

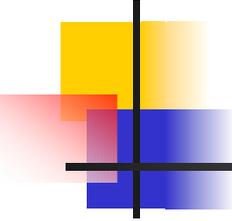
# Hypercomputing With the CORDIC Algorithm

---

Kristen Barr

Shaun Foley

Robert A. Lewis II



# Presentation Overview

---

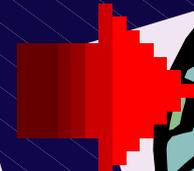
- Background of HAL project and CORDIC algorithm
- How the CORDIC algorithm works
- Demonstration of Viva
- Summary and conclusions
- Special Recognitions
- Questions and Answers

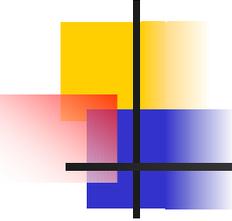
As the world surged towards the millennium 10 years ago

1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000



- Catastrophe!
- Computer Failure!
- Cataclysms!

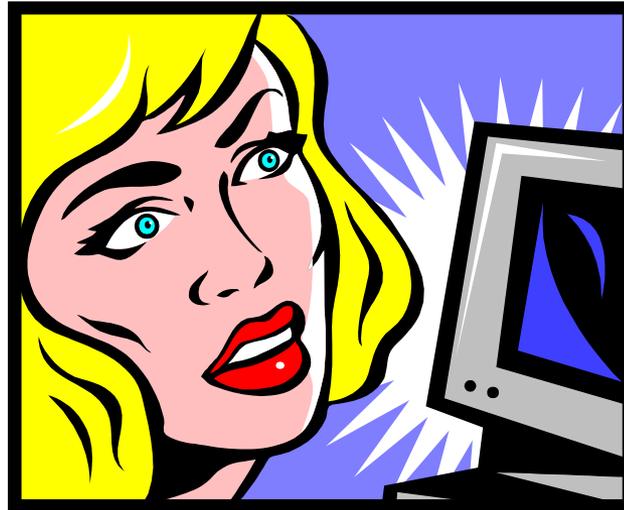




# So what does this mean?

---

Soaring without limits...  
Have we reached our limits  
When will we go  
for the ride again

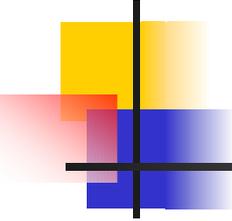


# “Re-configurable Computing”,

A phrase coined by Kent Gilson, refers to “the frequent remanufacture or morphing of the entire physical hardware, according to the demands of the user’s specific behavioral requirements”.



With re-configurable computing, you don't waste a lot of time moving in and out of memory, because all operations are performed on hardware, which makes things move very quickly. Kent Gilson refers to this as **hyper-computing**.



# The Potential For Efficient Computing is Greater

---

Because of the FPGA chip, or Field Programmable Gate Array chip.

An FPGA is a class of integrated circuits for which the logic function is defined by the customer after the IC has been manufactured and delivered to the end user.

FPGA's allow users to implement their algorithms at the chip level, as opposed to writing programs that are translated into machine level instructions.

This technique of programming uses dataflow algorithms at the chip level, which proves advantageous because it maximizes parallelism. (i.e. If an algorithm calls for 8 independent additions, then 8 adders can be implemented and all calculations run at once.)

# The Potential For Efficient Computing is Greater

Obtaining parallelism in processing would be a gigantic leap in programming, because it more closely depicts how things happen in the real world.



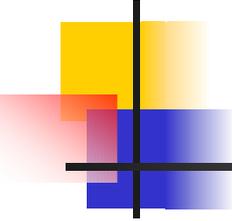
# The Potential For Efficient Computing is Greater

FPGA's are also produced much faster than the standard ASIC chip, (Application Specific Integrated Circuit), which makes them a good choice for the future of hyper-computing. (i.e. it can take a standard ASIC chip up to as long as 18 months to be produced.)



(Decrease production time + Increase in flexibility + Multi-processing) =

Cheaper, Faster, More Efficient Computing



# The 3-Points of HAL:

## Hyper Algorithmic Logic Computer

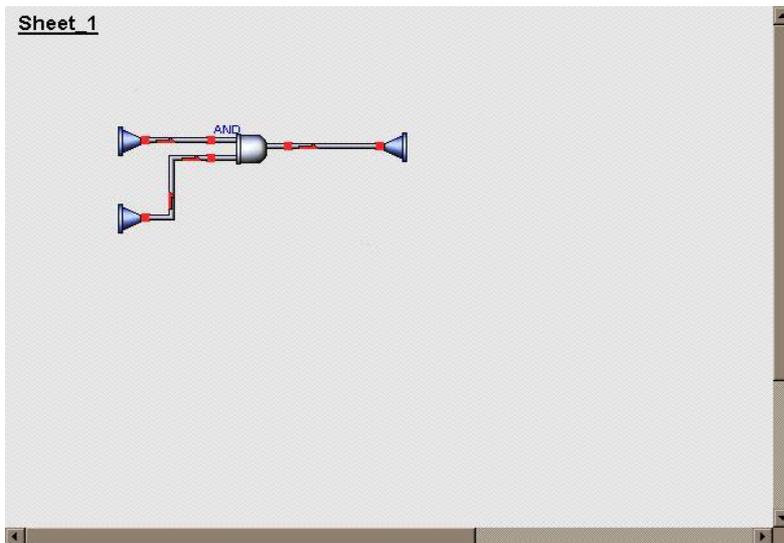
---

- The HAL 15 system here at NASA was the first system to be delivered by Star Bridge to an established high performance computer user. The HAL 15 uses a combination of an Intel based workstation, and a PCI board containing 10 Xilinx FPGA chips.
- IIADL (Implementation Independent Algorithm Description Language) a new programming language that makes it possible for an FPGA-based re-configurable computer to operate as a general-purpose computer system.
- Viva (Latin word for "life"), brings life to HAL and hyper-computing as an OS, compiler, and graphical user interface all in one.

# Viva

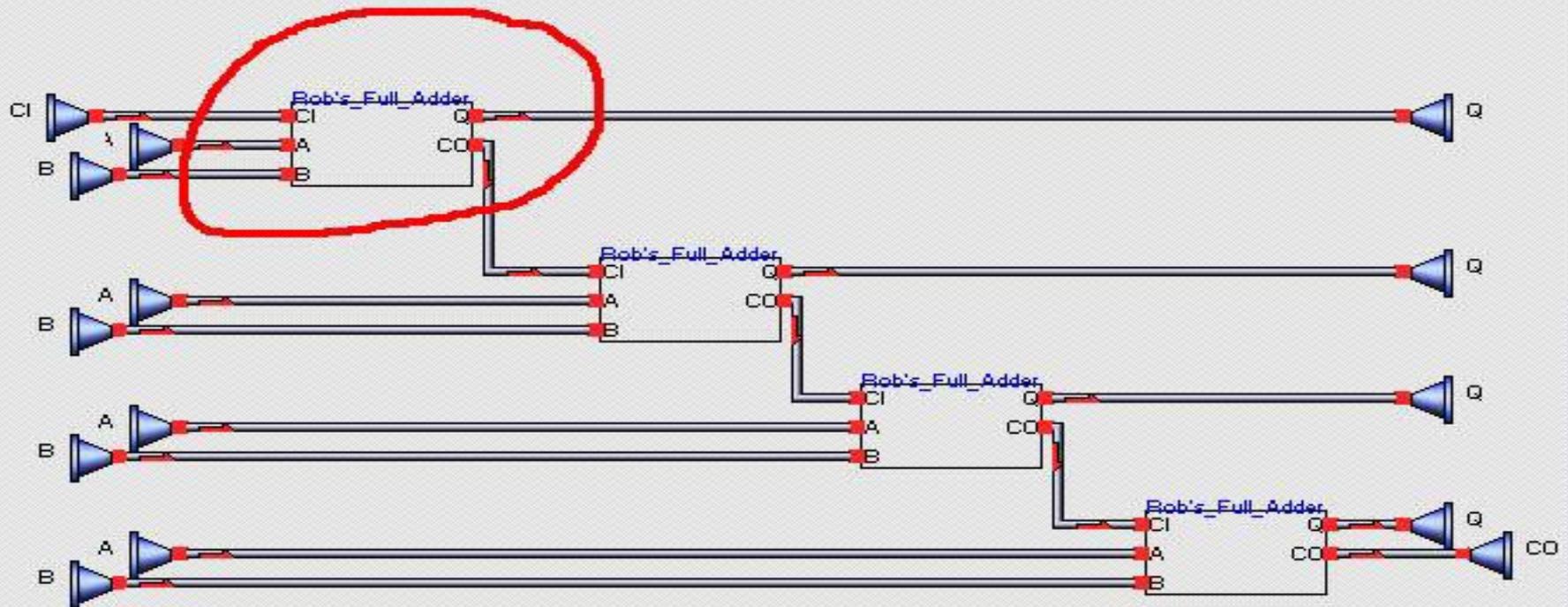
Viva, the OS of HAL, is a graphical programming interface that uses an object oriented programming model with data flow characteristics.

Reusable objects can be built up from primitive functions or multiple levels of hierarchy. Programs are written and compiled in Viva as well.



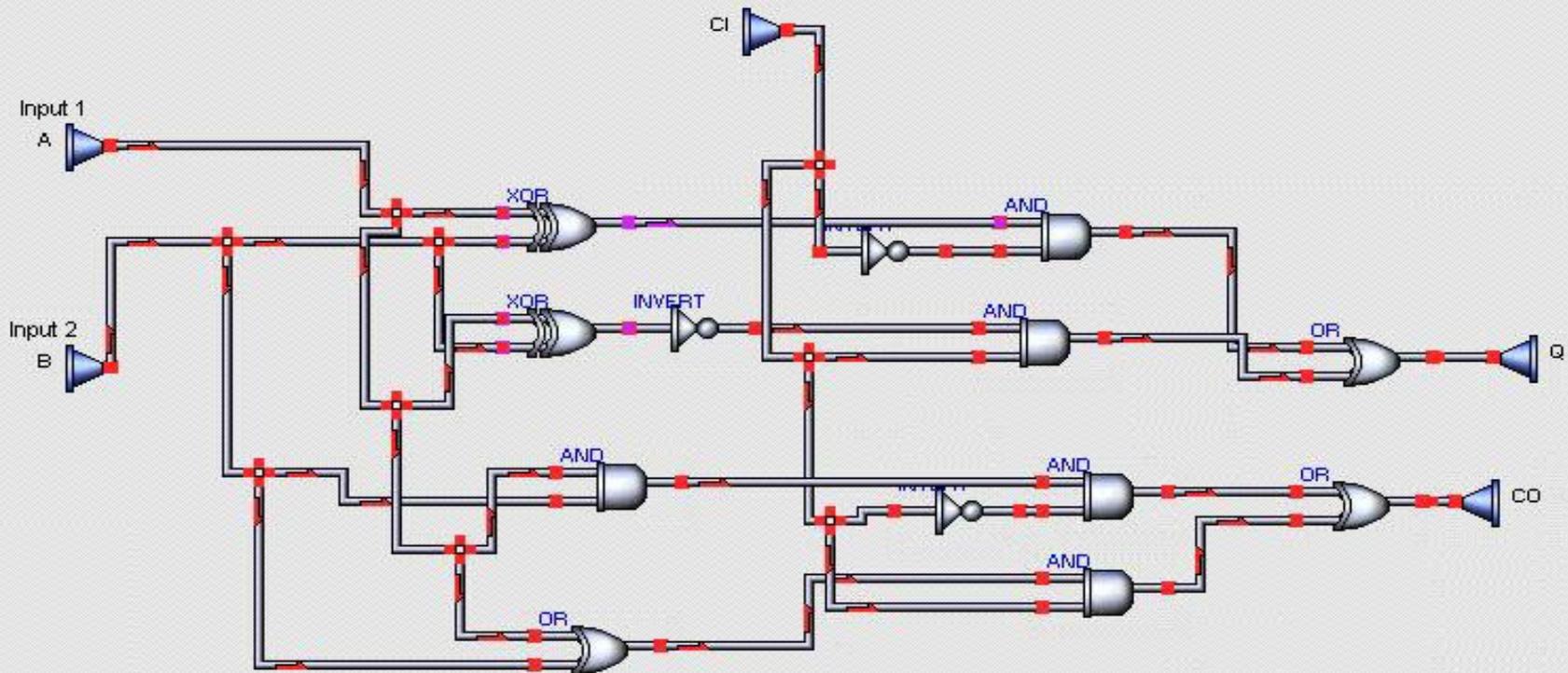
# Viva

## Rob's 4bitRipple Carry Adder



# Viva

## Rob's Full Adder



# So Where Did We Come In?

---

- Test Functions
- Build Libraries
- Innovate and Create!



# Our Goals

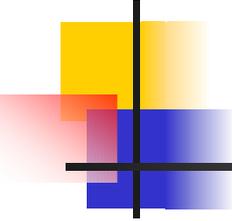
---

We initially tested out the primitive objects in Viva, and corrected some inconsistencies with the functions.

Once realizing that Viva had no transcendental functions we decided to implement them.

- Compute transcendental functions  
i.e.  $\sin$ ,  $\cos$ ,  $\exp$ ,  $\sinh$ ,  $\cosh$ ,  $\ln$ , square-root, etc.
- Create libraries





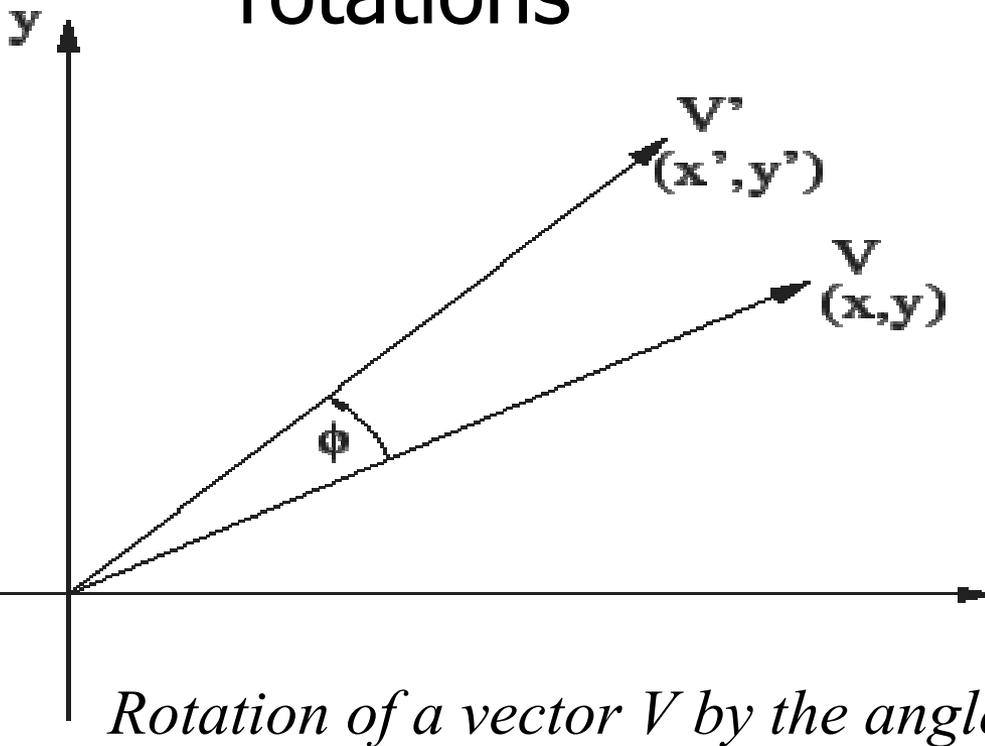
# CORDIC Algorithm

---

- ♦ **CO**ordinate **R**otational **DI**gital **C**omputer
- ♦ Jack E. Volder (1959)
- ♦ Primary concern was trig functions
- ♦ Extended by J. Walther in 1971
- ♦ Used by most calculators today
- ♦ Efficient shift add algorithm/ no multiplies needed

# How CORDIC Works

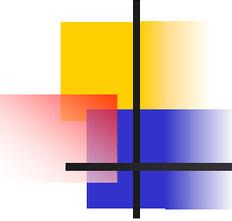
- General concept: iterative vector rotations



$$x' = x \cos \Phi - y \sin \Phi$$

$$y' = y \cos \Phi + x \sin \Phi$$

*Rotation of a vector  $V$  by the angle  $\phi$*

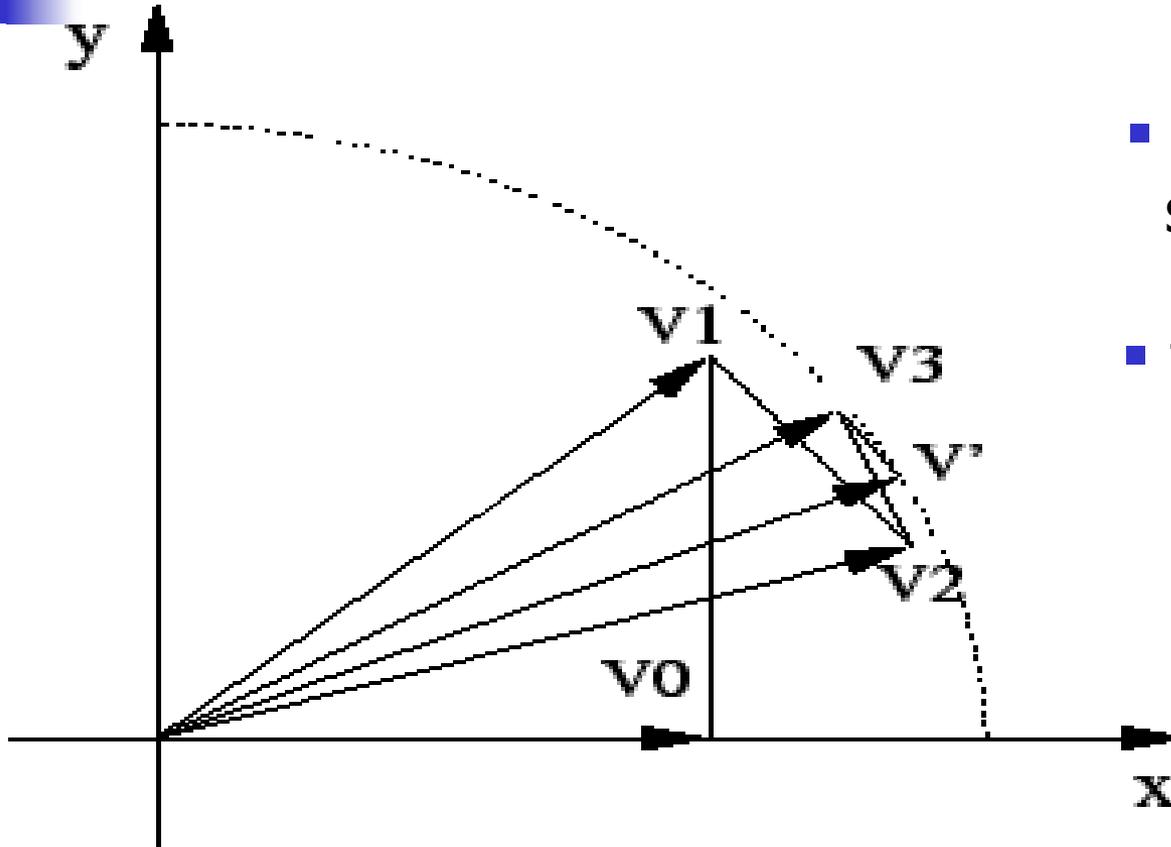


## How CORDIC Works (cont'd)

---

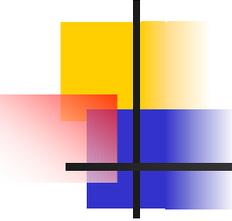
- At each step, accumulate corresponding  $x$  and  $y$  increments
- After a number of iterations,  $x$  and  $y$  increments converge to desired function values

# How CORDIC Works (cont'd)



- break up angle into smaller angles
- tangent of rotation angles =  $2^{-n}$

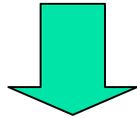
*Iterative vector rotation, initialized with  $V_0$*



# Basic CORDIC-equations

---

- $x' = x \cos \Phi - y \sin \Phi$
- $y' = y \cos \Phi + x \sin \Phi$

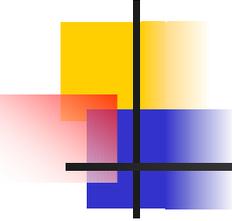


➤  $x_{n+1} = x_n - d_n y_n 2^{-n}$

➤  $y_{n+1} = y_n + d_n x_n 2^{-n}$

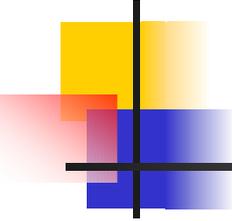
➤  $z_{n+1} = z_n - d_n \operatorname{atan}(2^{-n})$

- $x_{n+1} \rightarrow$  cosine
- $y_{n+1} \rightarrow$  sine
- $z_{n+1}$ : angle accumulator



# Basic CORDIC-equations (cont'd)

- $x_{n+1} = x_n - d_n y_n 2^{-n}$
- $y_{n+1} = y_n + d_n x_n 2^{-n}$
- $z_{n+1} = z_n - d_n \text{atan}(2^{-n})$
- $d_n$ : direction of rotation
  - $d_n = \begin{cases} -1, & \text{if } z_n < 0 \\ +1, & \text{if } z_n \geq 0 \end{cases}$
- $K = K_0 K_1 K_2 \dots K_n$   
where  $K_n = \cos(\text{atan}(2^{-n}))$
- $K \rightarrow 0.6073$

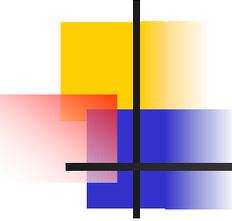


# CORDIC Example

---

- Computation of  $\cos(30^\circ)$  and  $\sin(30^\circ)$  in 8 iterations:

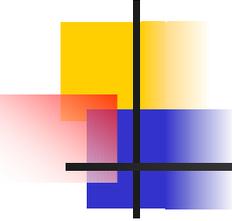
n	X (cos)	Y (sin)	Z (angle)	$\text{atan}(2^{-n})$
0	0.607	0	30	45
1	0.607	0.607	-15	26.565
2	0.91	0.303	11.565	14.036
3	0.835	0.531	-2.471	7.125
4	0.901	0.427	4.654	3.576
5	0.874	0.483	1.077	1.79
6	0.859	0.51	-0.712	0.895
7	0.867	0.497	0.183	0.448
8	0.863	0.504	-0.265	0.224



# Viva Demonstration

---

- Viva overview
- Implementation of sine and cosine functions in Viva.



# Summary and Conclusions

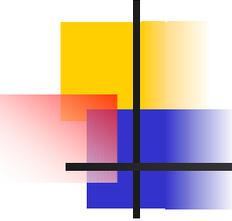
---

HAL computer at NASA

CORDIC algorithm (efficient shift-add algorithm)

Viva (graphical programming language)

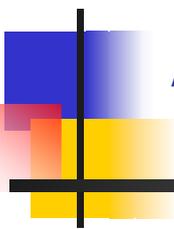
Implementations of transcendental functions to be in Langley Library for future use



# Thank You!!!

---

- Dr. Olaf Storaasli
- Dr. Ivatury Raju
- ACMB Branch



Any Questions???

---