Computer Systems, Inc.
MERCURY

Beamforming for Radar Systems on COTS Heterogeneous Computing Platforms

Jeffrey A. Rudin

Mercury Computer Systems, Inc.

High Performance Embedded Computing (HPEC) Conference

September 23, 2003

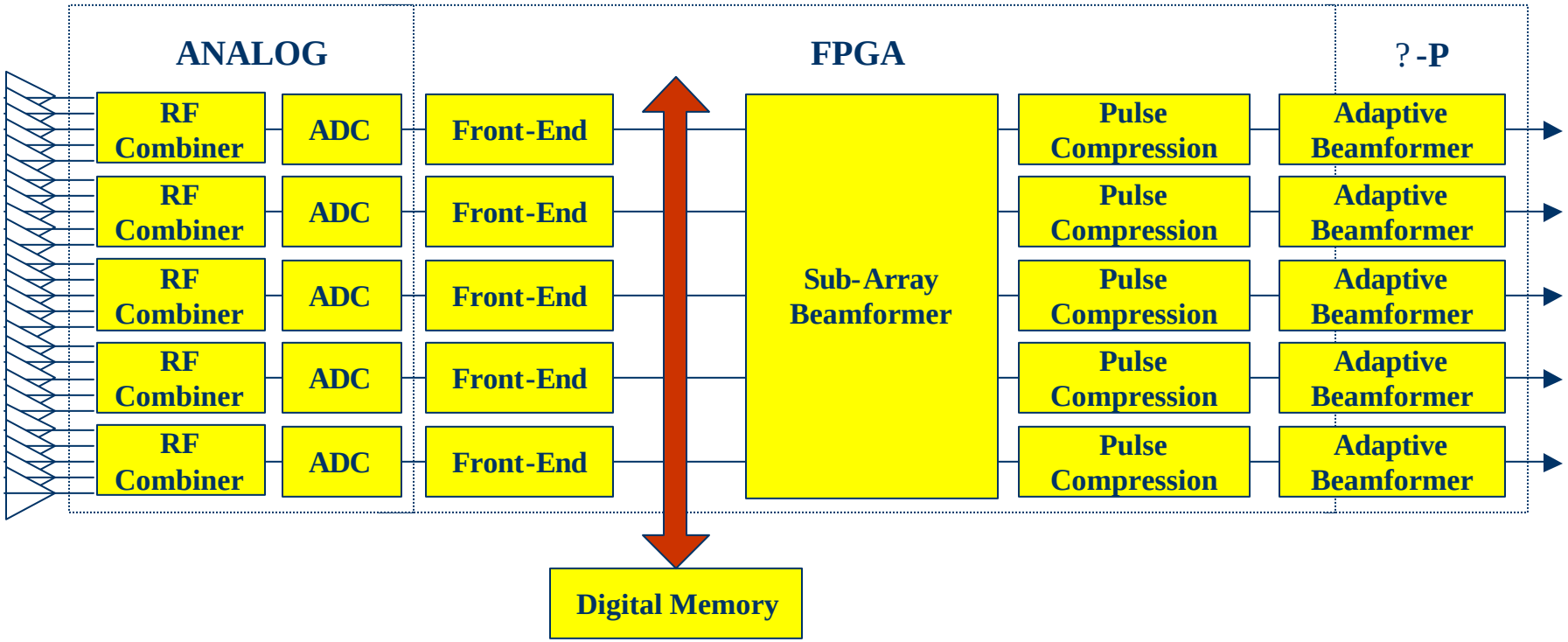
The Ultimate Performance Machine

Outline

- Beamforming Radar System Architecture
- Processing Resources
- Strawman System Analysis
  - Front-End Processing
  - Back-End Processing
  - Beamformer Architectures
- Summary

Radar System Architecture

- **Beamforming requires massive dataflow and computation**
  - **ADC precision and data rate are chosen to provide high dynamic range and and wide signal bandwidth**
  - **High number of input channels required in modern phased array radars to produce multiple beams and nulls**



Computer Systems, Inc.
MERCURY
The Ultimate Performance Machine

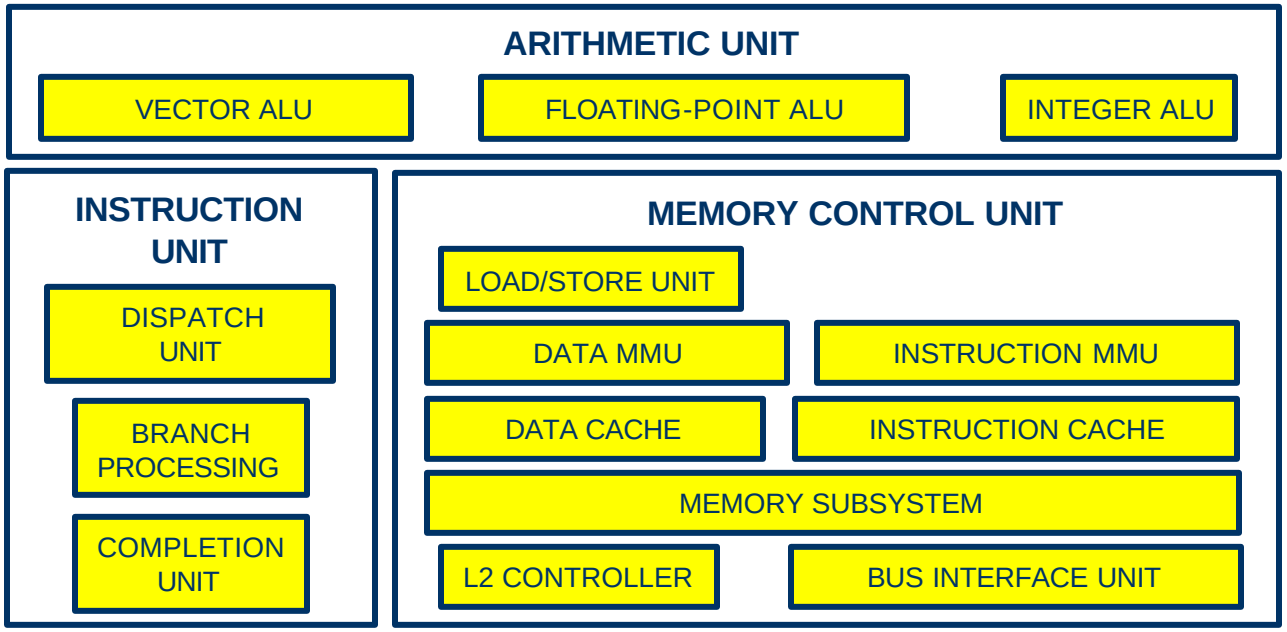
Processing Resources

- **Microprocessors**
  - Fixed processing, I/O, and memory architecture
  - Task context switch requires microseconds
  - Native floating-point available
  - Low interaction between code modules
- **FPGAs**
  - Customizable processing, I/O, and memory architecture
  - Task context switch requires reconfiguration -- milliseconds
  - Floating-point must be built or bought
  - Considerable interaction between IP cores
  - Signal propagation issues
  - Currently harder to program than microprocessors

Computer Systems, Inc.
MERCURY
The Ultimate Performance Machine

PowerPC Microprocessor

- 400 - 1000 MHz clock speeds
- 133 MHz system bus (MPC74xx) -- 851 MB/s
- 64-bit integer and floating-point units
- 128-bit AltiVec vector processing unit
- Pipelined instruction unit
- 32 kB instruction and data caches
- Up to 2 MB L2 cache



Virtex-II Pro FPGA

- Clock speeds lower than processors: 100 - 200 MHz clocks
- Up to 20 full-duplex multi-gigabit transceivers.
- Many DSP supporting features

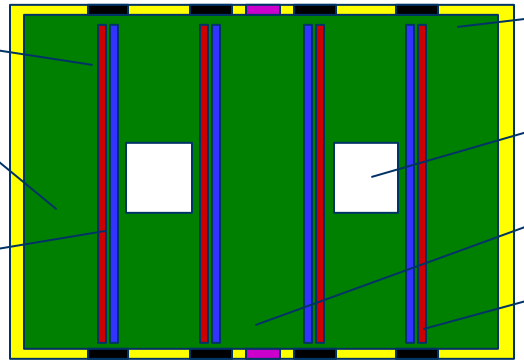
Device	Gigibit Tx/Rx	Logic Slices	18-Bit Multiplier	18K-Bit Block RAM	Clock Manager	I/O Pads	CPU Blocks
XC2VP40	12	19,392	192	192	8	804	2
XC2VP50	16	23,616	232	232	8	852	2
XC2VP70	20	33,088	328	328	8	996	2
XC2VP100	20	44,096	444	444	12	1,164	2

REGISTERS
LUT'S
CARRY LOGIC
MULTIPLEXERS
DISTRIBUTED RAM
SHIFT REGISTERS

DEDICATED MULTIPLIERS

CONFIGURABLE LOGIC

DUAL-PORT BLOCK RAM



I/O BLOCKS

PowerPC 405 CORE

CLOCK MANAGERS

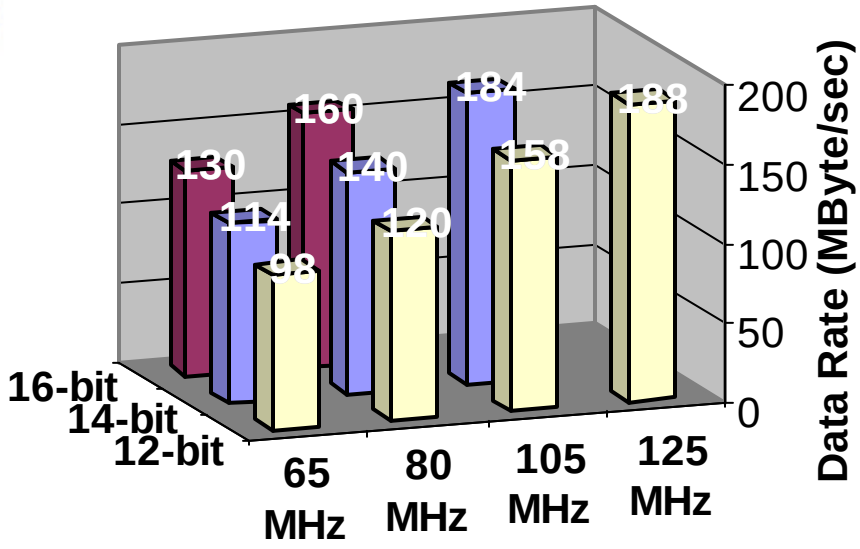
FULL DUPLEX TRANCEIVERS

Each block RAM contains two banks with independent sets of address and data lines
Gigabit transceivers provide over 240 MBps each direction -- over 4800 MBps throughput!

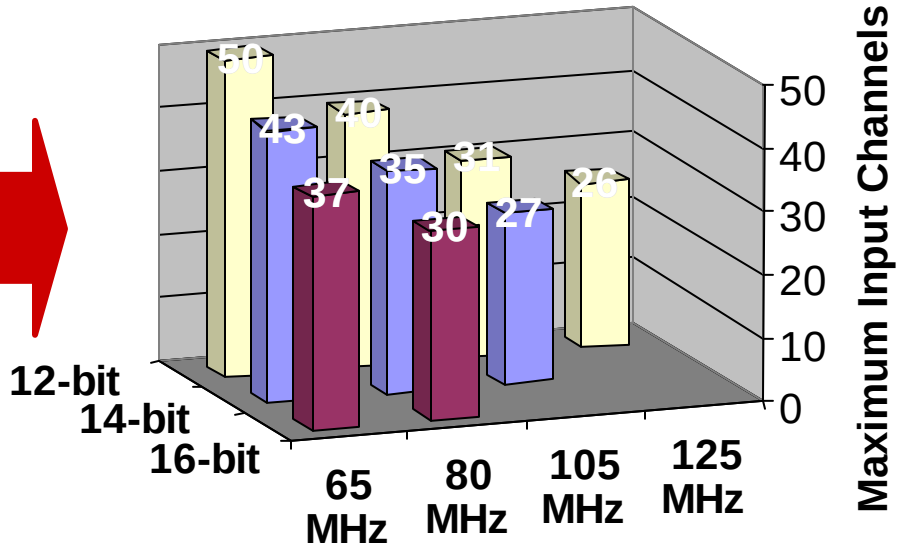
Strawman System Requirements

- Lots of channels -- 80+ input channels
- ADC with "good" bandwidth and dynamic range
  - 100 MSps -- 1.56 - 25 MHz bandwidth using $f_s/4$ sampling
  - 14-bit precision -- over 80 dB dynamic range
- Reasonable implementation risk -- 100 MHz clock

ANALOG/DIGITAL CONVERTER DATA RATES



INPUT CHANNELS PER FPGA USING GIGABIT Tx/Rx



ADC precision and rate and number of channels drive downstream requirements

Front-End Processing

- **Digital Down Converter**
  - **fs/4 IF & BW**
  - **4x decimation**

  } **Eliminates the need for numerically controlled oscillators (NCO)**

  - **31-tap complex FIR, real symmetric coefficients**
  - **Usually no bit growth**

$$G_{BIT} \geq \frac{1}{2}\log_2\left(\frac{SNR_o}{SNR_i}\right)$$

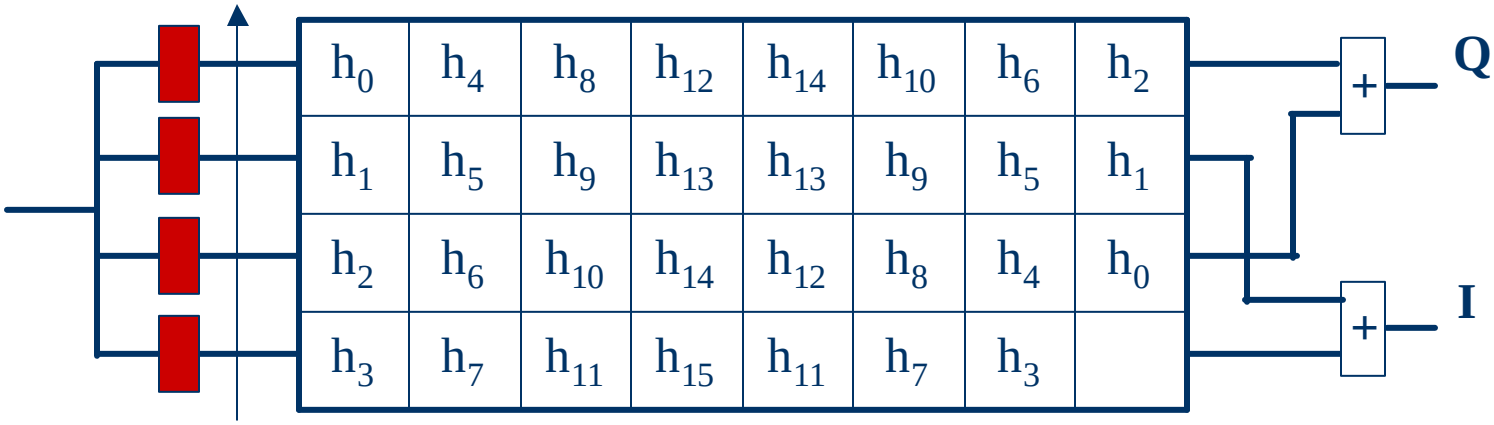
- **Lowpass Decimation Filter**
  - **1x (bypass), 2x, 4x, 8x, and 16x decimation rates**
  - **0, 16, 32, 64, 128 taps**
  - **Real coefficients**
  - **0 to 2 bits of bit growth**
- **Equalizer**
  - **16-tap, complex coefficients -- cannot generally exploit symmetry**
  - **Usually no bit growth**

Digital Down Converter

- Reduce complexity -- exploit fs/4 center frequency and bandwidth
  - Complex mixing reduces to polyphase commutation
    - Cosine and sine select even and odd samples respectively
      - cos(jn?/4) = 1, 0, -1, 0, 1,…; sin(jn?/4) = 0, j, 0, -j, 0,…
  - Exploit polyphase structure for decimation

$$y[n] = \sum_{i=0}^{N-1} h_3[i]x[n - 4i] + \sum_{i=0}^{N-1} h_1[i]x[n - 4i - 2] + j\left(\sum_{i=0}^{N-1} h_2[i]x[n - 4i - 1] + \sum_{i=0}^{N-1} h_0[i]x[n - 4i - 3]\right)$$

POLYPHASE fs/4 DDC



Odd number of taps creates symmetries in the FIR coefficients

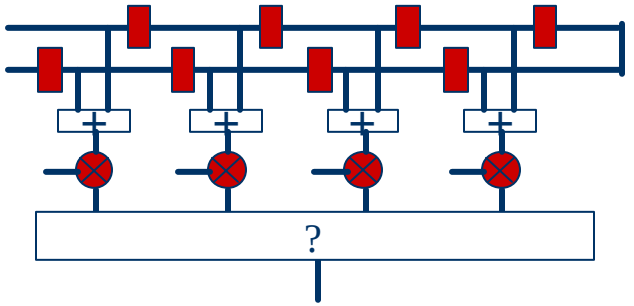
Digital Down Converter

- **Reduce complexity -- exploit filter symmetries**

Exploit symmetric filter structures for in-phase signal

$$y[n] = \sum_{i=0}^{N/2-1} h[i]x[n-i] + \sum_{i=N/2}^{N-1} h[N-1-i]x[n-i]$$

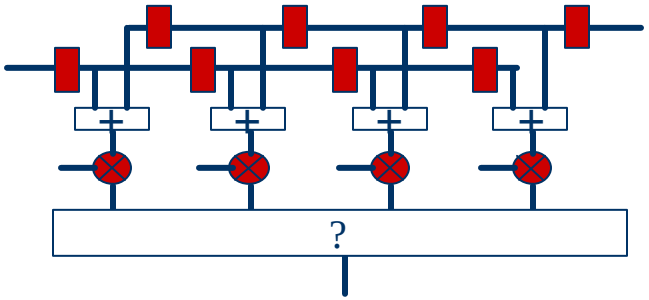
$$= \sum_{i=0}^{N/2-1} h[i](x[n-i] + x[n-(\tfrac{N}{2}-1-i)])$$



Exploit symmetry pair filters for quadrature signal

$$y[n] = \sum_{i=0}^{N-1} h[i]x_1[n-i] + \sum_{i=0}^{N-1} h[N-1-i]x_2[n-i]$$

$$= \sum_{i=0}^{N-1} h[i](x_1[n-i] + x_2[n-(N-1-i)])$$

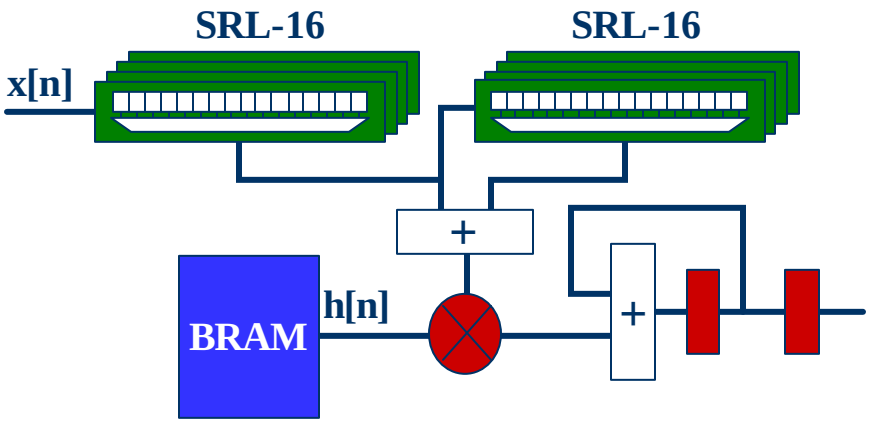


Each tap calculation involves one coefficient and two samples

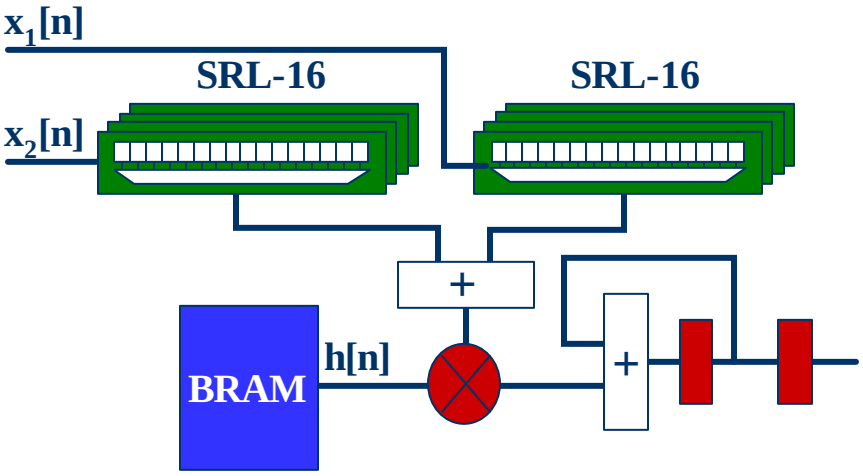
Digital Down Converter

- **Reduce complexity -- exploit 4x decimation**
  - **Use MAC-Engine to do 4 multiplies per input sample**
    - **Use fclk = 4 x fs to time share multipliers**
  - **Configure logic slices as shift registers (SRL's) to save BRAM**
    - **Need to store 3 sets of numbers -- need 2 BRAM's**
      - **Save BRAM by using logic slices to store both sets of samples**

Symmetric Filter



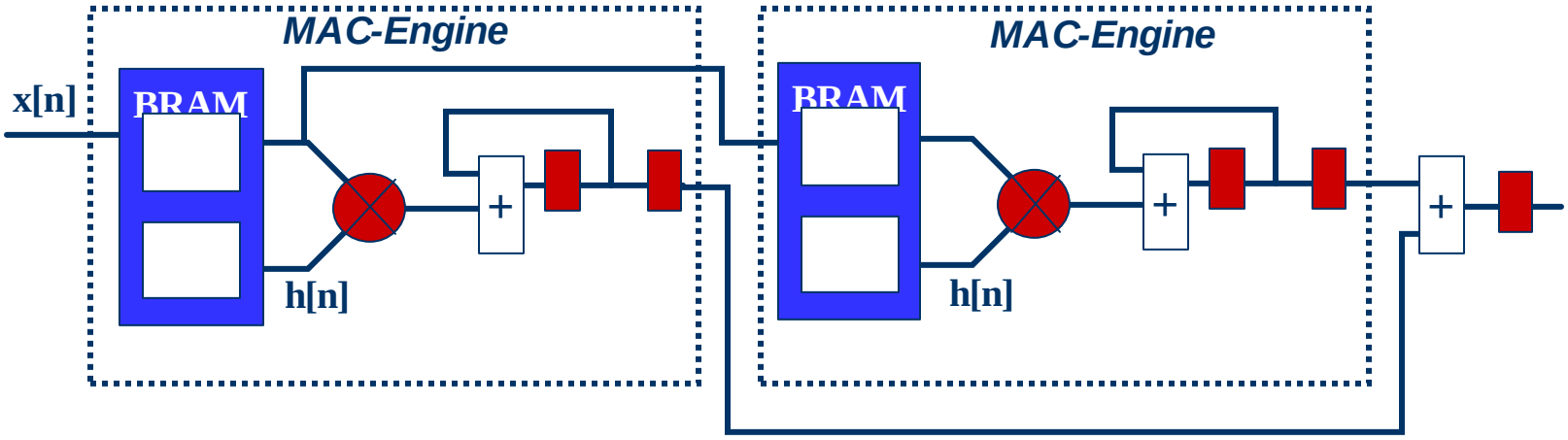
Symmetry Filter Pair



Low Pass Filter

- **Reduce complexity -- use MAC-Engine FIR implementation**
  - **Run multipliers at 4x sample rate -- time share multipliers**
  - **Exploit constant length-decimation product**
    - **Single structure handles multiple filter implementations**
    - **Single clock frequency**
  - **Use dual-bank feature of BRAM**
    - **First bank stores samples**
    - **Second bank stores FIR coefficients**

$$N_{MAC?Eng} ? N_{Taps} \frac{f_{out}}{f_{clk}}$$



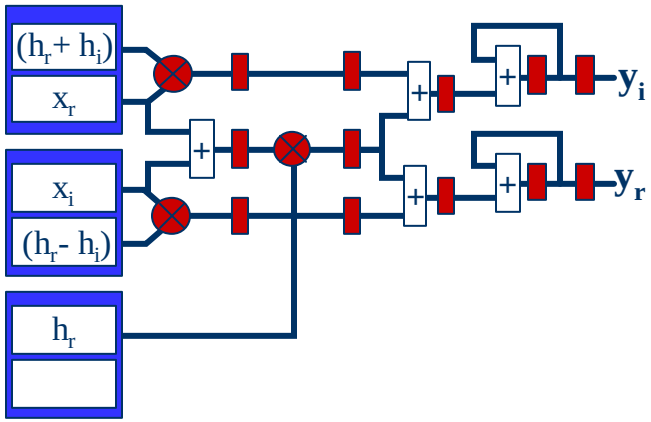
Equalizer

- **Reduce complexity -- reduce number of multipliers and BRAM's**
  - **Exploit f_{clk}/f_s -- use MAC-Engine**
  - **Implement complex multiply using only 3 MAC-Engines**
    - **Use common product term in complex multiply**

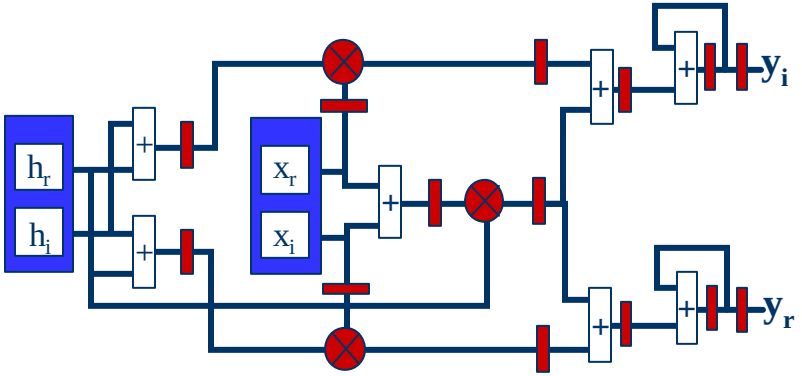
$$y_r = x_r h_r - x_i h_i = (x_r - x_i)h_r - x_i(h_r - h_i)$$

$$y_i = x_r h_i + x_i h_r = x_r(h_i + h_r) + (x_r - x_i)h_r$$

Trade logic slices for multipliers



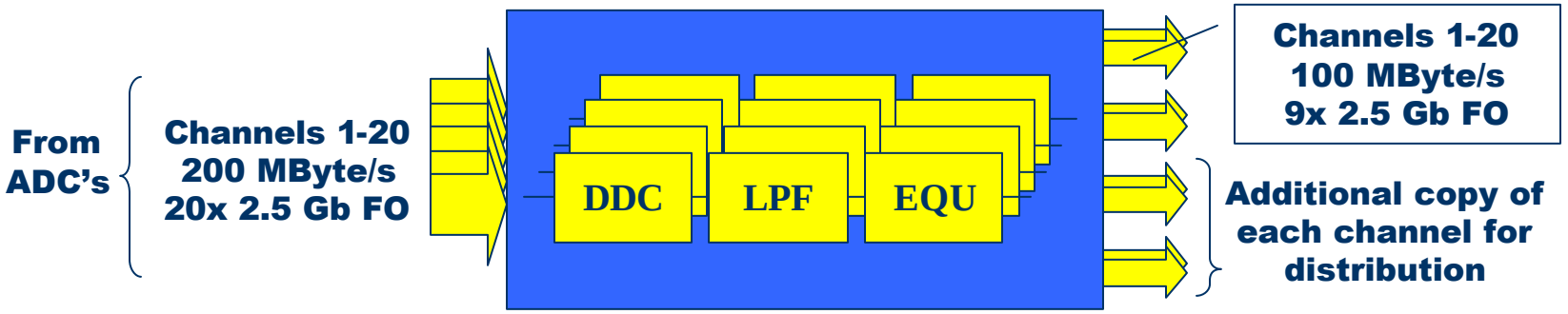
Trade logic slices for block RAM

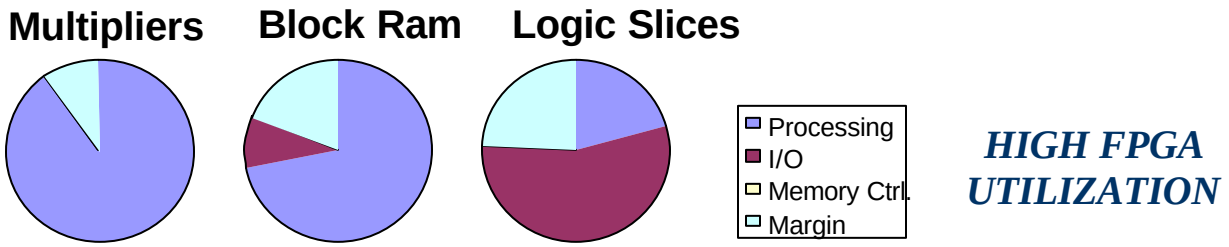


Front-End Realization

- FPGA features can be exploited to maximize utilization
  - Up to 20 100-MSps channels per FPGA
  - DDC with 31-Tap FIR using *only* 3 multipliers/channel
  - LPF 16-128 Tap decimating FIR using *only* 4 multipliers/channel
  - EQU 16-Tap complex FIR using *only* 12 multipliers/channel

Digital Receiver Module for 20x 100 MSps Channels on Virtex-II Pro 100





HIGH FPGA UTILIZATION

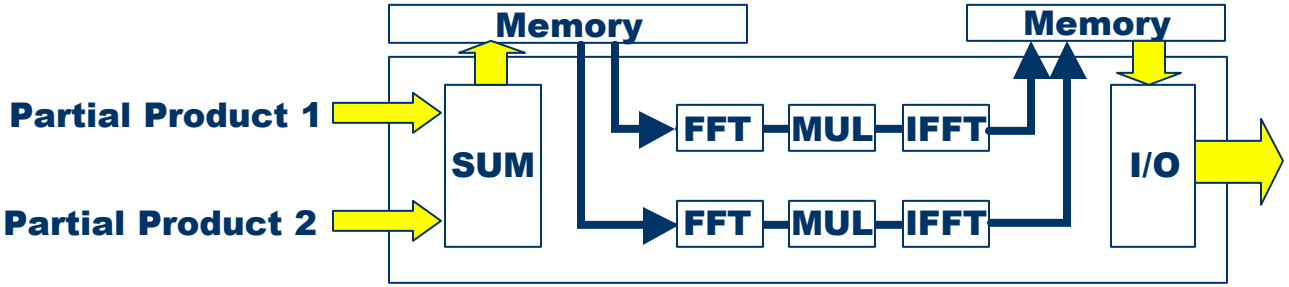
FPGA Utilization for 20x 100 MSps Channels

Back-End Processing

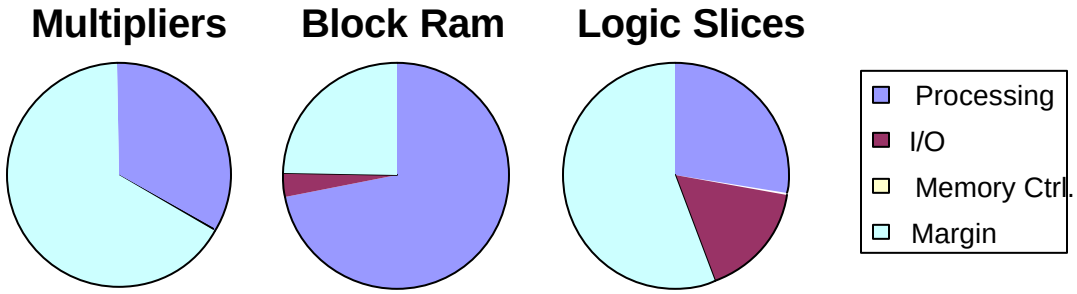
- FPGAs can be used to address data flow requirements that persist in the system until application of adaptive beamforming weights
  - Digital Pulse Compression
    - Fast convolution with FFT IP cores
  - Doppler Processing
    - FPGA FFT IP cores available
  - Adaptive Beamforming Weight Application
    - Similar advantages to those in sub-array beamformer
- FPGAs can augment weight computation
  - QR Decomposition
    - New FPGA solutions may replace microprocessors
  - Cholesky Decomposition
    - Possibly form covariance matrix in adjunct FPGA

Digital Pulse Compression

- FFT IP cores can be used to implement pulse compression
  - 8192-tap FFT @ 25 MSps/channel
  - 6 sub-array channels / FPGA
  - 3-stage pipelined convolver -- 2 convolvers / FPGA
  - Enough resources to sum partial products from beamformer



Doppler processing can be implemented using similar FFT cores



GOOD FPGA UTILIZATION

DIGITAL PULSE COMPRESSION FPGA UTILIZATION

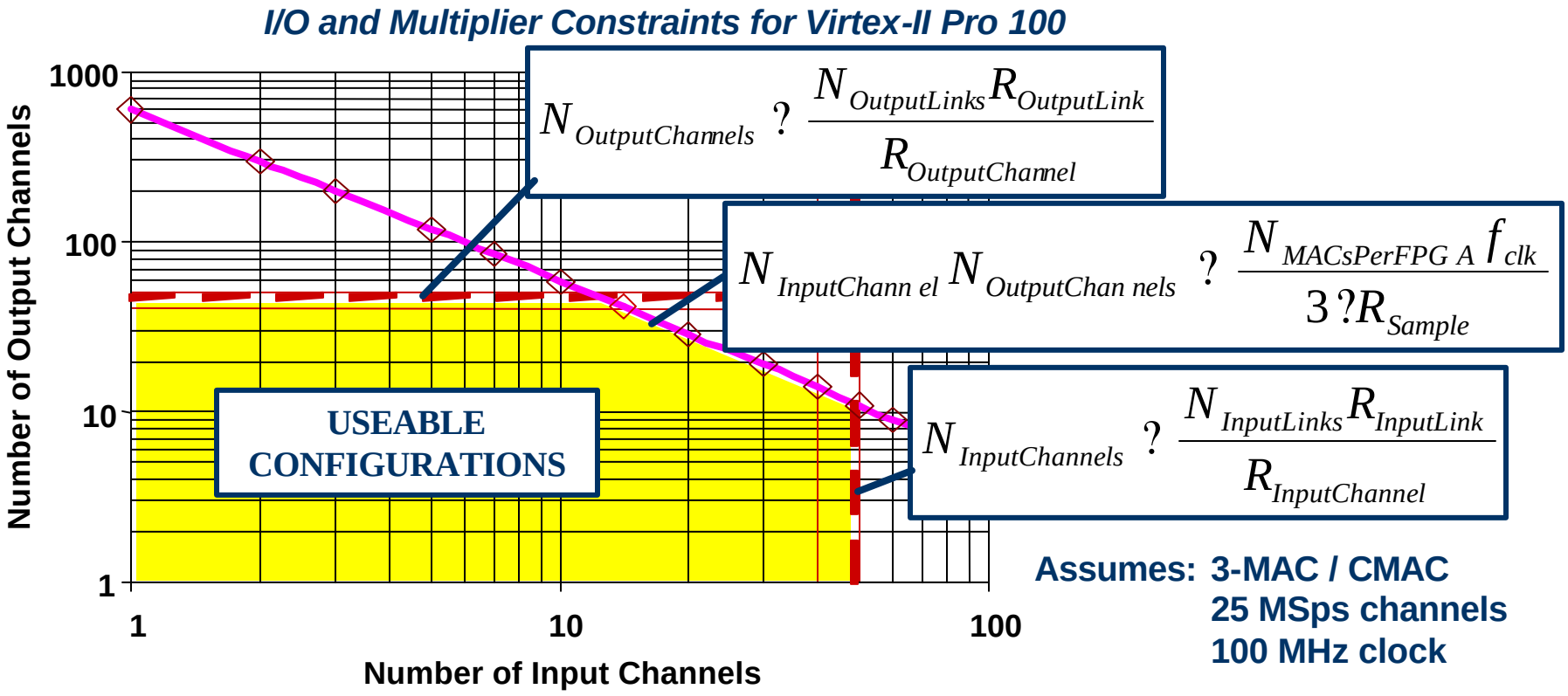
FFT cores tend to be BRAM hungry.

Beamformer Architectures

- **Unconstrained Linear Architecture**
  - All input channels contribute to each output
- **Constrained Linear Architecture**
  - A subset of input channels contributes to any output
- **Mesh Architecture**
  - All input channels contribute to each output

Beamformer Module Constraints

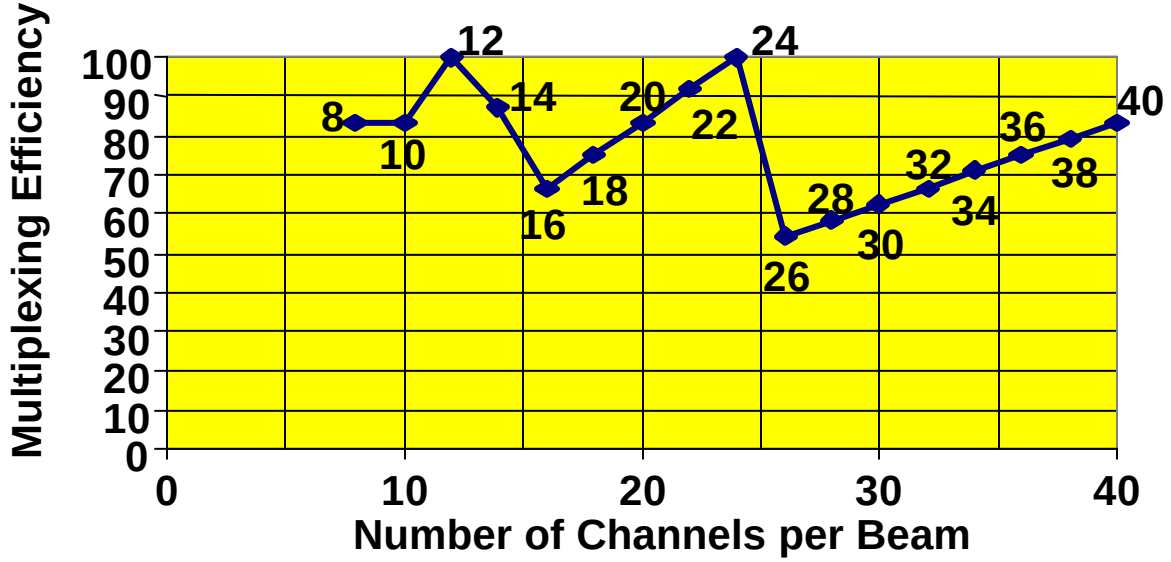
- Basic limits are imposed by I/O and number of multipliers
- Inputs over 18-bits can increase the number of multipliers
  - Keep watch on bit growth in front-end processing



Beamformer Module Constraints

- Multiplexing must be designed to maximize communication
  - Beam Partitioned output multiplexing may reduce efficiency
  - Alternate multiplexing methods may be necessary

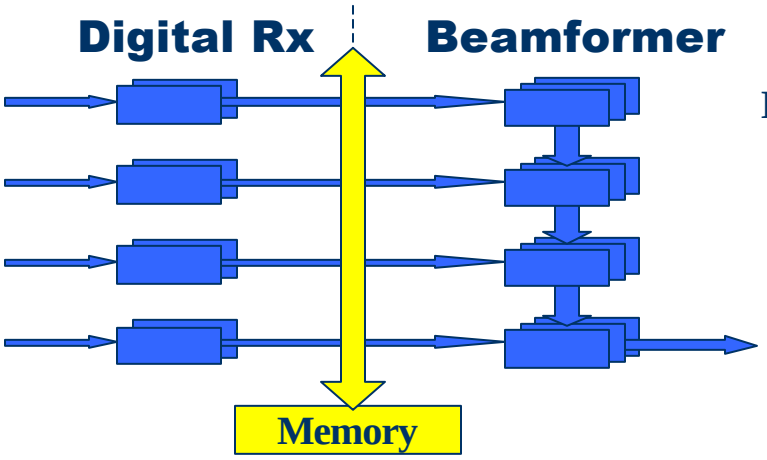
$$\eta_{MUX} \le \left\lfloor \frac{1}{N_{Links} R_{Link}} N_{Links} \right\rfloor \Big/ \left\lfloor \frac{N_{ChannelsPerBeam} R_{OutputChannel}}{R_{Link}} \right\rfloor$$

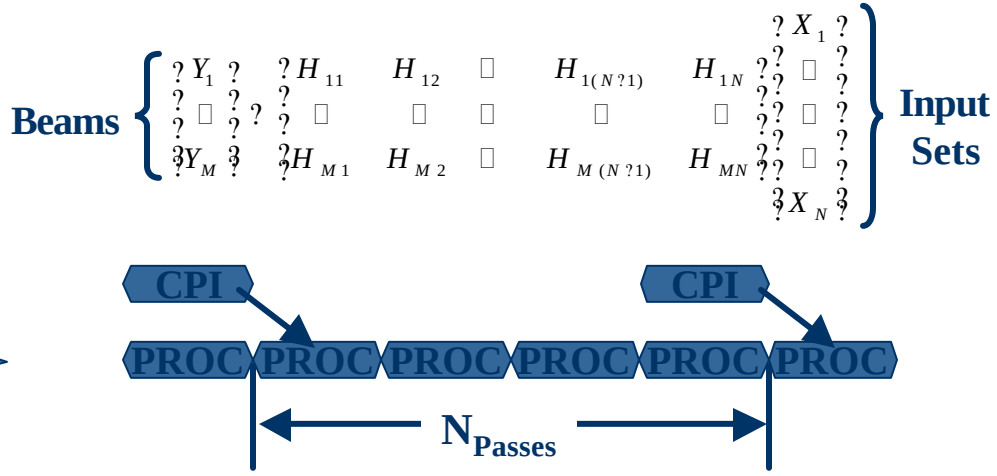


Data can also be partitioned by link: each link carried an integral number of channels

Unconstrained Linear Architecture

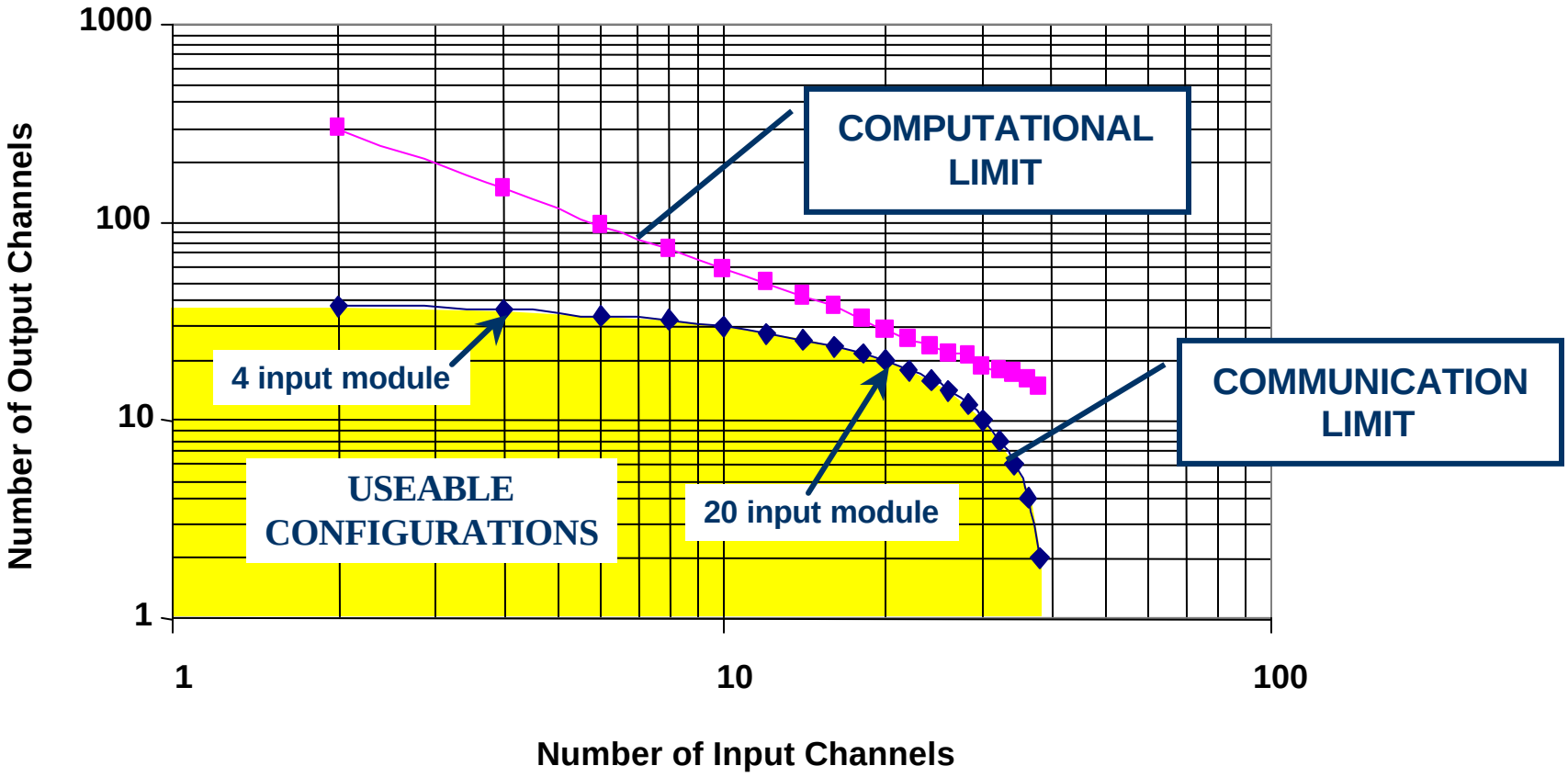
- Full MxN unconstrained complex matrix multiply
- Outputs only from a single module
- Processing throughput limited by beamformer module I/O
- Communication latency across beamformer is an issue
- Additional beams can be produced by multiple passes on data
  - Decreases overall radar duty cycle
  - Memory should be located in digital beamformer to save I/O bandwidth
  - Increased beamformer processing speed may be required





Unconstrained Linear Architecture

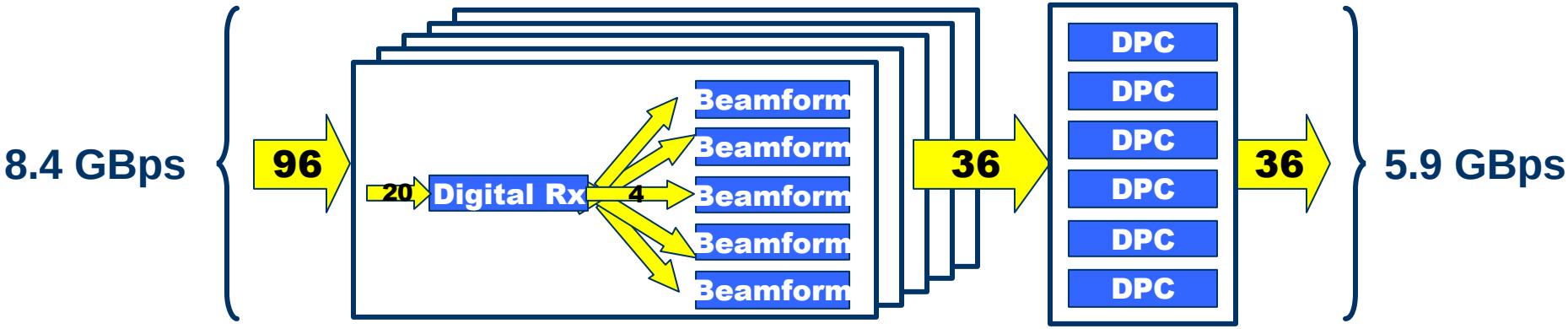
- **Unconstrained linear beamformer module is I/O bound**
  - **Total number of input links plus output links is constant**
  - **Choice of input to output balance affects utilization**

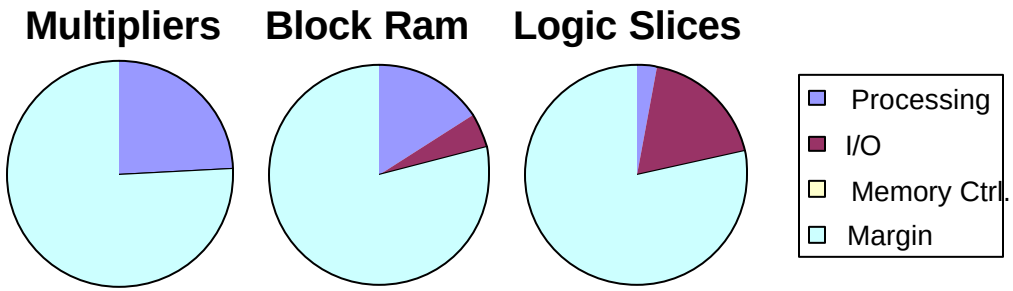


Note:adding additional non-MGT connections could potentially increase throughput

4 Input Module Realization

- I/O and compute bounds are not close -- low utilization
- 36 x 96 unconstrained matrix multiply
- 35 modules required for FPGA digital processor
  - Front-end – 5 modules
  - Small-array beamformer – 24 modules
  - Digital pulse compression – - 6 modules



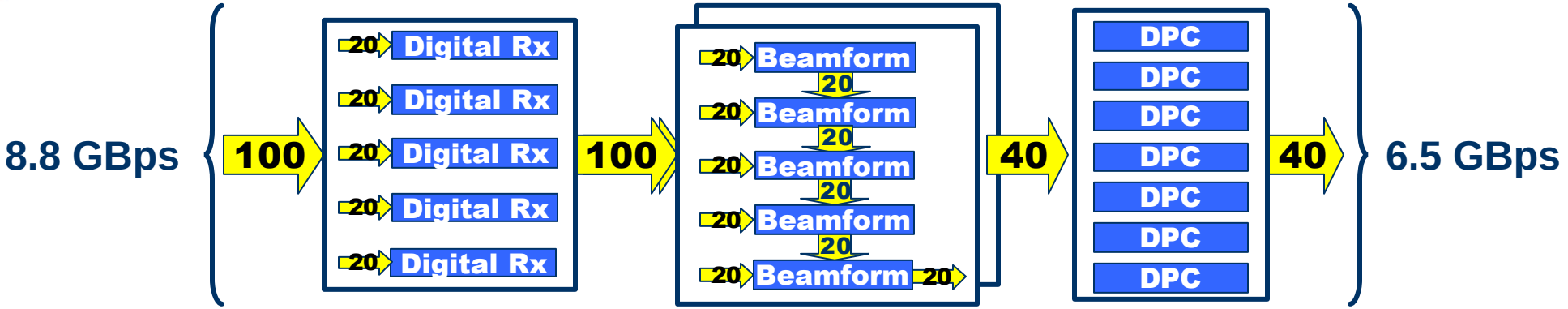


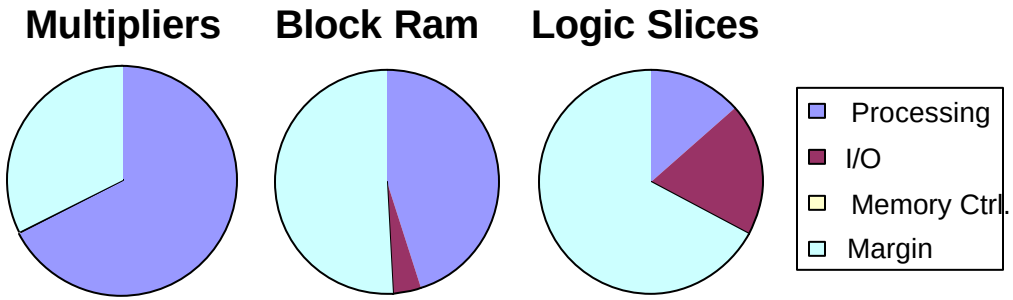
BEAMFORMER MODULE UTILIZATION

LOW FPGA UTILIZATION

20 Input Module Realization

- I/O and compute bounds are close -- good utilization
- 40 x 100 unconstrained matrix multiply
- 22 modules required for FPGA digital processor
  - Front-end – 5 modules
  - Small-array beamformer – 10 modules
  - Digital pulse compression – 7 modules



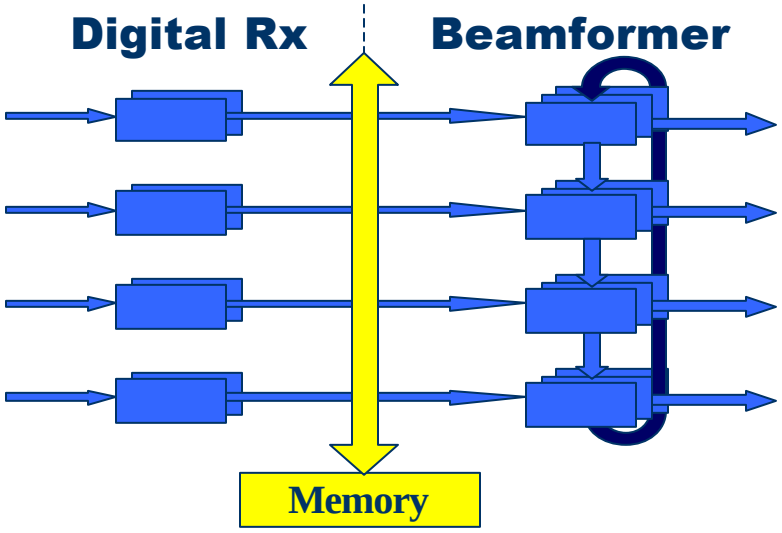


BEAMFORMER MODULE UTILIZATION

GOOD FPGA UTILIZATION

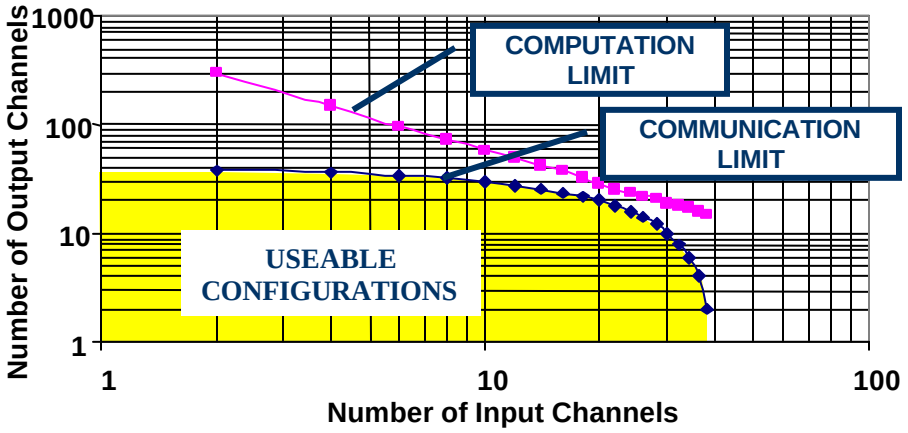
Constrained Linear Architecture

- Use each beamformer module to produce outputs
- MxN constrained complex matrix multiply
  - Use only a subset of inputs for each output
- I/O and computation bounds the as in the unconstrained case
  - Inputs and outputs must be balanced to maximize utilization



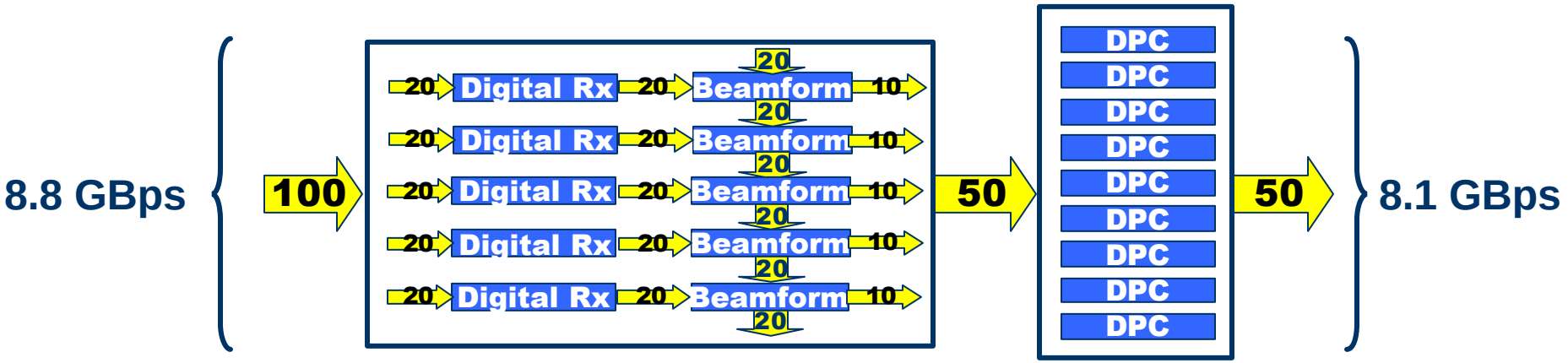
EXPLICIT ZEROS IN BEAMFORMING MATRIX

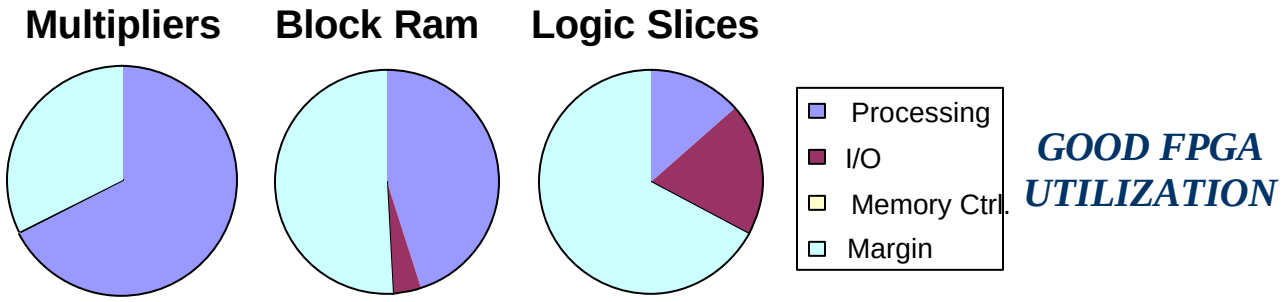
$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_M \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H_{(M-1)(N-1)} & H_{(M-1)N} & \cdots & 0 \\ 0 & 0 & \cdots & 0 & H_{MN} & \cdots & H_{M1} \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix}$$



20 Input Module Implementation

- Adding matrix constraints increases the number of outputs
- 50 x 100 constrained matrix multiply
- 19 modules required for FPGA digital processor
  - Front-end - 5 modules
  - Small-array beamformer – 5 modules
  - Digital pulse compression - 9 modules



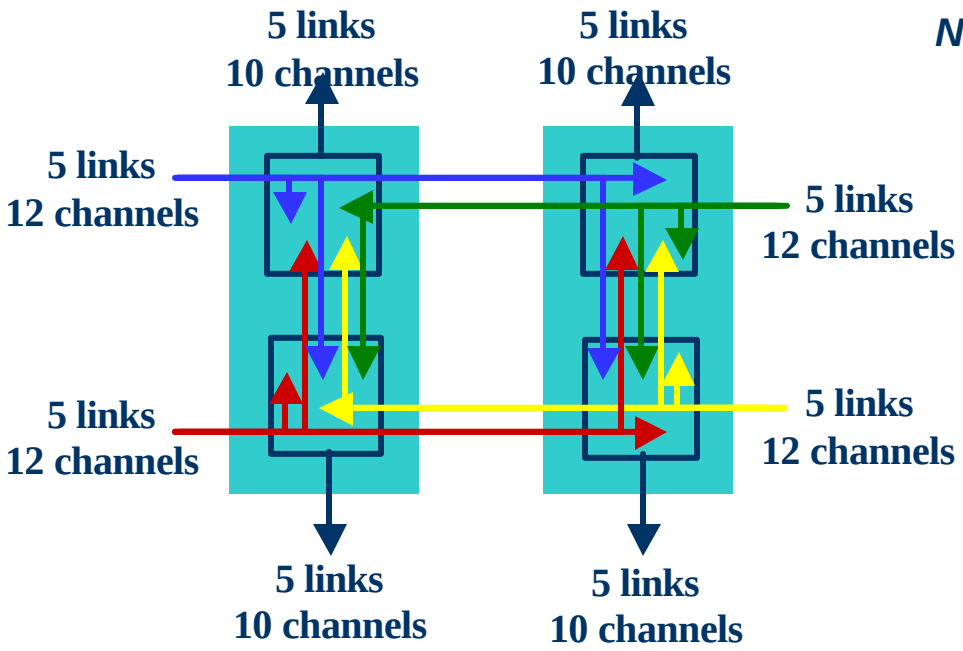


GOOD FPGA UTILIZATION

BEAMFORMER MODULE UTILIZATION

Mesh Architecture

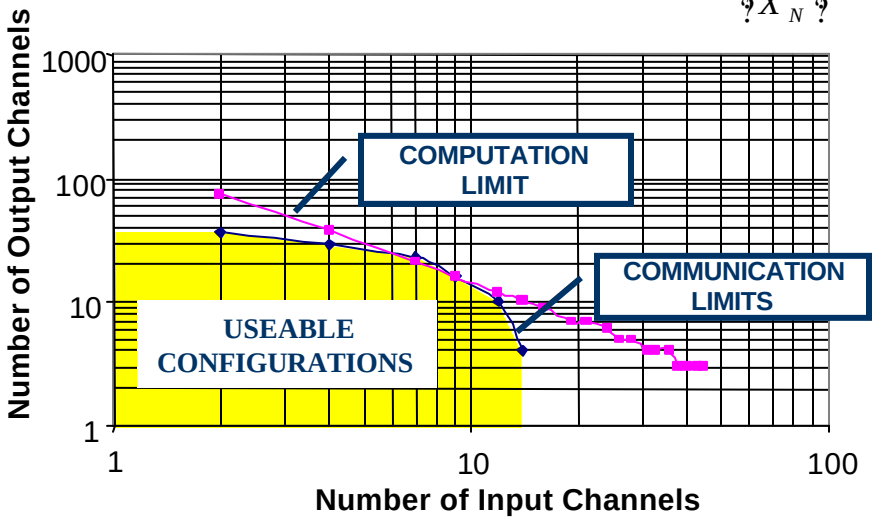
- Mesh architecture offers utilization enhancement
  - I/O and computation bounds touch
- Full unconstrained matrix multiply
- Partially formed beams sent forward for summing in DPC



40 x 48 CMAC using 4 modules

NO EXPLICIT ZEROS IN BEAMFORMING MATRIX

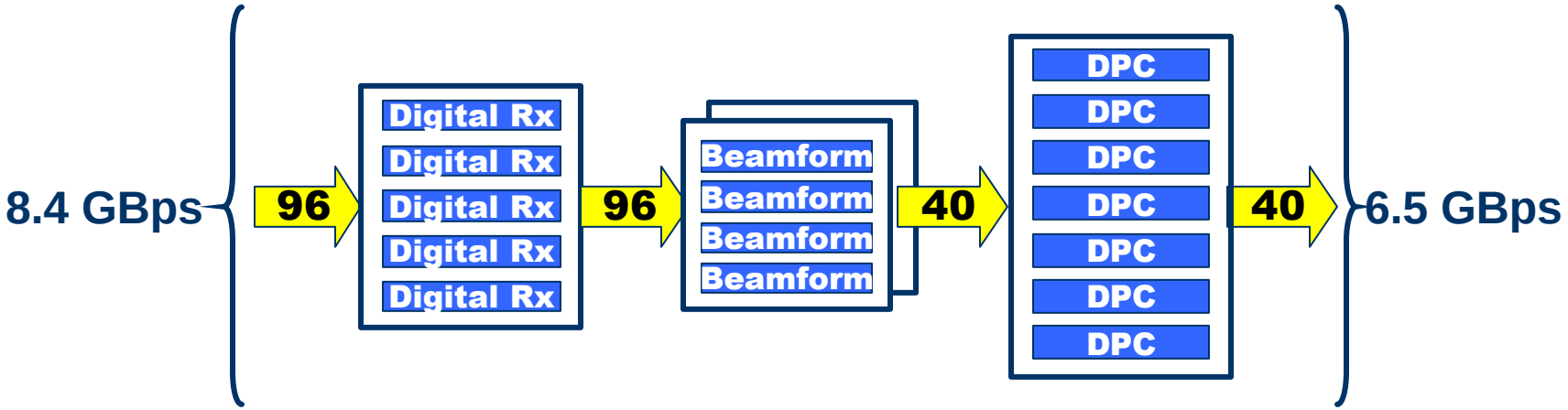
$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_M \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1(N-1)} & H_{1N} \\ \vdots & \vdots & & \vdots & \vdots \\ H_{M1} & H_{M2} & \cdots & H_{M(N-1)} & H_{MN} \end{bmatrix} \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix}$$

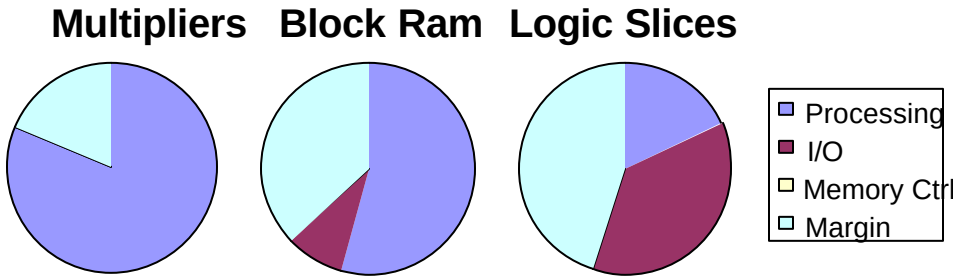


Note: Computation limit normalized for architecture

Mesh Implementation

- I/O and compute bounds touch -- high utilization
- 40 x 96 unconstrained matrix multiply
- 20 modules required for FPGA digital processor
  - Front-end – 5 modules
  - Small-array beamformer – 8 modules
  - Digital pulse compression – 7 modules





BEAMFORMER MODULE UTILIZATION

HIGH FPGA UTILIZATION

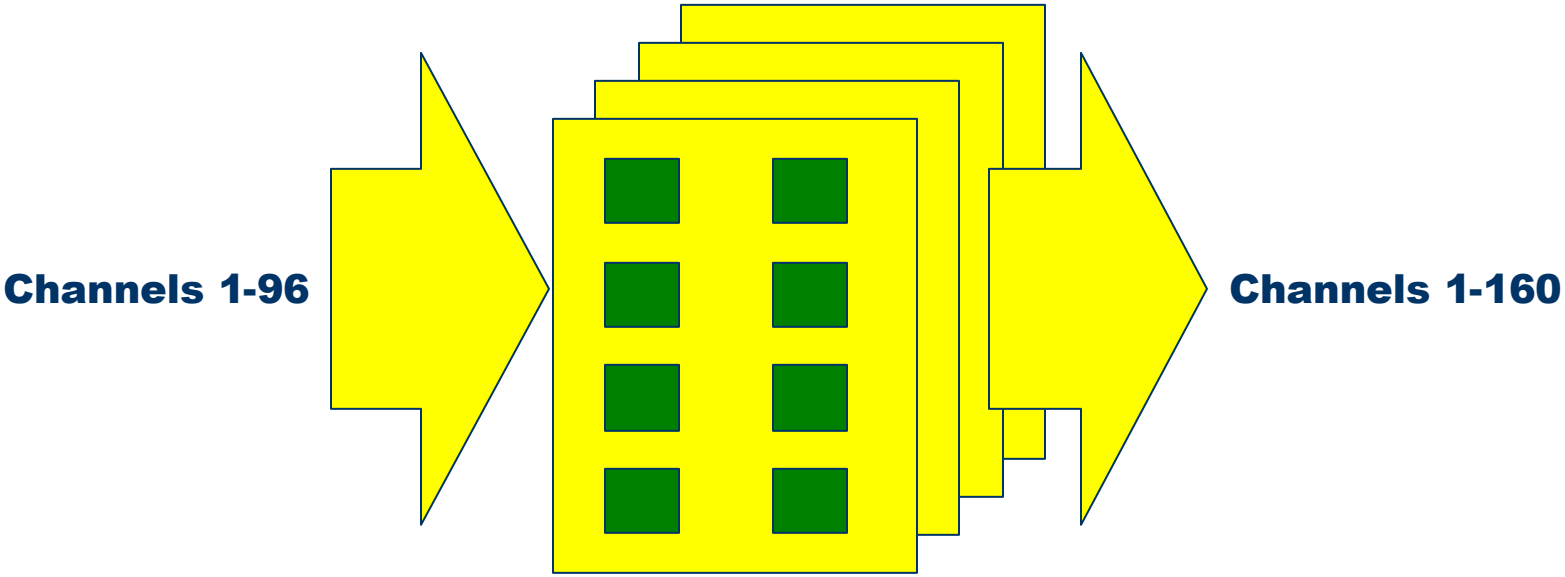
Architecture Comparison

- Mesh architecture gives highest multiplier utilization

	Unconstrained Linear	Unconstrained Linear	Constrained Linear	Mesh
Input Channels	96	100	100	96
Output Channels	36	40	50	40
Beamformer Modules	24	10	5	8
Inputs per Module	4	20	20	12
Multiplies per Output	96	100	40	96
Total Multiplies	3456	4000	2000	3840
Multiplies per Module	144	400	400	480

Large Systems

- Large systems can be created through layering beamformers
  - 8 beam system, 20 channels per beam -- 160 channels
  - 160 x 96 unconstrained matrix multiply
- 65 modules required for FPGA digital processor
  - Front-end - 5 modules
  - Small-array beamformer – 32 modules
  - Digital pulse compression - 28 modules



Computer Systems, Inc.
MERCURY
The Ultimate Performance Machine

Summary

- FPGAs can provide efficient I/O and computational power to address high input bandwidths of modern radar systems.
  - Front-end processing
  - Sub-array beamformer
  - Digital pulse compression
  - Adaptive beamforming
- System topologies that provide efficient utilization of computational and I/O resources change dramatically as system requirements scale.
  - Watch I/O and computation bounds
- Small changes in system requirements can dramatically increase complexity of FPGA implementations when computational bounds of embedded resources is exceeded.
  - Watch for symmetries in filters
  - Watch bit growth before 18-bit multipliers
- FPGAs should be used until application of adaptive beamforming weights due to high bandwidth dataflow.