

Linear Analysis and Optimization of Stream Programs

Andrew A. Lamb

William Thies

Saman Amarasinghe

The New Laboratory for Computer Science and Artificial Intelligence
Massachusetts Institute of Technology

Streaming Application Domain

- Based on audio, video, or data stream
- Increasingly prevalent and important
 - Embedded systems
 - Cell phones, handheld computers
 - Desktop applications
 - Streaming media
 - Software radio
 - Real-time encryption
 - Graphics packages
 - High-performance servers
 - Software routers (Example: Click)
 - Cell phone base stations
 - HDTV editing consoles

Properties of Stream Programs

- A large (possibly infinite) amount of data
 - Limited lifetime of each data item
 - Little processing of each data item
- Computation: apply multiple filters to data
 - Each filter takes an input stream, does some processing, and produces an output stream
 - Filters are independent and self-contained
- A regular, static computation pattern
 - Filter graph is relatively constant
 - A lot of opportunities for compiler optimizations

The StreamIt Language

■ Goals:

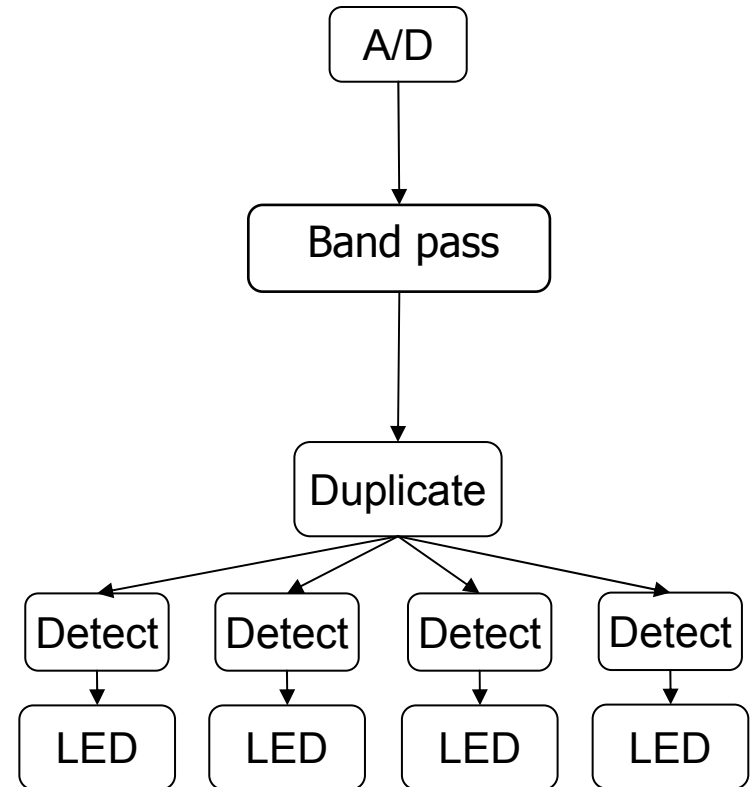
- Provide a High-Level Programming Paradigm
- Improve Programmer Productivity
- Match Performance of Hand-Hacked Assembly

■ Contributions

- Language Design, Structured Streams, Buffer Management (*CC 2002*)
- Exploiting Wire-Exposed Architectures (*ASPLOS 2002*)
- Scheduling of Static Dataflow Graphs (*LCTES 2003*)
- Domain Specific Optimizations (*PLDI 2003*)

Example: Freq band detection

- Used in...
 - metal detector
 - garage door opener
 - spectrum analyzer



Source:

Application Report SPRA414
Texas Instruments, 1999

Freq band detection in StreamIt

```
void->void pipeline FrequencyBand {  
    float sFreq = 4000;  
    float cFreq = 500/(sFreq*2*pi);  
    float wFreq = 100/(sFreq*2*pi);
```

```
    add D2ASource(sFreq);
```

```
    add BandPassFilter(1, cFreq-wFreq,  
                      cFreq+wFreq, 100);
```

```
    add splitjoin {
```

```
        split duplicate;
```

```
        for (int i=0; i<4; i++) {
```

```
            add Detector(i/4);
```

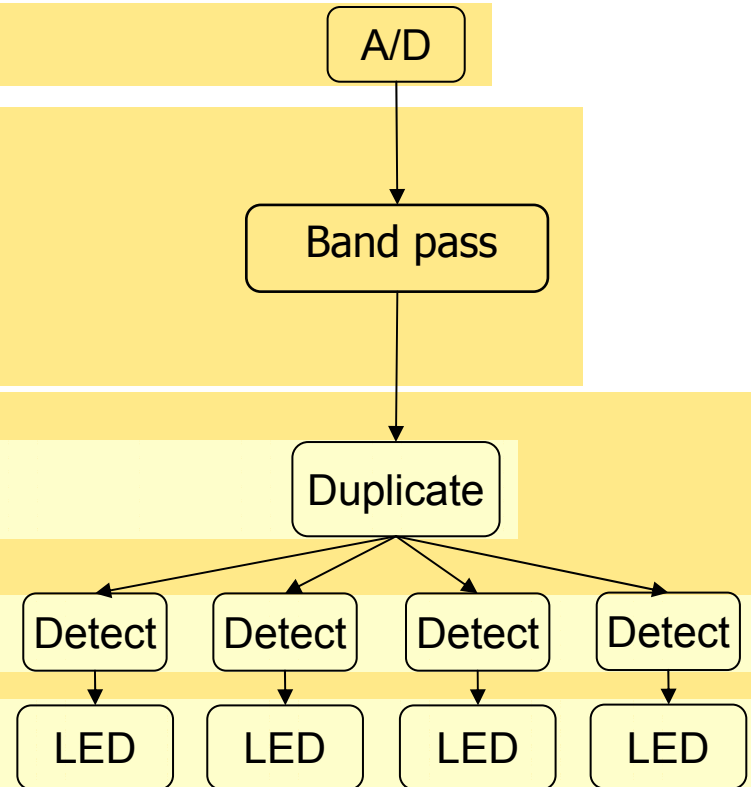
```
            add LEDOutput(i);
```

```
        }
```

```
        join roundrobin(0);
```

```
    }
```

```
}
```



Freq band detection in StreamIt

```
void->void pipeline FrequencyBand {  
    float sFreq = 4000;  
    float cFreq = 500/(sFreq*2*pi);  
    float wFreq = 100/(sFreq*2*pi);
```

```
    add D2ASource(sFreq);
```

```
    float->float pipeline BandPassFilter(float gain, float ws,  
                                          float wp, int num) {  
        add LowPassFilter(1, wp, num);  
        add HighPassFilter(gain, ws, num);  
    }
```

```
    add splitjoin {
```

```
        split duplicate;
```

```
        for (int i=0; i<4; i++) {
```

```
            add Detector(i/4);
```

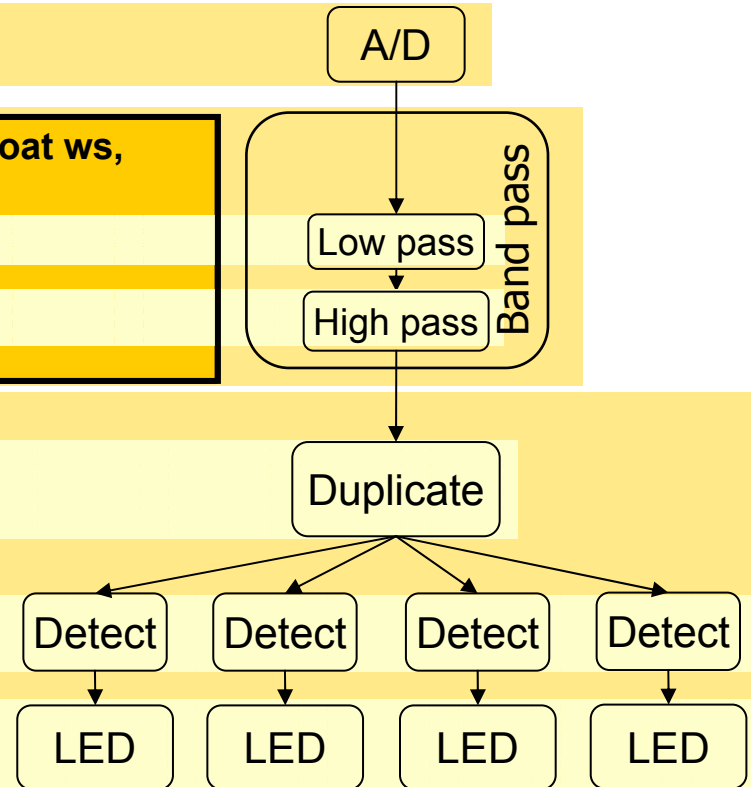
```
            add LEDOutput(i);
```

```
        }
```

```
        join roundrobin(0);
```

```
    }
```

```
}
```



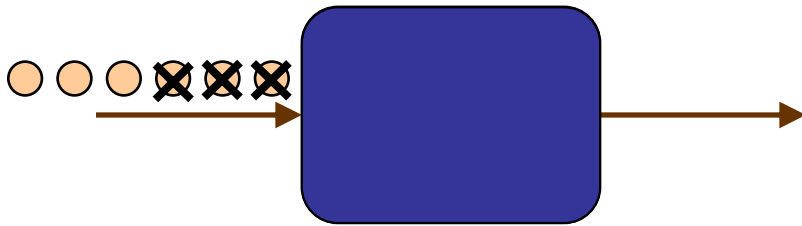
Freq band detection in StreamIt

```
add LowPassFilter(1, wp, num);
```

```
float->float filter LowPassFilter(float g,  
                                     float cFreq, int N)  
{  
  
    float[N] h;  
  
    init {  
        int OFF = N/2;  
        for (int i=0; i<N; i++) {  
            h[i] = g*sin(...);  
        }  
    }  
  
    work peek N pop 1 push 1 {  
        float sum = 0;  
        for (int i=0; i<N; i++) {  
            sum += h[i]*peek(i);  
        }  
        push(sum);  
        pop();  
    }  
}
```


Freq band detection in StreamIt

add LowPassFilter(1, wp, num);



```
float->float filter LowPassFilter(float g,  
                                   float cFreq, int N)
```

```
{
```

```
    float[N] h;
```

```
    init {
```

```
        int OFF = N/2;
```

```
        for (int i=0; i<N; i++) {
```

```
            h[i] = g*sin(...);
```

```
        }
```

```
    }
```

```
    work peek N pop 1 push 1 {
```

```
        float sum = 0;
```

```
        for (int i=0; i<N; i++) {
```

```
            sum += h[i]*peek(i);
```

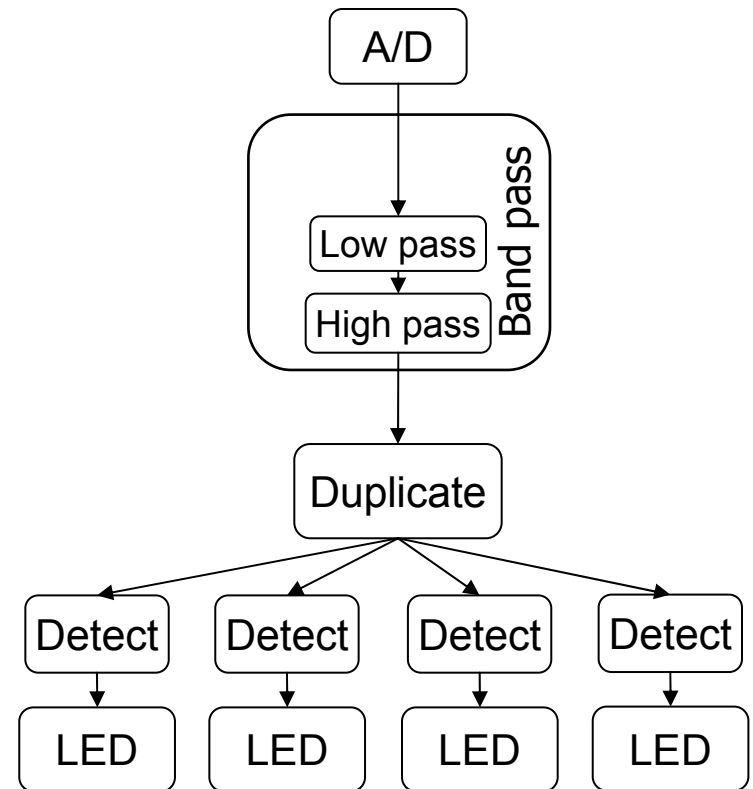
```
        }
```

```
        push(sum);
```

```
        pop();
```

```
    }
```

Freq band detection on a TI DSP



DSP Implementation

```
*****
; File Name: FIR0.ASM
; Originator: Digital control systems Apps group - Houston
; Target Sys: 'C24x Evaluation Board
;
; Description: FIR bandpass filter which detects the presence of a
; 500Hz signal. If the tone is detected an LED is
; lit by using the output port. Sampling Frequency
; forced to be 4kHz.
;
; Last Update: 9 June 1997
;
;-----
.include f240regs.h
;-----
; I/O Mapped EVM Registers
;-----
DAC0 .set 0000h ;Input data register for DAC0
DAC1 .set 0001h ;Input data register for DAC1
DAC2 .set 0002h ;Input data register for DAC2
DAC3 .set 0003h ;Input data register for DAC3
DACUPDATE .set 0004h ;DAC Update Register
;-----
; Variable Declarations for B2
;-----
.bss GPR0,1 ;General Purpose Register
.bss DAC0VAL,1 ;DAC0 Channel Value
.bss DAC1VAL,1 ;DAC1 Channel Value
.bss DAC2VAL,1 ;DAC2 Channel Value
.bss DAC3VAL,1 ;DAC3 Channel Value
;-----
; Vector address declarations
;-----
.sect ".vectors"
RSVECT B START ; Reset Vector
INT1 B PHANTOM ; Interrupt Level 1
INT2 B FIR_ISR ; Interrupt Level 2
INT3 B PHANTOM ; Interrupt Level 3
INT4 B PHANTOM ; Interrupt Level 4
INT5 B PHANTOM ; Interrupt Level 5
INT6 B PHANTOM ; Interrupt Level 6
RESERVED B PHANTOM ; Reserved
SW_INT8 B PHANTOM ; User S/W Interrupt
SW_INT9 B PHANTOM ; User S/W Interrupt
SW_INT10 B PHANTOM ; User S/W Interrupt
SW_INT11 B PHANTOM ; User S/W Interrupt
SW_INT12 B PHANTOM ; User S/W Interrupt
SW_INT13 B PHANTOM ; User S/W Interrupt
SW_INT14 B PHANTOM ; User S/W Interrupt
SW_INT15 B PHANTOM ; User S/W Interrupt
SW_INT16 B PHANTOM ; User S/W Interrupt
TRAP B PHANTOM ; Trap vector
NMINT B PHANTOM ; Non-maskable Interrupt
EMU_TRAP B PHANTOM ; Emulator Trap
SW_INT20 B PHANTOM ; User S/W Interrupt
SW_INT21 B PHANTOM ; User S/W Interrupt
SW_INT22 B PHANTOM ; User S/W Interrupt
SW_INT23 B PHANTOM ; User S/W Interrupt
```

```
*****
; M A I N C O D E - starts here
;-----
.text
NOP
START: SETC INTM ;Disable interrupts
SPLK #0002h,IMR ;Mask all core interrupts
; except INT2
LACC IFR ;Read Interrupt flags
SACL IFR ;Clear all interrupt flags
CLRC EXM ;Clear Sign Extension Mode
CLRC OVM ;Reset Overflow Mode
CLRC CNF ;Config Block B0 to Data mem
;-----
; Set up PLL Module
;-----
LDP #00E0h ;DP = 224; Address for
;7000h - 707fh
;The following line is necessary if a previous program set the PLL
;to a different setting than the settings which the application
;uses. By disabling the PLL, the CKCR1 register can be modified
;so that the PLL can run at the new settings when it is re-enabled.
SPLK #0000000001000001b,CKCR0 ;CLKRMD=PLL Disable
;SYSCLK=CPUCLK/2
; 5432109876543210
SPLK #0000000001011011b,CKCR1
;CLKIN(OSC)=10MHz,CPUCLK=20MHz
;CKCR1 - Clock Control Register 1
;Bits 7-4 (1011)CKINF(3)-CKINF(0) - Crystal or Clock-In Frequency
; Frequency = 10MHz
;Bit 3 (1) PLLDIV(2) - PLL divide by 2 bit
; Divide PLL input by 2
;Bits 2-0 (011) PLLFB(2)-PLLFB(0) - PLL Status. READ ONLY
; PLL Multiplication Ratio = 4
; 5432109876543210
SPLK #00000000011000011b,CKR0
;CLKRMD=PLL Enable, SYSCLK=CPUCLK/2
;CKR0 - Clock Control Register 0
;Bits 7-6 (11) CLKMD(1),CLKMD(0) - Operational mode of Clock
; Module
; PLL Enabled; Run on CLKIN on exiting low power mode
;Bits 5-4 (00) PLOCK(1),PLOCK(0) - PLL Status. READ ONLY
;Bits 3-2 (00) PLLPM(1),PLLPM(0) - Low Power Mode
; LPM0
;Bit 1 (0) ACLKENA - 1MHz ACLK Enable
; ACLK Enabled
;Bit 0 (1) PLLPS - System Clock Prescale Value
; f(sysclk)=f(cpuclk)/2
; 5432109876543210
SPLK #01000000011000000b,SYSCR ;CLKOUT=CPUCLK
;SYSCR - System Control Register
;Bit 15-14 (01) RESET1,RESET0 - Software Reset Bits
; No Action
;Bits 13-8 (000000) Reserved
;Bit 7-6 (11) CLKSRC1,CLKSRC0 - CLKOUT-Pin Source Select
; CPUCLK: CPU clock output mode
;Bit 5-0 (000000) Reserved
SPLK #006fh,WDCR ;Disable WD if VCCP=5V (JP5 in pos. 2-3)
KICK_DOG ;Reset Watchdog
```

```
*****
;-----
; Event Manager Module Reset
;
;This section resets all of the Event Manager Module Registers.
;This is necessary for silicon revision 1.1; however, for
;silicon revisions 2.0 and later, this is not necessary
;
;-----
LDP #232 ;DP=232 Data Page for the Event
;Manager
SPLK #0000h,GPTCON ;Clear General Purpose Timer Control
SPLK #0000h,T1CON ;Clear GP Timer 1 Control
SPLK #0000h,T2CON ;Clear GP Timer 2 Control
SPLK #0000h,T3CON ;Clear GP Timer 3 Control
SPLK #0000h,CMCON ;Clear Compare Control
SPLK #0000h,ACTR ;Clear Full Compare Action Control
;Register
SPLK #0000h,SACTR ;Clear Simple Compare Action Control
;Register
SPLK #0000h,DBTCON ;Clear Dead-Band Timer Control
;Register
SPLK #0FFFfh,EVIFRA ;Clear Interrupt Flag Register A
SPLK #0FFFfh,EVIFRB ;Clear Interrupt Flag Register B
SPLK #0FFFfh,EVIFRC ;Clear Interrupt Flag Register C
SPLK #0000h,CAPCON ;Clear Capture Control
SPLK #0000h,EVIMRA ;Clear Event Manager Mask Register A
SPLK #0000h,EVIMRB ;Clear Event Manager Mask Register B
SPLK #0000h,EVIMRC ;Clear Event Manager Mask Register C
;-----
; End of RESET section for silicon revision 1.1 *
;-----
; Set up Event Manager Module
;-----
T1COMPARE .set 2500
T1PERIOD .set 5000 ;Sets up period for 4kHz frequency
LDP #232 ;DP=232, Data Page for Event Manager
Addresses
SPLK #T1COMPARE,T1CMFPR;Compare value for 50% duty cycle
; 2109876543210
SPLK #0000001010101b,GPTCON
;GPTCON = GP Timer Control Register
;Bit 15 (0) T3STAT - GP Timer 3 Status. READ ONLY
;Bit 14 (0) T2STAT - GP Timer 2 Status. READ ONLY
;Bit 13 (0) T1STAT - GP Timer 1 Status. READ ONLY
;Bits 12-11 (00) T3TADC - ADC start by event of GP Timer 3
; No event starts ADC
;Bits 10-9 (00) T2TADC - ADC start by event of GP Timer 2
; No event starts ADC
;Bits 8-7 (00) T1TADC - ADC start by event of GP Timer 1
; No event starts ADC
;Bit 6 (1) TCMPOE - Compare output enable
; Enable all three GP timer compare outputs
;Bits 5-4 (01) T3PIN - Polarity of GP Timer 3 compare output
; Active Low
;Bits 3-2 (01) T2PIN - Polarity of GP Timer 2 compare output
; Active Low
;Bits 1-0 (01) T1PIN - Polarity of GP Timer 1 compare output
; Active Low
SPLK #T1PERIOD,T1PR ;Period value for 2kHz signal
SPLK #0000h,T1CNT ;Clear GP Timer 1 Counter
SPLK #0000h,T2CNT ;Clear GP Timer 2 Counter
SPLK #0000h,T3CNT ;Clear GP Timer 3 Counter
; 5432109876543210
SPLK #0001000000000010b,T1CON
```

Source: Application Report SPRA414, Texas Instruments, 1999

```

TTCCON = GP Timer 1 Control Register
;Bits 15-14(000) FREE,SOFT = Emulation Control Bits
; Stop immediately on emulation suspend
;Bits 13-11(010) TMODE2-TMODE0 = Count Mode Selection
; Continuous-Up Count Mode
;Bits 10-8 (000) TFS2-TFS0 = Input Clock Prescaler
; Divide by 1
;Bit 7 (0) Reserved
;Bit 6 (0) TENABLE = Timer Enable
; Disable timer operations
;Bits 5-4 (000) TCLKS1,TCLKS0 = Clock Source Select
; Internal Clock Source
;Bits 3-2 (000) TCLD1,TCLD0 = Timer Compare Register Reload
; Condition
; When counter is 0
;Bit 1 (1) TECMPR = Timer compare enable
; Enable timer compare operation
;Bit 0 (0) Reserved
; 5432109876543210
SPLK "0000000000000000000b,TTCCON
GP Timer 2 = Not Used
;TTCCON = GP Timer 2 Control Register
;Bits 15-14(000) FREE,SOFT = Emulation Control Bits
; Stop immediately on emulation suspend
;Bits 13-11(000) TMODE2-TMODE0 = Count Mode Selection
; Stop/Hold
;Bits 10-8 (000) TFS2-TFS0 = Input Clock Prescaler
; Divide by 1
;Bit 7 (0) TSWT1 = GP Timer 1 timer enable bit
; Use own TENABLE bit
;Bit 6 (0) TENABLE = Timer Enable
; Disable timer operations
;Bits 5-4 (000) TCLKS1,TCLKS0 = Clock Source Select
; Internal Clock Source
;Bits 3-2 (000) TCLD1,TCLD0 = Timer Compare Register Reload
; Condition
; When counter is 0
;Bit 1 (0) TECMPR = Timer compare enable
; Disable timer compare operation
;Bit 0 (0) SELTIPR = Period Register select
; Use own period register
; 5432109876543210
SPLK "0000000000000000000b,TTCCON
GP Timer 3 = Not Used
;TTCCON = GP Timer 3 Control Register
;Bits 15-14(000) FREE,SOFT = Emulation Control Bits
; Stop immediately on emulation suspend
;Bits 13-11(000) TMODE2-TMODE0 = Count Mode Selection
; Stop/Hold
;Bits 10-8 (000) TFS2-TFS0 = Input Clock Prescaler
; Divide by 1
;Bit 7 (0) TSWT1 = GP Timer 1 timer enable bit
; Use own TENABLE bit
;Bit 6 (0) TENABLE = Timer Enable
; Disable timer operations
;Bits 5-4 (000) TCLKS1,TCLKS0 = Clock Source Select
; Internal Clock Source
;Bits 3-2 (000) TCLD1,TCLD0 = Timer Compare Register Reload
; Condition
; When counter is 0
;Bit 1 (0) TECMPR = Timer compare enable
; Disable timer compare operation
;Bit 0 (0) SELTIPR = Period Register select
; Use own period register

```

```

; Set up Digital I/O Port
;-----
LDP #225 ;DP=225, Data Page to Configure OCRA
; 5432109876543210
SPLK #0011000000001111b,OCRA
;OCRA - Output Control Register A
;Bit 15 (0) CRA.15 - IOPE7
;Bit 14 (0) CRA.14 - IOPE6
;Bit 13 (1) CRA.13 - T2PWM/T2CMP
;Bit 12 (1) CRA.12 - T2PWM/T2CMP
;Bit 11 (1) CRA.11 - T1PWM/T1CMP
;Bit 10 (0) CRA.10 - IOPE2
;Bit 9 (0) CRA.9 - IOPE1
;Bit 8 (0) CRA.8 - IOPE0
;Bits 7-4 (0000)Reserved
;Bit 3 (1) CRA.3 - ADCIN3
;Bit 2 (1) CRA.2 - ADCIN9
;Bit 1 (1) CRA.1 - ADCIN1
;Bit 0 (1) CRA.0 - ADCIN0
;-----
; Set up ADC Module
;-----
LDP #224
; 5432109876543210
SPLK #1000100100000000b,ADCTRL1
;ADCTRL1 - ADC Control Register 1
;Bit 15 (1) Suspend-SOFT -
; Complete Conversion before halting emulator
;Bit 14 (0) Suspend-FREE -
; Operations is determined by Suspend-SOFT
;Bit 0 (0) ADCIMSTART - ADC start converting immediately
; No Action
;Bit 12 (0) ADCZEN - Enable/Disable ADC2
; Disable ADC2
;Bit 11 (1) ADC1EN - Enable/Disable ADC1
; Enable ADC1
;Bit 10 (0) ADCCONRUN - ADC Continuous Conversion Mode
; Disable Continuous Conversion
;Bit 9 (0) ADCINTEN - Enable ADC Interrupt
; Mask ADC Interrupt
;Bit 8 (1) ADCINTFLAG - ADC Interrupt Flag
; Clear Interrupt Flag Bit
;Bit 7 (0) ADCEOC - End of Conversion Bit READ ONLY
;Bits 6-4 (000) ADCCHSEL = ADC2 Channel Select
; Channel 8
;Bits 3-1 (000) ADC1CHSEL - ADC1 Channel Select
; Channel 0
;Bit 0 (0) ADCSOC - ADC Start of conversion bit
; No Action
; 5432109876543210
SPLK #0000000000000010b,ADCTRL2
;ADCTRL2 - ADC Control Register 2
;Bits 15-11 (00000)Reserved
;Bit 10 (0) ADCDEVSOC - Event Manager SOC mask bit
; Mask ADCDEVSO
;Bit 9 (0) ADCEXTSOC - External SOC mask bit
; Mask ADCEXTSO
;Bit 8 (0) Reserved
;Bits 7-6 (00) ADCFIPO1 - Data Register FFI01 Status READ
ONLY
;Bit 5 (0) Reserved
;Bits 4-3 (00) ADCFIPO2 - Data Register FFI02 Status READ
ONLY
;Bits 2-0 (101) ADCPSCALE - ADC Input Clock Prescaler
; Prescale Value is
; SYSCLK Period = 0.0usec
; 0.0usec x 16 x 6 = 9.6 usec == 6usec

```

[illegible]

Source: Application Report SPRA414, Texas Instruments, 1999

Cont. Cont.

```
;reset MAX values
.bss TEMP,1 ;Variable for temporary storage
;of values
.text
MAIN LAR ARI,#ADCFI01 ; ARI = ADCFI01 address
LAR AR2,#ADCTRL1 ; AR2 = ADCTRL1 address
LAR AR3,#BCOEFF ; AR3 = BCOEFF address
LAR AR5,#LEDS ; AR5 = LEDS Output
LDP #232
LACC EVIFRA ;ACC = Event Module Type A Interrupt
;Flags
SACL EVIFRA ;EVIFRA = ACC; Clears the current
;set flags
SPLK #0080h,EVIMRA ; Enable Timer 1 Period
; Interrupt
MAR *,AR2 ;ARP = AR2
LACC *,ACC = ADCTRL1
ADD #1 ;SET BIT FOR SINGLE CONVERSION
SACL *,0,ARI ;STARTS ADC CONVERSION
SBIT1 TICON,B6_MSK ;Sets Bit 6 of TICON; Starts
;the timer
LDP #0 ;DP = 0; Addresses 0000h - 007Fh
SPLK #0000h,LEDSOUT ;Clear the LEDS
OUT LEDSOUT,LEDS
SPLK #0E38h,THRESHOLD1;Q15 value for 1/9
SPLK #1C71h,THRESHOLD2;Q15 value for 2/9
SPLK #2AAAh,THRESHOLD3;Q15 value for 3/9
SPLK #38E3h,THRESHOLD4;Q15 value for 4/9
SPLK #471Ch,THRESHOLD5;Q15 value for 5/9
SPLK #5555h,THRESHOLD6;Q15 value for 6/9
SPLK #638Eh,THRESHOLD7;Q15 value for 7/9
SPLK #71C7h,THRESHOLD8;Q15 value for 8/9
SPLK #0000h,MAXIN ;Initialize Maximum input
;value
SPLK #0000h,MAXOUT ;Initialize Maximum FIR output
;value
SPLK #WINDOW,RESET_MAX ;Initialize the maximum
;reset counter
CLR INTM ;Enable Interrupts
WAIT B WAIT ;Wait for interrupt
;-----
; INTERRUPT SERVICE ROUTINES FOR FIR FILTER
;-----
FIR_ISR LAR AR4,#XVALUE+100 ;AR4 = DATA ADDRESS
MAR *,AR1 ;ARP = ARI = ADCFI01
LACC *,0,AR4 ;ACC = ADCFI01; ARP =
AR4
LDP #0 ;DP = 0
;Addresses 0000h - 007Fh
SACL DAC1VAL ;DAC1VAL = ADCFI01
RPT #7 ;Shift ADC value 8 places
; - Reduce to 8 bit value
SFR ;larger bit values produced
;large results
SUB #7Fh ;Subtract the equivalent 8 bit
;DC offset
LDP #04h ;DP = 4; Address 0200h - 027Fh
SACL XVALUE ;XVALUE = ADCFI01 / 256;
LACC #0h ;Initialize the ACCUMULATOR
MPY #0h ;Initialize the PROD REG
RPT #100 ;Calculate Y
MACD BCOEFF, *- ;Multiply X with B, and add
APAC ;final accumulation
LDP #0
RPT #7 ;Shift the result 8 places to left
SFL
SACH DACOVAL,1 ;DACOVAL = Y * 2; shift to
;remove extra sign bit
;FIR result to output
```

```
;Multiply the values by 5/4 because the maximum gain is 4/5
LT DACOVAL ;TREG = DACOVAL
MPY #5 ;PREG = DACOVAL * 5
PAC ;ACC = PREG = DACOVAL * 5
SFR ;ACC = DACOVAL * 5 / 2
SFR ;ACC = DACOVAL * 5 / 4
SACL DACOVAL ;DACOVAL = DACOVAL * 5/4
LACC DACOVAL ;ACC = DACOVAL
RPT #3 ;Shift right 4 times
; = 16 bit value to
SFR ;12 bit value because
;DAC is 12bits
ADD #7Ffh ;Add DC offset
AND #0FFFh ;Ensure 12 bits
SACL DACOVAL ;Store value for output on the DAC
LDP #7 ;DP=7; Address for 0380h to 03FFh
SACL VALUEOUT ;Store value to find maximum
;value of the output values
LAR AR6,#(VALUEOUT+127-1) ;AR6 = End of VALUE OUT
;buffer
LAR AR7,#126 ;AR7 = 127 - 1; Number of
;values to move
MAR *,AR6 ;ARP = AR6
SHIFT1 DMOV *- ;AR7 ;Move all of the values in
;the VALUEOUT
BANZ SHIFT1,-,AR6 ;Data Buffer to the next
;higher address
LDP #0 ;DP = 0; Addresses 0000h - 007Fh
LACC DAC1VAL ;ACC = DAC1VAL = Input Value
RPT #3 ;Shift the value to the
;right 4 times
SFR ;Convert the value from 16
;bits to 12 bits
SACL DAC1VAL ;DAC1VAL = 12 bit value for DAC
LDP #6 ;DP = 6; Addresses 0300h - 037Fh
SACL VALUEIN ;VALUEIN = DAC1VAL
LAR AR6,#(VALUEIN+127-1) ;AR6 = End of VALUE IN
;buffer
LAR AR7,#126 ;AR7 = 127 - 1; Number of
;values to move
MAR *,AR6 ;ARP = AR6
SHIFT2 DMOV *- ;AR7 ;Move all of the values in
;the VALUEIN
BANZ SHIFT2,-,AR6 ;Data Buffer to the next
;higher address
;Outputs the FIR results and the original value
;DACD has the FIR results and DAC1 has the original value
LDP #0
OUT DACOVAL,DACO ;DACO = DACOVAL; FIR result on
;DAC channel 0
OUT DAC1VAL,DACL ;DAC1 = DAC1VAL; Input value
;on DAC channel 1
OUT DACOVAL,DACUPDATE
;Update the values on the DAC
;Find the maximum value among VALUEIN and VALUEOUT for the LEDS
LACC RESET_MAX ;ACC = RESET_MAX
; Max Reset Counter
SUB #1 ;Decrement by 1
SACL RESET_MAX ;Store new value for RESET_MAX
BCND NO_RESET,GT ;If not WINDOWth value, don't
;reset counter
SPLK #WINDOW,RESET_MAX
;Else reset the max reset counter
SPLK #0000h,MAXIN ;Reset the MAXIN value
SPLK #0000h,MAXOUT ;Reset the MAXOUT value
NO_RESET LAR AR6,#VALUEIN ;AR6 = VALUEIN; Beginning of
```

```
;Data In Buffer
LAR AR7,#127 ;AR7 = 128 - 1; Counter to find
;max value in
MAR *,AR6 ;ARP = AR6
FIND_MAXIN LACC *- ,0,AR7 ;ACC = Value pointed by AR6
SUB MAXIN ;Subtract MAXIN
BCND RESUME1,LEQ ;If the value results in a
;value less than 0,
;then the value is smaller
;than MAXIN, else the
;value is larger than MAXIN
ADD MAXIN ;ACC = Value pointed by AR6
SACL MAXIN ;Store new MAXIN value
RESUME1 BANZ FIND_MAXIN,-,AR6 ;If smaller than MAXIN,
;decrement loop counter
; (AR7), move to next value in
;buffer
LAR AR7,#127 ;Since VALUEIN buffer is
;adjacent to
;VALUEOUT buffer, only AR7
;needs to be reset
;ARP is already AR6
FIND_MAXOUT LACC *- ,0,AR7 ;ACC = Value pointed by AR6
SUB MAXOUT ;Subtract MAXOUT
BCND RESUME2,LEQ ;If the value results in a
;value less than 0,
;then the value is smaller than
;MAXOUT, els
;the value is larger than
;MAXOUT
ADD MAXOUT ;ACC = Value pointed by AR6
SACL MAXOUT ;Store new MAXOUT value
RESUME2 BANZ FIND_MAXOUT,-,AR6 ;If smaller than MAXOUT,
;decrement loop counter (AR7),
;move to next value in buffer
;The following section determines if the value meets the threshold
;requirement
LDP #0 ;DP = 0; Addresses 0000h to 007Fh
;All variables used are in B2
;Need to remove the DC offset because if the FIR result is 0 it will
;equal 7Ffh which is already 50% of the maximum input value
LACC MAXIN ;ACC = MAXIN
SUB #7Ffh ;Subtract the DC offset
SACL DIFFIN ;DIFFIN = MAXIN - 7Ffh
LACC MAXOUT ;ACC = MAXOUT
SUB #7Ffh ;Subtract the DC offset
SACL DIFFOUT ;DIFFOUT = MAXOUT - 7Ffh
;Check if the output exceeds the middle threshold value, THRESHOLD4
LT DIFFIN,TREG = DIFFIN
TH4 MPY THRESHOLD4 ;PREG = DIFFIN * THRESHOLD4
PAC ;ACC = PREG
SACH TEMP,1 ;TEMP = ACC*2; Shift to remove
;extra sign bit
LACC TEMP ;ACC = TEMP
SUB DIFFOUT ;Subtract DIFFOUT
BCND ABOVE4,LT ;If DIFFOUT is greater than
;TEMP, then the FIR result is
;greater than VALUEIN * THRESHOLD4,
;else, it is below THRESHOLD4 value
;Output is below THRESHOLD4. Check if above THRESHOLD2
BELOW4 LT DIFFIN
TH2 MPY THRESHOLD2
PAC
SACH TEMP,1
LACC TEMP
SUB DIFFOUT
BCND ABOVE2,LT
;Output is below THRESHOLD4 & THRESHOLD2. Check if above THRESHOLD1
BELOW2 LT DIFFIN
TH1 MPY THRESHOLD1
PAC
SACH TEMP,1
LACC TEMP
SUB DIFFOUT
BCND ABOVE1,LT
```

Source: Application Report SPRA414, Texas Instruments, 1999

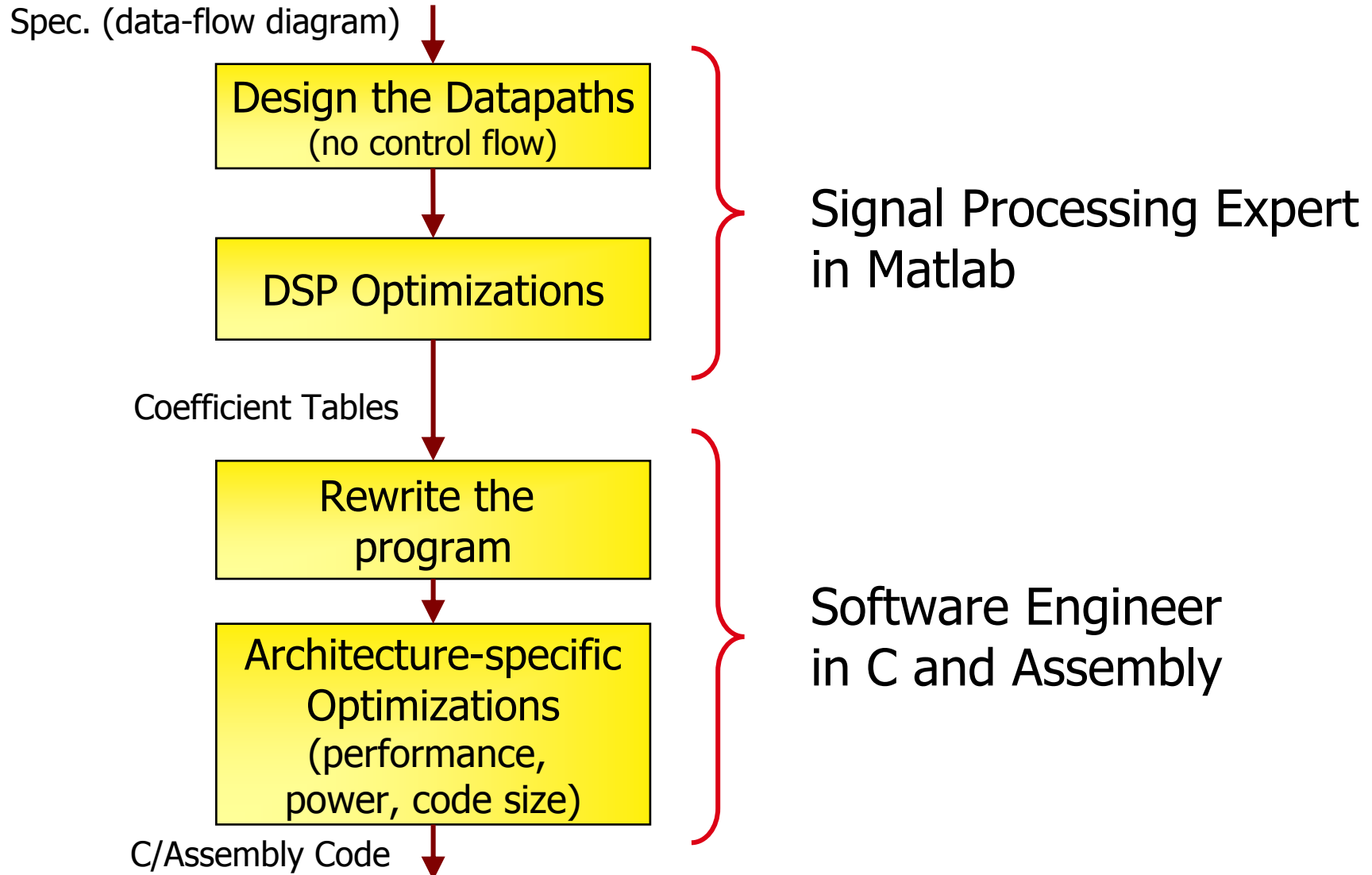
Cont. Cont. Cont.

```
;Output is below THRESHOLD4, THRESHOLD2, & THRESHOLD1. Turn off LEDS
BELOW1 SPLK #0000h,LEDSOUT
B OUTLEDS
;Output is below THRESHOLD4, THRESHOLD2, but above THRESHOLD1. Turn
;on DS1
ABOVE1 SPLK #0001h,LEDSOUT
B OUTLEDS
;Output is below THRESHOLD4, but above THRESHOLD2. Check if above
;THRESHOLD3
ABOVE2 LT DIFFIN
TH3 MPY THRESHOLD3
PAC
SACH TEMP,1
LACC TEMP
SUB DIFFOUT
BCND ABOVE3,LT
;Output is below THRESHOLD4 and THRESHOLD3, but above THRESHOLD2.
;Turn on DS1-DS2
BELOW3 SPLK #0003h,LEDSOUT
B OUTLEDS
;Output is below THRESHOLD4, but above THRESHOLD3 and THRESHOLD2.
;Turn on DS1-DS3
ABOVE3 SPLK #0007h,LEDSOUT
B OUTLEDS
;Output is above THRESHOLD4. Check if above THRESHOLD6
ABOVE4 LT DIFFIN
TH6 MPY THRESHOLD6
PAC
SACH TEMP,1
LACC TEMP
SUB DIFFOUT
BCND ABOVE6,LT
;Output is above THRESHOLD4, but below THRESHOLD6. Check if above
;THRESHOLD5
BELOW6 LT DIFFIN
TH5 MPY THRESHOLD5
PAC
SACH TEMP,1
LACC TEMP
SUB DIFFOUT
BCND ABOVE5,LT
;Output is above THRESHOLD4, but below THRESHOLD6 & THRESHOLD5. Turn
;on DS1-DS4
BELOW5 SPLK #000Fh,LEDSOUT
B OUTLEDS
;Output is above THRESHOLD4 & THRESHOLD5, but below THRESHOLD6.
;Turn on DS1-DS5
ABOVE5 SPLK #001Fh,LEDSOUT
B OUTLEDS
;Output is above THRESHOLD4 & THRESHOLD6. Check if above THRESHOLD8.
ABOVE6 LT DIFFIN
TH8 MPY THRESHOLD8
PAC
SACH TEMP,1
LACC TEMP
SUB DIFFOUT
BCND ABOVE8,LT
;Output is above THRESHOLD4 & THRESHOLD6, but below THRESHOLD8.
;Check if above THRESHOLD7.
BELOW8 LT DIFFIN
TH7 MPY THRESHOLD7
PAC
SACH TEMP,1
LACC TEMP
SUB DIFFOUT
BCND ABOVE7,LT
```

```
;Output is above THRESHOLD4 & THRESHOLD6, but below THRESHOLD8 &
;THRESHOLD7. Turn on DS1-DS6
BELOW7 SPLK #003Fh,LEDSOUT
B OUTLEDS
;Output is above THRESHOLD4, THRESHOLD6, & THRESHOLD7, but below
;THRESHOLD8. Turn on ;DS1-DS7
ABOVE7 SPLK #007Fh,LEDSOUT
B OUTLEDS
;Output is above THRESHOLD4, THRESHOLD6, & THRESHOLD8. Turn on
;DS1-DS8
ABOVE8 SPLK #00FFh,LEDSOUT
OUTLEDS OUT LEDSOUT,LEDS ;Turn on the LEDS
RESTART_ADC MAR *,AR2 ;ARP = AR2
LACC * ;ACC = ADCTRL1
ADD #1h ;Set bit to restart the ADC
SACL * ;Start converting next value
LDP #232
LACC EVIFRA ;Clear the flag register of
;Event Manager
SACL EVIFRA
CLRC INTM ;ENABLE INTERRUPTS
RET ;Return to main line
;=====
; I S R - PHANTOM
;
; Description: Dummy ISR, used to trap spurious interrupts.
;
; Modifies:
;
; Last Update: 16-06-95
;=====
PHANTOM B PHANTOM
```

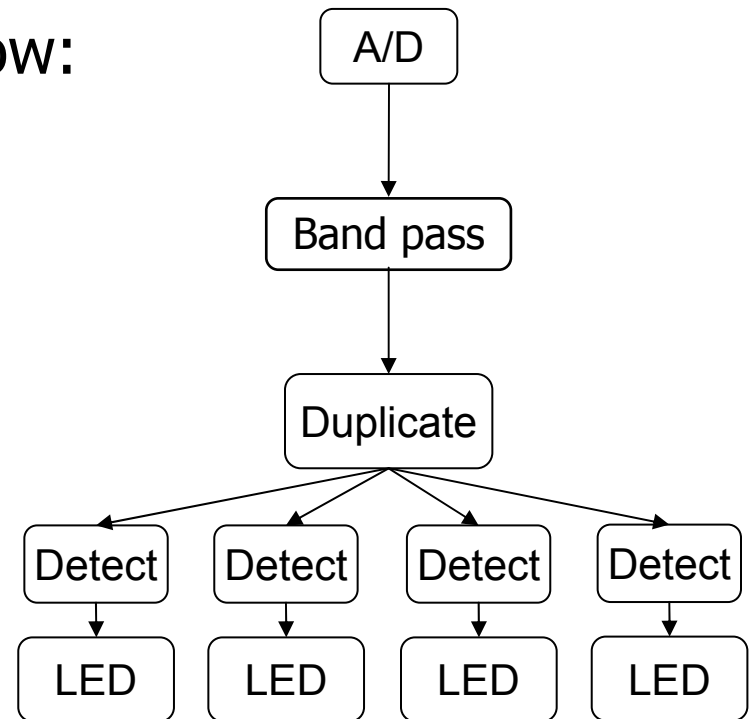
Source: Application Report SPRA414, Texas Instruments, 1999

Conventional DSP Design Flow

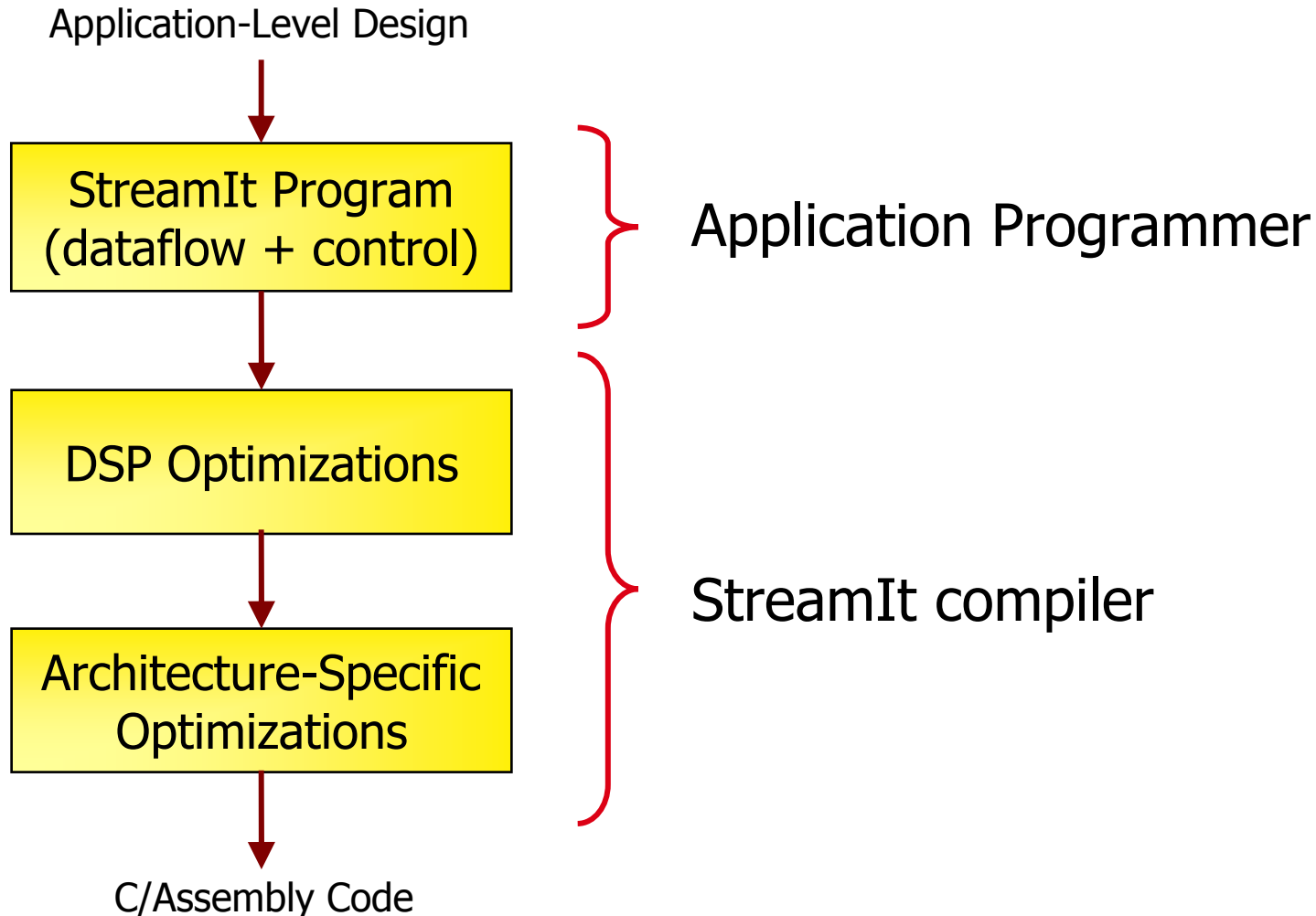


Any Design Modifications?

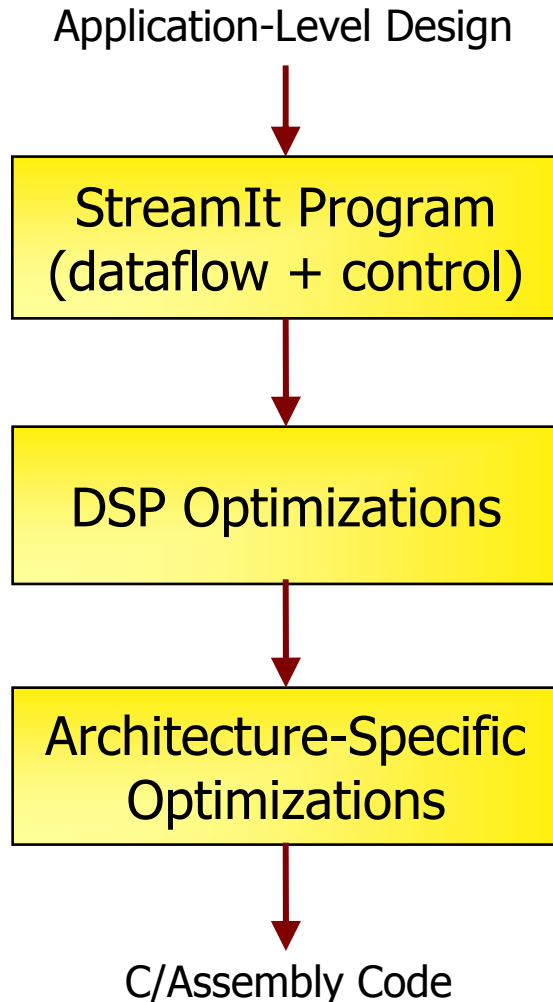
- Center frequency from 500 Hz to 1200 Hz?
 - According to TI,
in the conventional design-flow:
 - Redesign filter in MATLAB
 - Cut-and-paste values to EXCEL
 - Recalculate the coefficients
 - Update assembly
 - If using StreamIt
 - Change one constant
 - Recompile



Design Flow with StreamIt



Design Flow with StreamIt



- Benefits of programming in a single, high-level abstraction
 - Modular
 - Composable
 - Portable
 - Malleable
- The Challenge: Maintaining Performance
 - Replacing Expert DSP Engineer
 - Replacing Expert Assembly Hacker

Our Focus: Linear Filters

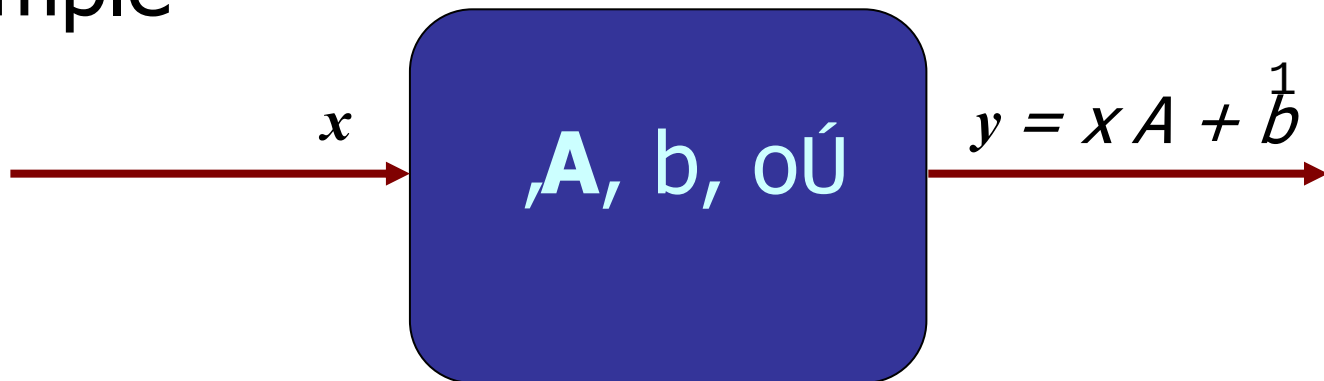
- Most common target of DSP optimizations
 - FIR filters
 - Compressors
 - Expanders
 - DFT/DCT

} Output is weighted sum of inputs
- Example optimizations:
 - Combining Adjacent Nodes
 - Translating to Frequency Domain

Representing Linear Filters

- A linear filter is a tuple $\langle \mathbf{A}, b, o \rangle$
 - $\mathbf{A}$: matrix of coefficients
 - b : vector of constants
 - o : number of items popped

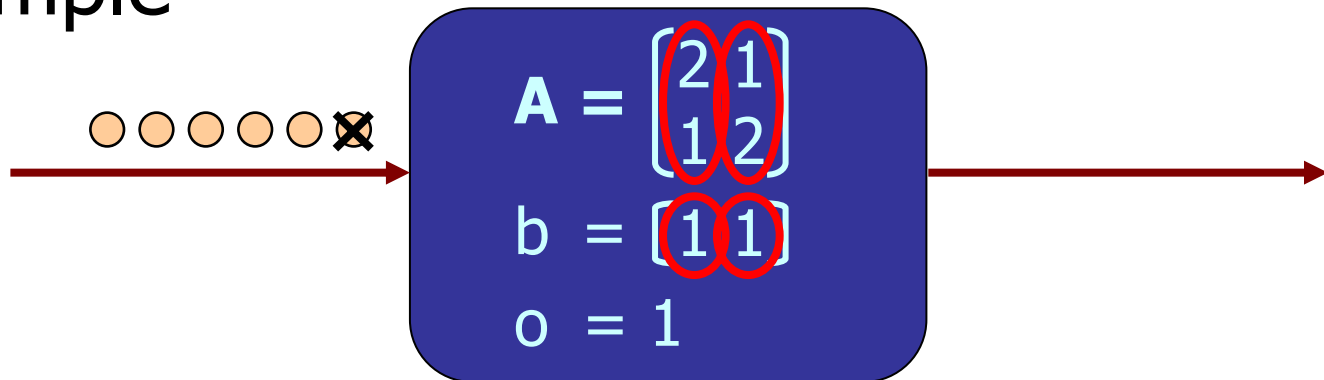
- Example



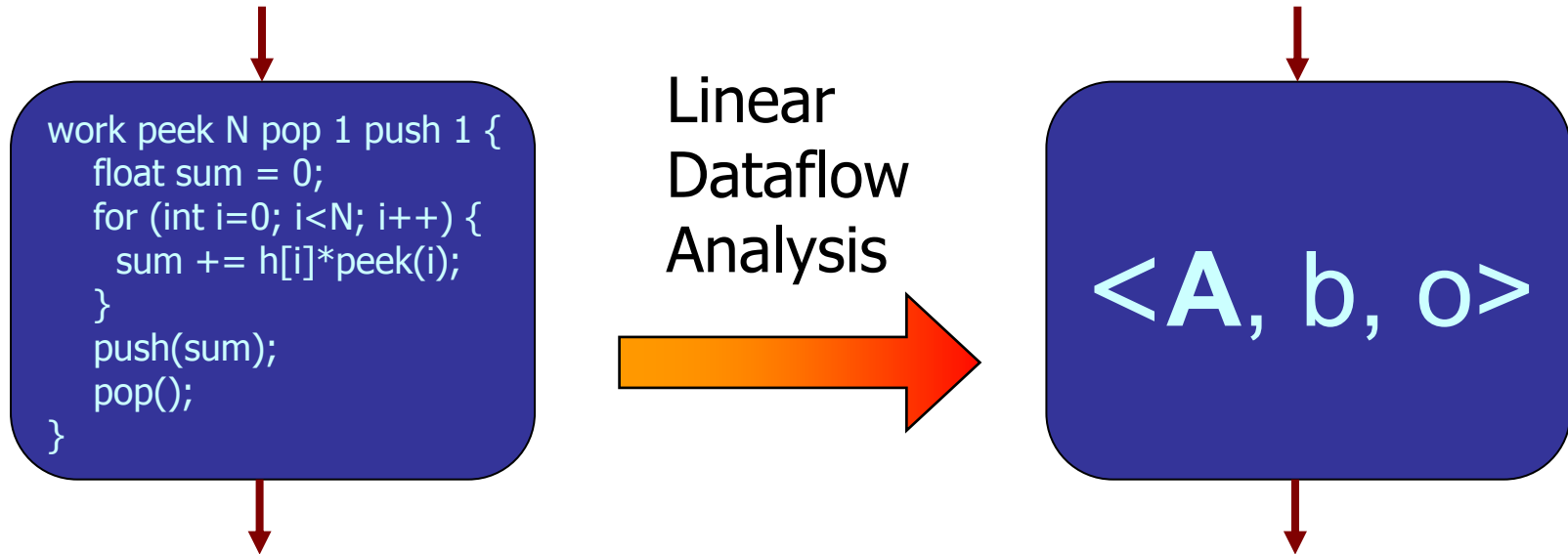
Representing Linear Filters

- A linear filter is a tuple $\langle \mathbf{A}, b, o \rangle$
 - $\mathbf{A}$: matrix of coefficients
 - b : vector of constants
 - o : number of items popped

- Example



Extracting Linear Representation



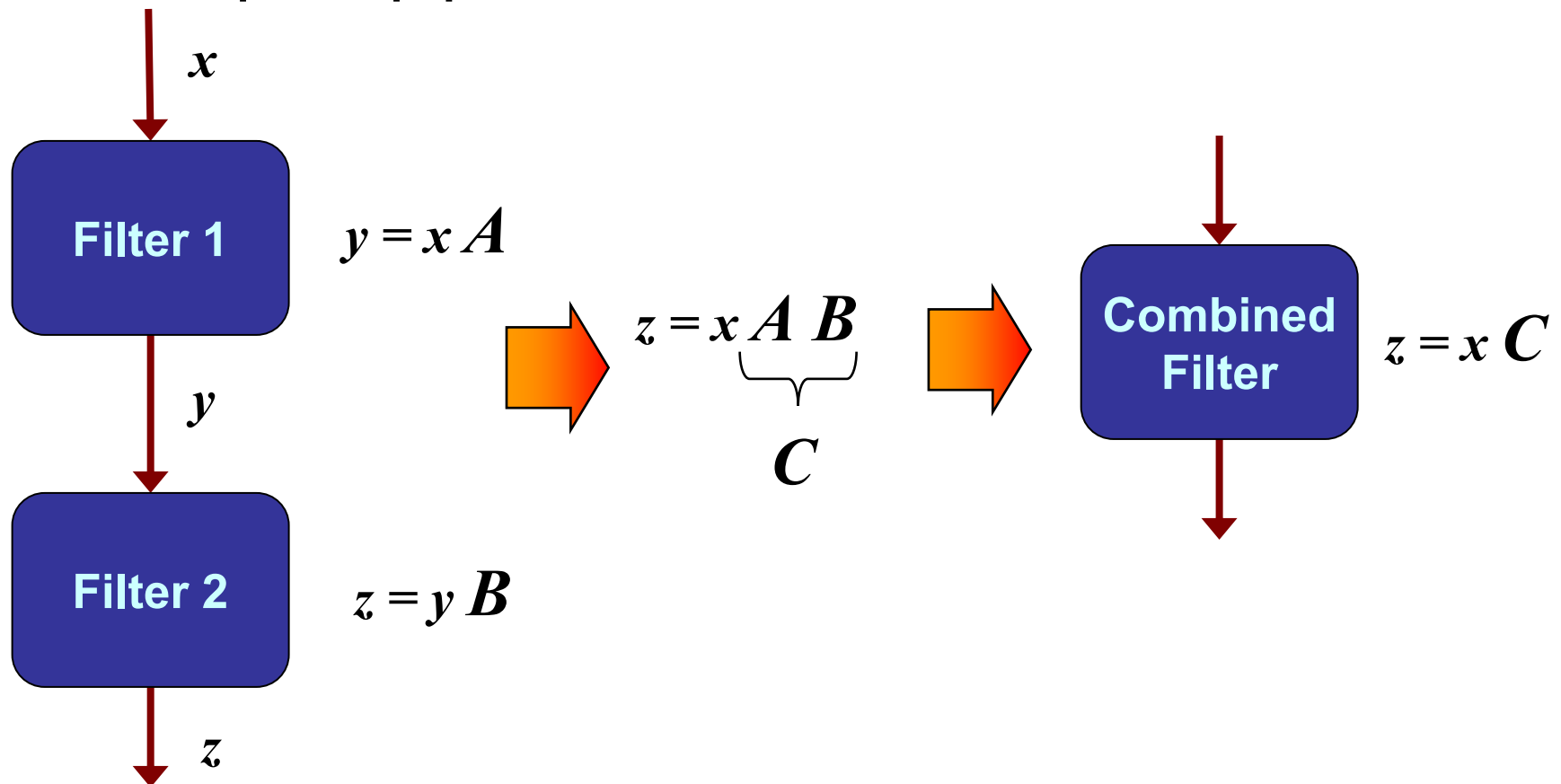
- Resembles constant propagation
- Maintains linear form $\langle v, b \rangle$ for each variable
 - Peek expression: generate fresh v^1
 - Push expression: copy v^1 into $\mathbf{A}$
 - Pop expression: increment o

Optimizations using Linear Analysis

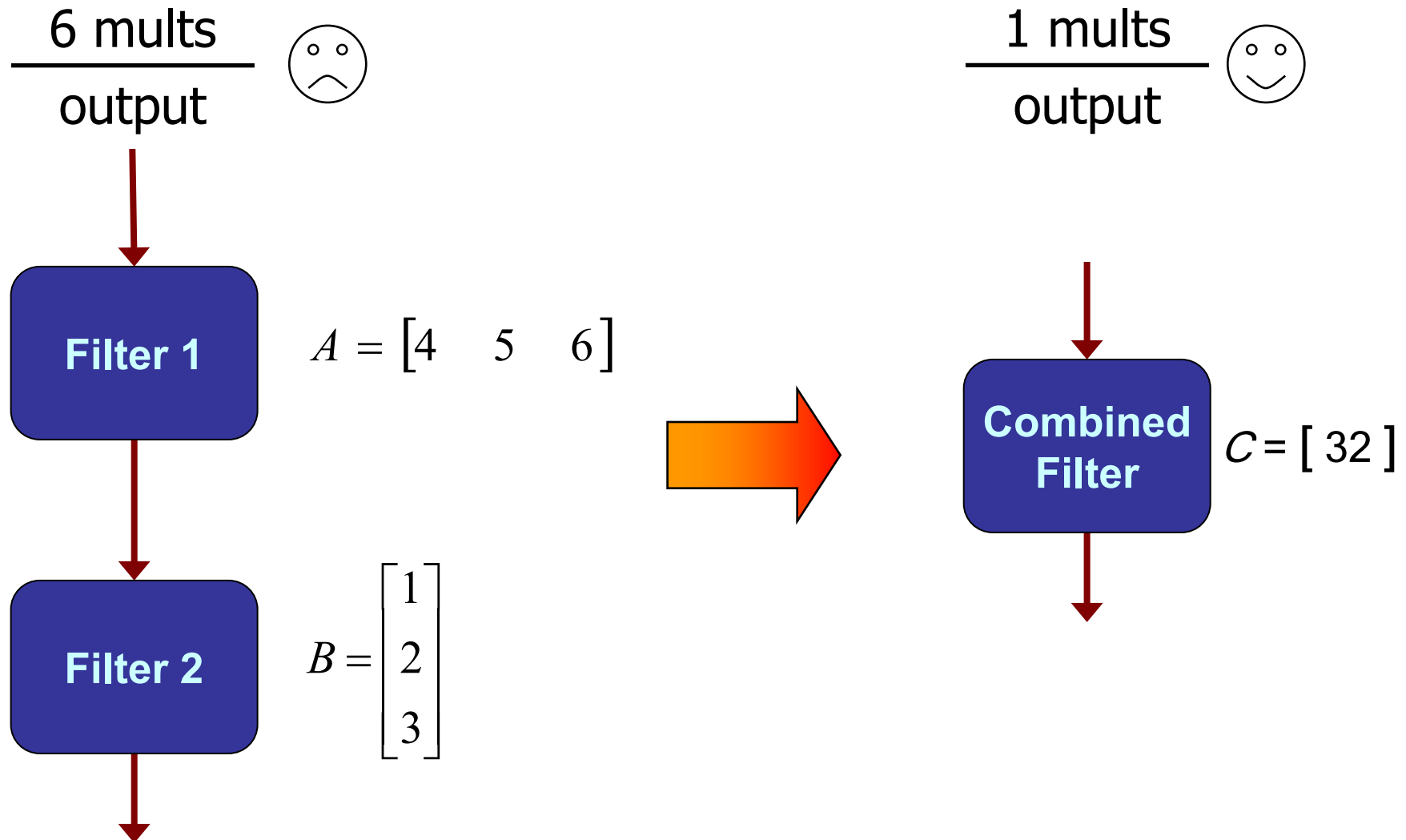
- 1) Combining adjacent linear structures
- 2) Shifting from time to the frequency domain
- 3) Selection of 'optimal' set of transformations

1) Combining Linear Filters

- Pipelines and splitjoins can be collapsed
- Example: pipeline



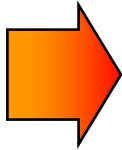
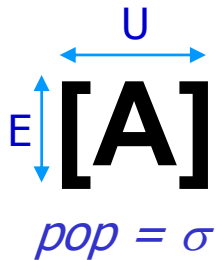
Combination Example



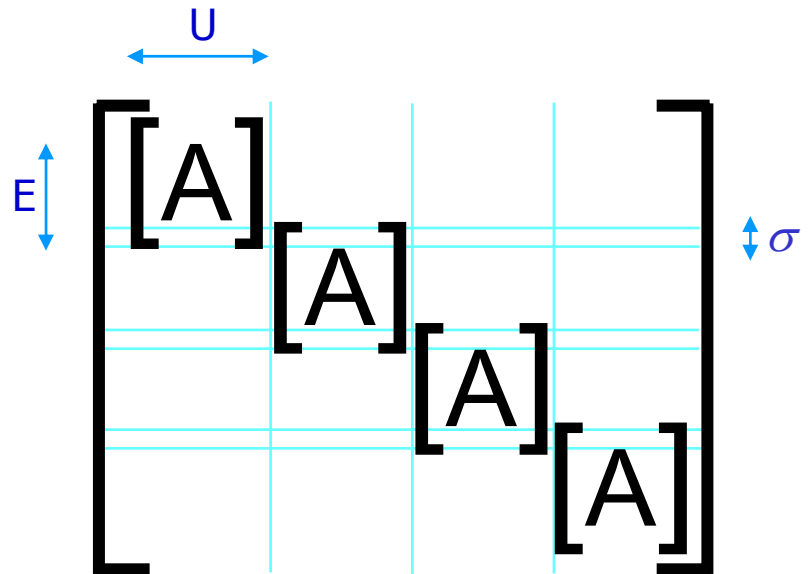
AB for any A and B??

■ Linear Expansion

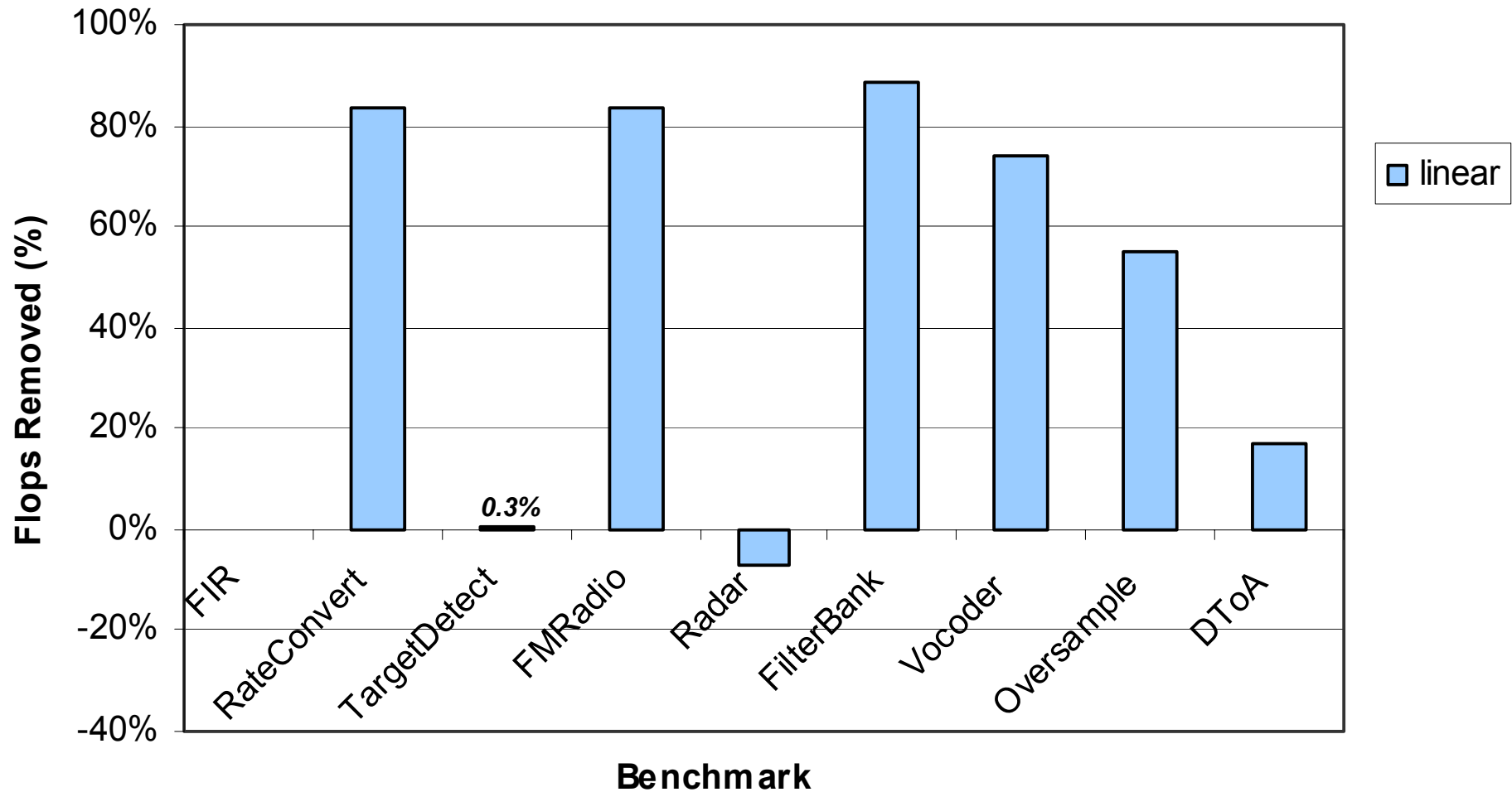
Original



Expanded

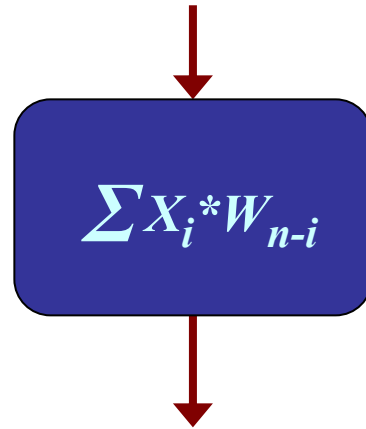


Floating-Point Operations Reduction



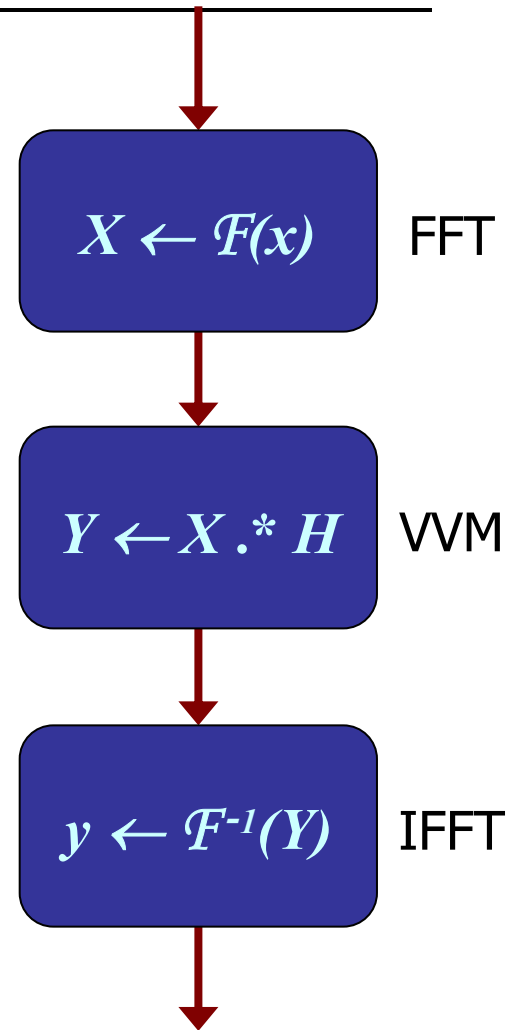
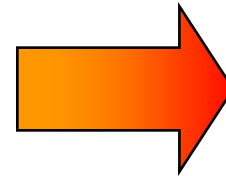
2) From Time to Frequency Domain

- Convolutions can be done cheaply in the Frequency Domain

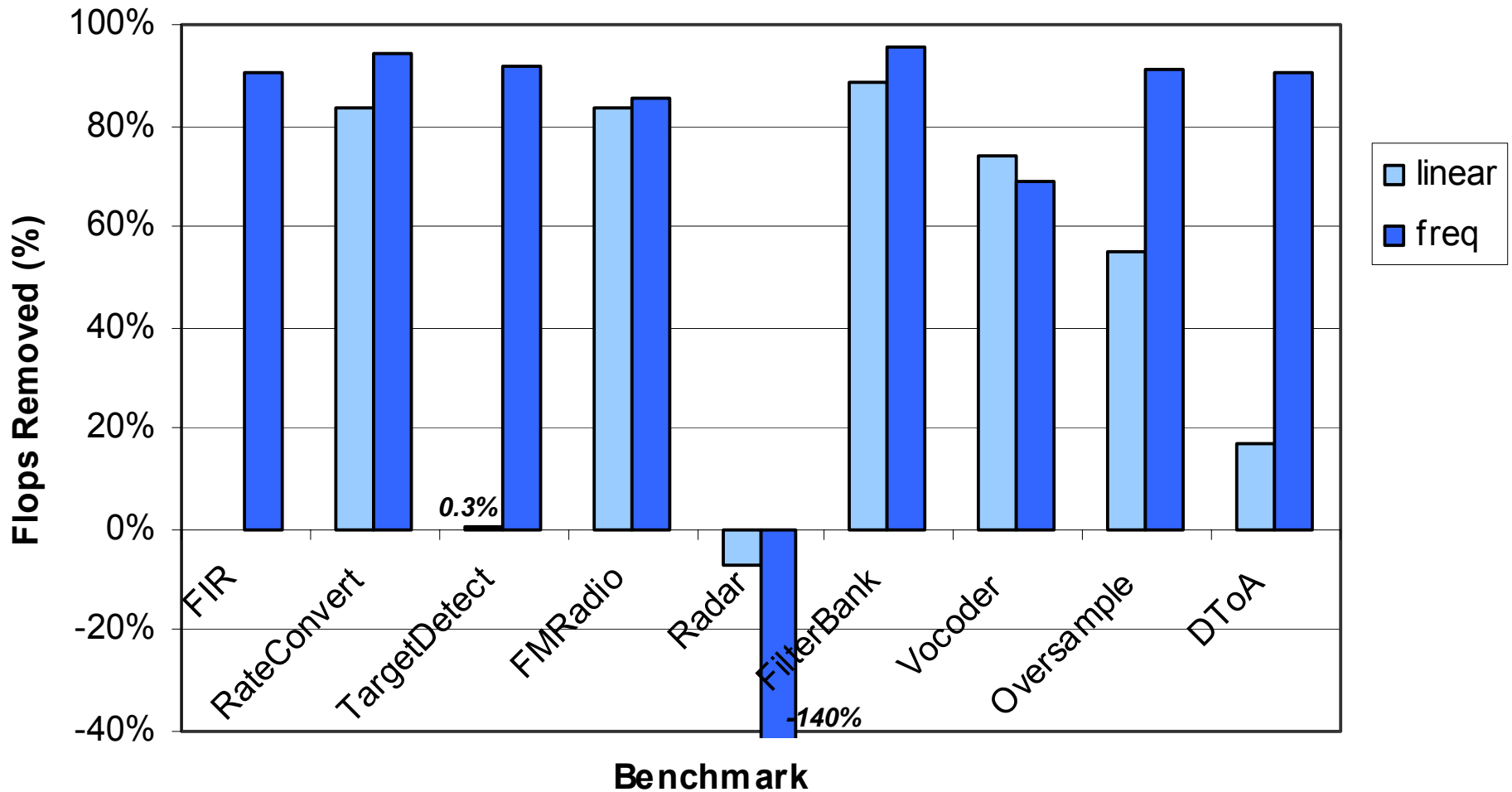


A blue rounded rectangle containing the equation $\sum X_i * W_{n-i}$. A red arrow points down to it from above, and another red arrow points down from it.

- Painful to do by hand
 - Blocking
 - Coefficient calculations
 - Startup etc.



Floating-Point Operations Reduction



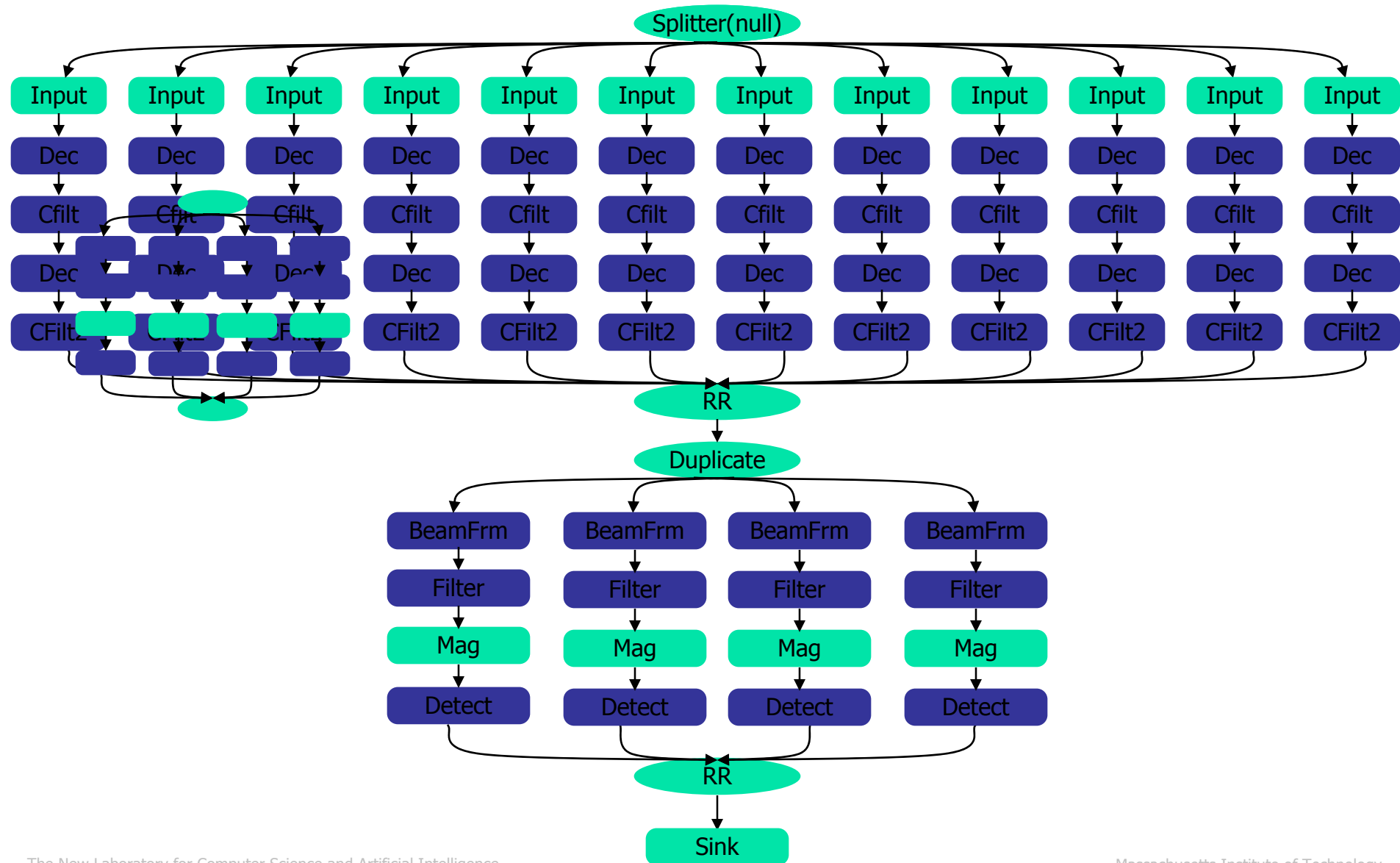
3) Transformation Selection

- When to apply what transformations?
 - Linear filter combination can increase the computation cost
 - Shifting to the Frequency domain is expensive for filters with $\text{pop} > 1$
 - Compute all outputs, then decimate by pop rate
 - Some expensive transformations may later enable other transformations, reducing the overall cost

Selection Algorithm

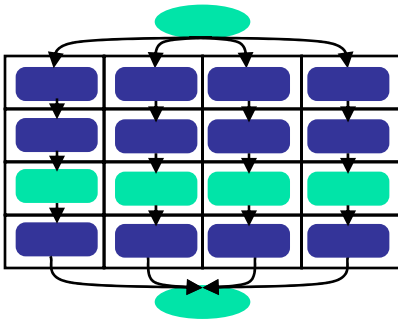
- Estimate minimal cost for each structure:
 - Linear combination
 - Frequency translation
 - No transformation
 - If hierarchical, consider all possible groupings of children
- } Cost function based on profiler feedback
- Overlapping sub-problems allows efficient dynamic programming search

Radar (Transformation Selection)



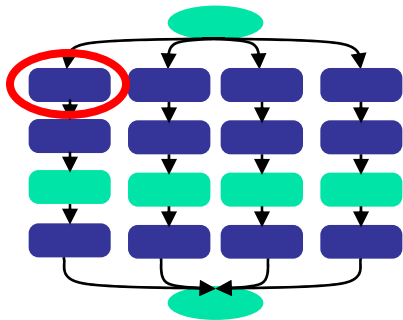
Radar (Transformation Selection)

First compute cost of individual filters:

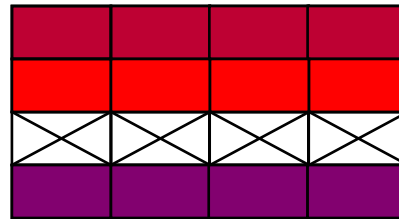


Radar (Transformation Selection)

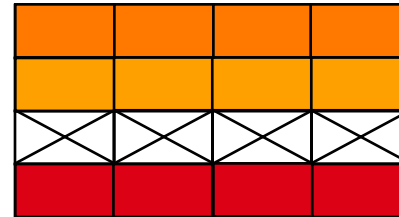
First compute cost of individual filters:



Linear Combination



Frequency

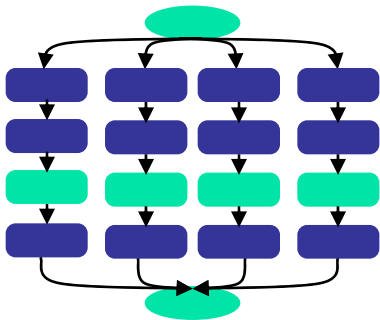


No Transform

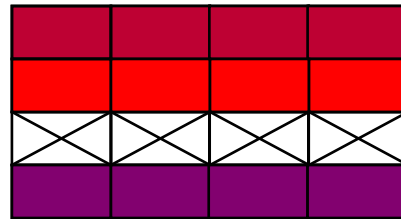


Radar (Transformation Selection)

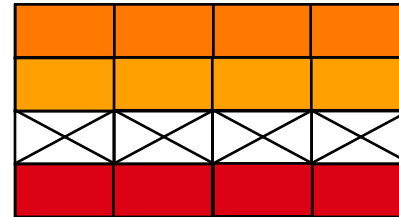
First compute cost of individual filters:



Linear Combination



Frequency



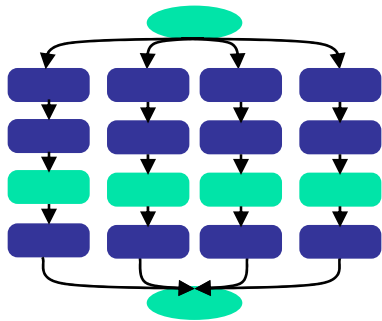
No Transform



1x1

Radar (Transformation Selection)

Then, compute cost of 1x2 nodes:

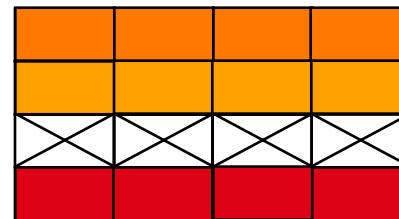
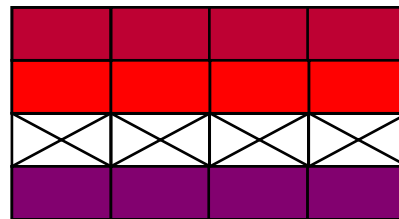


Linear Combination

Frequency

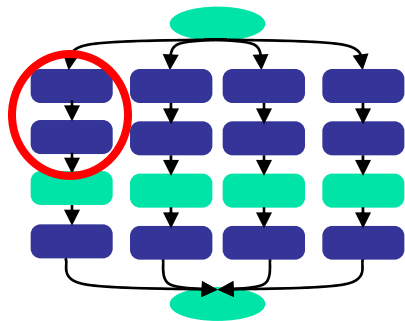
No Transform

1x1

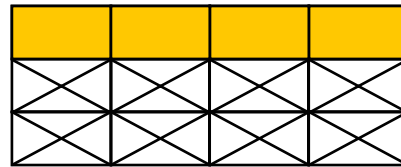


Radar (Transformation Selection)

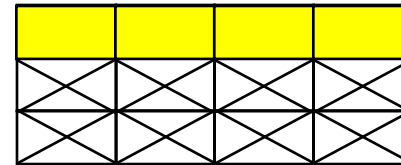
Then, compute cost of 1x2 nodes:



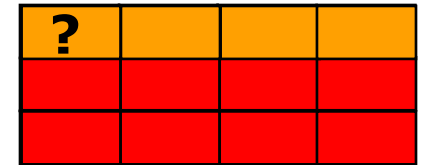
Linear Combination



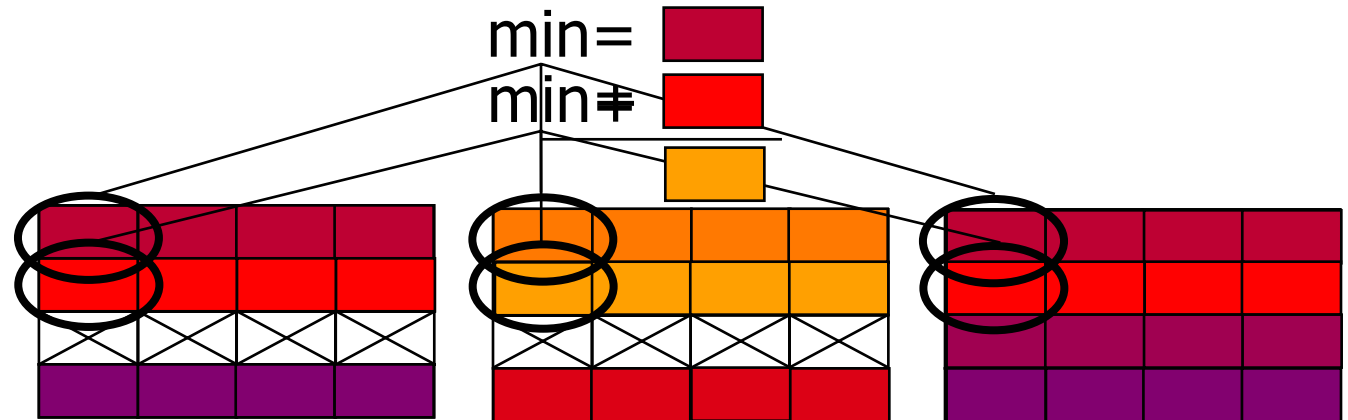
Frequency



No Transform

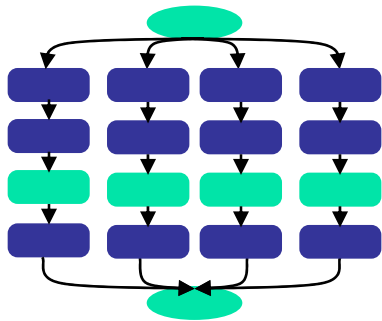


1x1

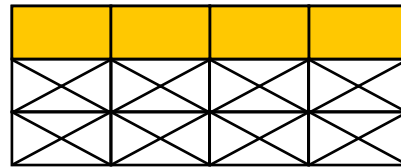


Radar (Transformation Selection)

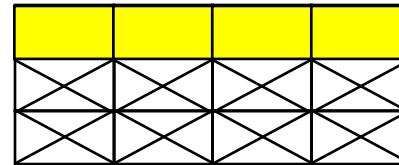
Then, compute cost of 1x2 nodes:



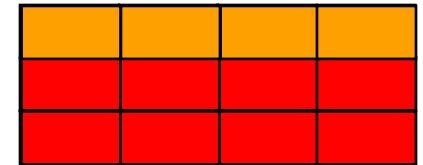
Linear Combination



Frequency

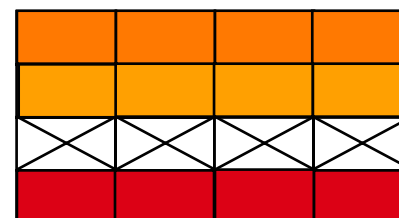
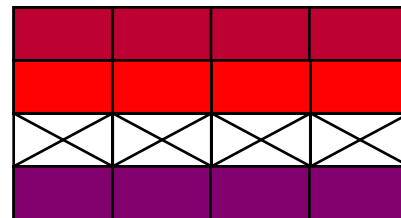


No Transform



1x2

1x1



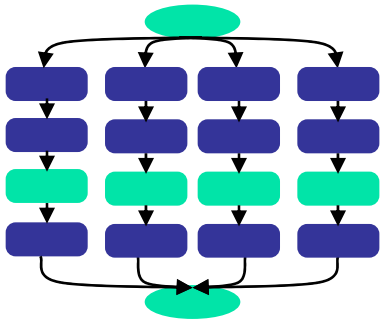
Radar (Transformation Selection)

Continue with 1x3 2x1 3x1 4x1

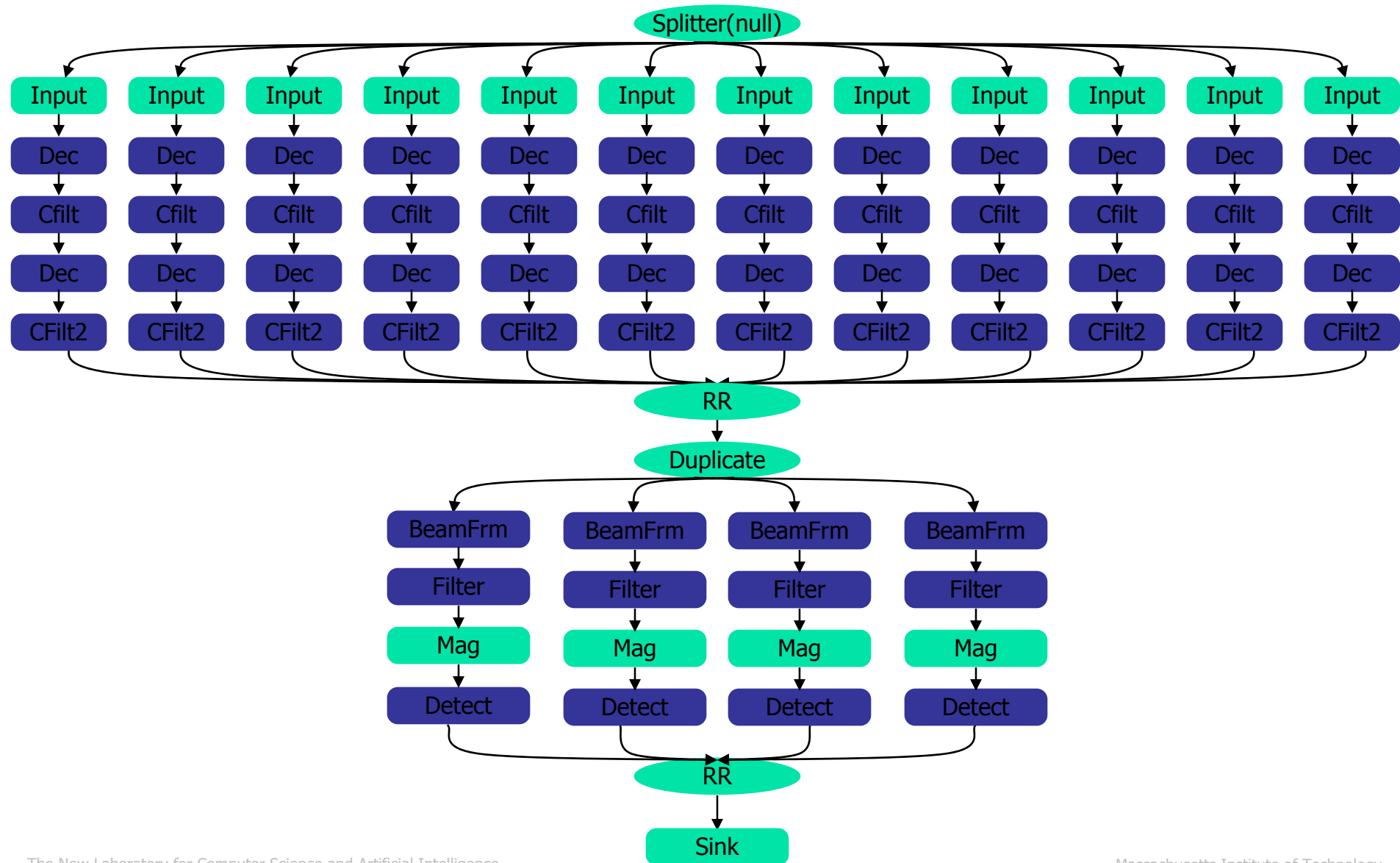
1x4 2x2 3x2 4x2

2x3 3x3 4x3

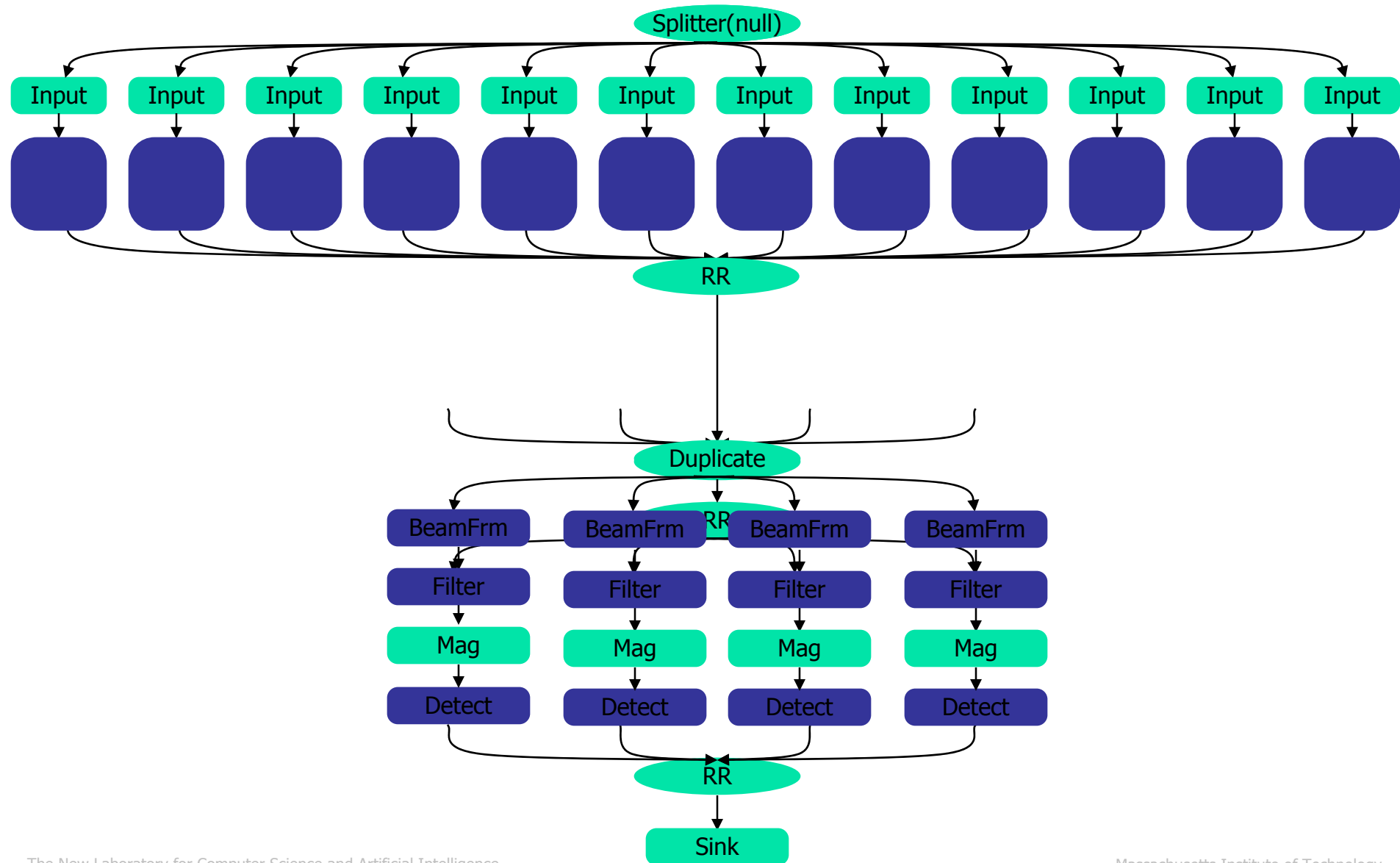
2x4 3x4 **4x4** → Overall solution



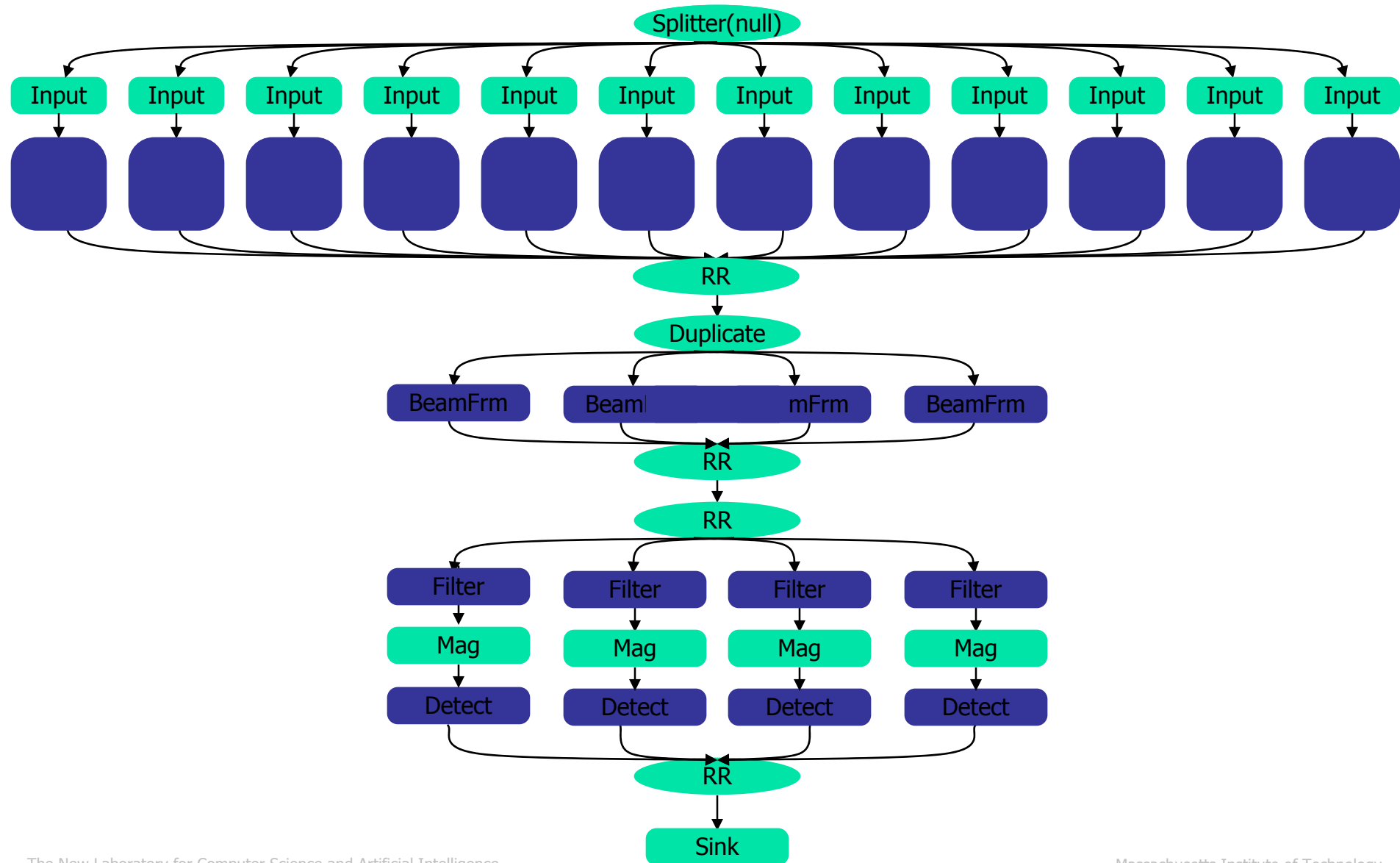
Radar (Transformation Selection)



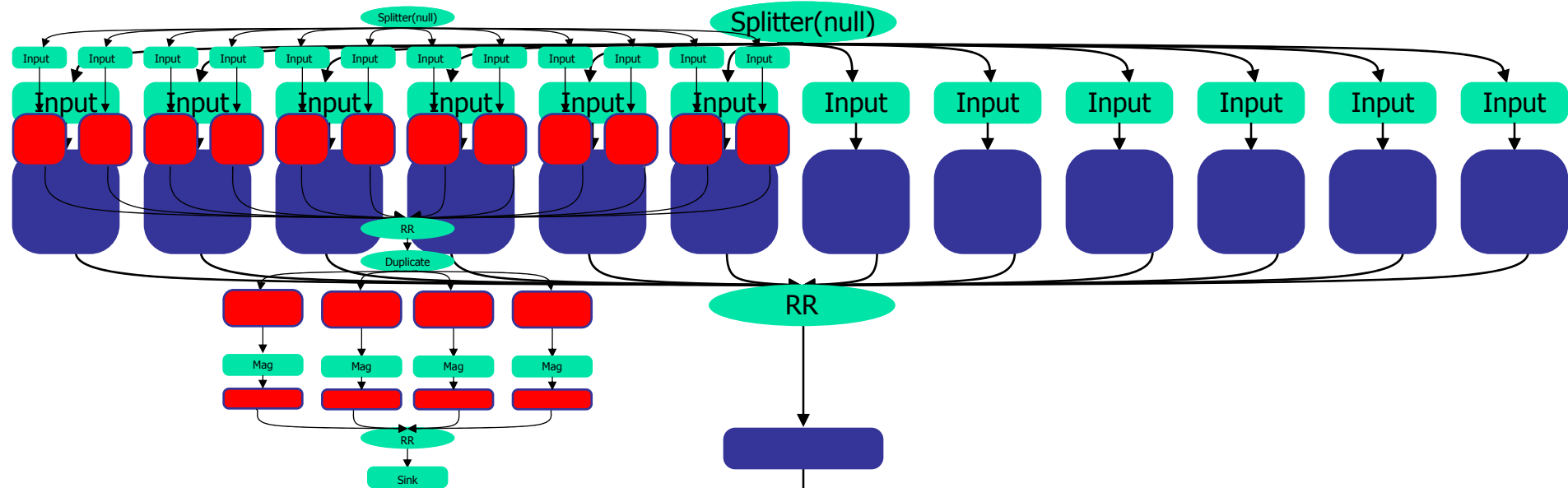
Radar (Transformation Selection)



Radar (Transformation Selection)

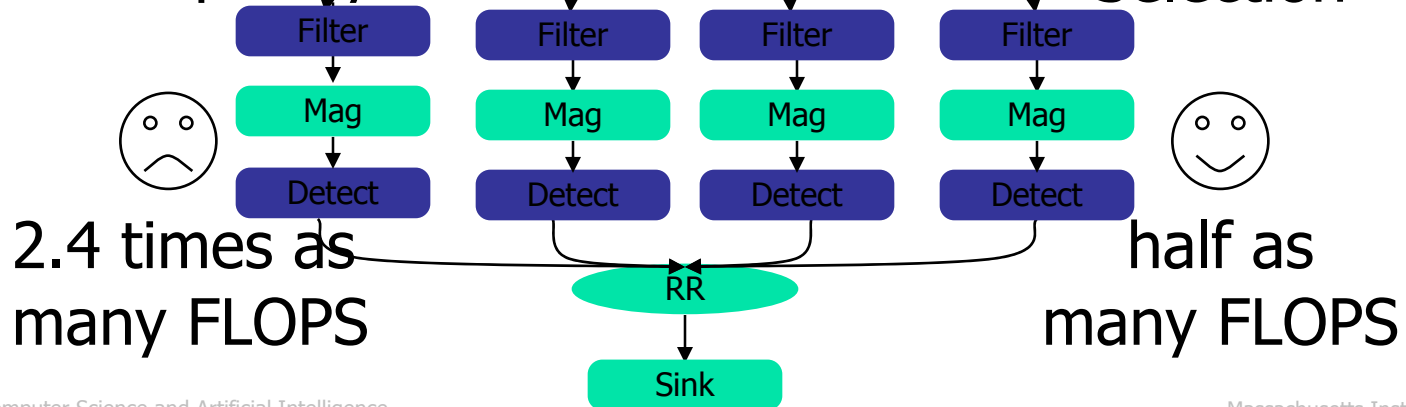


Radar

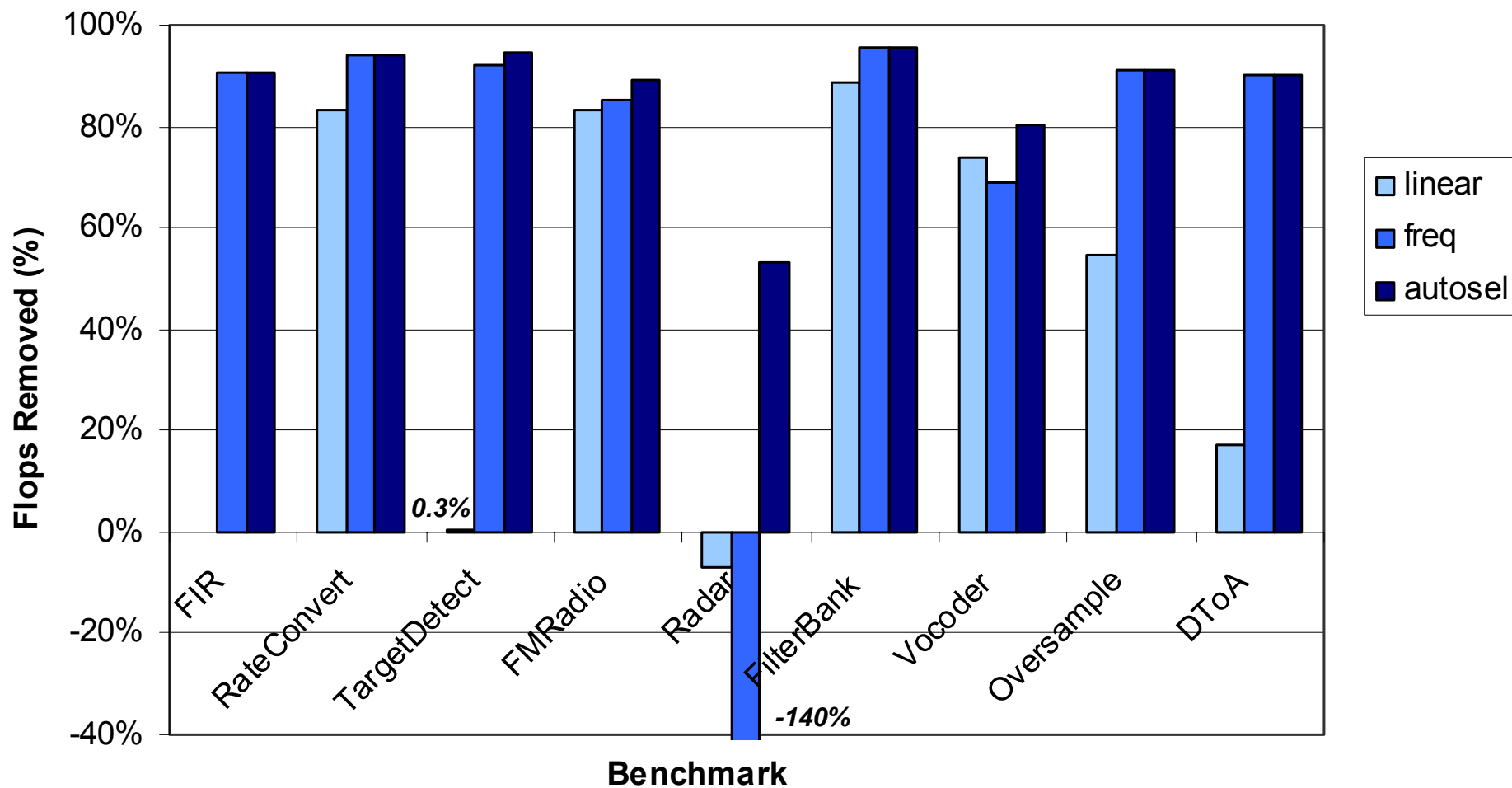


Maximal Combination and
Shifting to Frequency Domain

Using Transformation
Selection



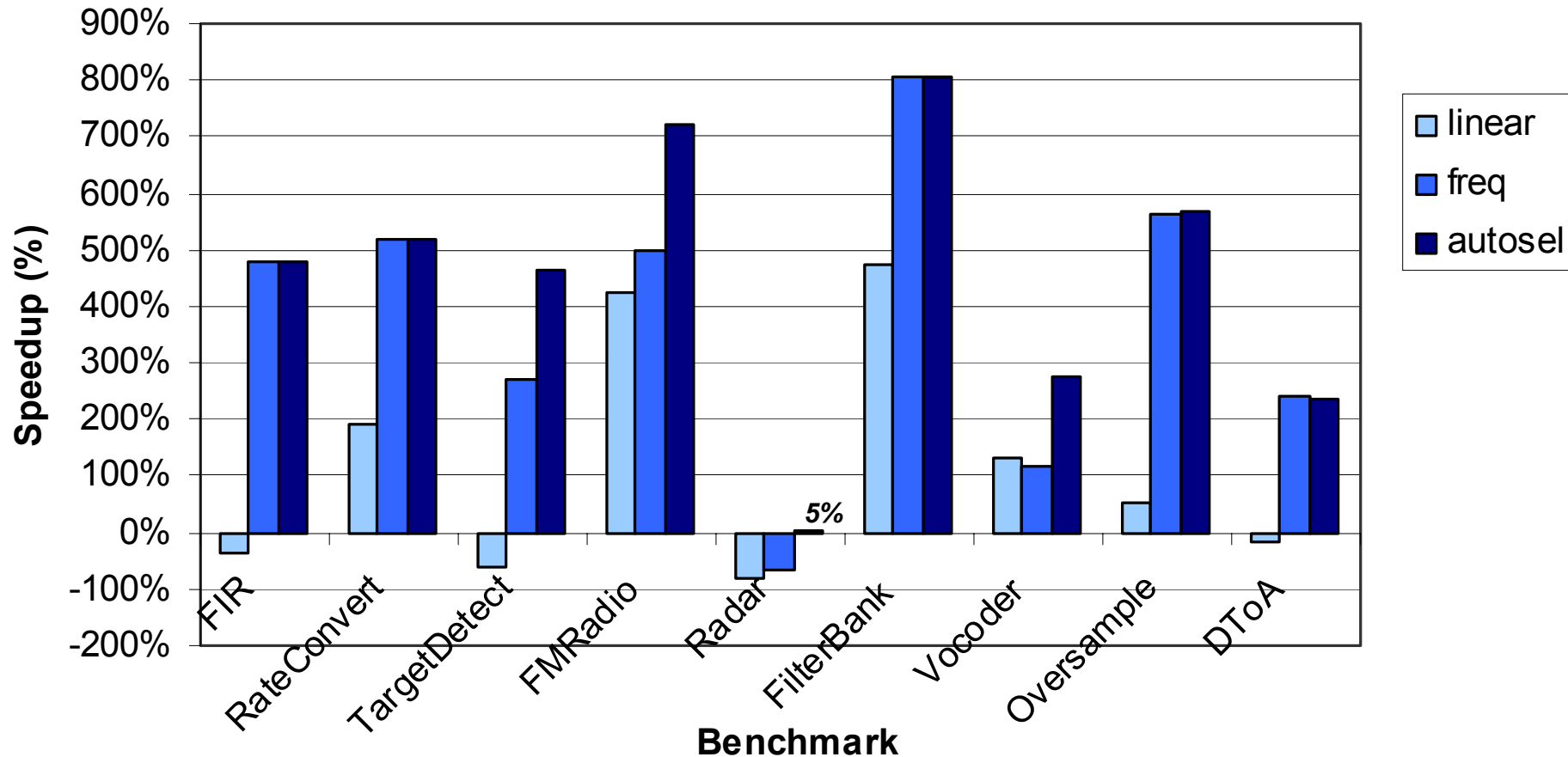
Floating-Point Operations Reduction



Experimental Results

- Fully automatic implementation
 - StreamIt compiler
- StreamIt to C compilation
 - FFTW for shifting to the frequency domain
- Benchmarks all written in StreamIt
- Measurements
 - Dynamic floating-point instruction counting
 - Speedups on a general purpose processor

Execution Speedup



On a Pentium IV

Related Work

- SPIRAL/SPL (Püschel et. al)
 - Automatic derivation of DSP transforms
- FFTW (Friego et. al)
 - Wicked fast FFT
- ADE (Covell, MIT PhD Thesis, 1989)
- Affine Analysis (Karr, Acta Informatica, 1976)
 - *Affine relationships among variables of a program*
- Linear Analysis (Cousot, Halbwachs, POPL, 1978)
 - *Automatic discovery of linear restraints among variables of a program*

Conclusions

- A DSP Program Representation: *Linear Filters*
 - A dataflow analysis that recognizes linear filters
- Three Optimizations using Linear Information
 - Adjacent Linear Structure Combination
 - Time Domain to Frequency Domain Transformation
 - Automatic Transformation Selection
- Effective in Replacing the DSP Engineer from the Design Flow
 - On the average 90% of the FLOPs eliminated
 - Average performance speedup of 450%
- StreamIt: A Unified High-level Abstraction for DSP Programming
 - Increased abstraction does not have to sacrifice performance

<http://cag.lcs.mit.edu/linear/>