

# Resilient



OFDM Implementation with  
Elemental Computing Arrays

# Computing

# Agenda



- OFDM overview & Growth
- Elemental Computing Architecture
- Radix-2 FFT implementation
- Viterbi decoder implementation
- Summary



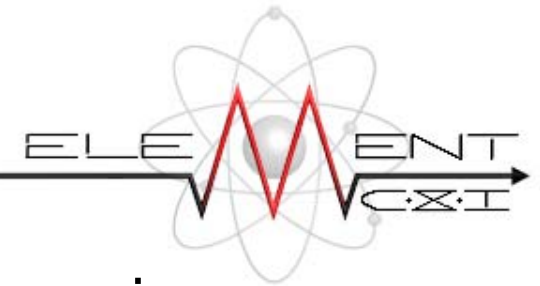
If the only tool you have is a hammer...  
every problem looks like a nail.



# The Multi-core solution...

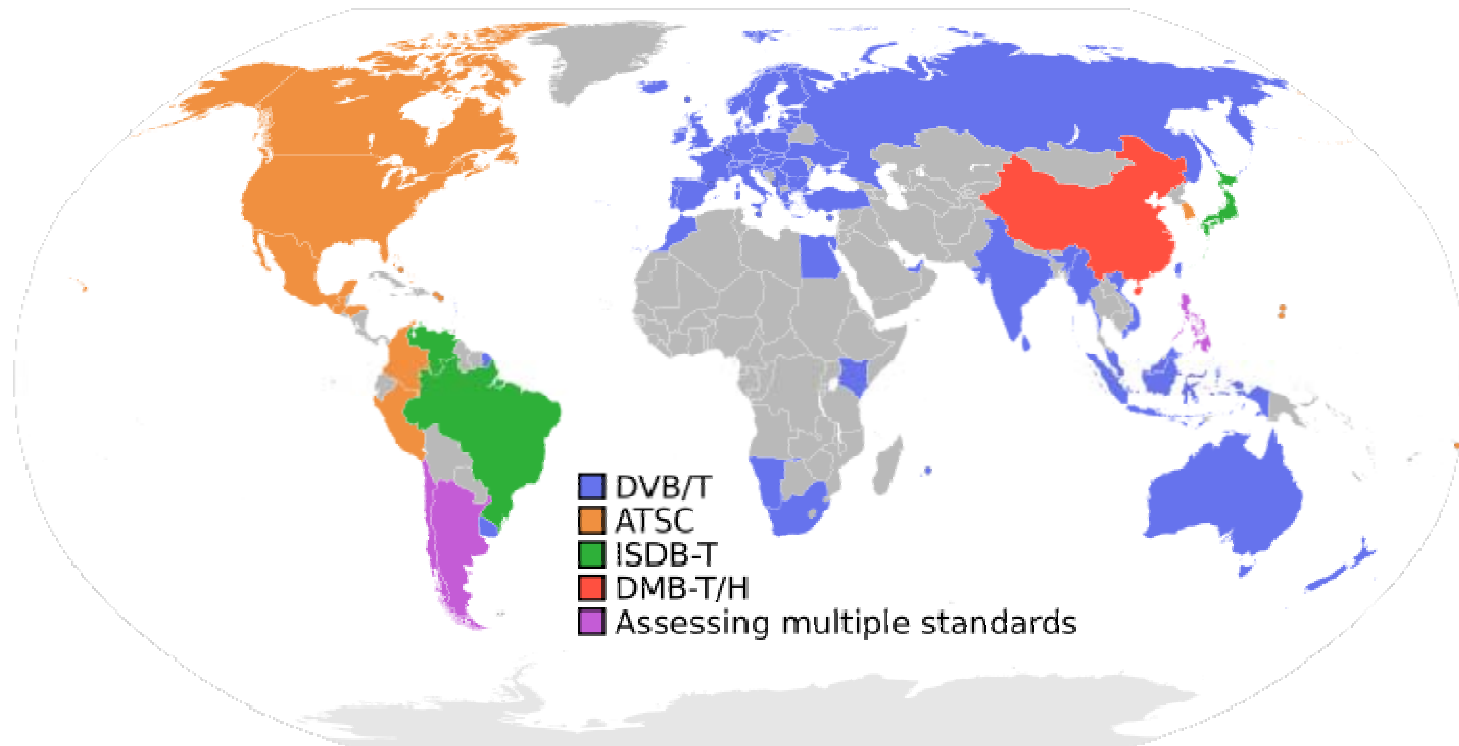


# Overview

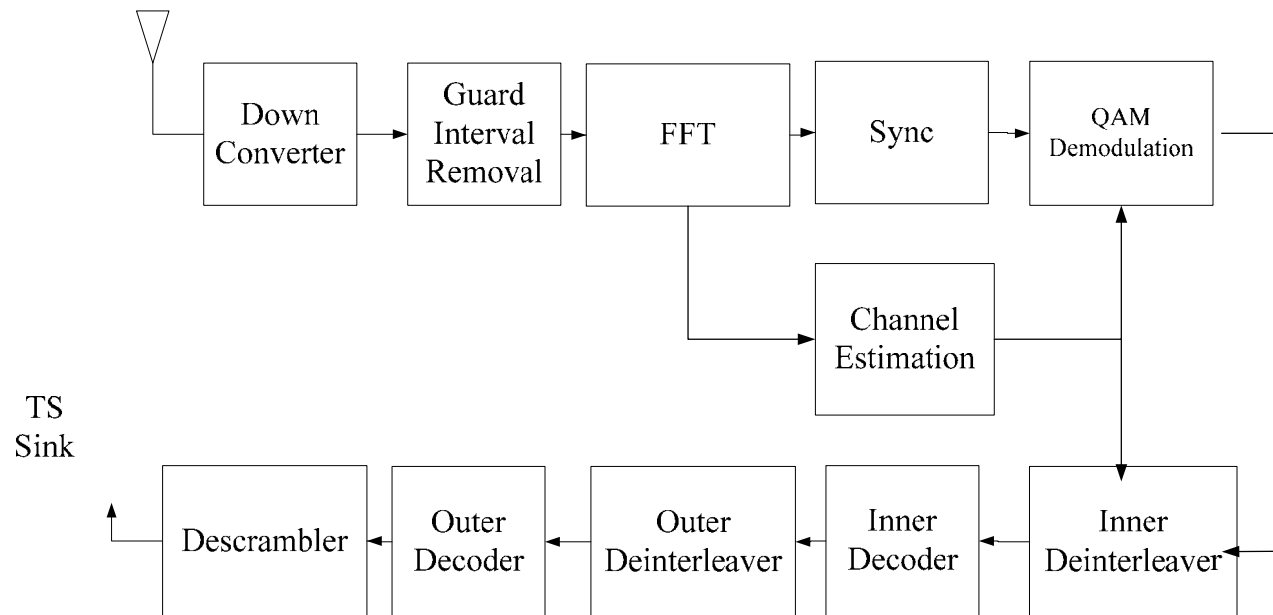


- OFDM-based communication standards growing worldwide
  - Wide area and local area networks
  - Digital Television and music
- Implementation challenge faces barriers of cost, power, and time-to-market
- Elemental Computing Architecture breakthrough delivers extreme performance needed for OFDM
  - Consumer price-points
  - Low power
  - Programmable

# Many Global Standards

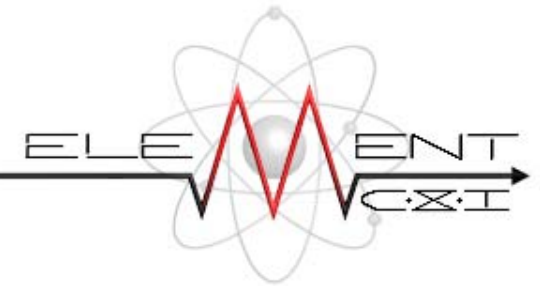


# OFDM Receiver Schematic



- Many blocks demand extremely high MIPS or custom acceleration
  - FFT
  - Inner Convolution Decoder

# Implementation Alternatives

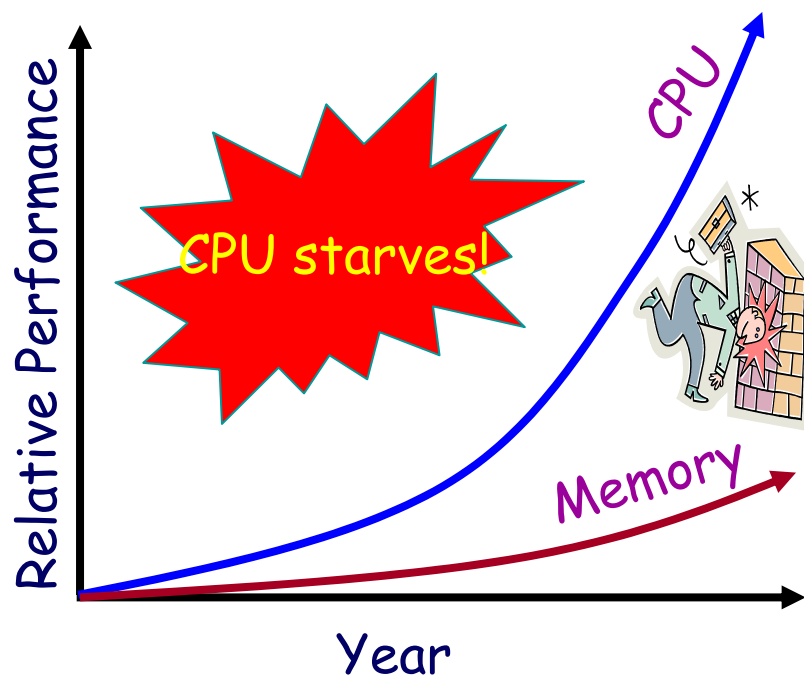


- DSP or multiple DSPs - Inefficient
- DSP + accelerators - Limited Flexibility
- Pure ASIC - No Flexibility + high NRE
- FPGAs - Expensive, High Power
  
- Elemental Computing
  - *Programmable*
  - *Extreme performance, low cost*
  - *Scalability*

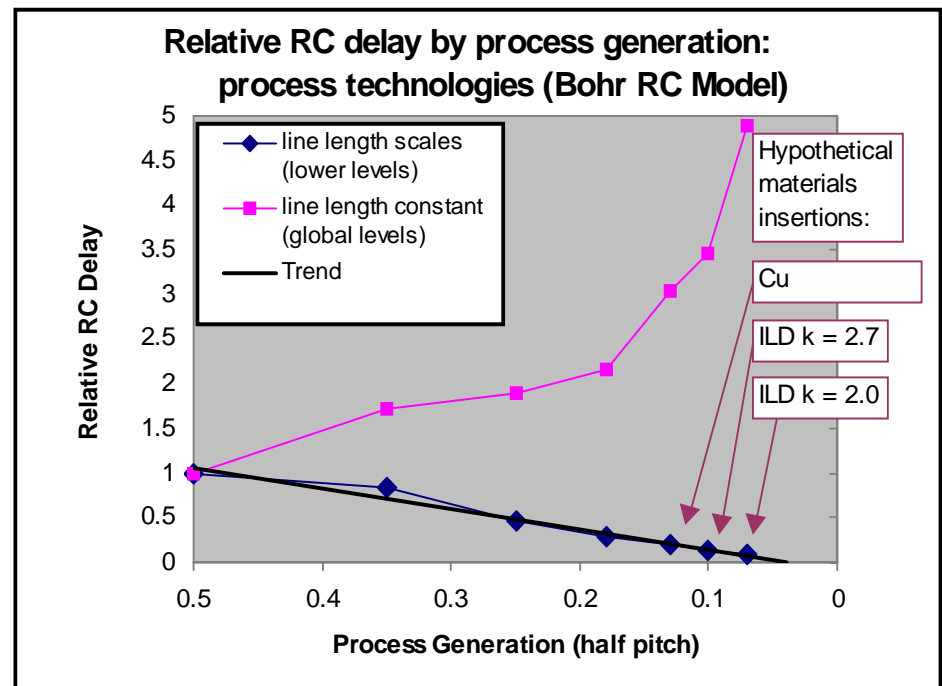
# Process Technology Drives Change



## Memory Bottleneck



## Interconnect Bottleneck



*These twin factors drive the need to smaller, operator level processing elements*

# Introducing Elemental Computing



- 4 Clusters of 16 Elements
  - Homogeneous clusters
- Elements snap together like LEGO<sup>R</sup> blocks
  - Algorithms get the number of resources needed
- Hyper-scaling
  - “Memory-like” attributes
  - Resiliency



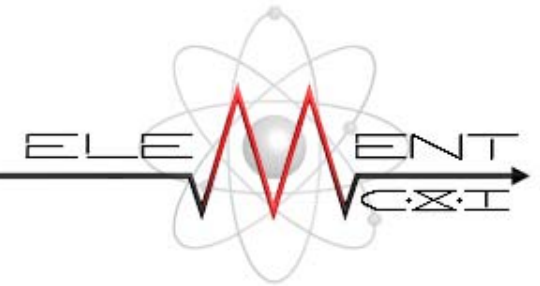
# Element Types



Name	Description	Capabilities
<b>MULT</b>	Multiplier	Dual 8x8 multiply accumulate (32 bit), 16 X 16 multiply accumulate (64 bit), 32 x 32 bit multiply accumulate (64 bit)
<b>TALU</b>	Triple ALU	16 bit sorts, compares, ANDs, ORs, XORs, ADDs, SUBs, ABS, masking, detecting, Add-Compare-Select
<b>SALU</b>	Super ALU	16/32 bit sorts, compares, ANDs, Ors, XORs, ADDs, SUBs, ABS, masking, detecting, leading 1's, leading 0's
<b>BSHF</b>	Barrel Shifter	16/32/48 bit barrel shift, left shift, right shift, logical shift, arithmetic shift, circular, concatenation
<b>BREO</b>	Bit Reorderer	(un)packing, (de)interleaving, (de)puncturing, bit extraction, simple conditionals
<b>SME</b>	State Machine Element	Sequential instruction based element for executing arbitrary control code
<b>MEMU</b>	R/W Memory Unit	Data Storage, Look Up Table (LUT), Random Access, Block mode access via Address Generator



# ECA-64 = 512 Virtual Elements



The ECA isn't two dimensional  
Each Element has 8 contexts  
7 contexts are virtual elements  
Accessible in the same clock cycle  
data becomes available

True meaning of

Low power

Low cost

High performance & flexibility

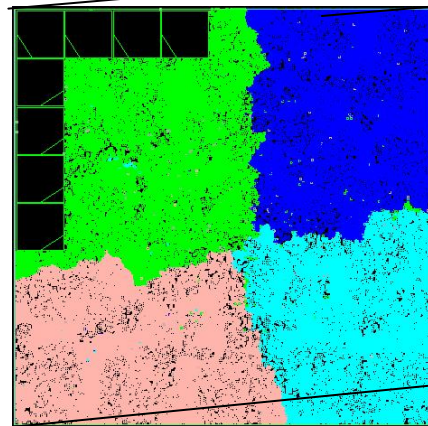
There are ~6,277,101,735,  
386,680,000,000,000,000,  
000,000,000,000,000,000,  
000,000,000,000 possible Element combinations



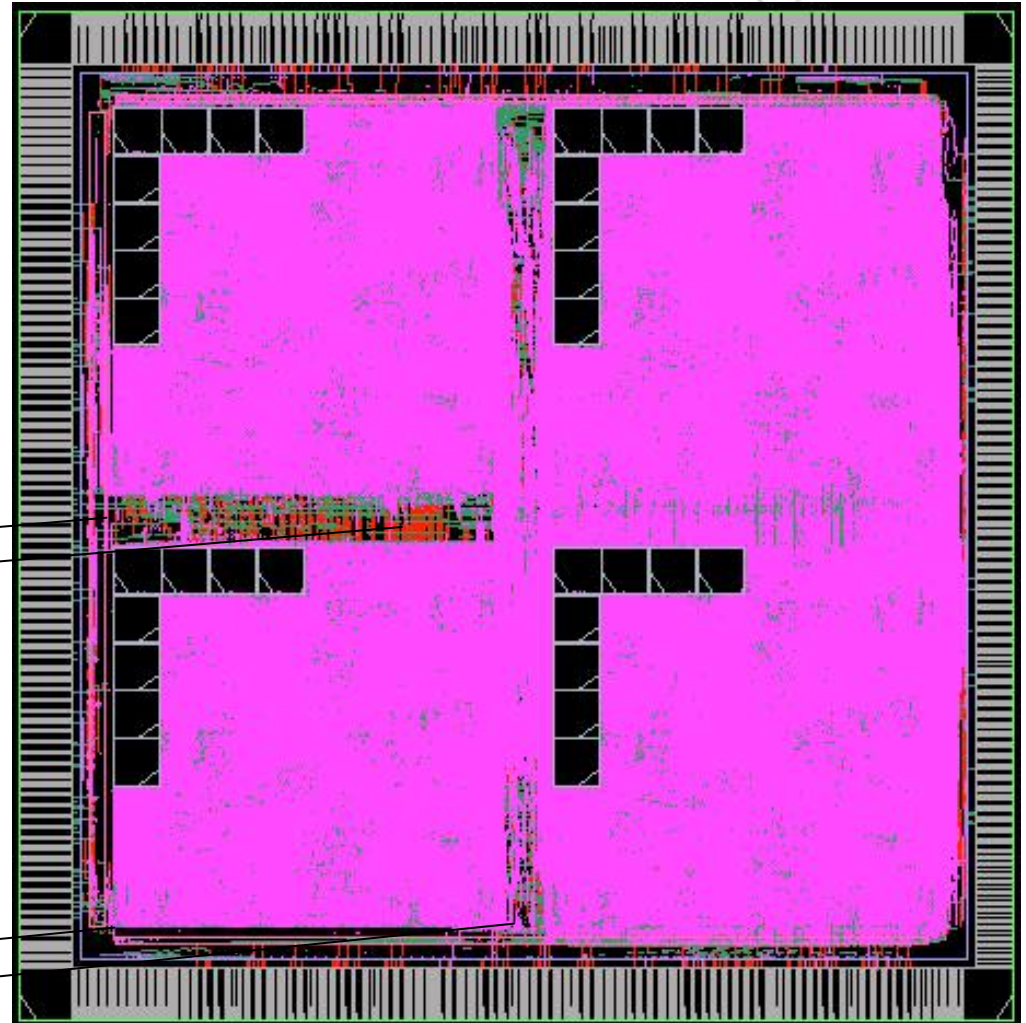
# Physical View



- ~ 4 Million gates (4 cluster)
- 240 MHz @ core <1 Watts
- TSMC 90nm G process
- Standard Cell, 8 LM
- P&R at Cluster level  
(16 elements) replicated 4x



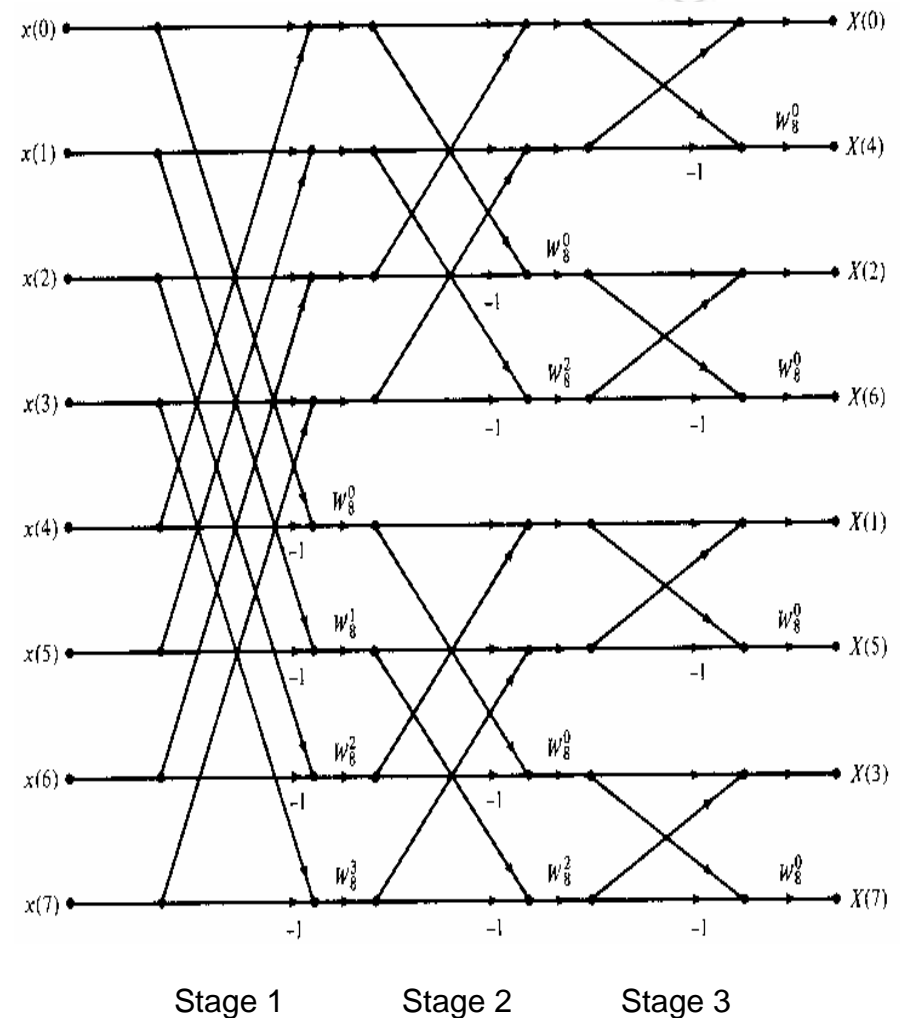
Cluster of  
16 elements



# FFT Radix-2 Implementation



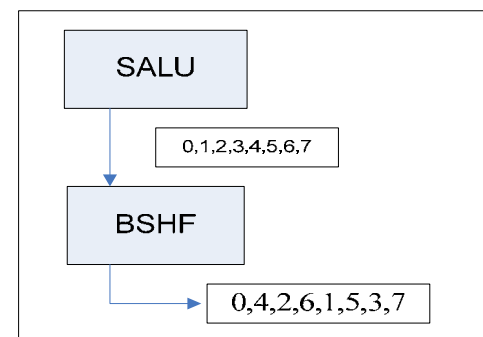
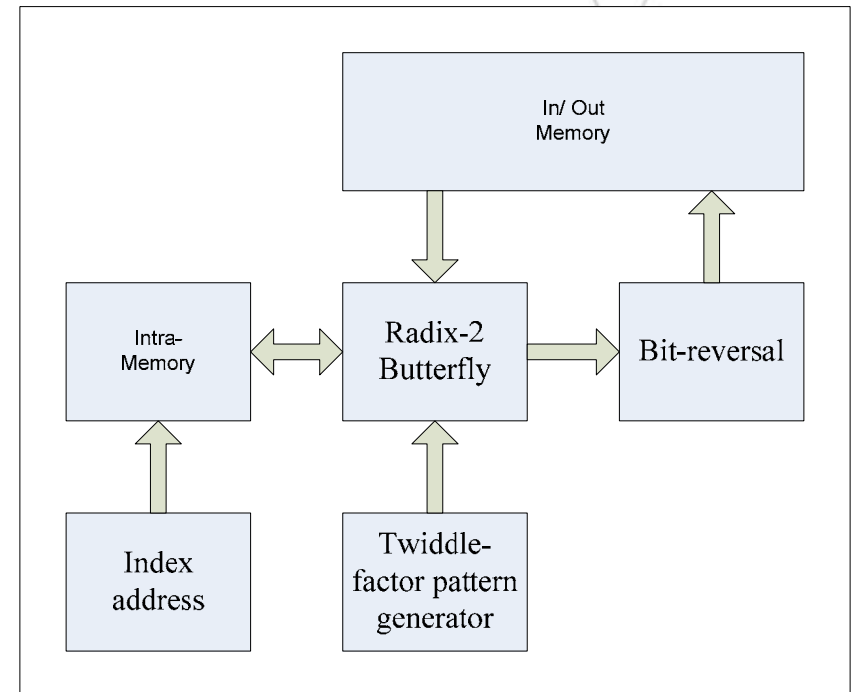
- Decimate in Frequency
- Simultaneous read of 2 real and 2 imaginary, 2 constants, and write of next 2 real and 2 imaginary
  - Block-mode of MEMU eliminates need for address calc
- Complete data-flow based control (no uP control)
- Not a hard-coded FFT engine
  - All elements can be reused for other needs



# Overall Flow and Bit Reversal



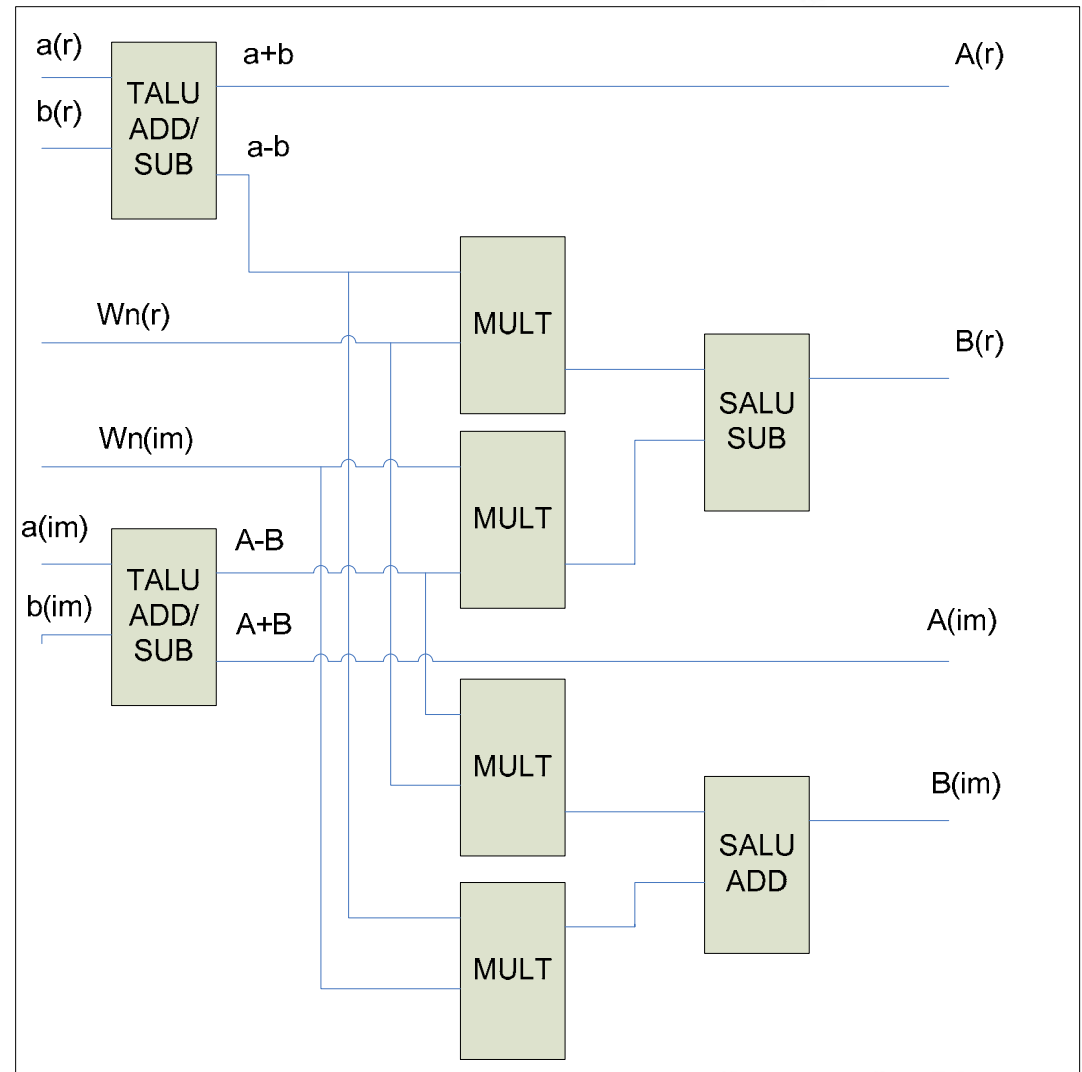
- Input, Output stored in memory
  - Separate memory banks used as 'ping-pong' for intermediate pass
  - Memory banks allow for 10 simultaneous memory transactions
- Memory addressing automatic with MEMU address generator
- Natural output order is produced by bit-reverse address in last pass
  - Simple circuit uses 1 SALU, 1 BSHF context



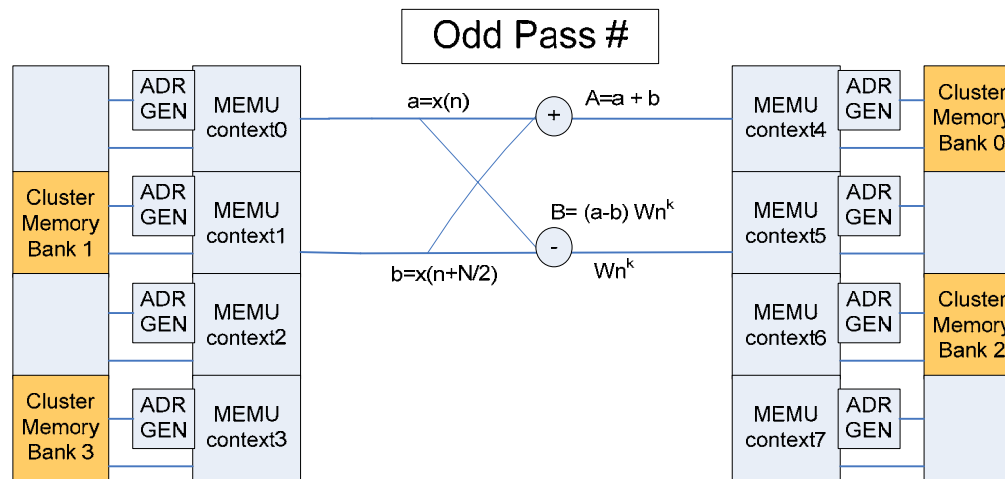
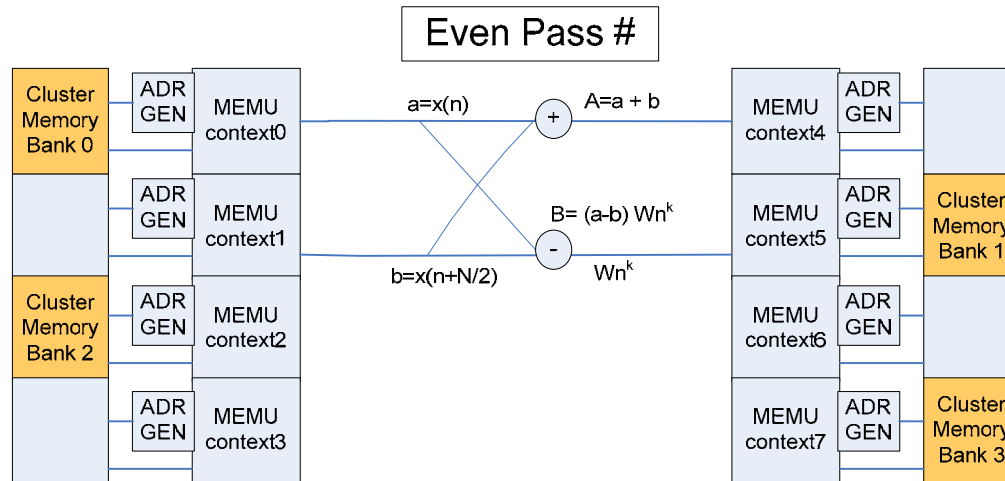
# Butterfly Implementation



- Pipelined Operation
  - 1 Butterfly per clock
- TALU : 2 functions
  - A+B and A-B
- MULT:
  - 16x16 multiplication
- SALU:
  - A+B
  - A-B



# Memory Implementation



Inputs	stage 0	stage 1	stage 2
x0	0 1 2 3	0 4 1 5	0 2 4 6
x1	4 5 6 7	2 6 3 7	1 3 5 7

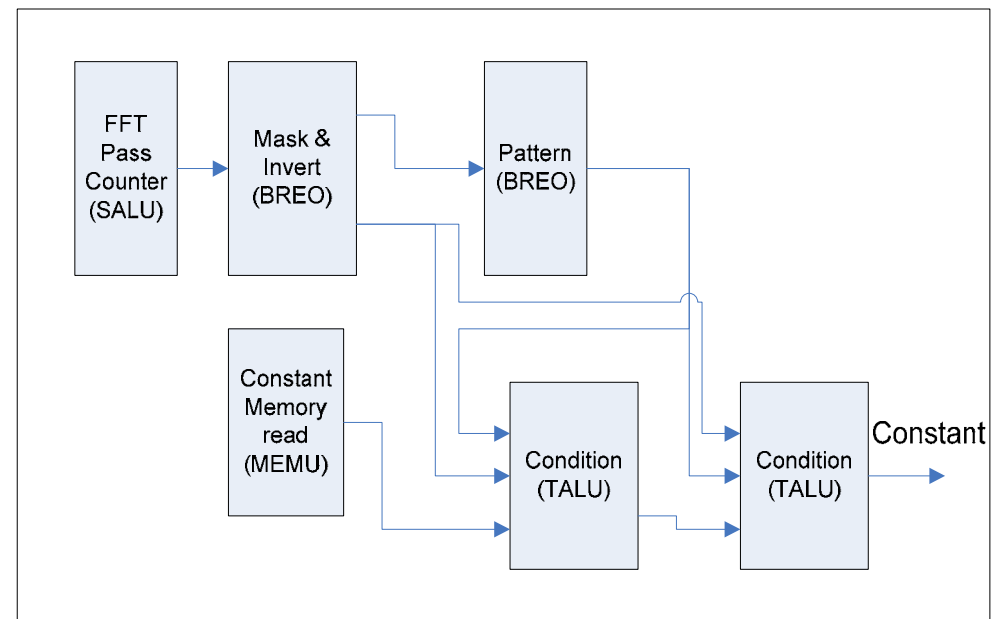
# FFT Twiddle Factor Control



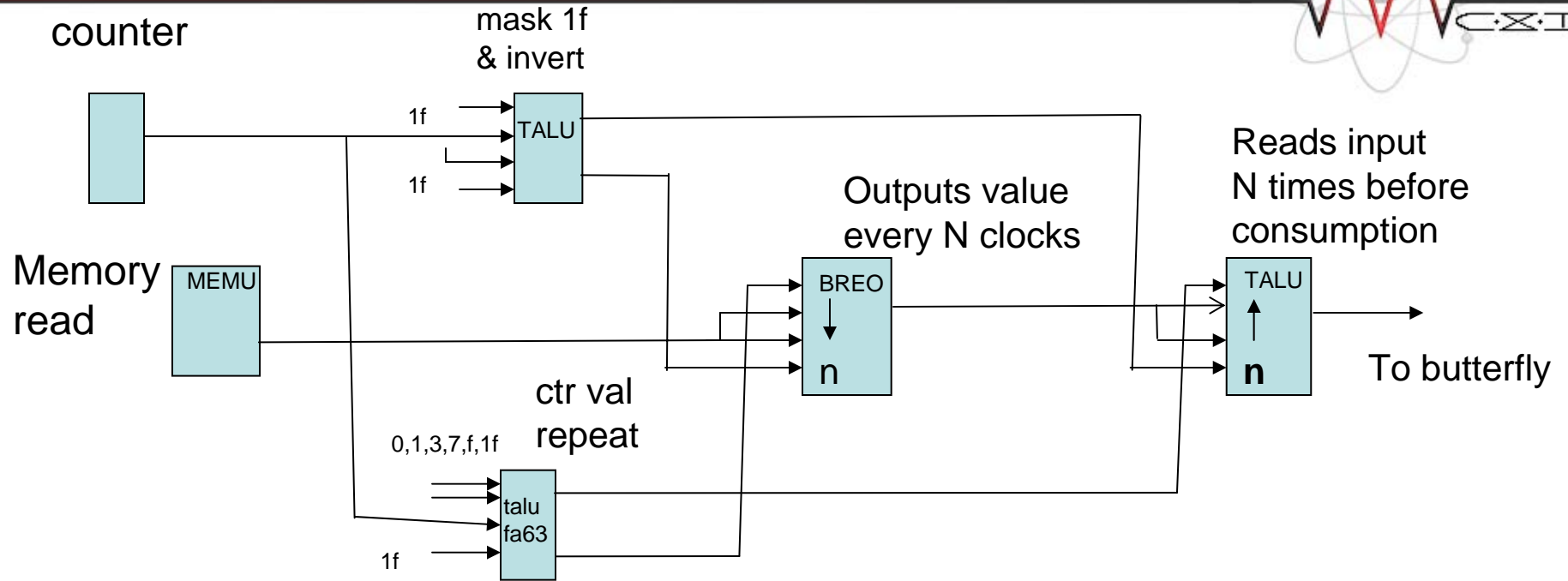
- Butterfly constants are re-used each pass
  - After first pass, values are repeated
  - Circuit for constant repeat control should be scalable for N size
- **Elemental solution**
  - Elements used to up-sample and down-sample constant stream
  - General solution for multiple apps

Stage	Twiddle indices								
0	0	1	2	3	4	5	6	7	
1	0	0	2	2	4	4	6	6	
2	0	0	0	0	4	4	4	4	
3	0	0	0	0	0	0	0	0	

Twiddle indices for 16 point FFT



# Down/Up Sampling Circuit



# FFT Performance and Resource Use



Standard	ISDB-T	DVB-H	DVB-T	FLO	CMMB	DTMB	T-DMB	WiMax (Mob)	WiMax (Fixed)
<b>FFT Size (points)</b>	2048 4096 8192	2048 4096 8192	2048 8192	4096	1024 4096	1024 4096	256 512 1024 2048	128 512 1024 2048	256
<b>Duration (µs)</b>	252 504 1008	224 448 896	224 896	738.02	409.6	500	250	91.4	64
<b>Single FFT Cycles</b>	28.2k 61.4k 133.12k	28.2k 61.4k 133.1k	28.2k 133.1k	61.4k	12.8k 61.4k	12.8k 61.4k	2.56k 5.76k 12.8k 28.2k	1.12k 5.76k 12.8k 28.2k	2.56k
<b>Symbol Number</b>	4K 2K 1K	4.5K 2.2K 1.1K	4.5K 1.1K	1.4K	2.4K	2K	4K	11K	15.63K
<b>MIPS</b>	112.80 132.80 133.12	127 135 146	127 146	83.2	30.72 147	25.6 132.8	10.24 23.04 51.20 112.2	12.3212. 63.36 140.80 310.20	40.01

FFT Requirements for Standards

FFT SIZE	µs
2048	102 (64.5)
4096	229
8192	469

ECA Performance

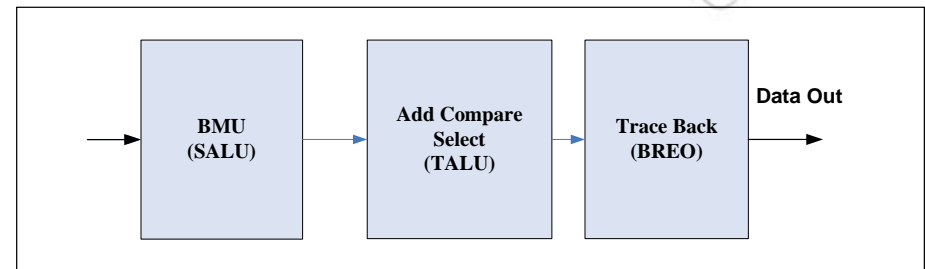
Element	Contexts Used				Total	Used / Available
	C0	C1	C2	C3		
<b>Type</b>						
SALU	2	2	5	4	13	
TALU	4	4	4	4	16	
MULT	2	2	2	2	8	
BSHF	2	2	2	2	8	
BREO	2	2	5	5	14	
MEMU	6	6	5	5	22	
	18	18	23	22	81	81/480

Contexts used in ECA-64

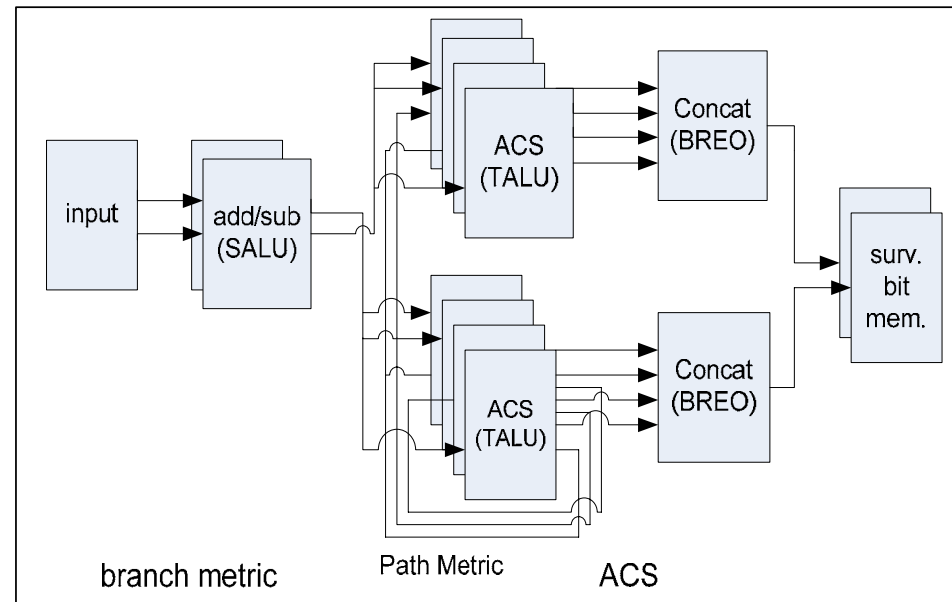
# Viterbi Decoder Implementation



- Viterbi k=7 implementation
- Implemented by many parallel elements, no microcontroller sequencing
- Computationally intensive ACS is executed by TALU
  - 16 TALU elements perform 64 ACS in 4 cycles
  - Surviving bit information concatenated into 4 16-bit words in memory



Overview of Viterbi Sections

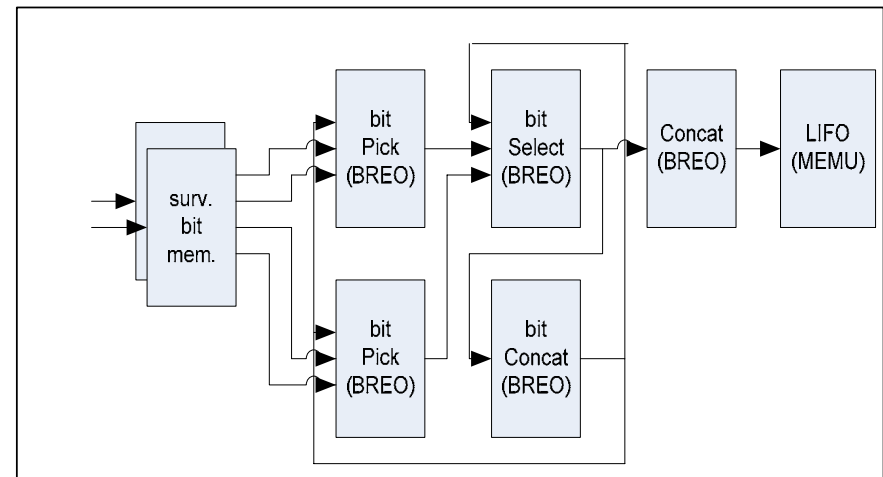


Detail of Branch Metric and ACS

# Viterbi Decoder Trace back



- Derives previous survivor state based on current state
- Survival bits (4 16-bit words) read from memory
  - BREO conditional mode picks candidate bit
- LIFO implementation
  - Block mode memory write with negative stride value



Look Ahead Trace back

# Viterbi Performance Summary



- Highly parallel processing for under 10 cycles/bit with 64 element ECA-64
- Regular, scalable architecture of ECA allows trade-off of element usage and performance
  - TALU context usage determines ACS speed

Element Utilization						Ratio
Element	Contexts Used					Used vs. total
Type	C0	C1	C2	C3	total	
SALU	2	2	2	4	10	
TALU	16	16	16	16	64	
MULT	2	2	2	6	8	
BSHF	0	0	0	0	0	
BREO	5	5	5	5	20	
MEMU	1	0	0	6	7	
	26	25	25	37	109	109/480

Resource utilization for Viterbi Decoder

# Summary



- Elemental Computing meets the high throughput needs of OFDM signal processing
  - Programmable, reconfigurable operator-level elements meet demands of FFT, Viterbi decode
    - Current 512-virtual element silicon meets FFT performance for WiMax and ISDB-t standards
    - Current 512-virtual element silicon meets Viterbi performance for ISDB-t
- Elemental Computing fabric is scalable across algorithms, ICs, and process technology
- Directly applicable to OFDM implementations and Software Defined Radio
  - Programmable, dynamic reconfigurable, low power solution for:
    - Space- time coding, diversity techniques, convolutional coding/decoding, demapping, synchronization