

---

# ECE 697F

## Reconfigurable Computing

### Lecture 13

### *Reconfigurable Computing Applications I*

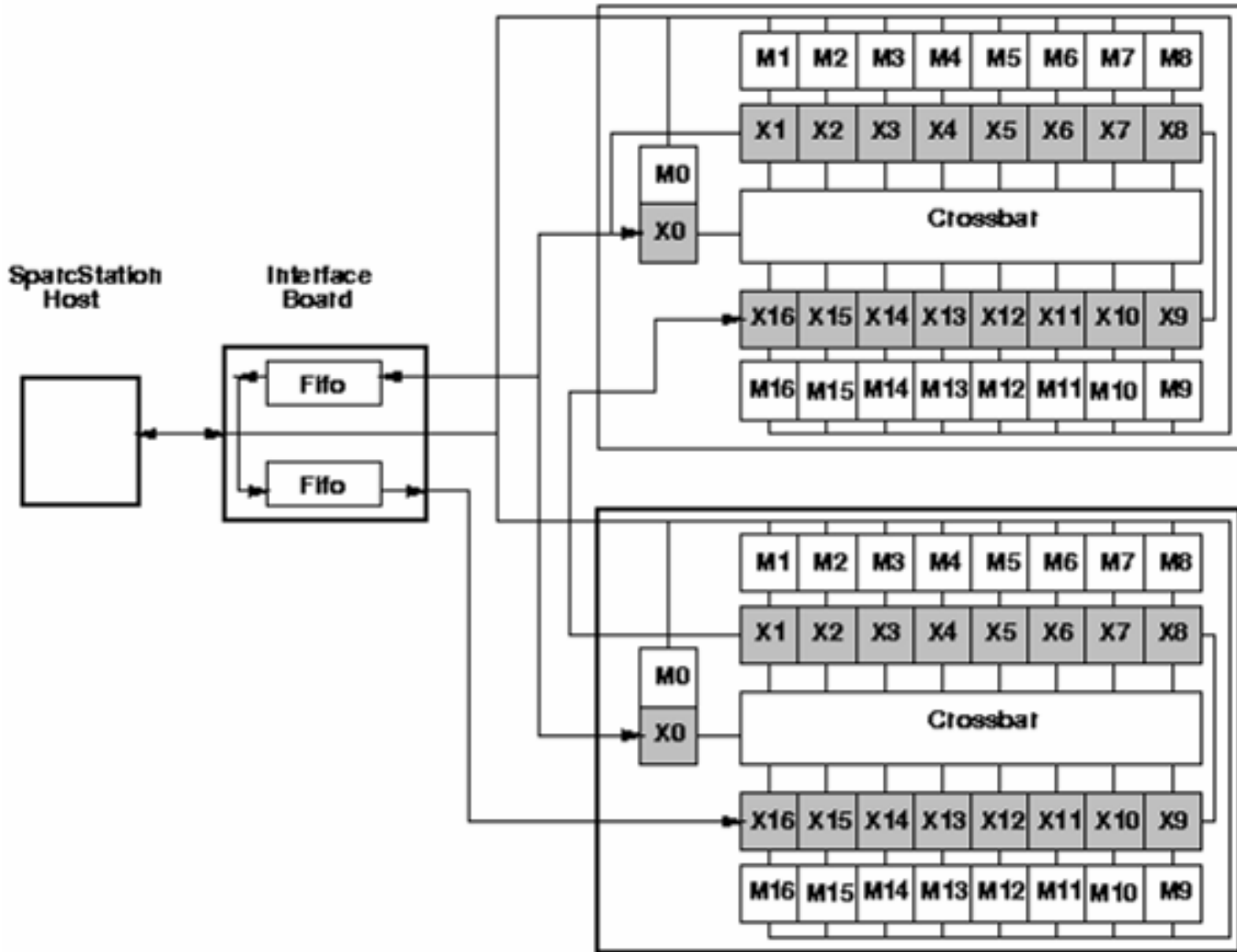


# Overview

---

- Perhaps the most well-known reconfigurable computer is **Splash/Splash 2**
- Implemented as linear, systolic array
- Developed at **Supercomputing Research Center (1990-1994)**
- Memory tightly coupled with each FPGA
- Multiple **Splash** boards can be combined to form larger system.
- **Radar signal processing**

# Splash 2 Architecture

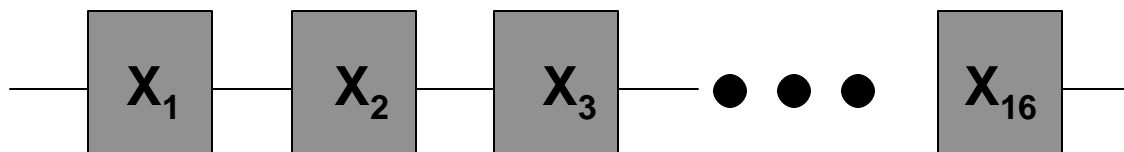


# Splash 2 Models of Computations

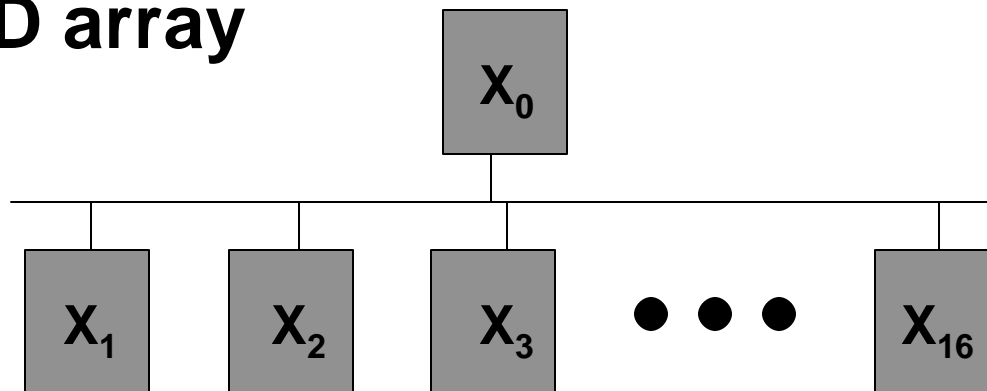
---

- **Linear (systolic) array**

- All near-neighbor communication, pipelined
- Very fast (at the time) of 20-30MHz achieved
- All FPGAs have same program



- **SIMD array**



- Instructions fanned out to all processing element
- Data across all elements collected at the end

# **Splash 2 Programming Environment**

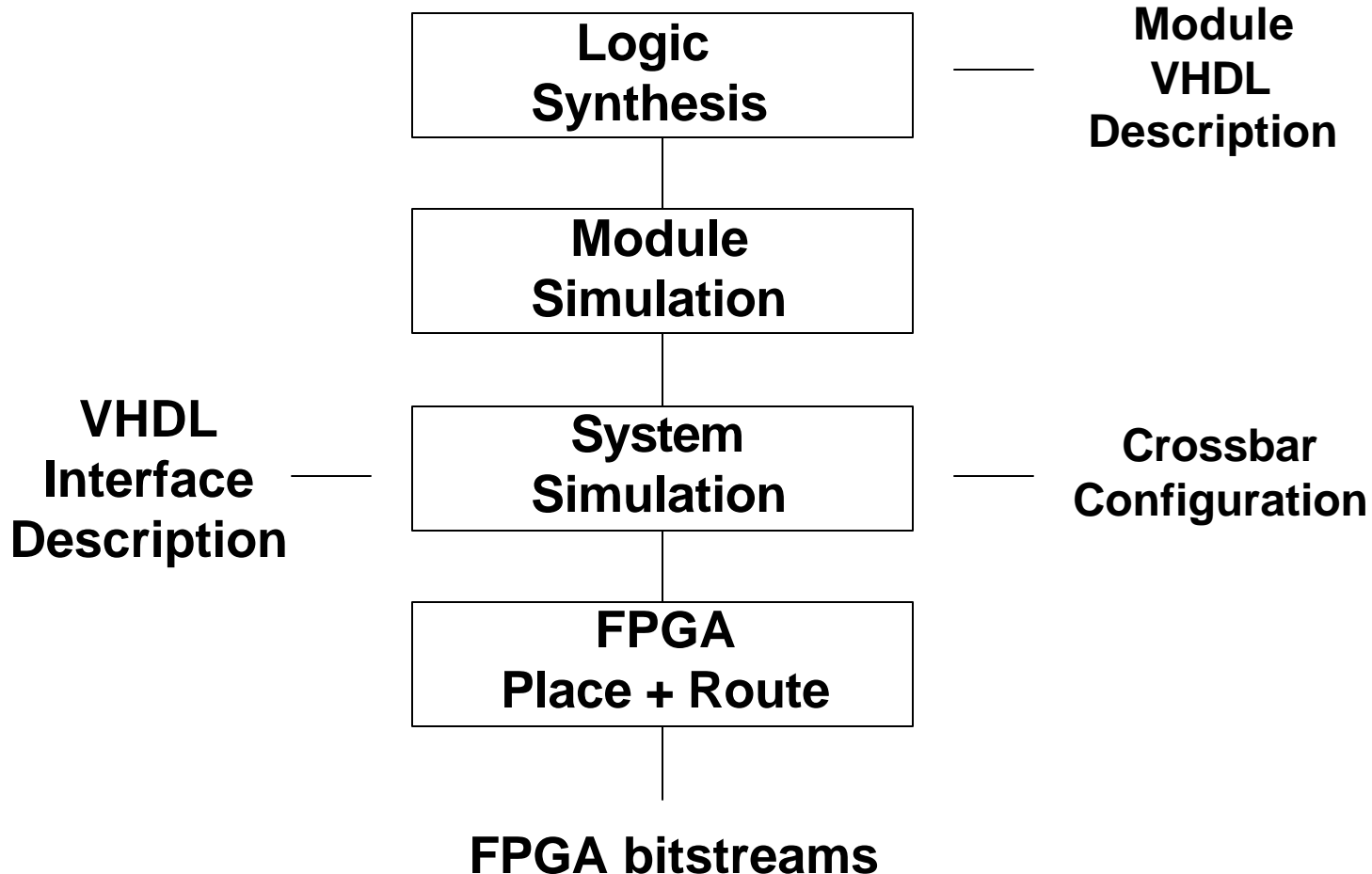
---

- **Three components to be programmed**
  - **Splash board -> crossbar configurations and FPGA configurations determined individually**
  - **Splash interface -> FIFO controls data flow to boards**
  - **Host interface -> driver software controls application execution and collection of results**
  
- **Somewhat less automated than PAM**
  - **Typically comparable to programming a parallel multiprocessor system.**

# Example Application Flow

---

- Frequently an iterative process



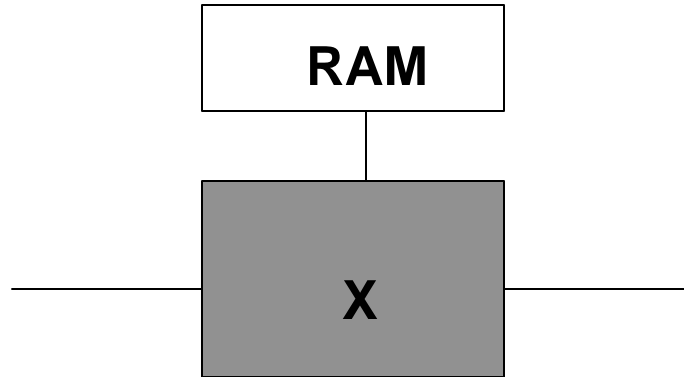
# **Application #1: Text Searching**

---

- **Search through dictionary of words for data hit**
- **Applicable to internet search engines/databases**
- **Opportunities for search parallelism**
- **Splash implementation uses systolic communication**

# Data Access

---



- Each FPGA used to look into local memory.
- Longer data words “hashed” into 18 bit address
- Valid bit in memory indicates if data value is currently stored.
- Could be stored in several locations

# Example Hash Function

---

Shift amount: 7 bits

Hash function: 1100 1000 1010 0011

00 0000 0000 0000 0000 0000

01 1010 0001 1101 00

-----

10 1000 0011 0101 1100 0000

Clear hash register  
Input the letters "th"

Temporary Result

10 0000 0101 0000 0110 1011

00 0000 0001 1001 01

-----

01 0010 0110 0001 1110 1011

Result for "th"  
Input for letters "e\_"

Temporary result

10 0101 1010 0100 1100 0011

Result for "the\_"

- XOR two character value with temp result and hash function
- Rotate result
- Different hash function for each FPGA

# Text Searching Tips

---

- **Distribute dictionary in parallel to all memories**
- **Collect word values in FIFOs**
- **Distribute words two characters at a time across all devices.**
- **Perform local hashing and lookup in parallel**
- **Collect “hit” result at end**

# Results

---

- **Splash 2 implementation runs at 25 MHz**
- **Three phases needed**
  - **Fetch 2 bit-sliced characters**
  - **Perform hash**
  - **Table look-up**
- **Takes advantage of both systolic and SIMD modes.**

# Application #2: Genetic Pattern Matching

- Evaluate similarities between pairs of genetic sequences
- Edit distance defined as similarity between sequences

abqrt

acqsdh

- Operations include deleting characters, inserting characters, substituting characters
- Existing approach iterative (dynamic program) comparing one position at a time.

# Base Comparison Cell

---

Let  $S = [s_1 s_2 \cdots s_m]$  be the source sequence,  $T = [t_1 t_2 \cdots t_n]$  the target sequence, and  $d_{i,j}$  the distance between the subsequences  $[s_1 s_2 \cdots s_i]$  and  $[t_1 t_2 \cdots t_j]$ . Then for  $1 \leq i \leq m, 1 \leq j \leq n$ ,

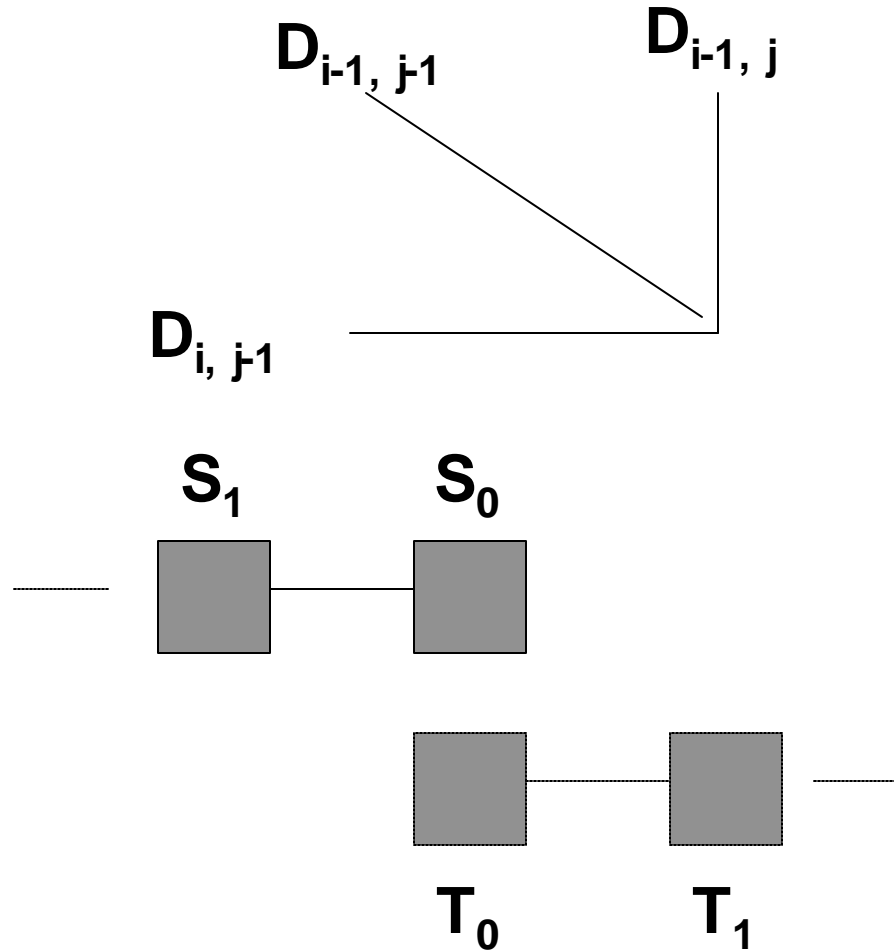
$$\begin{aligned}d_{0,0} &= 0 , \\d_{i,0} &= d_{i-1,0} + c(s_i, \emptyset) , \\d_{0,j} &= d_{0,j-1} + c(\emptyset, t_j) ,\end{aligned}$$

and

$$d_{i,j} = \min \begin{cases} d_{i-1,j} + c(s_i, \emptyset) \\ d_{i,j-1} + c(\emptyset, t_j) \\ d_{i-1,j-1} + c(s_i, t_j) \end{cases} , \quad (1)$$

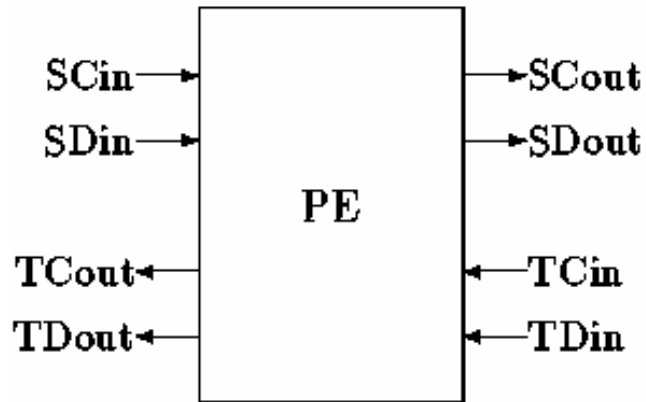
where  $c(s_i, \emptyset)$  is the cost of deleting  $s_i$ ,  $c(\emptyset, t_j)$  is the cost of inserting  $t_j$ , and  $c(s_i, t_j)$  is the cost of substituting  $t_j$  for  $s_i$ . The edit distance between  $S$  and  $T$  is simply  $d_{m,n}$ .

# Genetic Search Implementation



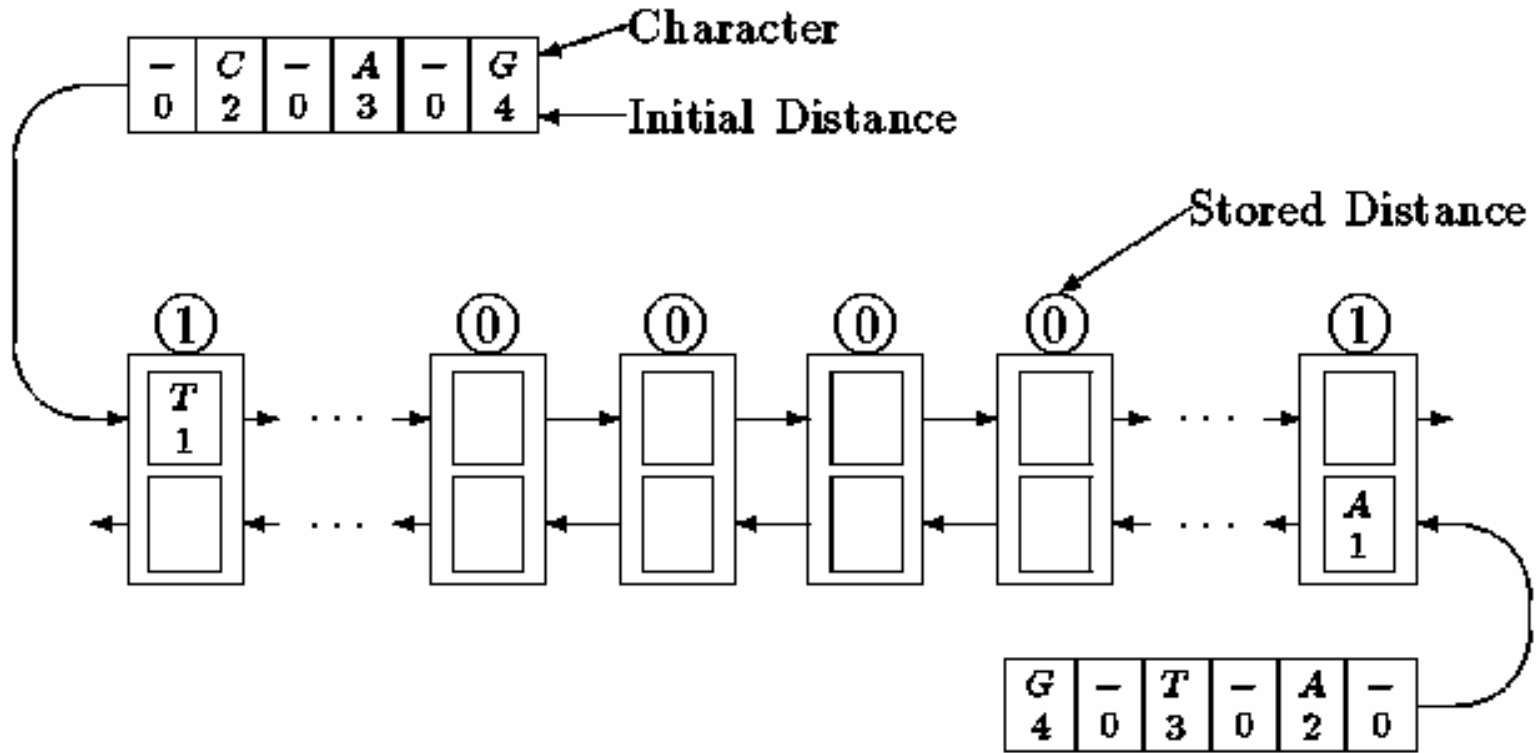
- Bidirectional linear array used to transfer information back and forth
- Run time set at  $O(mn)$  for compares/accumulates.

# Splash 2 Data Flow



```
loop
  if (SCin  $\neq$   $\emptyset$ ) and (TCin  $\neq$   $\emptyset$ ) then
    PEDist  $\leftarrow$  min { PEDist+c(SCin,TCin),
                        TDin+c(SCin, $\emptyset$ ),
                        SDin+c( $\emptyset$ ,TCin)
    }
    SDout  $\leftarrow$  PEDist
    TDout  $\leftarrow$  PEDist
  elsif (SCin  $\neq$   $\emptyset$ ) then
    SDout  $\leftarrow$  SDin
    TDout  $\leftarrow$  SDin
  elsif (TCin  $\neq$   $\emptyset$ ) then
    SDout  $\leftarrow$  TDin
    TDout  $\leftarrow$  TDin
  else
    SDout  $\leftarrow$  PEDist
    TDout  $\leftarrow$  PEDist
  endif
  SCout  $\leftarrow$  SCin
  TCout  $\leftarrow$  TCin
endloop
```

# Splash 2 Data Flow



# Genetic Search Result

---

- Nearly linear scaling in cell updates per second (CUPS)
- Need to reuse array for large patterns

Table 1: Benchmark of DNA sequence comparison  
(Values are rounded to two decimal places.)

Hardware	Specifics	CUPS
Splash 2	unidir; 16 boards	43,000M
Splash 2	bidir; 16 boards	34,000M
Splash 2	unidir; 1 board	3,000M
Splash 2	bidir; 1 board	2,100M
Splash 1	bidir; 746 PE's	370M
MP-1 [13]	8K PE's	32M
CM-2 [14]	16K PE's	5.9M
BSYS [14]	100 PE's	2.9M
SPARC 10/30GX	gcc -O2	1.2M
P-NAC [4]		1.1M
VAX 6620	VMS; CC	1.0M
SPARC 1	gcc -O2	0.87M
486DX-50 PC	DOS; gcc -O2	0.67M

# **Application #3: Building Pyramids**

- **Reconfigurable computers well suited to image processing due to high parallelism and specialization (filtering)**
- **Algorithms change sufficiently fast such that ASIC implementations become outdated.**
- **Examine two issues with Splash**
  - **Image compression and image error estimation**
- **Parallelize across array in SIMD and systolic fashion**

# Pyramid Operations

---

- **Gaussian Pyramid**

- **Down sample image to compress image size for communication.**

$$g_k(i, j) = \sum_{m=-2}^2 \sum_{n=-2}^2 w(m, n) g_{k-1}(2i+m, 2j+n)$$

- **Average over a set of points to create new point**



- **Laplacian Pyramid**

- **Determine error found from Gaussian Pyramid**
- **Expand contracted picture and compare with original**

# Gaussian Pyramid Implementation

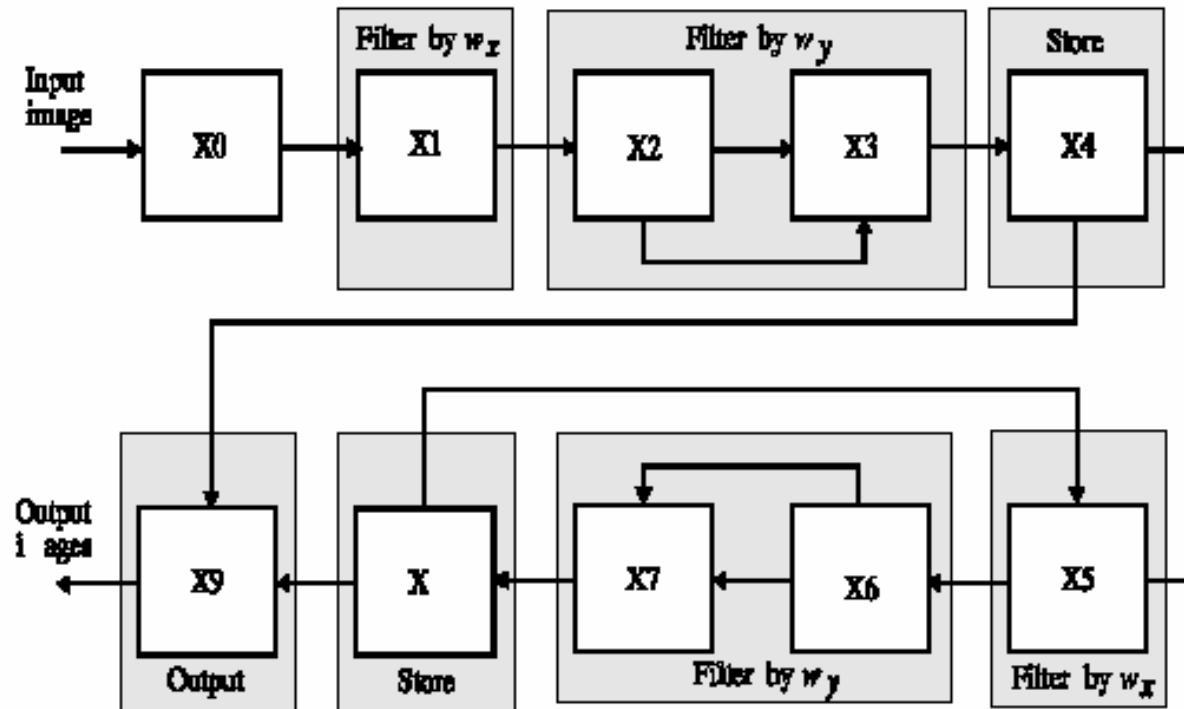


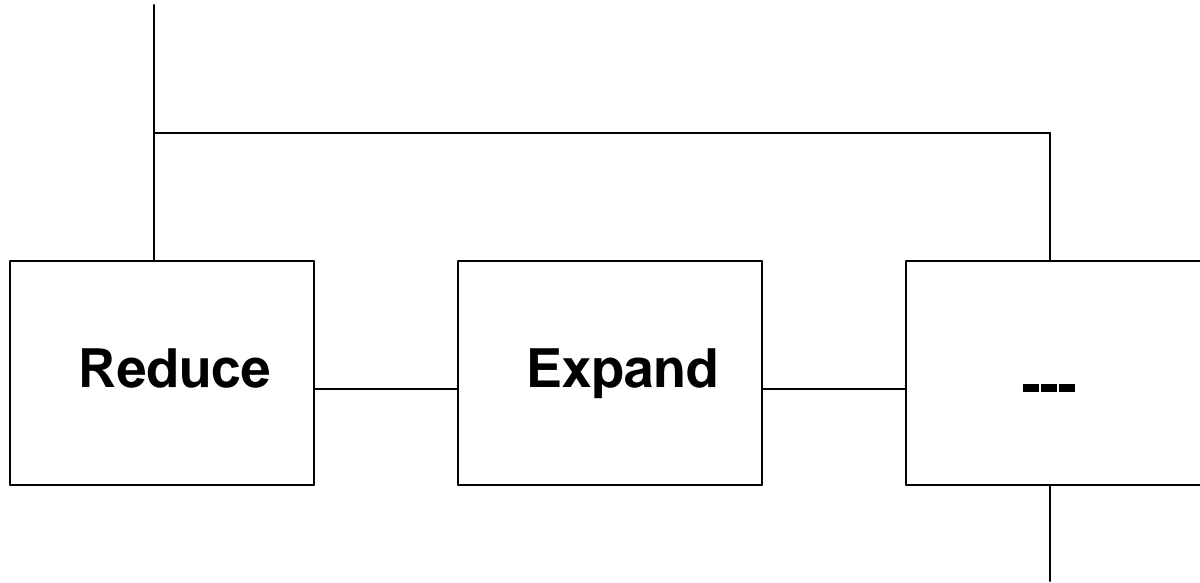
Figure 6. *SPLASH 2* implementation of a 5-level Gaussian pyramid generator. This design can generate the upper 4 levels at video frame rates of 30 images/second. Xilinx chips X5-X8 duplicate the circuitry in X1-X4.

- **Systolic array in which each device performs a separate function.**
- **Limited by clock rate of slowest device.**

# Laplacian Pyramid

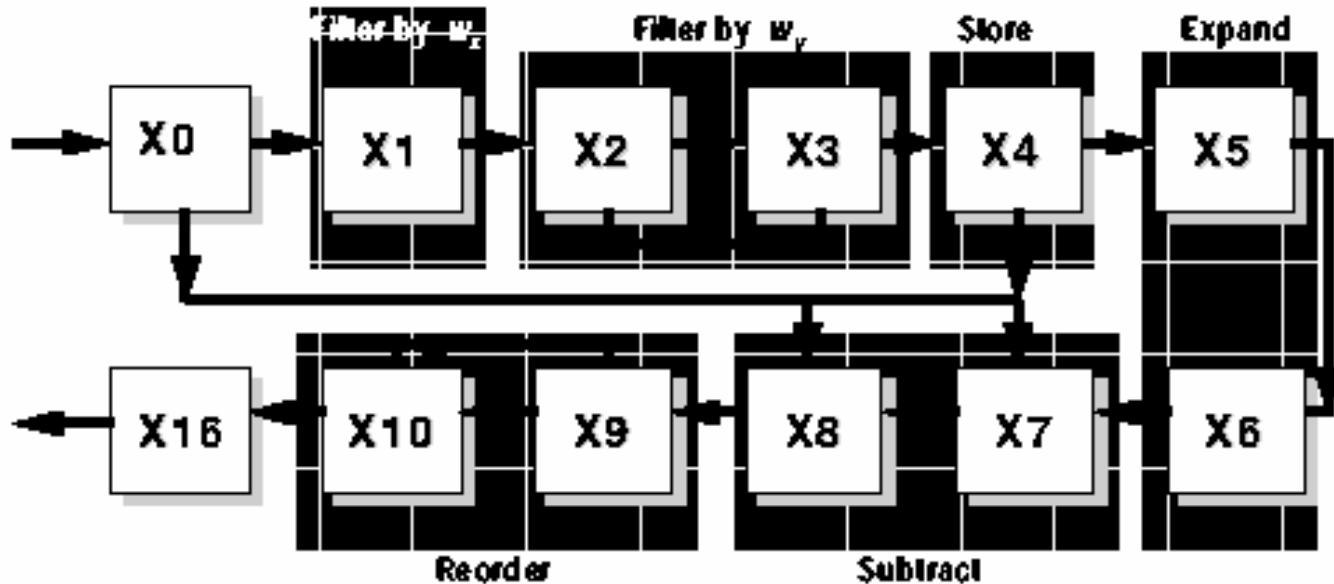
---

Original image



- Use interpolation to expand reduced image
- Error calculation can be used to tune reduction operation (filtering)

# Gaussian/Laplacian Pyramid Flow



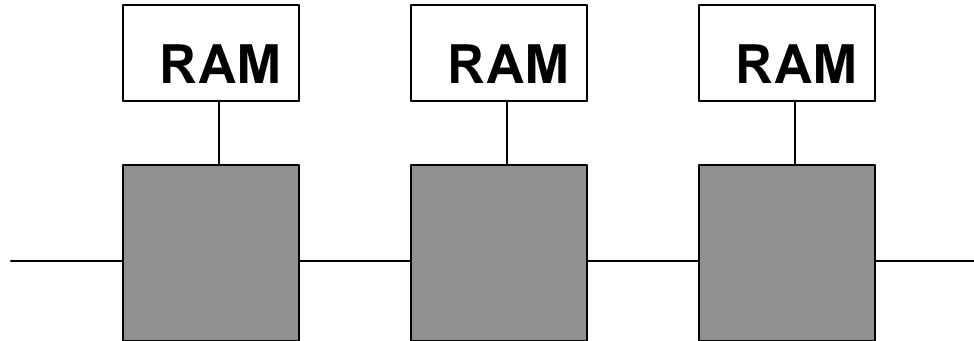
LEGEND:  $X_n$  = XC4010 and 256k x 16 RAM

- Generates both Gaussian and Laplacian pyramid for 512 x 480 image in 22.7 ms at 15.7 MHz
- Comparable to custom devices.

# Other Image Processing

---

- Target recognition
- Break image into “chips”
- Each chip passed through linear array in attempt to match with stored image



- Images can be rotated, mirrored.
- Zoom in if suspicious object found.

# BEE2 Benchmark applications

---

- **1024 channel dual polarization Polyphase Filter Bank (PFB) with 8K tap filter coefficients**
- **1024 channel 2 input dual polarization cross correlator (XMAC)**
- **256 million channel PFB based spectrometer**
- **All optimizations were performed at high level**

Courtesy: Chen

# Radio Astronomy Driver Applications

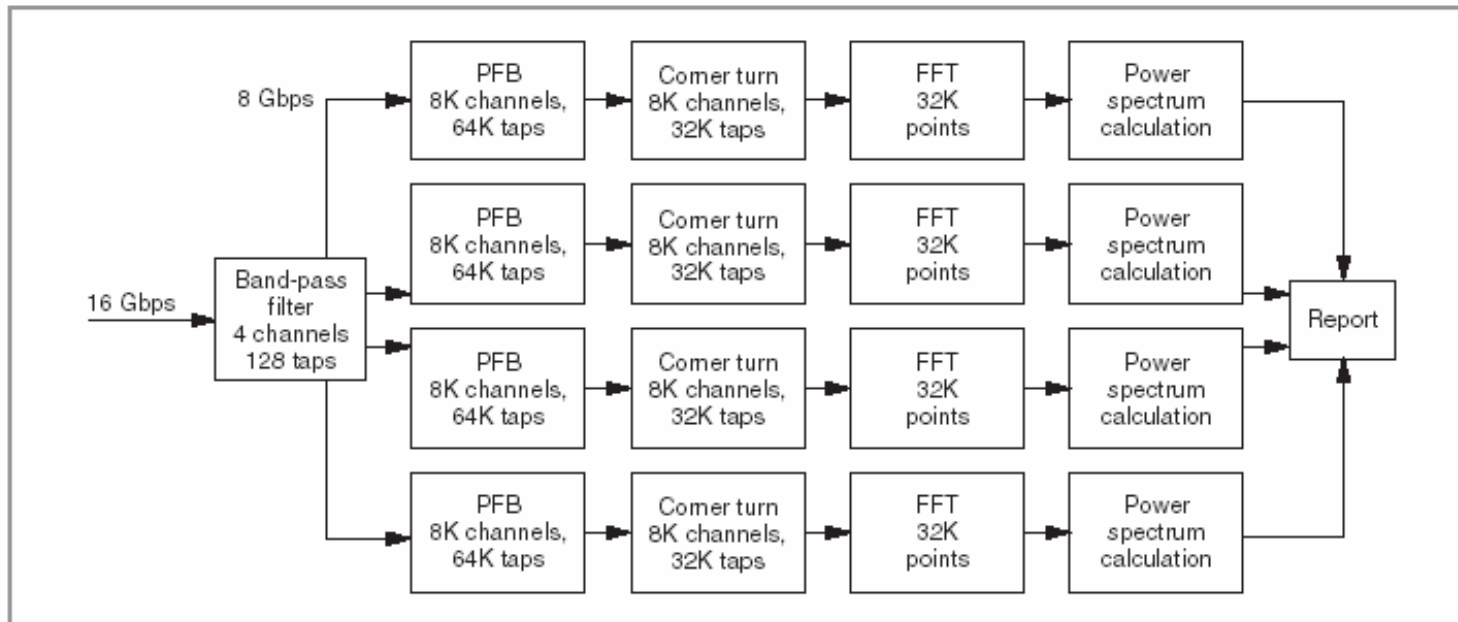
---

## ◦ SETI Spectrometer

- 800~1000 MHz input bandwidth (4 bit I & Q)
- 1 billion channel spectrometer (0.745 Hz resolution)
- In a single BEE2 module
- Approach requires significant multiply-accumulates
- Requires converting signals from time domain to frequency domain
- BEE2 implements a bandpass filter

Courtesy: Chen

# Billion Channel Spectrometer

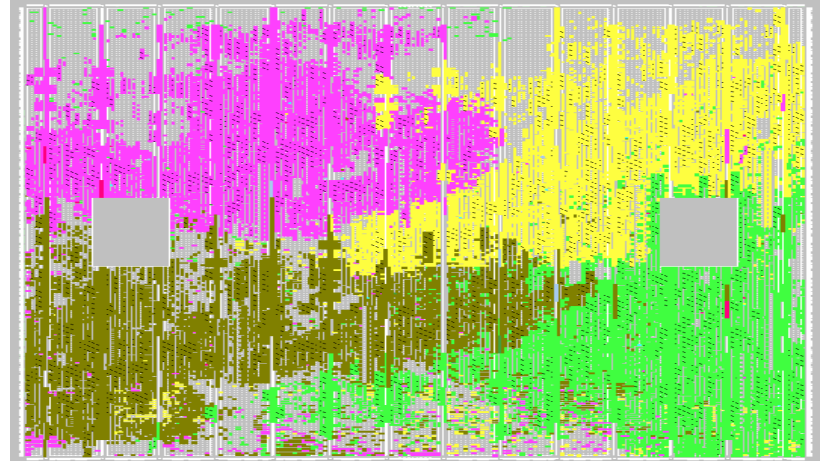


- **Performs frequency isolation**
  - 256 million 0.745 MHz Hz channels
- **Four data streams**
  - Computation takes place in each FPGA

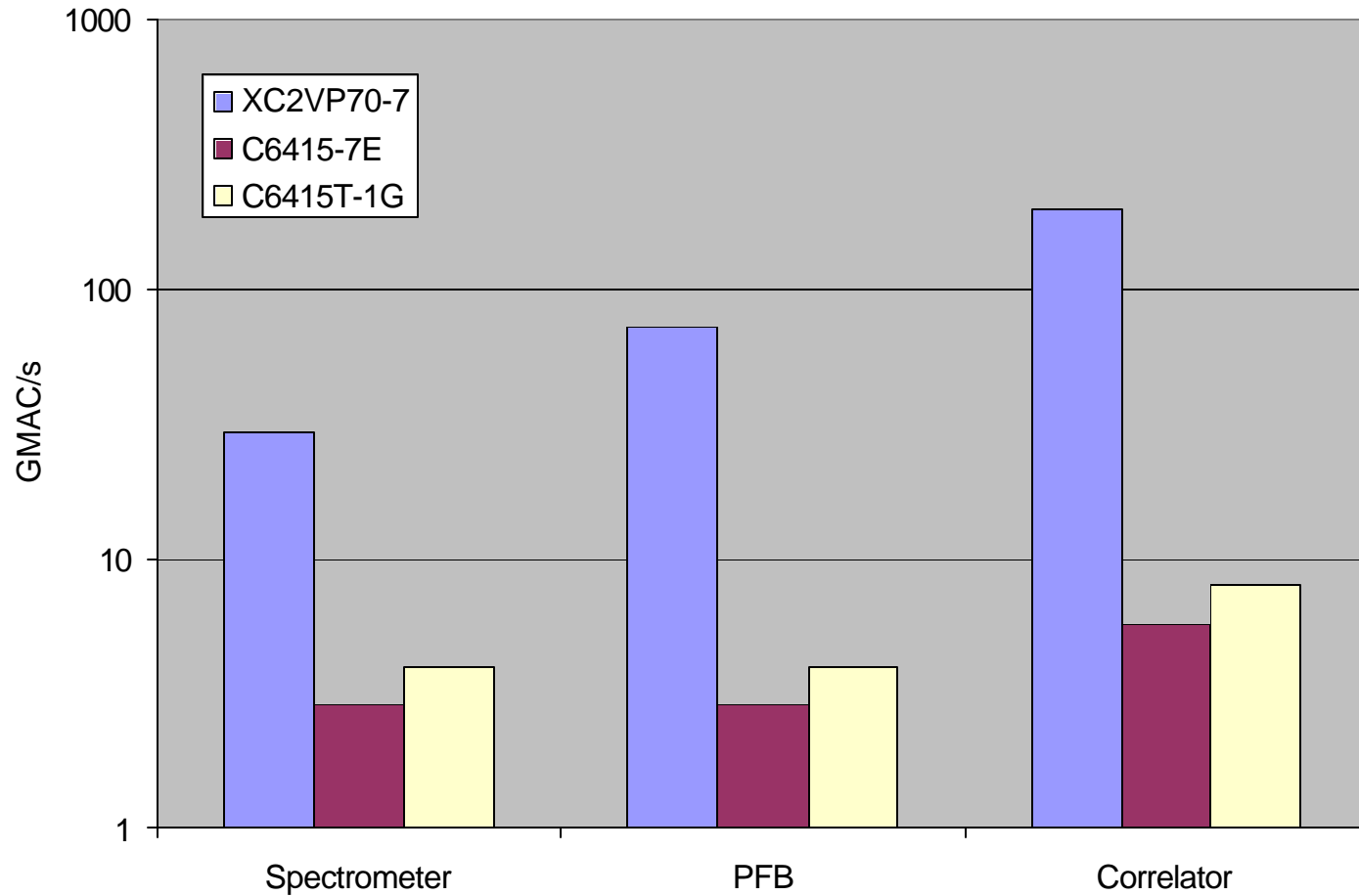
# PFB1K (4 instances in 1 FPGA)

---

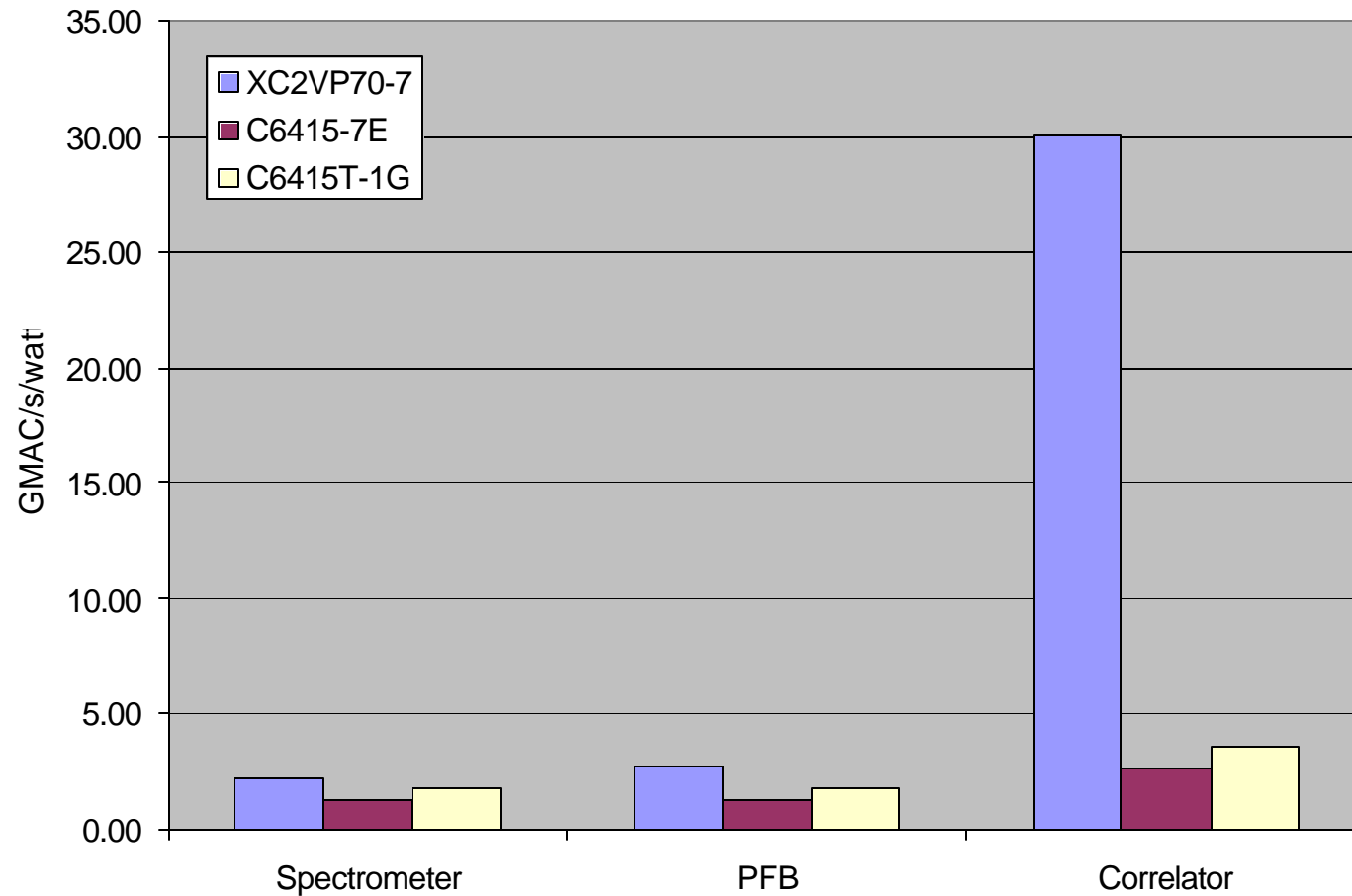
- **Resource Utilization:**
  - Flip Flops: 45,856 (69%)
  - LUTs: 14,816 (22%)
  - Slices: 25,380 (76%)
  - Block RAMs: 216 (65%)
  - MULT18X18s: 256 (78%)
- **Max clock rate:**
  - 252.8MHz (2VP70-7)
  - 72GMAC/s per FPGA @250MHz
- **Power consumption: 26.5W**
- **Tool Flow run-time/Mem**
  - Matlab/XSG: 10min/303MB
  - Synth: ~2 min/250MB
  - XFLOW: 84 min/1GB



# Sustained throughput: FPGA 10~34 times faster

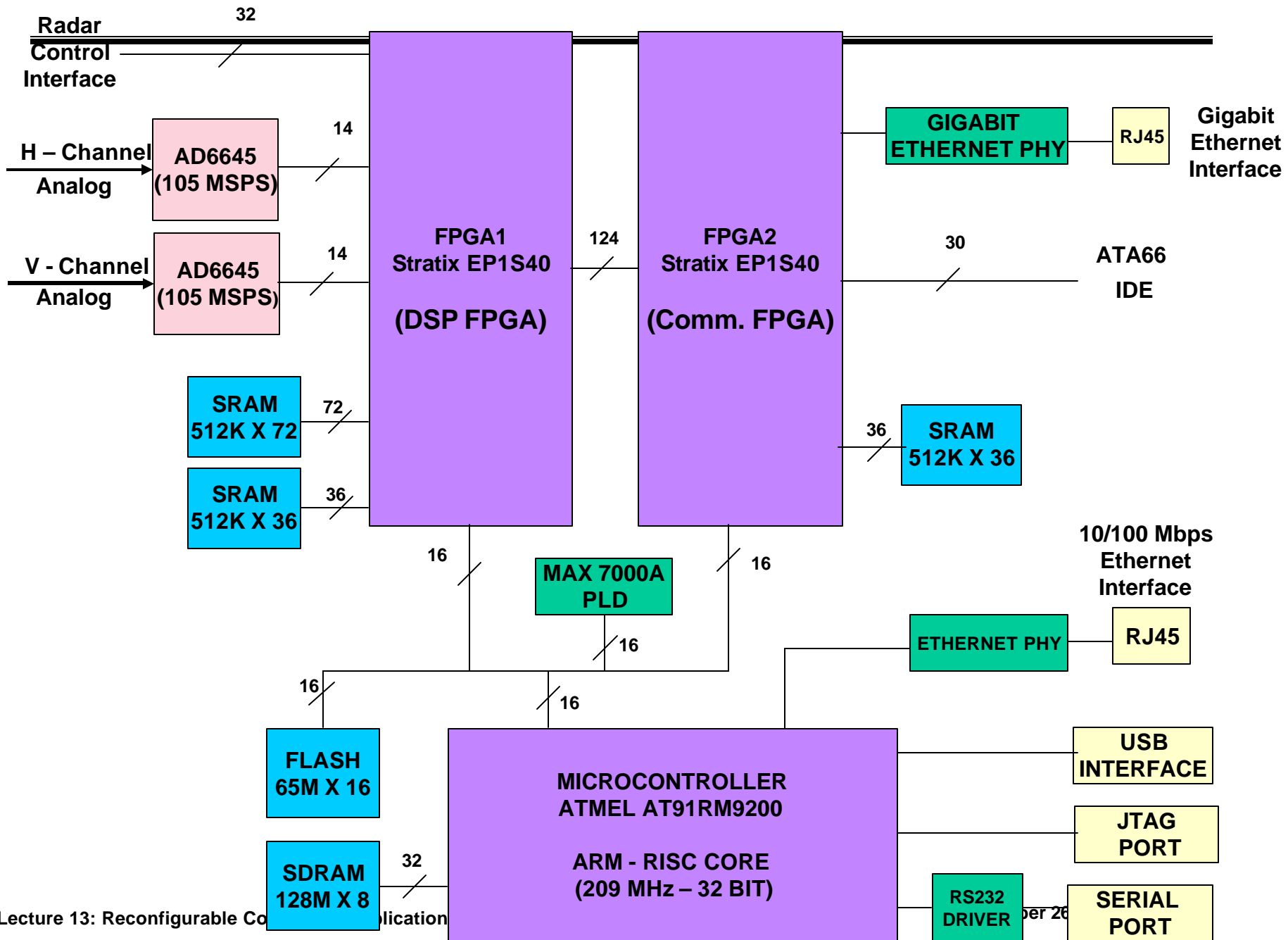


# Throughput / Power consumption



**FPGA is 72% to 11X more efficient**

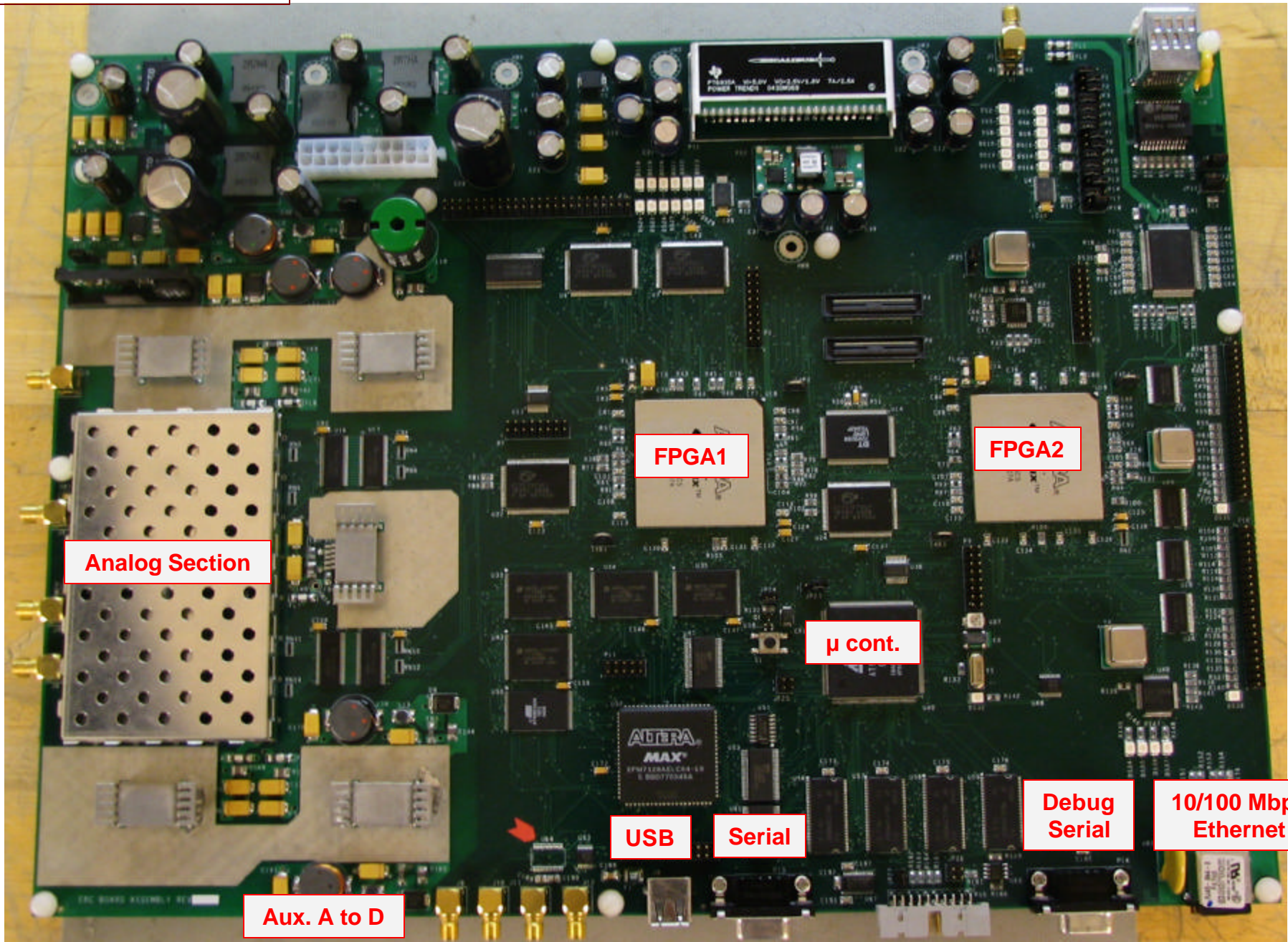
# Data Acquisition System



No. of layers: 10  
Dimensions: 15" x 11"

# Data Acquisition System

Gigabit Ethernet



Analog Section

FPGA1

FPGA2

μ cont.

USB

Serial

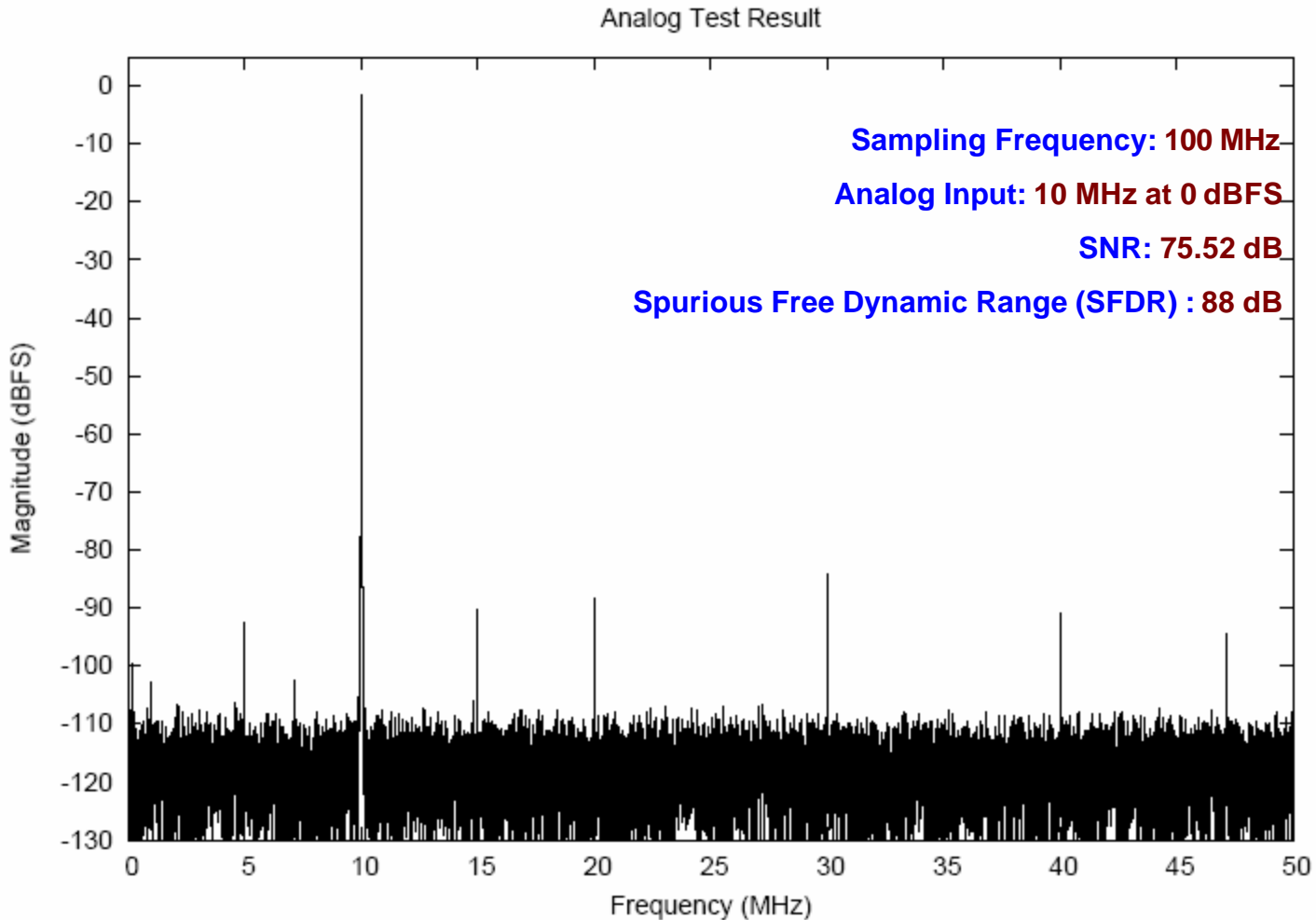
Debug  
Serial

10/100 Mbps  
Ethernet

Aux. A to D

# System Hardware -- Testing

## Analog Front End Single Tone Response



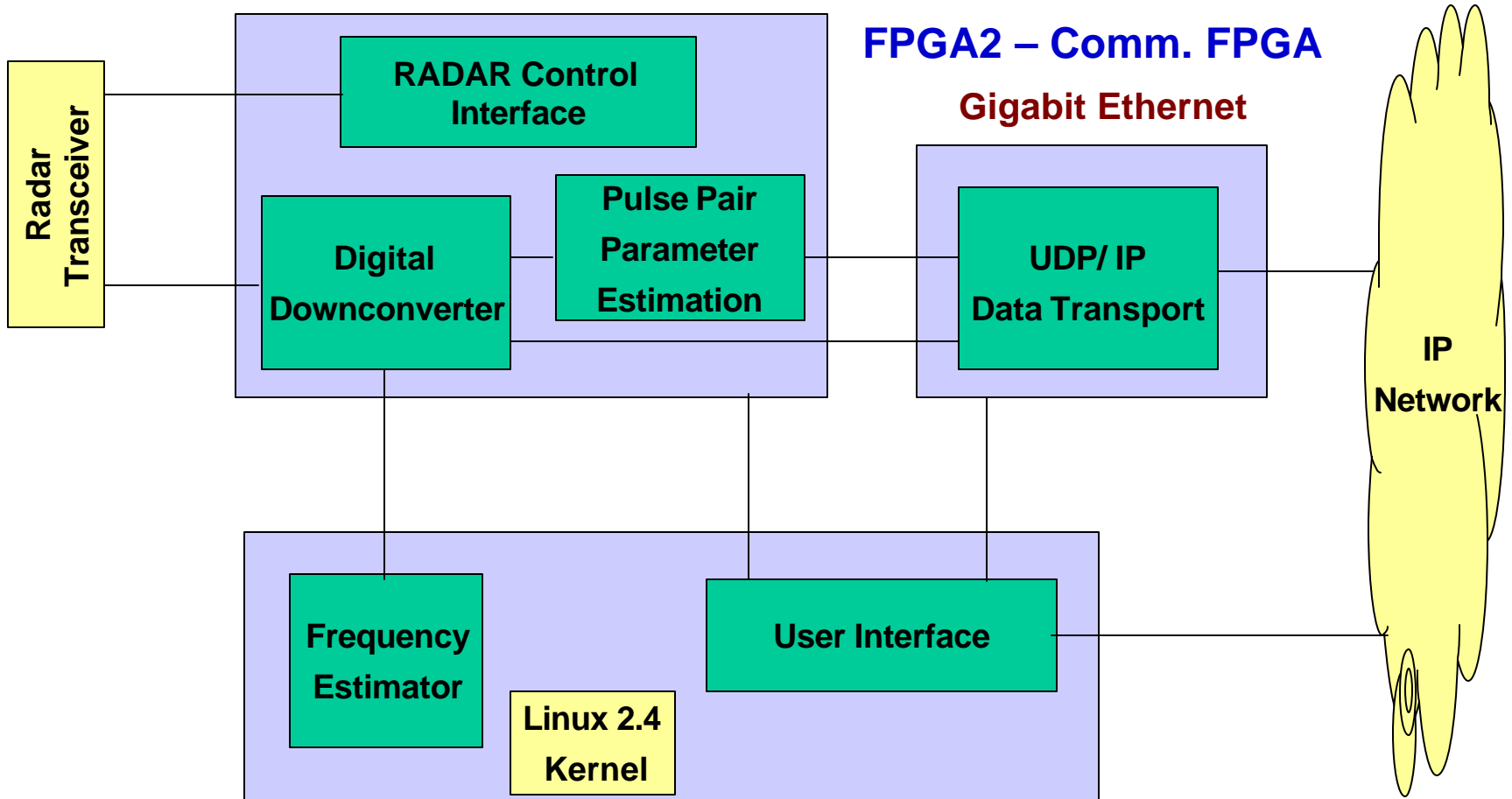
# System Firmware

## FPGA1 – DSP FPGA

Software Radio Functions for weather radar processing

## FPGA2 – Comm. FPGA

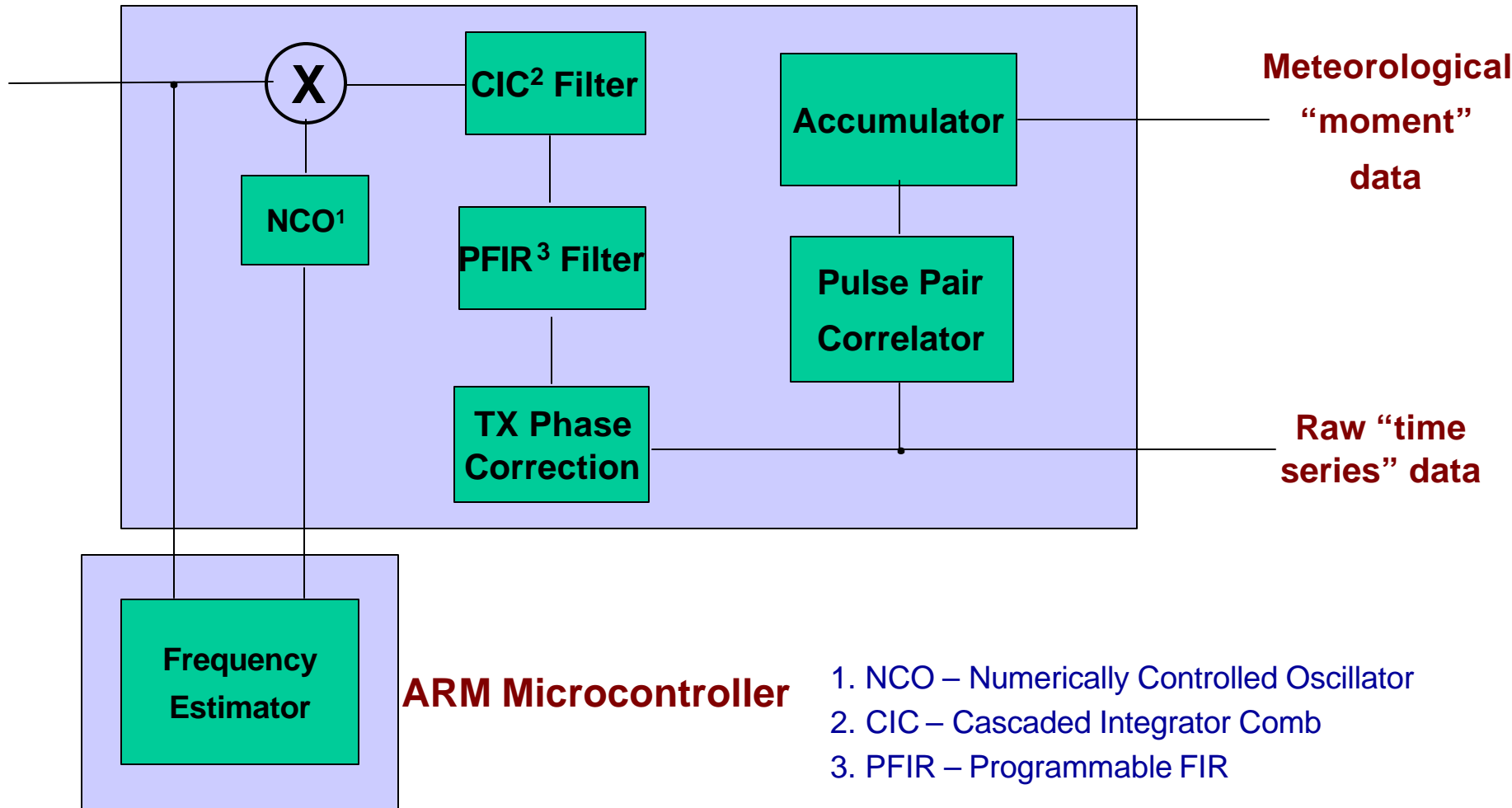
Gigabit Ethernet



# System Firmware *cont'd.*

## DSP Software

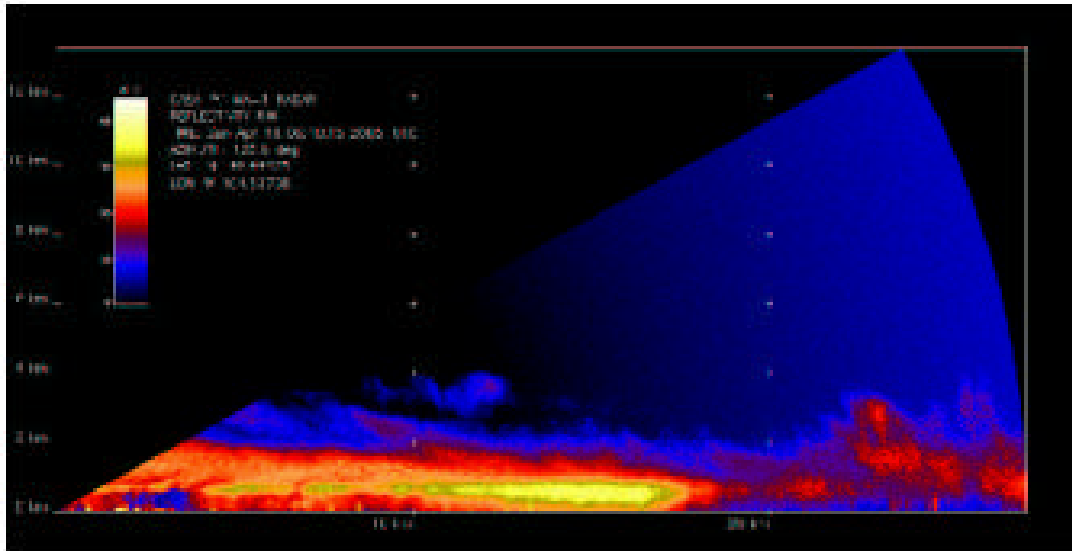
### FPGA1



# Results

---

## Radar Reflectivity Image of Winter Storm in Colorado



**Received Power in a vertical cross section of a winter snow event.**

**Testing and calibration at Colorado State University's CHILL\* radar facility.**

\* *Colorado State University – CHILL National Radar Facility - <http://chill.colostate.edu/>*

# Summary

---

- **Splash 2 effective due to scalability and programming model.**
- **Parameterizable applications benefit that are regular and distributed**
- **High bandwidth effective for searching/signal processing**
- **Challenges remain in software development.**
- **Radar signal processing has been shown to be an effective reconfigurable computing application**