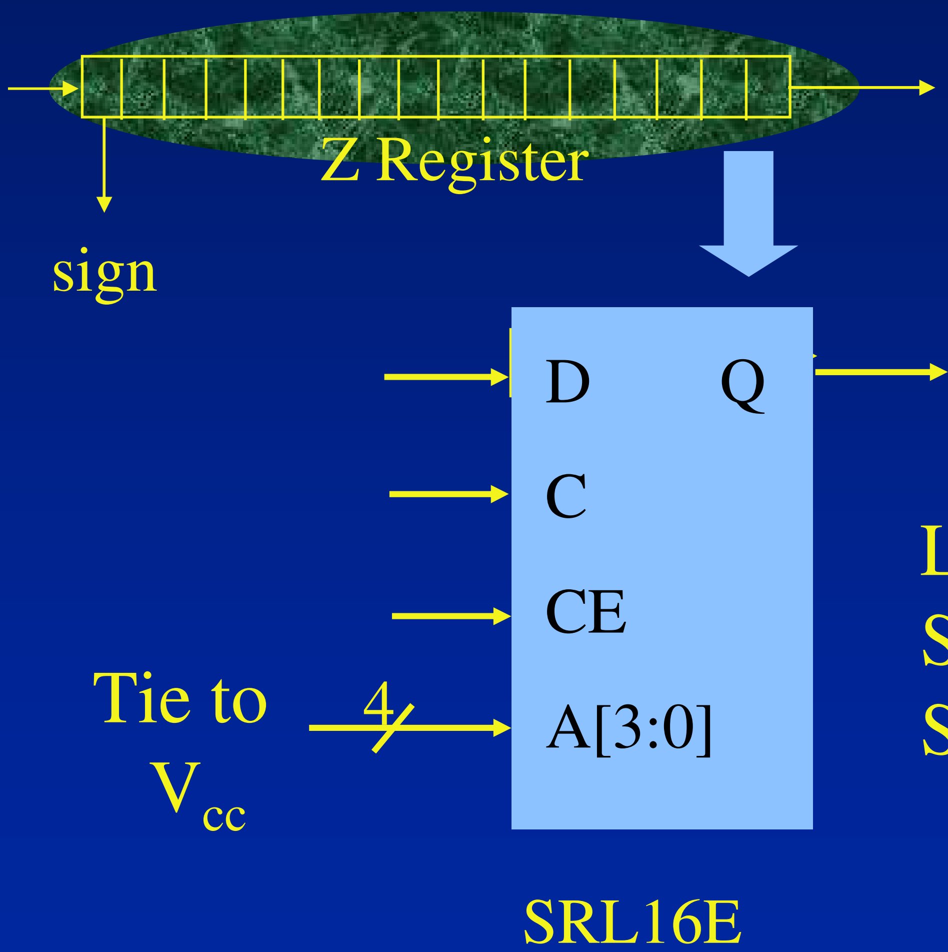
# Unrolled Bit Parallel



# Bit Serial Implementation

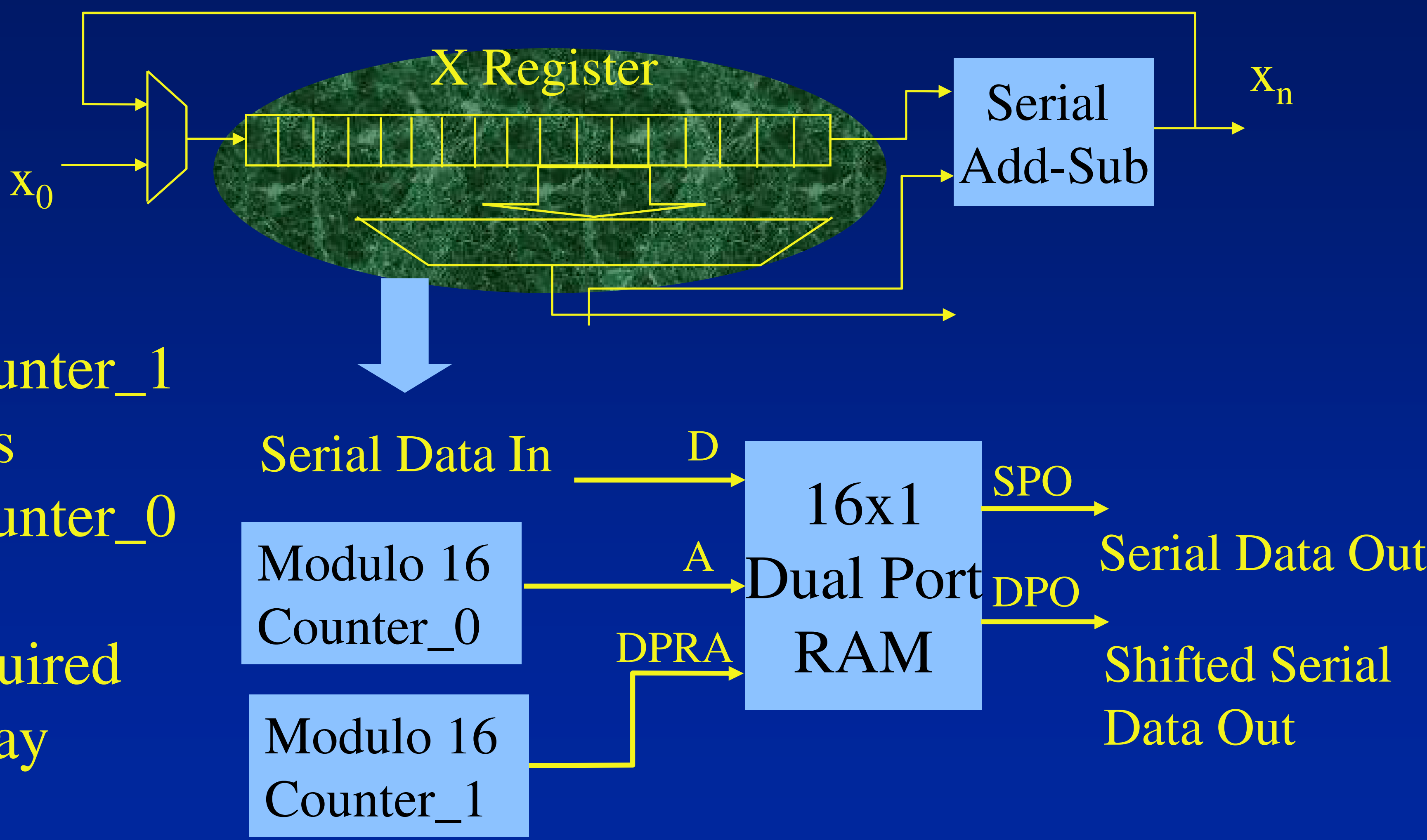


# Bit Serial Implementation in Virtex




LUT used as
Serial In Serial Out
Shift Register

# Bit Serial Implementation in Virtex




Counter_1 lags Counter_0 for required delay

# Conclusion

- Good hardware algorithm
  - calculate various trigonometric and hyperbolic functions
  - Rotate Vectors appropriately
  - Convert from one coordinate system to another
- Pay attention to the convergence of the algorithm.

# References

Andraka, R.J., "A survey of CORDIC algorithms for FPGA based computers", FPGA '98, Proc. Of the 1998 ACM/SIGDA sixth international symposium on Field Programmable Gate Arrays, Feb 22-24, 1998, Monterey CA, pp 191-200.

Volder, J., "The CORDIC Trigonometric Computing Technique", IRE Trans. Electronic Computing, Vol EC-8, pp330-334, Sept 1959.

Walther, J.S., "A unified algorithm for elementary functions", Spring Joint Computer Conf., pp. 379-385, proc., 1971.

Volder, J., "Binary Computation algorithms for coordinate rotation and function generation", Convair Report IAR-1 148 Aeroelectrics Group, June 1956.