

$$\sum_{k=0}^{n/2} (m_{2k-1} + k_{2k-1})(m_{2k} + k_{2k})r^{-k}$$

**ECE 699—Digital Signal Processing  
Hardware Implementations**

**Lecture 8**

CORDIC, DDFS, Bit-Level Arithmetic  
4/1/09

1

**Outline**

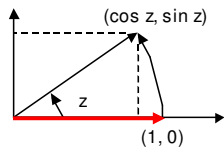
- CORDIC
- Direct Digital Frequency Synthesis
- Bit-Level Arithmetic (Addition, Multiplication)

2

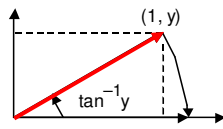
**CORDIC**

3

### 22.1 Rotations and Pseudorotations



start at (1, 0)  
rotate by z  
get cos z, sin z



start at (1, y)  
rotate until y = 0  
rotation amount is tan<sup>-1</sup>y

#### Key ideas in CORDIC

**CO**ordinate **R**otation  
**D**igital **C**omputer used  
this method in 1950s;  
modern electronic  
calculators also use it

If we have a computationally efficient way of rotating a vector,  
we can evaluate cos, sin, and tan<sup>-1</sup> functions

Rotation by an arbitrary angle is difficult, so we:

- Perform pseudorotations that require simpler operations
- Use special angles to synthesize the desired angle z

$$z = \alpha^{(1)} + \alpha^{(2)} + \dots + \alpha^{(m)}$$

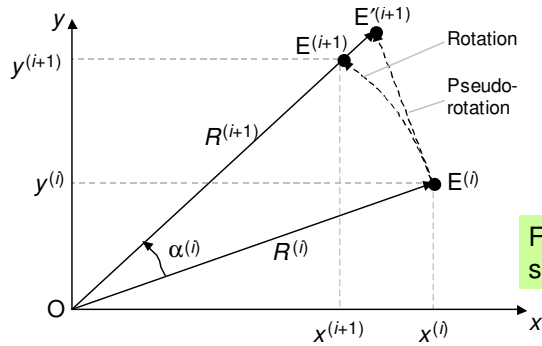
### Rotating a Vector $(x^{(i)}, y^{(i)})$ by the Angle $\alpha^{(i)}$

$$x^{(i+1)} = x^{(i)} \cos \alpha^{(i)} - y^{(i)} \sin \alpha^{(i)} = (x^{(i)} - y^{(i)} \tan \alpha^{(i)}) / (1 + \tan^2 \alpha^{(i)})^{1/2}$$

$$y^{(i+1)} = y^{(i)} \cos \alpha^{(i)} + x^{(i)} \sin \alpha^{(i)} = (y^{(i)} + x^{(i)} \tan \alpha^{(i)}) / (1 + \tan^2 \alpha^{(i)})^{1/2}$$

$$z^{(i+1)} = z^{(i)} - \alpha^{(i)}$$

Recall that  $\cos \theta = 1 / (1 + \tan^2 \theta)^{1/2}$



**Our strategy:** Eliminate the terms  $(1 + \tan^2 \alpha^{(i)})^{1/2}$  and choose the angles  $\alpha^{(i)}$  so that  $\tan \alpha^{(i)}$  is a power of 2; need two shift-adds

Fig. 22.1 A pseudorotation step in CORDIC

Source: Parhami

5

### Pseudorotating a Vector $(x^{(i)}, y^{(i)})$ by the Angle $\alpha^{(i)}$

$$\left. \begin{aligned} x^{(i+1)} &= x^{(i)} - y^{(i)} \tan \alpha^{(i)} \\ y^{(i+1)} &= y^{(i)} + x^{(i)} \tan \alpha^{(i)} \\ z^{(i+1)} &= z^{(i)} - \alpha^{(i)} \end{aligned} \right\}$$

**Pseudorotation:** Whereas a real rotation does not change the length  $R^{(i)}$  of the vector, a pseudorotation step increases its length to:  
 $R^{(i+1)} = R^{(i)} / \cos \alpha^{(i)} = R^{(i)} (1 + \tan^2 \alpha^{(i)})^{1/2}$

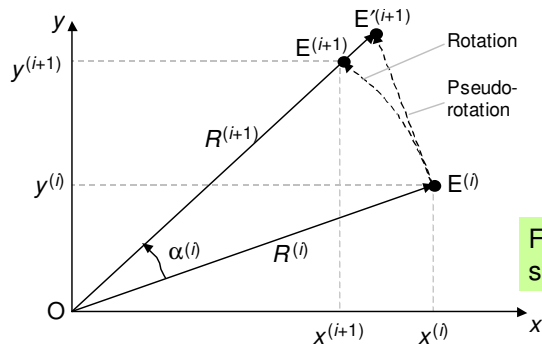


Fig. 22.1 A pseudorotation step in CORDIC

Source: Parhami

6

## A Sequence of Rotations or Pseudorotations

$$\left. \begin{aligned} x^{(m)} &= x \cos(\sum \alpha^{(i)}) - y \sin(\sum \alpha^{(i)}) \\ x^{(m)} &= y \cos(\sum \alpha^{(i)}) + x \sin(\sum \alpha^{(i)}) \\ z^{(m)} &= z - (\sum \alpha^{(i)}) \end{aligned} \right\}$$

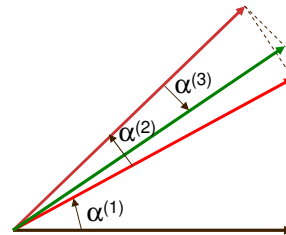
After  $m$  real rotations by  $\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(m)}$ , given  $x^{(0)} = x, y^{(0)} = y$ , and  $z^{(0)} = z$

$$\left. \begin{aligned} x^{(m)} &= K(x \cos(\sum \alpha^{(i)}) - y \sin(\sum \alpha^{(i)})) \\ y^{(m)} &= K(y \cos(\sum \alpha^{(i)}) + x \sin(\sum \alpha^{(i)})) \\ z^{(m)} &= z - (\sum \alpha^{(i)}) \end{aligned} \right\}$$

After  $m$  pseudorotations by  $\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(m)}$ , given  $x^{(0)} = x, y^{(0)} = y$ , and  $z^{(0)} = z$

where  $K = \prod (1 + \tan^2 \alpha^{(i)})^{1/2}$  is a constant if angles of rotation are always the same, differing only in sign or direction

**Question:** Can we find a set of angles so that any angle can be synthesized from all of them with appropriate signs?



Source: Parhami

7

## 22.2 Basic CORDIC Iterations

$$\left. \begin{aligned} x^{(i+1)} &= x^{(i)} - d_i y^{(i)} 2^{-i} \\ y^{(i+1)} &= y^{(i)} + d_i x^{(i)} 2^{-i} \\ z^{(i+1)} &= z^{(i)} - d_i \tan^{-1} 2^{-i} \\ &= z^{(i)} - d_i e^{(i)} \end{aligned} \right\}$$

**CORDIC iteration:** In step  $i$ , we pseudorotate by an angle whose tangent is  $d_i 2^{-i}$  (the angle  $e^{(i)}$  is fixed, only direction  $d_i$  is to be picked)

| $i$ | $e^{(i)}$ in degrees (approximate) | $e^{(i)}$ in radians (precise) |
|-----|------------------------------------|--------------------------------|
| 0   | 45.0                               | 0.785 398 163                  |
| 1   | 26.6                               | 0.463 647 609                  |
| 2   | 14.0                               | 0.244 978 663                  |
| 3   | 7.1                                | 0.124 354 994                  |
| 4   | 3.6                                | 0.062 418 810                  |
| 5   | 1.8                                | 0.031 239 833                  |
| 6   | 0.9                                | 0.015 623 728                  |
| 7   | 0.4                                | 0.007 812 341                  |
| 8   | 0.2                                | 0.003 906 230                  |
| 9   | 0.1                                | 0.001 953 123                  |

Table 22.1 Value of the function  $e^{(i)} = \tan^{-1} 2^{-i}$ , in degrees and radians, for  $0 \leq i \leq 9$

Example:  $30^\circ$  angle

$$\begin{aligned} 30.0 &\cong 45.0 - 26.6 + 14.0 \\ &\quad - 7.1 + 3.6 + 1.8 \\ &\quad - 0.9 + 0.4 - 0.2 \\ &\quad + 0.1 \\ &= 30.1 \end{aligned}$$

Source: Parhami

8

### Choosing the Angles to Force z to Zero

$$\begin{aligned} x^{(i+1)} &= x^{(i)} - d_i y^{(i)} 2^{-i} \\ y^{(i+1)} &= y^{(i)} + d_i x^{(i)} 2^{-i} \\ z^{(i+1)} &= z^{(i)} - d_i \tan^{-1} 2^{-i} \\ &= z^{(i)} - d_i e^{(i)} \end{aligned}$$

| $i$ | $z^{(i)}$ | $-$ | $d_i e^{(i)}$ | $=$ | $z^{(i+1)}$ |
|-----|-----------|-----|---------------|-----|-------------|
| 0   | +30.0     | -   | 45.0          | =   | +30.0       |
| 1   | -15.0     | +   | 26.6          | =   | -15.0       |
| 2   | +11.6     | -   | 14.0          | =   | +11.6       |
| 3   | -2.4      | +   | 7.1           | =   | -2.4        |
| 4   | +4.7      | -   | 3.6           | =   | +4.7        |
| 5   | +1.1      | -   | 1.8           | =   | +1.1        |
| 6   | -0.7      | +   | 0.9           | =   | -0.7        |
| 7   | +0.2      | -   | 0.4           | =   | +0.2        |
| 8   | -0.2      | +   | 0.2           | =   | -0.2        |
| 9   | +0.0      | -   | 0.1           | =   | +0.0        |

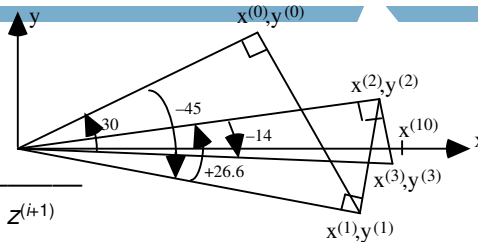


Fig. 22.2 The first three of 10 pseudorotations leading from  $(x^{(0)}, y^{(0)})$  to  $(x^{(10)}, 0)$  in rotating by  $+30^\circ$ .

Table 22.2 Choosing the signs of the rotation angles in order to force  $z$  to 0

Source: Parhami

9

### Why Any Angle Can Be Formed from Our List

**Analogy:** Paying a certain amount while using all currency denominations (in positive or negative direction) exactly once; red values are fictitious.

\$20 \$10 \$5 **\$3** \$2 \$1 \$.50 \$.25 **\$.20** \$.10 \$.05 **\$.03** **\$.02** \$.01

**Example:** Pay \$12.50

\$20 - \$10 + \$5 - **\$3** + \$2 - \$1 - \$.50 + \$.25 - **\$.20** - \$.10 + \$.05 + **\$.03** - **\$.02** - \$.01

Convergence is possible as long as each denomination is no greater than the sum of all denominations that follow it.

Domain of convergence:  $-\$42.16$  to  $+\$42.16$

We can guarantee convergence with actual denominations if we allow multiple steps at some values:

\$20 \$10 \$5 **\$2** **\$2** \$1 \$.50 \$.25 **\$.10** **\$.10** \$.05 **\$.01** **\$.01** **\$.01** **\$.01**

**Example:** Pay \$12.50

\$20 - \$10 + \$5 - **\$2** - **\$2** + \$1 + \$.50 + \$.25 - **\$.10** - **\$.10** - \$.05 + **\$.01** - **\$.01** + **\$.01** - **\$.01**

We will see later that in hyperbolic CORDIC, convergence is guaranteed only if certain "angles" are used twice.

Source: Parhami

10

### Using CORDIC in Rotation Mode

$$x^{(i+1)} = x^{(i)} - d_i y^{(i)} 2^{-i}$$

$$y^{(i+1)} = y^{(i)} + d_i x^{(i)} 2^{-i}$$

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

$$= z^{(i)} - d_i e^{(i)}$$

Make  $z$   
converge to 0  
by choosing  
 $d_i = \text{sign}(z^{(i)})$

$$x^{(m)} = K(x \cos z - y \sin z)$$

$$y^{(m)} = K(y \cos z + x \sin z)$$

$$z^{(m)} = 0$$

where  $K = 1.646\ 760\ 258\ 121 \dots$

For  $k$  bits of precision in results,  
 $k$  CORDIC iterations are needed,  
because  $\tan^{-1} 2^{-i} \cong 2^{-i}$  for large  $i$

Start with

$$x = 1/K = 0.607\ 252\ 935 \dots$$

and  $y = 0$

to find  $\cos z$  and  $\sin z$

Convergence of  $z$  to 0 is possible because each of the angles  
in our list is more than half the previous one or, equivalently,  
each is less than the sum of all the angles that follow it

Domain of convergence is  $-99.7^\circ \leq z \leq 99.7^\circ$ , where  $99.7^\circ$  is the sum  
of all the angles in our list; the domain contains  $[-\pi/2, \pi/2]$  radians

Source: Parhami

11

### Using CORDIC in Vectoring Mode

$$x^{(i+1)} = x^{(i)} - d_i y^{(i)} 2^{-i}$$

$$y^{(i+1)} = y^{(i)} + d_i x^{(i)} 2^{-i}$$

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

$$= z^{(i)} - d_i e^{(i)}$$

Make  $y$  converge  
to 0 by choosing  
 $d_i = -\text{sign}(x^{(i)} y^{(i)})$

$$x^{(m)} = K(x^2 + y^2)^{1/2}$$

$$y^{(m)} = 0$$

$$z^{(m)} = z + \tan^{-1}(y/x)$$

where  $K = 1.646\ 760\ 258\ 121 \dots$

For  $k$  bits of precision in results,  
 $k$  CORDIC iterations are needed,  
because  $\tan^{-1} 2^{-i} \cong 2^{-i}$  for large  $i$

Start with

$$x = 1 \text{ and } z = 0$$

to find  $\tan^{-1} y$

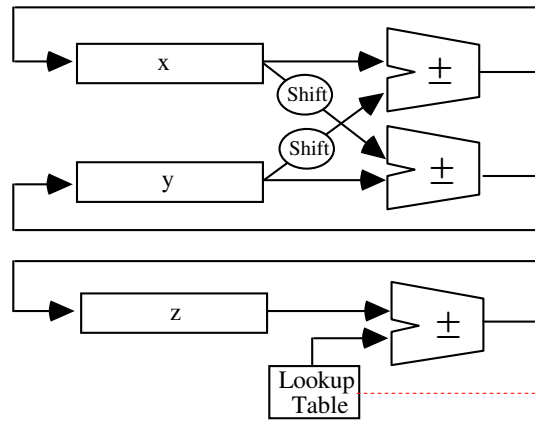
Even though the computation above always converges,  
one can use the relationship  $\tan^{-1}(1/y) = \pi/2 - \tan^{-1} y$   
to limit the range of fixed-point numbers encountered

Other trig functions:  $\tan z$  obtained from  $\sin z$  and  $\cos z$  via division;  
inverse sine and cosine ( $\sin^{-1} z$  and  $\cos^{-1} z$ ) discussed later

Source: Parhami

12

### 22.3 CORDIC Hardware



$$\begin{aligned}
 x^{(i+1)} &= x^{(i)} - d_i y^{(i)} 2^{-i} \\
 y^{(i+1)} &= y^{(i)} + d_i x^{(i)} 2^{-i} \\
 z^{(i+1)} &= z^{(i)} - d_i \tan^{-1} 2^{-i} \\
 &= z^{(i)} - d_i e^{(i)}
 \end{aligned}$$

If very high speed is not needed (as in a calculator), a single adder and one shifter would suffice

k table entries for k bits of precision

Fig. 22.3 Hardware elements needed for the CORDIC method.

Source: Parhami

13

### 22.4 Generalized CORDIC

$$\begin{aligned}
 x^{(i+1)} &= x^{(i)} - \mu d_i y^{(i)} 2^{-i} \\
 y^{(i+1)} &= y^{(i)} + d_i x^{(i)} 2^{-i} \\
 z^{(i+1)} &= z^{(i)} - d_i e^{(i)}
 \end{aligned}$$

$\mu = 1$  Circular rotations  
(basic CORDIC)  
 $e^{(i)} = \tan^{-1} 2^{-i}$

$\mu = 0$  Linear rotations  
 $e^{(i)} = 2^{-i}$

$\mu = -1$  Hyperbolic rotations  
 $e^{(i)} = \tanh^{-1} 2^{-i}$

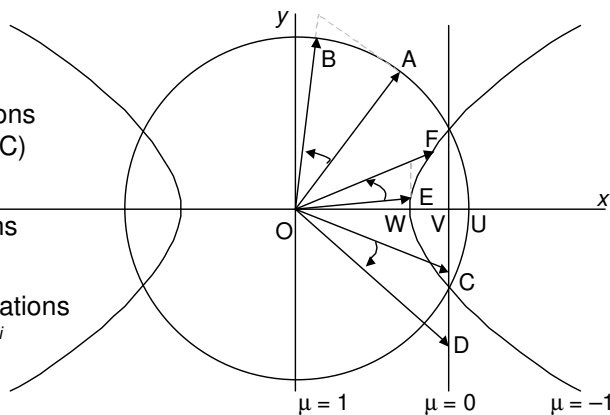


Fig. 22.4 Circular, linear, and hyperbolic CORDIC.

Source: Parhami

14

## 22.5 Using the CORDIC Method

|  |  |  |  |
|--|--|--|--|
| $x^{(i+1)} = x^{(i)} - \mu d_i y^{(i)} 2^{-i}$ $y^{(i+1)} = y^{(i)} + d_i x^{(i)} 2^{-i}$ $z^{(i+1)} = z^{(i)} - d_i e^{(i)}$<br>$\mu \in \{-1, 0, 1\}$ $d_i \in \{-1, 1\}$ $K = 1.646\ 760\ 258\ 121\ \dots$ $1/K = .607\ 252\ 935\ 009\ \dots$ $K' = .828\ 159\ 360\ 960\ 2\ \dots$ $1/K' = 1.207\ 497\ 067\ 763\ \dots$ | Mode $\rightarrow$   | Rotation: $d_i = \text{sign}(z^{(i)})$ , $z^{(i)} \rightarrow 0$   | Vectoring: $d_i = -\text{sign}(x^{(i)} y^{(i)})$ , $y^{(i)} \rightarrow 0$   |
|  | $\mu = 1$<br>Circular<br><br>$e^{(i)} = \tan^{-1} 2^{-i}$  | <br>For cos & sin, set $x = 1/K$ , $y = 0$<br>$\tan z = \sin z / \cos z$   | <br>For $\tan^{-1}$ , set $x = 1$ , $z = 0$<br>$\cos^{-1} w = \tan^{-1}[\sqrt{1-w^2}/w]$<br>$\sin^{-1} w = \tan^{-1}[w/\sqrt{1-w^2}]$  |
|  | $\mu = 0$<br>Linear<br><br>$e^{(i)} = 2^{-i}$  | <br>For multiplication, set $y = 0$  | <br>For division, set $z = 0$  |
|  | $\mu = -1$<br>Hyperbolic<br><br>$e^{(i)} = \tanh^{-1} 2^{-i}$  | <br>For cosh & sinh, set $x = 1/K'$ , $y = 0$<br>$\tanh z = \sinh z / \cosh z$<br>$\exp(z) = \sinh z + \cosh z$<br>$w^t = \exp(t \ln w)$ | <br>For $\tanh^{-1}$ , set $x = 1$ , $z = 0$<br>$\ln w = 2 \tanh^{-1}[(w-1)/(w+1)]$<br>$\sqrt{w} = \sqrt{(w+1/4)^2 - (w-1/4)^2}$<br>$\cosh^{-1} w = \ln(w + \sqrt{1-w^2})$<br>$\sinh^{-1} w = \ln(w + \sqrt{1+w^2})$ |
| Note $\rightarrow$   | In executing the iterations for $\mu = -1$ , steps 4, 13, 40, 121, $\dots$ , $j$ , $3j+1$ , $\dots$ must be repeated. These repetitions are incorporated in the constant $K'$ below. |  |  |

Fig. 22.5 Summary of generalized CORDIC algorithms.

Source: Parhami

15

## CORDIC Speedup Methods

$$x^{(i+1)} = x^{(i)} - \mu d_i y^{(i)} 2^{-i}$$

$$y^{(i+1)} = y^{(i)} + d_i x^{(i)} 2^{-i}$$

$$z^{(i+1)} = z^{(i)} - d_i e^{(i)}$$

### Skipping some rotations

Must keep track of expansion via the recurrence:

$$(K^{(i+1)})^2 = (K^{(i)})^2 (1 \pm 2^{-2i})$$

This additional work makes *variable-factor* CORDIC less cost-effective than *constant-factor* CORDIC

$$x^{(k)} = x^{(k/2)} - y^{(k/2)} z^{(k/2)}$$

$$y^{(k)} = y^{(k/2)} + x^{(k/2)} z^{(k/2)}$$

$$z^{(k)} = z^{(k/2)} - z^{(k/2)}$$

### Early termination

Do the first  $k/2$  iterations as usual, then combine the remaining  $k/2$  into a single multiplicative step:

For very small  $z$ , we have  $\tan^{-1} z \cong z \cong \tan z$

Expansion factor not an issue because contribution of the ignored terms is provably less than *ulp*

$$d_i \in \{-2, -1, 1, 2\} \text{ or}$$

$$\{-2, -1, 0, 1, 2\}$$

### High-radix CORDIC

Source: Parhami

16



# Direct Digital Frequency Synthesis

17

## Overview of Frequency Synthesizers

- A frequency synthesizer is a device which generates many output frequencies from a single input reference frequency using direct, indirect, or digital synthesis techniques
- Three different types of frequency synthesizers
  - Indirect Frequency Synthesizer
    - Produce an output frequency from a secondary oscillator frequency, usually a voltage controlled oscillator (VCO) phase locked to a primary frequency
  - Direct Frequency Synthesizer
    - Produces multiple output frequencies from a single frequency standard using a series of mixing, multiplication, division, and filtering stages
  - Direct Digital Frequency Synthesizer also called a Numerically Controlled Oscillator (NCO)
    - A digital synthesizer, as suggested by its name, utilizes digital circuitry to generate output frequencies
    - Direct Digital Frequency Synthesizer first describe in a paper by J. Tierney, C. M. Rader, and B. Gold in 1971

Courtesy: D. Wilson

18

## Advantages and Disadvantages of a DDFS

- Advantages
  - Fine Frequency Resolution (sub-Hertz)
  - Lower Power Consumption
  - Fast Switching Speed
  - Wide Tuning Bandwidth
  - Low Phase Noise
  - Continuous Phase Switching Response
- Disadvantages
  - A DDFS generates a sinc(x) output frequency spectrum containing the desired output frequency plus harmonics which must be filtered out
  - A DDFS produces spurious frequencies or spurs resulting from phase word truncation and imperfections in the digital-to-analog converter (DAC)
  - A DDFS requires a digital-to-analog converter (DAC)
    - DACs are the greatest cause of spurs in high-speed and high-resolution (>10 bits, >50 MHz) DDFS applications
    - DACs susceptible to spurs created by clock feedthrough, intermodulation, and glitch energy

Courtesy: D. Wilson

19

## DDFS Applications

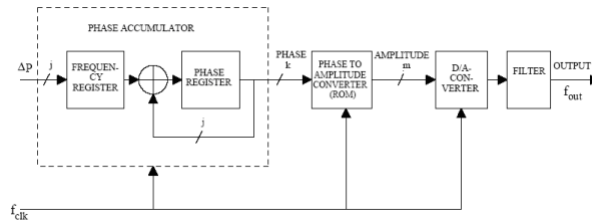
- Satellite communications systems
  - Example given earlier of a space qualified DDFS used in a GPS satellite RVS Atomic Clock design
- Medical imaging applications
- RADAR
- Spread spectrum frequency hopping communication systems
- Software radio

Courtesy: D. Wilson

20

## Basic DDFS Architecture

- Structure of a DDFS is fairly simple
  - Major components are: a Phase Accumulator, Phase to Amplitude Converter, a D/A Converter, and a Low Pass or Inverse Sinc Filter



Courtesy: D. Wilson

21

## Basic DDFS Architecture

- Phase accumulator
  - Typically 32-bits or 48-bits
  - DDFS output frequency is equal to the phase accumulator overflow rate
  - Phase accumulator overflow rate is determined by the phase increment word (also called the frequency control word) stored in frequency register
  - DDFS output frequency calculated by following equation:

$$f_{\text{out}} = (\Delta P \cdot f_{\text{clk}}) / 2^j$$

$\Delta P$  is the phase increment word,  $f_{\text{clk}}$  is the DDFS clock frequency, and  $j$  is the phase accumulator number of bits

- DDFS frequency resolution is  $f_{\text{out}}$  calculated for  $\Delta P$  equal to 1
- The DDFS bandwidth is  $f_{\text{clk}}/2$

Courtesy: D. Wilson

22

## Basic DDFS Architecture

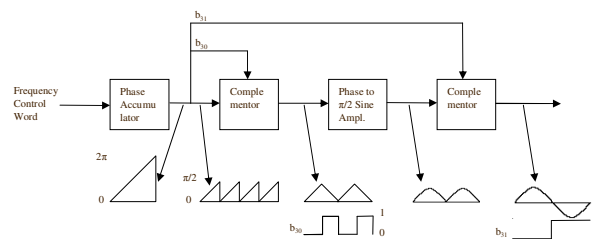
- Phase to Amplitude Converter
  - Usually a sine ROM LUT containing digitized samples of sine function
  - If all 32-bits of the phase accumulator were used to address a sine ROM LUT with a 12 bit word width the storage capacity would be  $2^{32} \times 12$  bits or 5.154 giga-bits
  - A ROM 5.154 giga-bits is obviously impractical for most applications so phase accumulator is typically truncated to 16 most significant bits
- DAC
  - Produces a quantized sinusoidal output
- Low pass filter
  - Removes high frequency harmonics from quantized sinusoidal output to produce a spectrally pure output

Courtesy: D. Wilson

23

## Hutchison DDFS

- 32-bit Phase Accumulator truncated to 14 MSB's
- 50 MHz clock Frequency, 0.0116 Hz frequency resolution, and 25 MHz bandwidth
- Two MSB's control quadrant select logic
- Remaining 12 bits input to the phase to  $\pi/2$  sine amplitude block



Hutchison DDFS Block Diagram and Signal Flow

Courtesy: D. Wilson

24

## Hutchison DDFS

- If 12 bits were used to address a single sine ROM LUT the storage capacity would be 4K
- The 4K sine ROM LUT storage capacity reduced by partitioning the sine ROM LUT into a coarse angle ( $\theta_C$ ) ROM and a fine angle ( $\theta_F$ ) ROM
- The fine and coarse ROM contents are determined from the trigonometric equation  $\sin(\theta) = \sin(\theta_C)\cos(\theta_F) + \cos(\theta_C)\sin(\theta_F)$ 
  - If  $\theta_F$  is assumed to be small, which is a valid assumption in this case since  $\theta_F < (\pi/2)(1/256)$ , then above equation simplifies to:

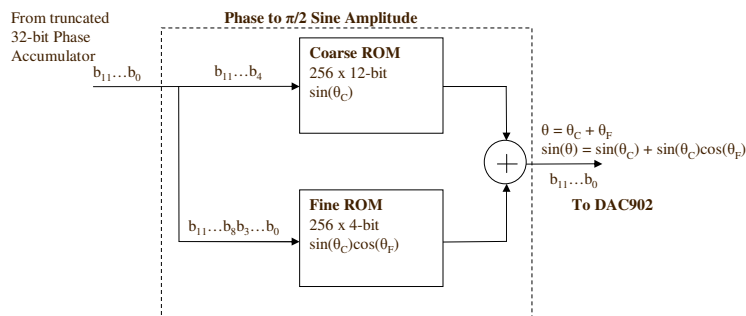
$$\cos(\theta_F) \approx 1 \text{ and } \sin(\theta) \approx \sin(\theta_C) + \cos(\theta_C)\sin(\theta_F)$$

- The values for  $\sin(\theta_C)$  and  $\cos(\theta_C)\sin(\theta_F)$  are stored in a 256 x 12-bit coarse angle ROM and 256 x 4-bit fine angle ROM respectively

Courtesy: D. Wilson

25

## Hutchison Phase to $\pi/2$ Sine Amplitude Converter



Courtesy: D. Wilson

26

## Sunderland DDFS

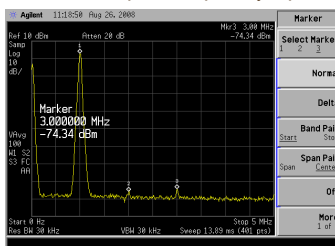
- The Sunderland DDFS architecture is identical to the Hutchison DDFS architecture
- The only difference is the coarse and fine angle ROM contents
- Sunderland DDFS divides phase angle  $\theta$  for  $0 \leq \theta \leq \pi/2$  into three angles,  $\theta_C$ ,  $\theta_S$ , and  $\theta_F$ 
  - $\theta_C = (\pi/2)(A/2^4)$  for  $0 \leq A \leq 2^4 - 1$
  - $\theta_S = (\pi/2)(B/2^3)$  for  $0 \leq B \leq 2^4 - 1$
  - $\theta_F = (\pi/2)(C/2^{12})$  for  $0 \leq C \leq 2^4 - 1$
- The Sunderland coarse and fine angle ROM contents are calculated from the following trigonometric equation:
  - $\sin(\theta) = \sin(\theta_C + \theta_S + \theta_F)$
  - $= \sin(\theta_C + \theta_S)\cos(\theta_F) + \cos(\theta_C)\cos(\theta_S)\sin(\theta_F) - \sin(\theta_C)\sin(\theta_S)\sin(\theta_F)$
- $\sin(\theta) \approx \sin(\theta_C + \theta_S) + \cos(\theta_C + \theta_{Savg})\sin(\theta_F + \theta_{offset})$
- where  $\theta_{Savg} = (1/16)(0/256 + 1/256 + \dots + 16/256) = 1/32$  and  $\theta_{offset} = (\pi/2)(2^{-13})$
- The Sunderland coarse angle ROM contains the quantity  $\sin(\theta_C + \theta_S)$  and the fine angle ROM contains  $\cos(\theta_C + \theta_{Savg})\sin(\theta_F + \theta_{offset})$

Courtesy: D. Wilson

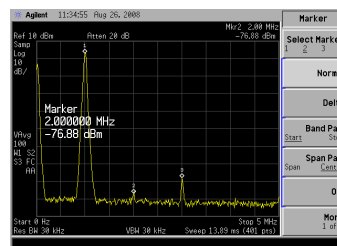
27

## DDFS Spectrum Analyzer and Oscilloscope Displays

- DDFS Output Frequency Spectrum at 1 MHz



Hutchison DDFS



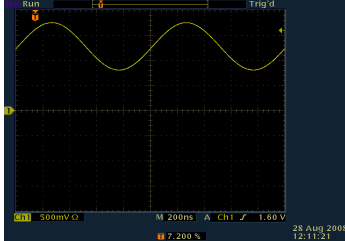
Sunderland DDFS

Courtesy: D. Wilson

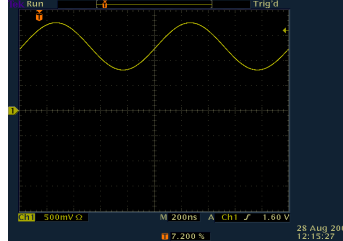
28

## DDFS Spectrum Analyzer and Oscilloscope Displays

- DDFS Oscilloscope Output at 1 MHz



Hutchison DDFS



Sunderland DDFS

Courtesy: D. Wilson

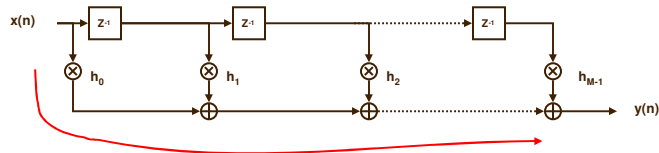
29



## Bit-Level Arithmetic: Ripple-Carry Adder Trees

30

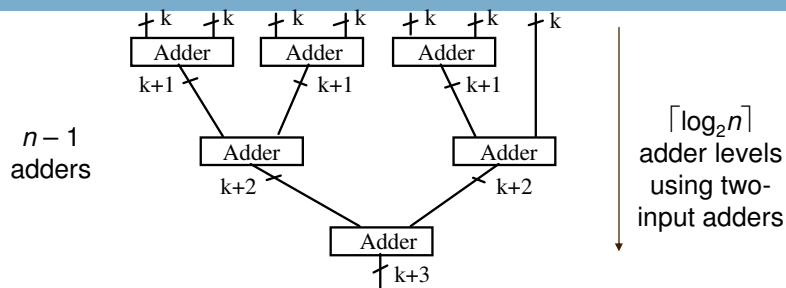
## Direct Form FIR Filters



- M-tap FIR filter in direct form
- Critical path:
  - $T_A$  = delay through adder
  - $T_M$  = delay through multiplier
  - Critical path delay:  $1 T_M + (M-1) T_A$
- What is the true critical path delay at the implementation level?

31

## Binary Adder Trees



$$T_{\text{tree-ripple-multi-add}} = O(k + (k + 1) + \dots + (k + \lceil \log_2 n \rceil - 1))$$

explanation:  $\log_2(n)$  stages, each stage increasing by one bit

This can be optimized to:

$$= O(k + \log n)$$

[Justified on the next slide]

Source: Parhami

32

## Elaboration on Tree of Ripple-Carry Adders

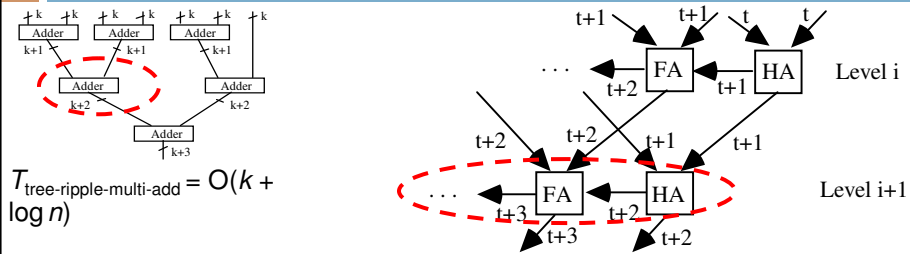


Fig. 8.5 Ripple-carry adders at levels  $i$  and  $i + 1$  in the tree of adders used for multi-operand addition.

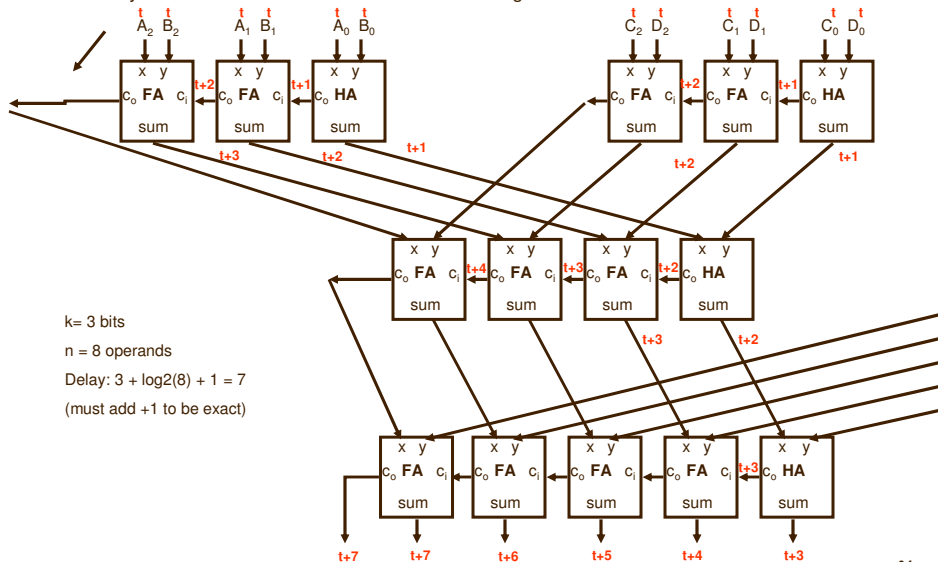
- Sometimes ripple-carry tree can have advantage over fast adder in that the computations can "overlap" in time between levels
- May or may not be true with fast adders, depending on architecture
  - In general, fast adders need previous stage to complete before going to next stage

Source: Parhami

33

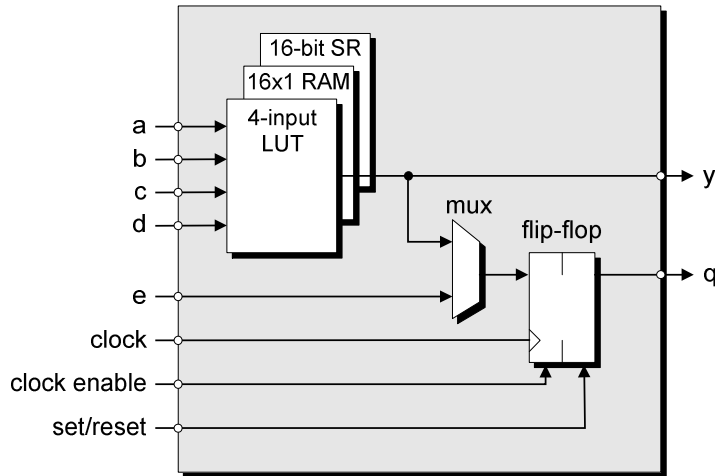
## Example: Adding 8 three bit numbers

recall: carry-out becomes the msb of the new sum in unsigned arithmetic



34

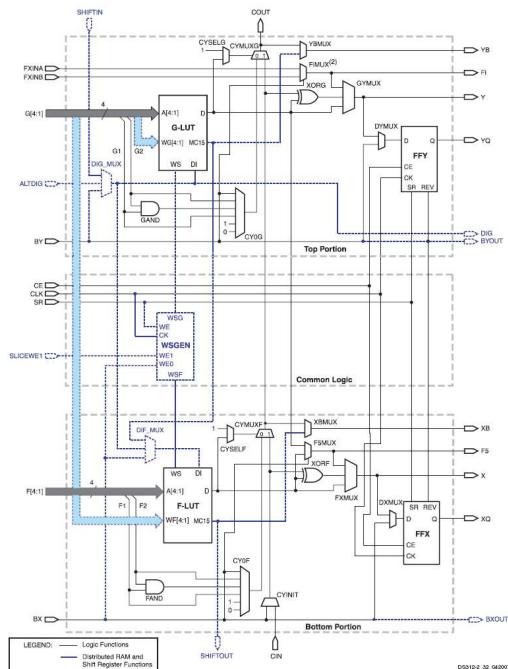
## Simplified view of a Xilinx Logic Cell



The Design Warrior's Guide to FPGAs  
 Devices, Tools, and Flows. ISBN 0750676043  
 Copyright © 2004 Mentor Graphics Corp. (www.mentor.com)

35

## Spartan 3 Slice



LEGEND:  
 — Logic Functions  
 — Distributed RAM and Shift Register Functions

DS10-2-36-0007  
 36

## Ripple-Carry Adders in Xilinx FPGAs

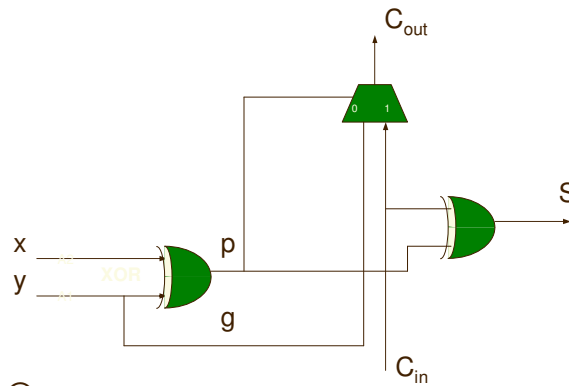
Implementation used to generate fast carry logic  
in Xilinx FPGAs

| x | y | $c_{out}$ |
|---|---|-----------|
| 0 | 0 | $y$       |
| 0 | 1 | $c_{in}$  |
| 1 | 0 | $c_{in}$  |
| 1 | 1 | $y$       |

$$p = x \oplus y$$

$$g = y$$

$$s = p \oplus c_{in} = x \oplus y \oplus c_{in}$$



37

## Bit-Level Arithmetic: Multiplication

38

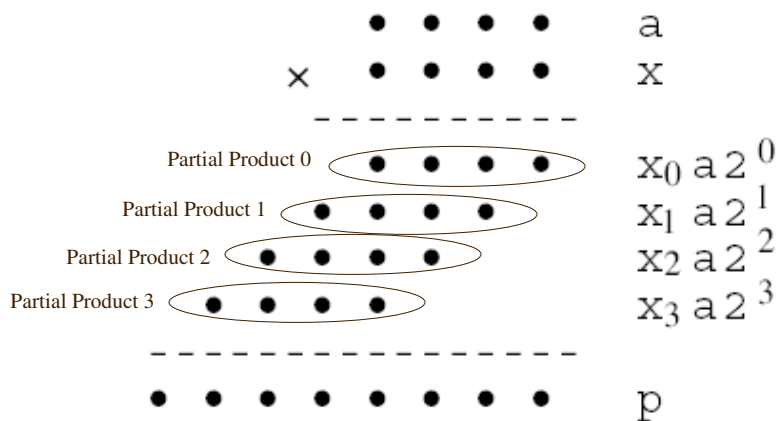
## Notation

|   |                 |                                       |
|---|-----------------|---------------------------------------|
| a | Multiplicand    | $a_{k-1} a_{k-2} \dots a_1 a_0$       |
| x | Multiplier      | $x_{k-1} x_{k-2} \dots x_1 x_0$       |
| p | Product (a · x) | $p_{2k-1} p_{2k-2} \dots p_2 p_1 p_0$ |

If multiplicand and multiplier different sizes, usually multiplier is the smaller size

39

## Multiplication of Two 4-bit Unsigned Binary Numbers in Dot Notation



Number of partial products = number of bits in multiplier x  
 Bit-width of each partial product = bit-width of multiplicand a

40

## Basic Multiplication Equations

$$p = a \cdot x \qquad x = \sum_{i=0}^{k-1} x_i \cdot 2^i$$

$$\begin{aligned} p = a \cdot x &= \sum_{i=0}^{k-1} a \cdot x_i \cdot 2^i = \\ &= x_0 a 2^0 + x_1 a 2^1 + x_2 a 2^2 + \dots + x_{k-1} a 2^{k-1} \end{aligned}$$

41

## Unsigned Multiplication

|        |       |   |       |           |           |           |           |           |       |       |       |       |
|--------|-------|---|-------|-----------|-----------|-----------|-----------|-----------|-------|-------|-------|-------|
|        |       |   |       | $a_4$     | $a_3$     | $a_2$     | $a_1$     | $a_0$     |       |       |       |       |
|        |       |   | $x$   | $x_4$     | $x_3$     | $x_2$     | $x_1$     | $x_0$     |       |       |       |       |
|        |       |   |       |           |           |           |           |           |       |       |       |       |
| $ax_0$ | $2^0$ |   |       | $a_4 x_0$ | $a_3 x_0$ | $a_2 x_0$ | $a_1 x_0$ | $a_0 x_0$ |       |       |       |       |
| $ax_1$ | $2^1$ | + |       | $a_4 x_1$ | $a_3 x_1$ | $a_2 x_1$ | $a_1 x_1$ | $a_0 x_1$ |       |       |       |       |
| $ax_2$ | $2^2$ |   |       | $a_4 x_2$ | $a_3 x_2$ | $a_2 x_2$ | $a_1 x_2$ | $a_0 x_2$ |       |       |       |       |
| $ax_3$ | $2^3$ |   |       | $a_4 x_3$ | $a_3 x_3$ | $a_2 x_3$ | $a_1 x_3$ | $a_0 x_3$ |       |       |       |       |
| $ax_4$ | $2^4$ |   |       | $a_4 x_4$ | $a_3 x_4$ | $a_2 x_4$ | $a_1 x_4$ | $a_0 x_4$ |       |       |       |       |
|        |       |   |       |           |           |           |           |           |       |       |       |       |
|        |       |   | $p_9$ | $p_8$     | $p_7$     | $p_6$     | $p_5$     | $p_4$     | $p_3$ | $p_2$ | $p_1$ | $p_0$ |

42

## Two's Complement Multiplication

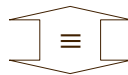
| ----- Extended positions ----- |           |           |           |           | Sign      | Magnitude positions ----- |           |           |         |  |
|--------------------------------|-----------|-----------|-----------|-----------|-----------|---------------------------|-----------|-----------|---------|--|
| $x_{k-1}$                      | $x_{k-1}$ | $x_{k-1}$ | $x_{k-1}$ | $x_{k-1}$ | $x_{k-1}$ | $x_{k-2}$                 | $x_{k-3}$ | $x_{k-4}$ | $\dots$ |  |
| $y_{k-1}$                      | $y_{k-1}$ | $y_{k-1}$ | $y_{k-1}$ | $y_{k-1}$ | $y_{k-1}$ | $y_{k-2}$                 | $y_{k-3}$ | $y_{k-4}$ | $\dots$ |  |
| $z_{k-1}$                      | $z_{k-1}$ | $z_{k-1}$ | $z_{k-1}$ | $z_{k-1}$ | $z_{k-1}$ | $z_{k-2}$                 | $z_{k-3}$ | $z_{k-4}$ | $\dots$ |  |

- Recall, for two's complement multi-operand addition, must first sign extend all partial products to result bit-width, then add
- This can be wasteful in terms of extra full adder cells requires
- Two solutions:
  - Remove redundant full adder cells for sign extension
  - Use "negative weight" representation  $\rightarrow$  Baugh-Wooley multiplier

43

## Two's Complement Negative Weight Representation

$$\begin{array}{rcccccc}
 & & -2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
 & & \mathbf{a}_4 & a_3 & a_2 & a_1 & a_0 \\
 x & & \mathbf{x}_4 & x_3 & x_2 & x_1 & x_0 \\
 \hline
 \end{array}$$



$$\begin{array}{rcccccc}
 & & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
 & & -\mathbf{a}_4 & a_3 & a_2 & a_1 & a_0 \\
 x & & -\mathbf{x}_4 & x_3 & x_2 & x_1 & x_0 \\
 \hline
 \end{array}$$

44



## Implementing Partial Products

$$\bar{z} = 1 - z$$

$$z = 1 - \bar{z}$$

$$-a_j x_i = -a_j (1 - \bar{x}_i) = a_j \bar{x}_i - a_j = a_j \bar{x}_i + a_j - 2 a_j$$

$$-a_j x_i = -(1 - \bar{a}_j) x_i = \bar{a}_j x_i - x_i = \bar{a}_j x_i + x_i - 2 x_i$$

$$-a_j x_i = -(1 - \bar{a}_j \bar{x}_i) = \bar{a}_j \bar{x}_i - 1 = \bar{a}_j \bar{x}_i + 1 - 2$$

$$-a_j = -(1 - \bar{a}_j) = \bar{a}_j - 1 = \bar{a}_j + 1 - 2$$

$$-x_i = -(1 - \bar{x}_i) = \bar{x}_i - 1 = \bar{x}_i + 1 - 2$$

47

$$-a_4 x_0$$

$$-a_4 x_1$$

$$-a_4 x_2$$

$$+ \quad -a_4 x_3$$


---

|        |                 |       |  |  |
|--------|-----------------|-------|--|--|
| $-a_4$ | $a_4 \bar{x}_0$ |       |  |  |
| $-a_4$ | $a_4 \bar{x}_1$ | $a_4$ |  |  |
| $-a_4$ | $a_4 \bar{x}_2$ | $a_4$ |  |  |
| $-a_4$ | $a_4 \bar{x}_3$ | $a_4$ |  |  |
|        | $a_4$           |       |  |  |

---

|             |                 |                 |                 |                 |
|-------------|-----------------|-----------------|-----------------|-----------------|
| $\bar{a}_4$ | $a_4 \bar{x}_3$ | $a_4 \bar{x}_2$ | $a_4 \bar{x}_1$ | $a_4 \bar{x}_0$ |
| $-1$        |                 |                 |                 | $a_4$           |

48

$$\begin{array}{r}
 + \quad -a_3x_4 - a_2x_4 - a_1x_4 - a_0x_4 \\
 \hline
 \begin{array}{cccc}
 & & -x_4 & \bar{a}_0x_4 \\
 & & -x_4 & \bar{a}_1x_4 & x_4 \\
 & -x_4 & \bar{a}_2x_4 & x_4 \\
 -x_4 & \bar{a}_3x_4 & x_4 & \\
 & x_4 & & 
 \end{array} \\
 \hline
 \bar{x}_4 & \bar{a}_3x_4 & \bar{a}_2x_4 & \bar{a}_1x_4 & \bar{a}_0x_4 \\
 -1 & & & & x_4
 \end{array}$$

49

|        |             |                |                |                |                |                |
|--------|-------------|----------------|----------------|----------------|----------------|----------------|
| $2^9$  | $2^8$       | $2^7$          | $2^6$          | $2^5$          | $2^4$          |                |
|        | $\bar{a}_4$ | $a_4\bar{x}_3$ | $a_4\bar{x}_2$ | $a_4\bar{x}_1$ | $a_4\bar{x}_0$ |                |
|        | $-1$        |                |                | $a_4$          |                |                |
|        | $\bar{x}_4$ | $\bar{a}_3x_4$ | $\bar{a}_2x_4$ | $\bar{a}_1x_4$ | $\bar{a}_0x_4$ |                |
|        | $-1$        |                |                | $x_4$          |                |                |
| $-1$   | $\bar{a}_4$ | $a_4\bar{x}_3$ | $a_4\bar{x}_2$ | $a_4\bar{x}_1$ | $a_4\bar{x}_0$ |                |
|        | $\bar{x}_4$ | $\bar{a}_3x_4$ | $\bar{a}_2x_4$ | $\bar{a}_1x_4$ | $\bar{a}_0x_4$ |                |
|        |             |                |                | $a_4$          |                |                |
|        |             |                |                | $x_4$          |                |                |
|        | $1$         | $\bar{a}_4$    | $a_4\bar{x}_3$ | $a_4\bar{x}_2$ | $a_4\bar{x}_1$ | $a_4\bar{x}_0$ |
|        |             | $\bar{x}_4$    | $\bar{a}_3x_4$ | $\bar{a}_2x_4$ | $\bar{a}_1x_4$ | $\bar{a}_0x_4$ |
|        |             |                |                | $a_4$          |                |                |
|        |             |                |                | $x_4$          |                |                |
| $-2^9$ |             |                |                |                |                |                |

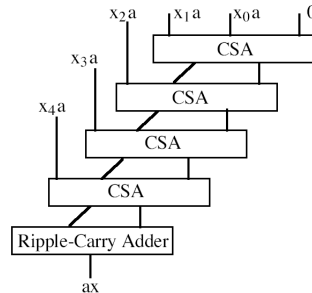
combine

remap sign bit to negative weight

50



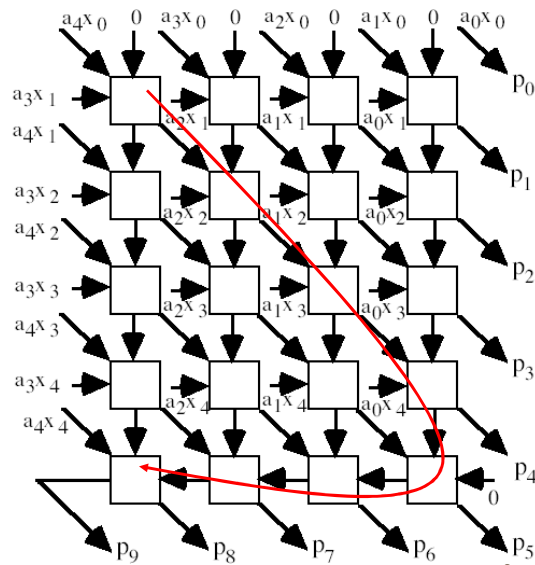
## Basic 5 x 5 Unsigned Array Multiplier



- Critical path is longer
  - For M-bit multiplier and M-bit multiplicand, critical path passes through approximately  $(M - 1) + (M - 1)$  full adder cells
- Also good for computing  $ax + y$ 
  - Replace zero on top CSA with  $y$
  - Helpful for inner product calculations and multiply-accumulate functions

53

## 5 x 5 Array Multiplier



Critical path  
(assuming sum  
and carry delays  
the same)

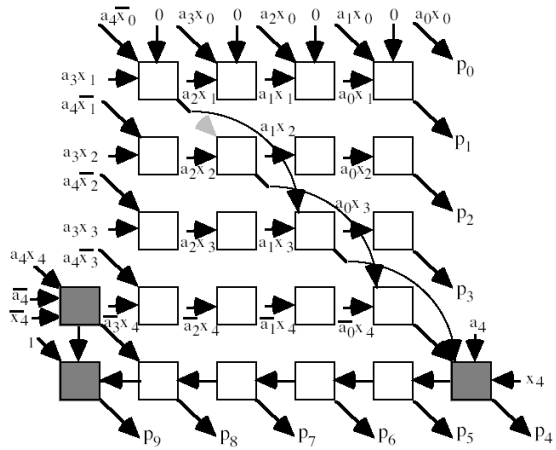
Source: Parhami

54



## Modifications in 5 x 5 Array Multiplier

- Shaded cells indicate additional cells for Baugh-Wooley
- When sum logic longer delay than carry logic, critical path goes through diagonal then along last row
  - Can reduce this by causing some sum signals to skip rows
  - These are shown as curved arcs in figure



Source: Parhami