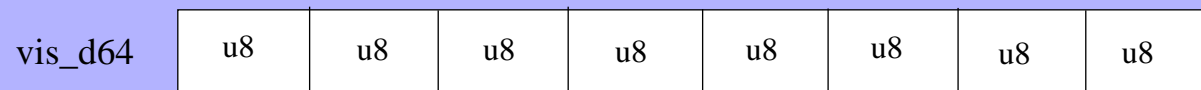
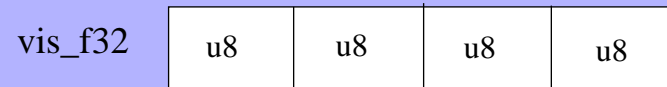
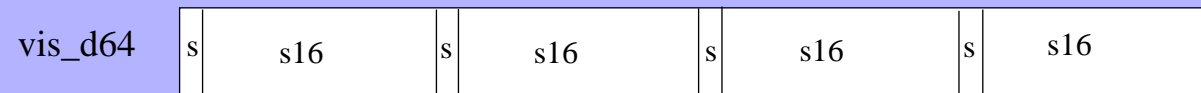
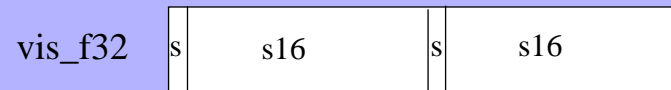


# VIS data formats

## *Pixel Data*



## *Audio Data*



# Performance Results

With no fixed-point scaling  
With scaling: 2.0

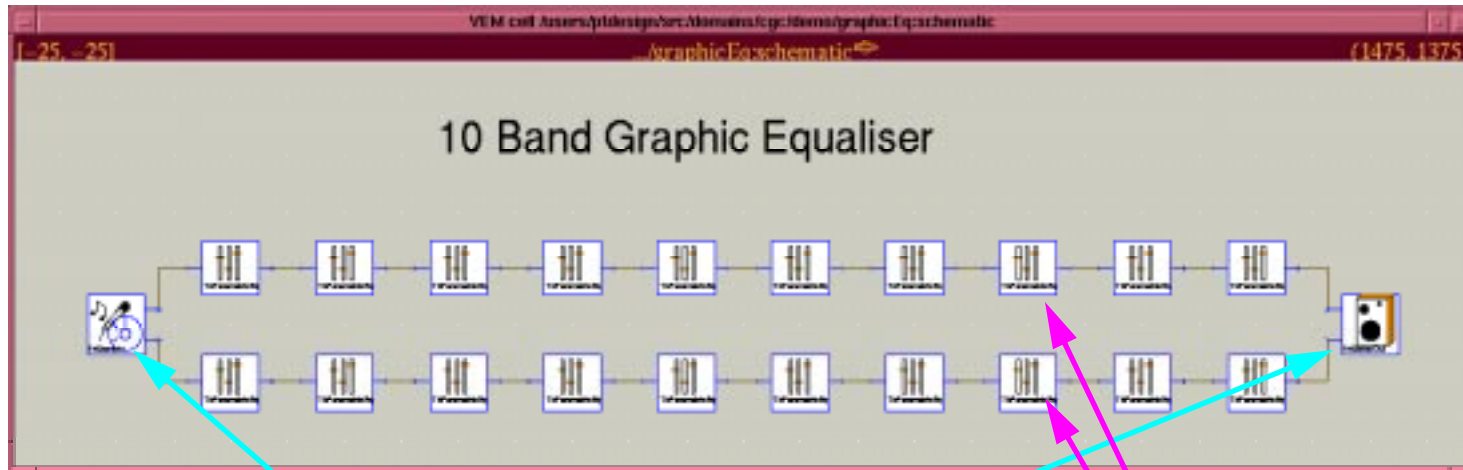
*Realistic comparison*

Kernels	VIS/Float	VIS/Integer
FIR	3.43	6.33
FFT	1.28	N/A
Biquad	0.99	2.76

Recursive algorithm prevents parallelism

Marginal improvement

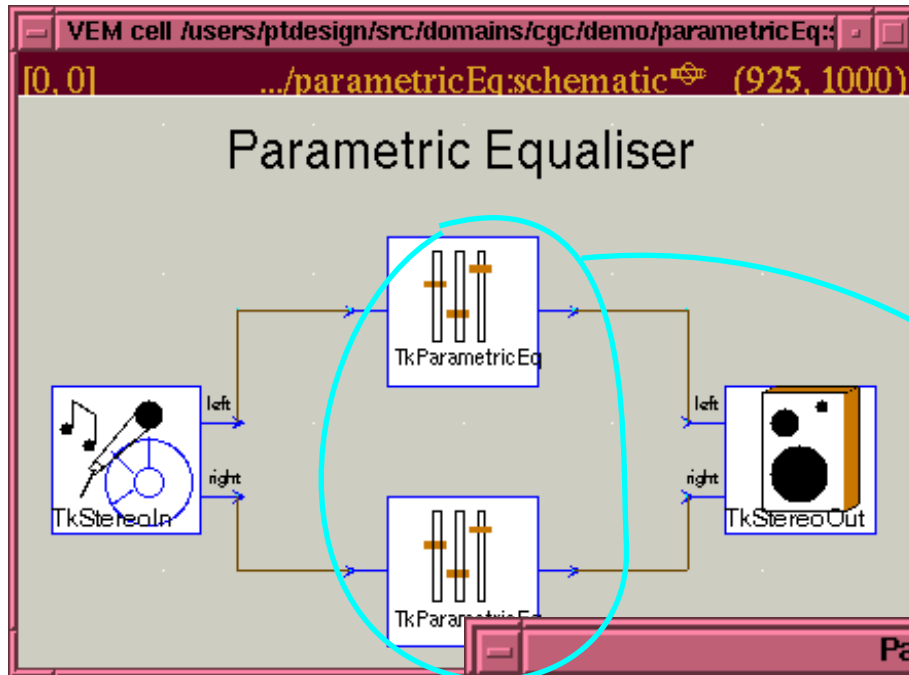
# Graphic equalizer in Ptolemy



The control window, titled "Graphic Equalizer", displays a list of source and destination options on the left and ten frequency sliders in the center. The sliders are labeled with frequencies: Rec, Bal, Play, 31, 63, 125, 250, 500, 1k, 2k, 4k, 8k, and 16k. The 4k slider is circled in magenta. On the right side, there are buttons for Go, Pause, Stop, Quit, and Help.

Source:	1.3	0.0	7.0	7	0	-2	0	0	-2	3	-1	0	3
Line In													
Microphone													
CD													
Destination:	Rec	Bal	Play	31	63	125	250	500	1k	2k	4k	8k	16k
Line Out													
Speaker													

# Parametric equalizer in Ptolemy



Star parameters bound to custom control panel by name

The control panel, titled "Parametric Equalizer", includes a "Source:" section with radio buttons for "Line In", "Microphone", and "CD". Below this is a "Destination:" section with radio buttons for "Line Out" and "Speaker". The main control area contains several sliders and a numeric field:

- Three sliders labeled "Rec", "Bal", and "Play" with values 5.0, 0.0, and 5.0 respectively.
- A numeric field with values 10000 and 100.
- Three sliders labeled "Freq", "BW", and "Gain" with values 4800, 2.0, and 8 respectively.
- A vertical stack of buttons: "Go", "Pause", "Stop", "Quit", and "Help".

A red circle highlights the "Freq", "BW", and "Gain" sliders, with a line pointing to the schematic above.

# Closed-form biquadratic filter

## Analog Bandpass Filter

$$H(s) = \frac{s^2 + \frac{s\omega_C}{Q_Z} + \omega_C^2}{s^2 + \frac{s\omega_C}{Q_P} + \omega_C^2}$$

$$Q_P = \frac{\sqrt{k}\omega_C\omega_1}{\omega_C^2 - \omega_1^2}$$

$$Q_Z = \frac{Q_P}{k}$$

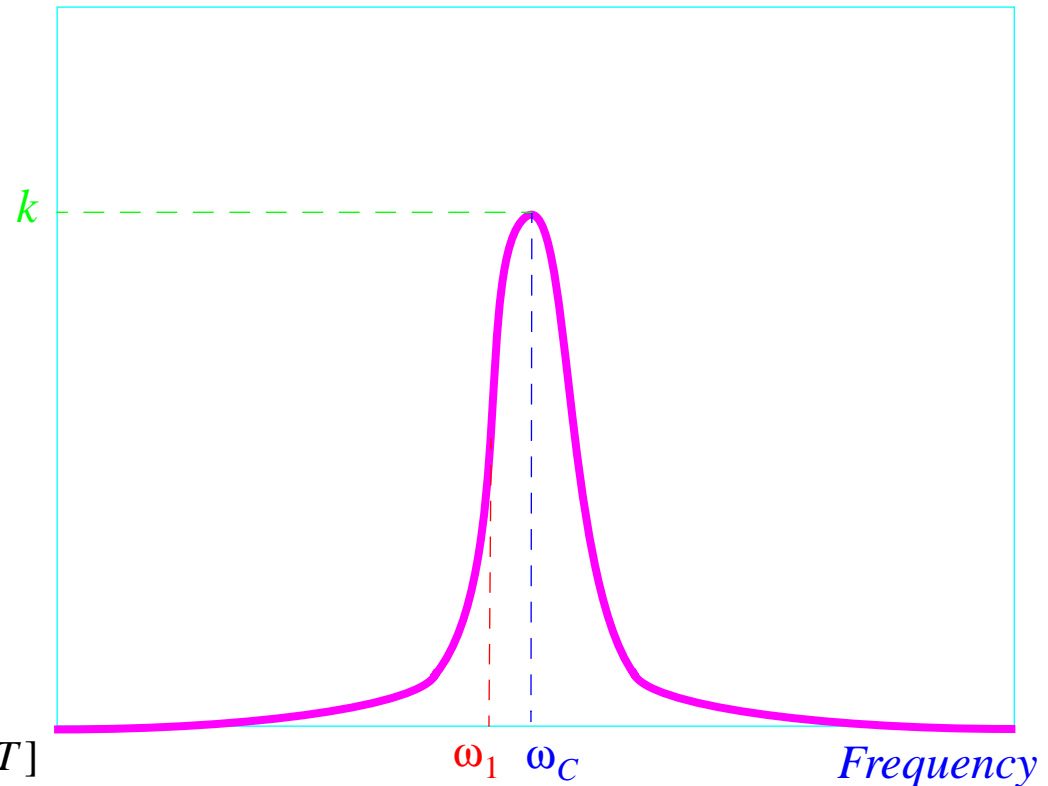
## Digital Bandpass Filter

$$a_2 = Q_P[Q_Z(4 + \omega_C^2 T^2) + 2\omega_C T]$$

$$a_1 = \frac{Q_P Q_Z (-4 + \omega_C^2 T^2)}{a_2}$$

...

Gain



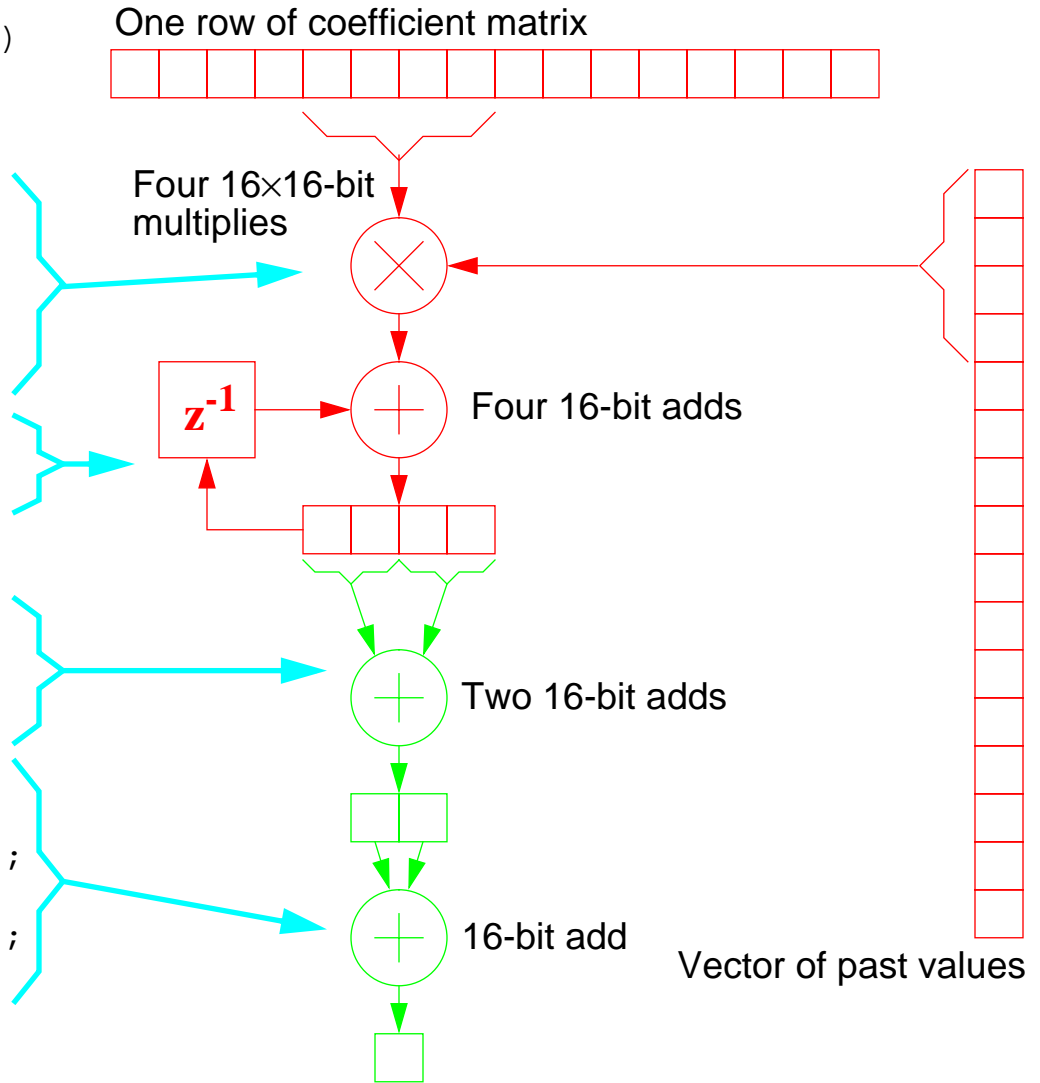
# FIR filter code

```

for (outerloop = 0;
    outerloop < n; outerloop++)
{
    data = src[nminusk];
    tapvalue = *tapptr0++;
    pairhi = vis_fmulsux16(data,
                          tapvalue);
    pairlo = vis_fmulsulx16(data,
                           tapvalue);
    pair = vis_fpadd16(pairhi,
                      pairlo);
    accum0 = vis_fpadd16(accum0,
                        pair);
    ...
}

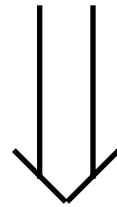
splithi = vis_read_hi(accum0);
splitlo = vis_read_lo(accum0);
split = vis_fpadd16s(splithi,
                    splitlo);
accum0u = *((vis_u32*) &split);
splithihi = (short)
            ((accum0u >> 16));
splitlolo = (short)
            (accum0u & 0xffff);
y11 = splithihi + splitlolo;
...

```



## FIR filter formulation

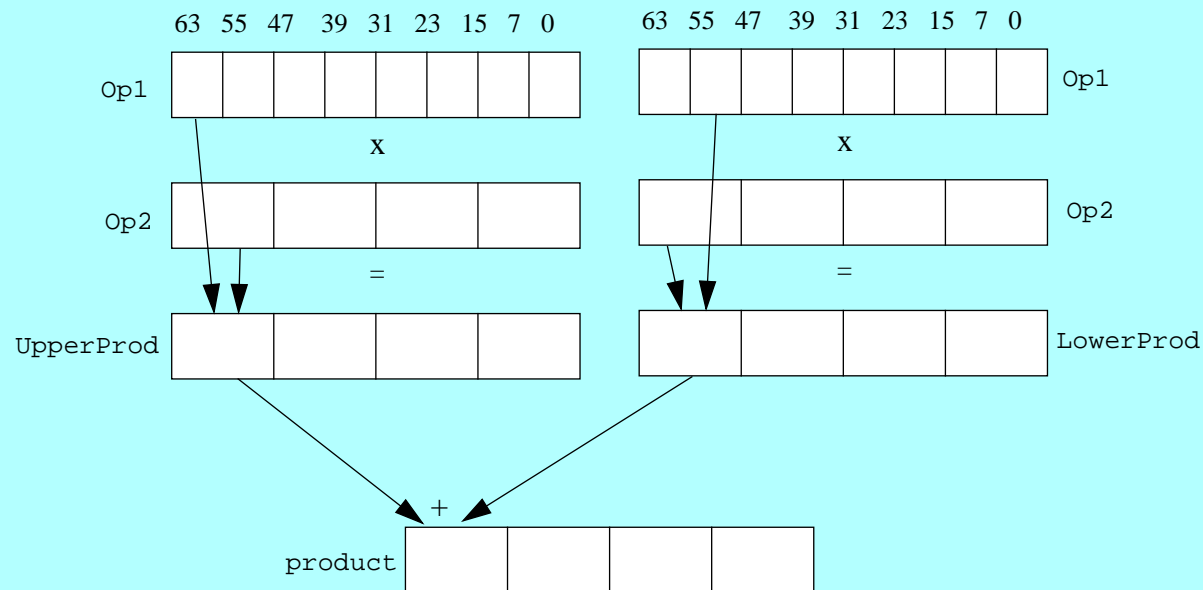
$$y[n] = \sum_{k=0}^n h[k]x[n-k]$$



$$\begin{bmatrix} y[11] \\ y[10] \\ y[9] \\ y[8] \end{bmatrix} = \begin{bmatrix} h[0] & h[1] & h[2] & h[3] & h[4] & h[5] & h[6] & h[7] & h[8] & h[9] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & h[0] & h[1] & h[2] & h[3] & h[4] & h[5] & h[6] & h[7] & h[8] & h[9] & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h[0] & h[1] & h[2] & h[3] & h[4] & h[5] & h[6] & h[7] & h[8] & h[9] & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & h[0] & h[1] & h[2] & h[3] & h[4] & h[5] & h[6] & h[7] & h[8] & h[9] & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x[11] \\ x[10] \\ x[9] \\ x[8] \\ x[7] \\ x[6] \\ x[5] \\ x[4] \\ x[3] \\ x[2] \\ x[1] \\ x[0] \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

# Parallel multiply, 4 x 16-bit

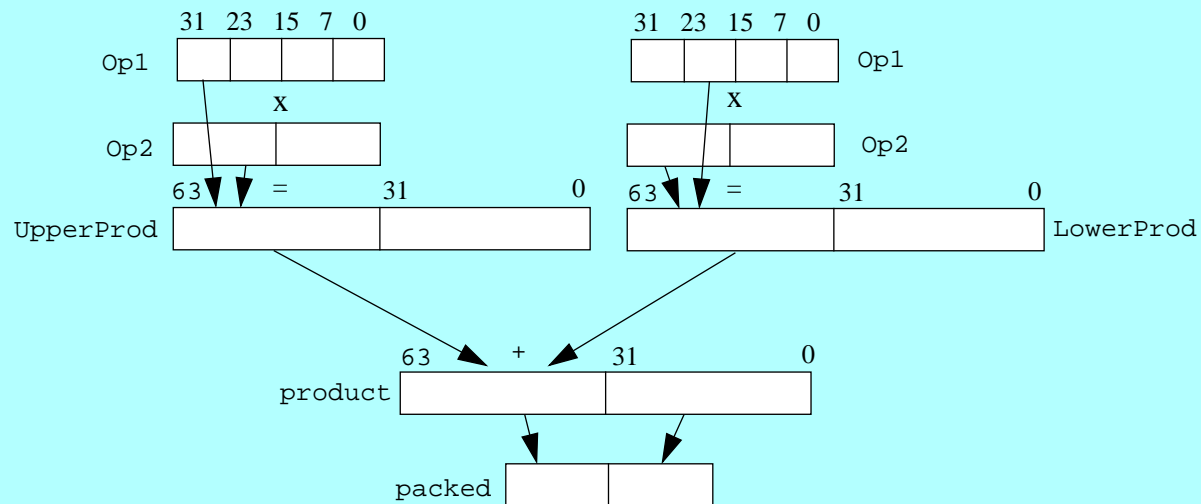
```
vis_d64 Op1, Op2, product;  
vis_d64 UpperProd, LowerProd;  
LowerProd = vis_fpmul8ul16(Op1,Op2);  
UpperProd = vis_fpmul8sux16(Op1,Op2);  
product = vis_fpadd16(LowerProd,UpperProd);
```





## Parallel multiply: 2 x 16-bit

```
vis_f32 Op1, Op2, packed;  
vis_d64 UpperProd, LowerProd, product;  
vis_write_gsr(1<<3);  
LowerProd = vis_fpmuld8ull16(Op1,Op2);  
UpperProd = vis_fpmuld8sux16(Op1,Op2);  
product = vis_fpadd32(LowerProd,UpperProd);  
packed = vis_fpack16(product);
```

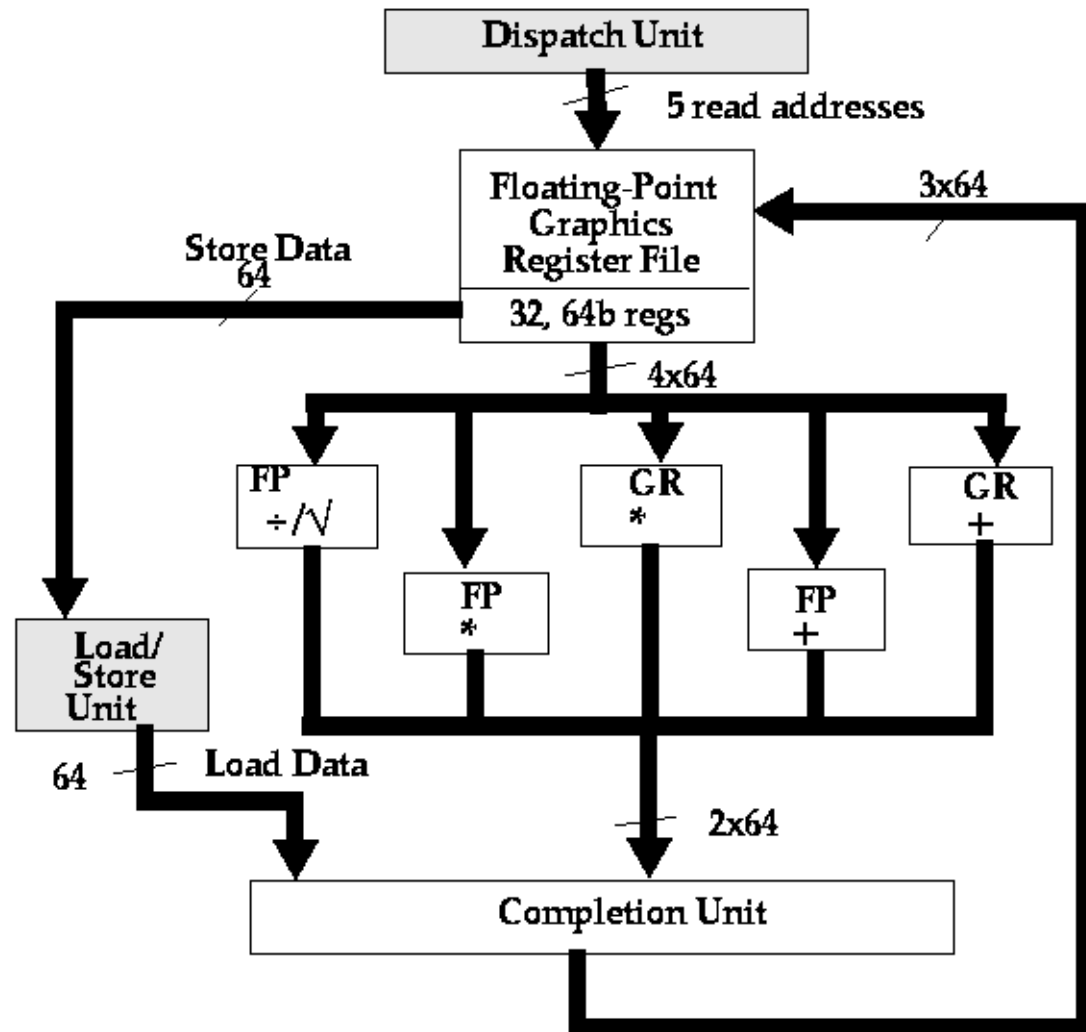


## VIS instructions

- *VIS Utility Inlines*
  - set gsr: `vis_write_gsr()`, `vis_read_gsr()`
  - manipulate vis\_d64 register: `vis_read_hi()`, `vis_read_lo()`, `vis_write_hi()`, `vis_write_lo()`
- *VIS Arithmetic*
  - add/subtract: `vis_fpadd16()`, `vis_fpsub16`
  - multiply: `vis_fmul8sux16()`, `vis_fmul8ulx16()`
- *VIS Data Formatting*
  - load/store 16 bit data: `vis_st_u16()`, `vis_ld_u16`
  - memory alignment: `vis_faligndata()`

## VIS architecture

- Four-instruction issue
- Two integer units, three FP units, two “graphics” units



## Issues

- **Other algorithms**

# Native Signal Processing on the UltraSparc in the Ptolemy Environment



**William Chen**

**John Reekie**

**Sunil Bhawe**

**Edward A. Lee**

**UC Berkeley  
Dept. of EECS**