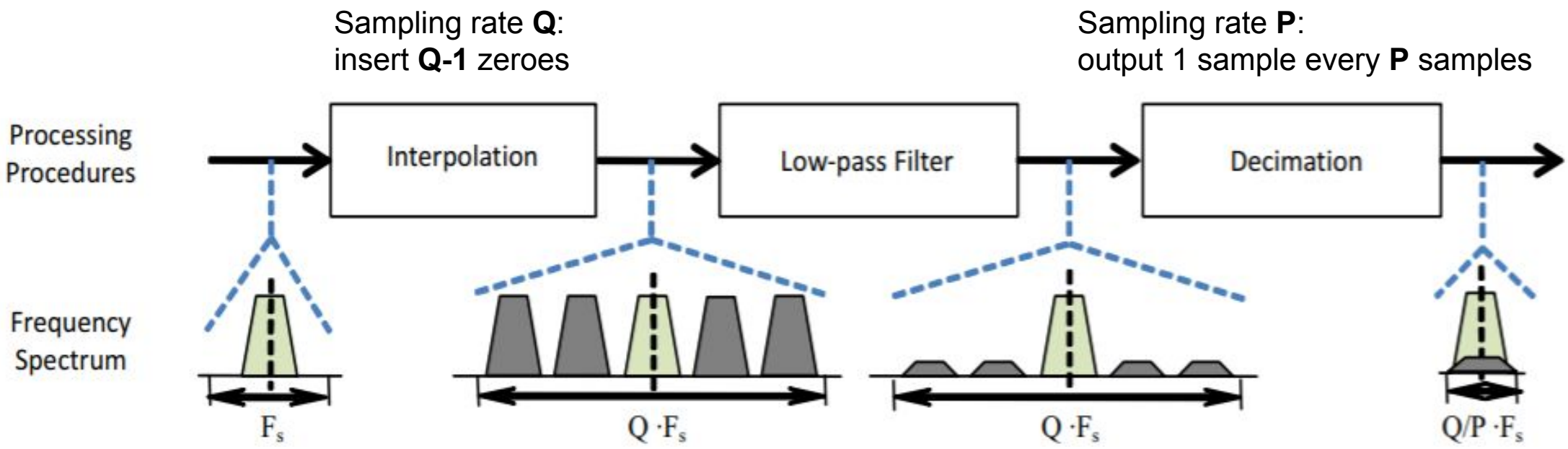
Lab A: Multi-rate Filtering

Team 7 朱祐葳 蘇著傑 吳宜凡

Outline

- Introduction
- 10.4 Using Decimation in Filters
- 10.5 Using Interpolation in Filters
- 10.6 Multi-stage Decimation

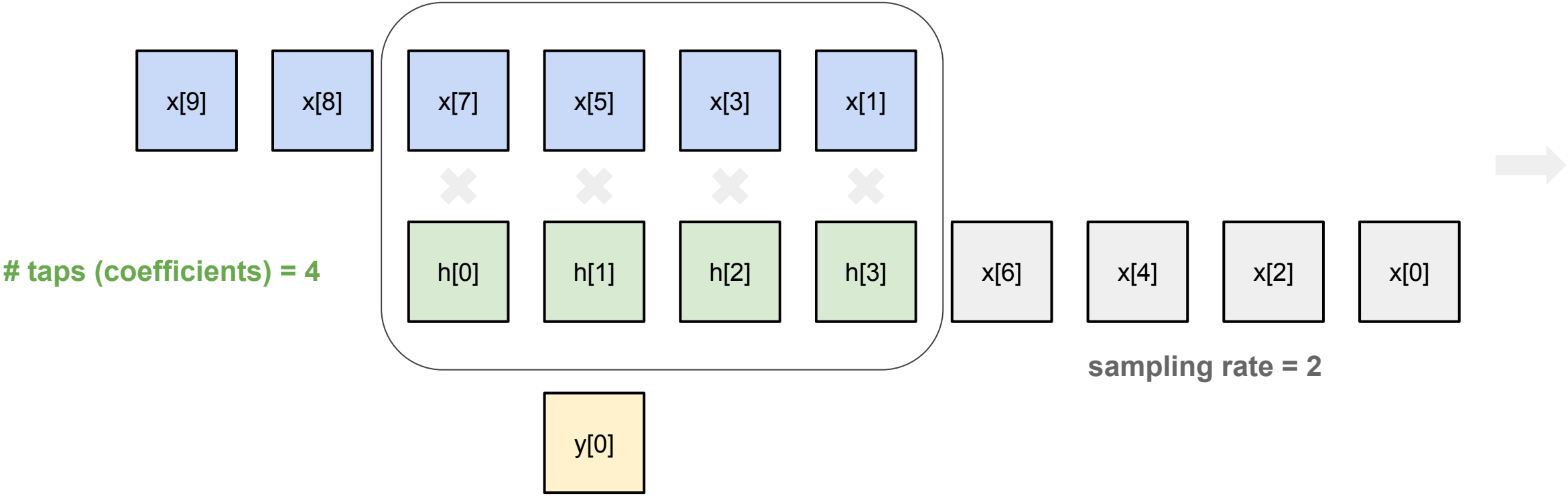
Introduction



10.4 Using Decimation in Filters

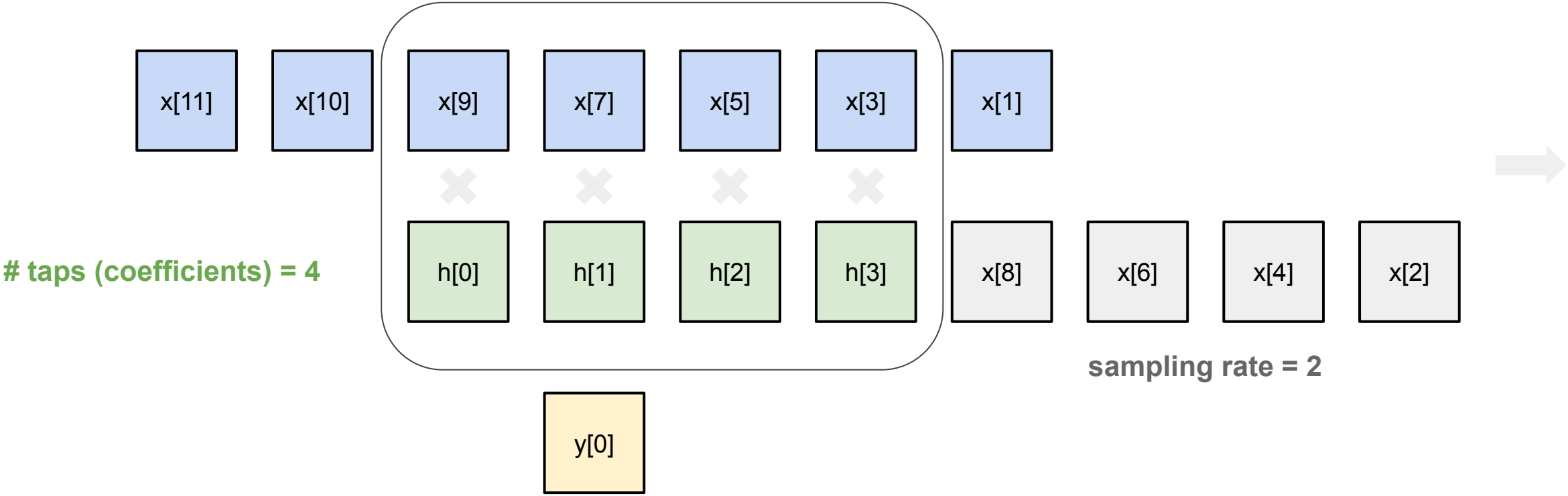
Using Decimation in Filters

- **Down** sampling + lowpass FIR filter

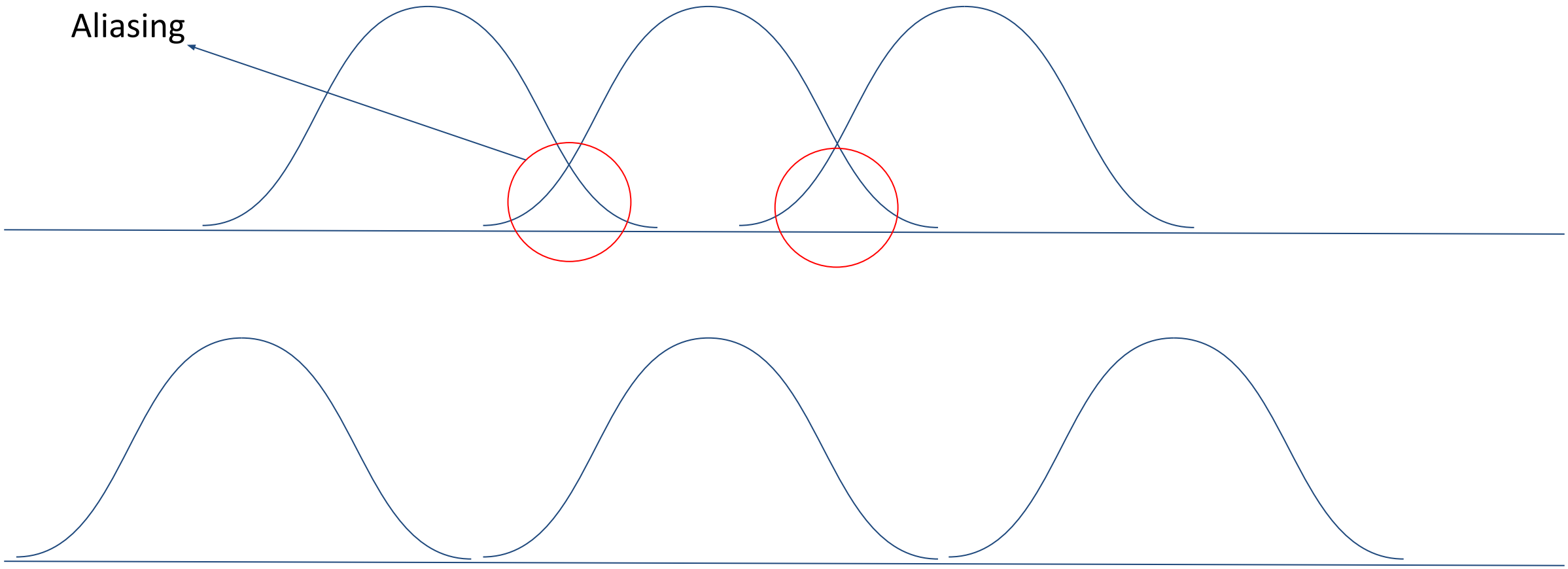


Using Decimation in Filters

- **Down** sampling + lowpass FIR filter



Frequency Aliasing



Algorithmic Decimation Design Parameters

- Pipeline II = down sampled rate
- Read data in original rate for efficiency
- Minimise the area

Algorithmic Decimation

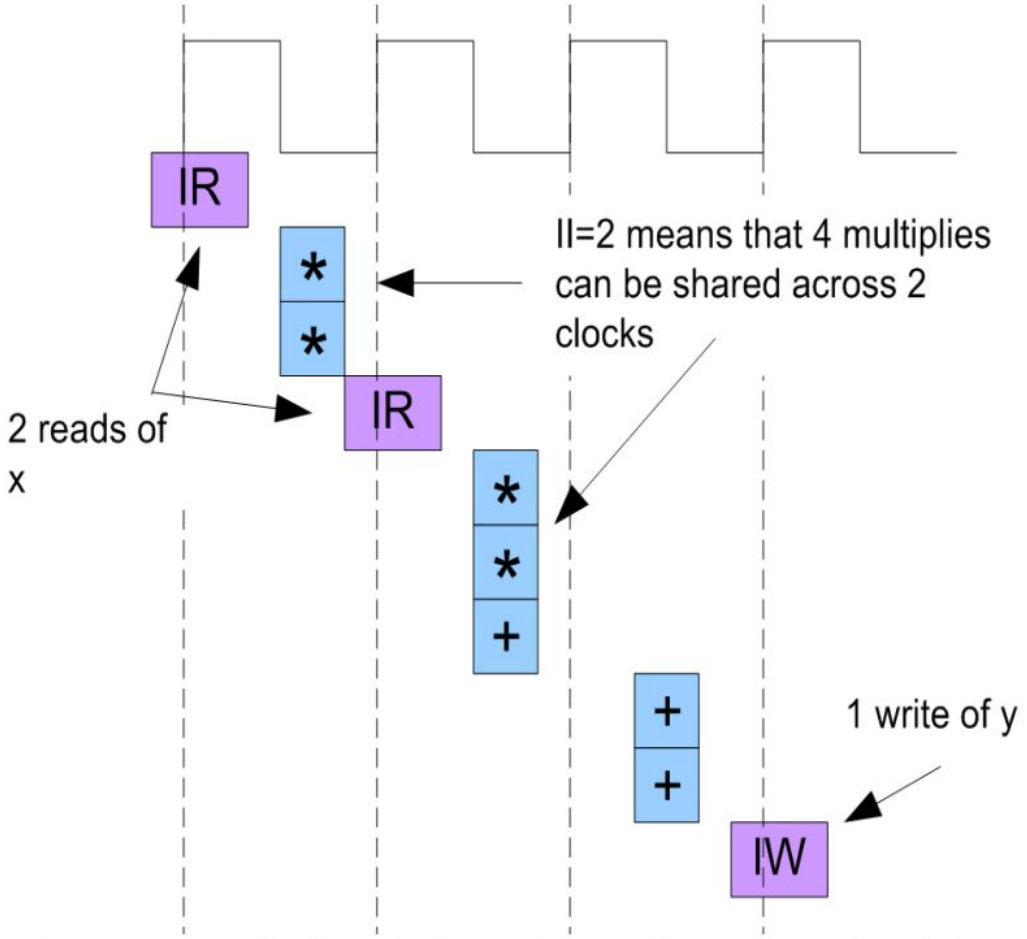


Illustration 145: Schedule of Decimation by Two

Algorithmic Decimation : READ Pipeline

original

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	3.455 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	27	5.000 ns	0.135 us	1	27	none

READ pipeline

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	3.455 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	21	5.000 ns	0.105 us	1	21	none

MAC unroll

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.702 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	23	10.000 ns	0.230 us	1	23	none

loop unroll

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.702 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	9	10.000 ns	90.000 ns	1	9	none

Algorithmic Decimation

```
if ( x->available(RATE)){
    READ:for(int i=0;i<RATE;i++)
    x_int = x->read();
    regs << x_int;
    MAC:for (int i = 0; i<N; i++) {
        acc += h[i]*regs[i];
    }
    y->write(acc);
}
```

Main loop

MAC loop

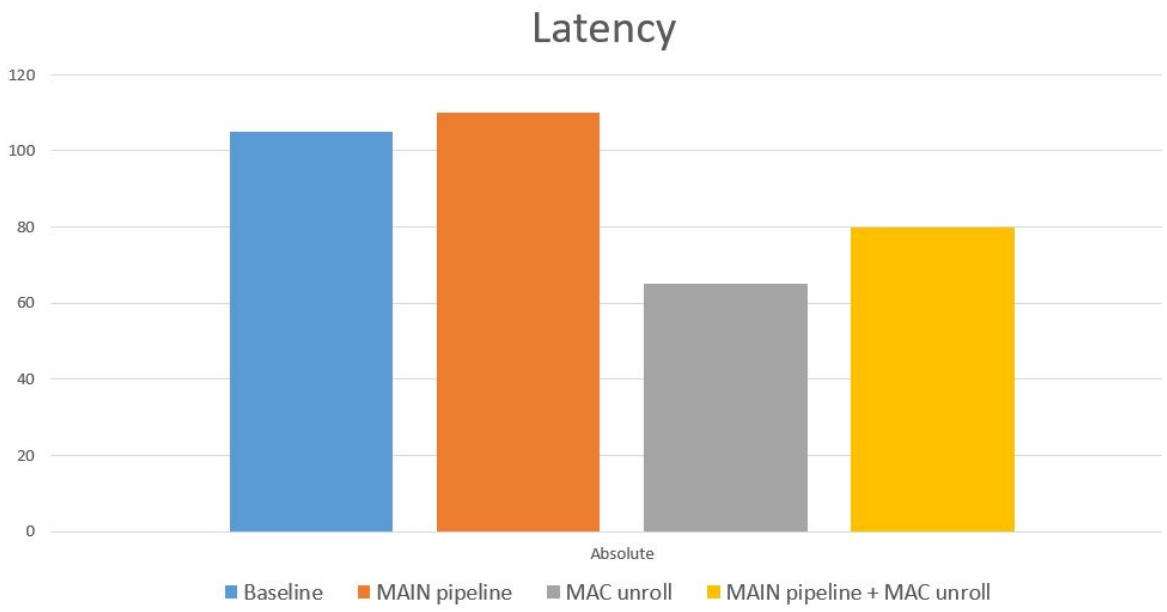
Testbench

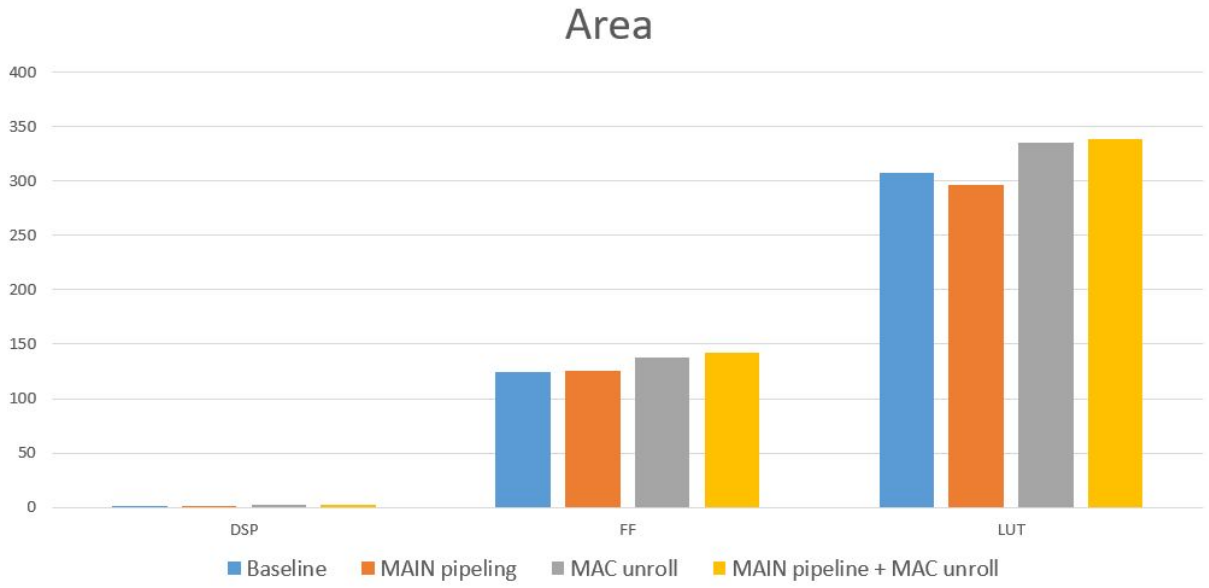
```
for (i = 0; i < NUM_SAMPLES; i++) {
    signal = 0.98 * sin(2 * pi * i / 64);
    input[i] = signal;
    strmInput.write(signal);
    fir_filter(&strmInput, taps, &strmOutput);
}
```

Input Signal

Filter

Algorithmic Decimation





Manual Decimation

```
 1 #include <ac_channel.h>
 2 #include <ac_fixed.h>
 3 #include "shift_class.h"
 4 void dec2(ac_channel< ac_fixed<8,1> > &x,
 5                 ac_fixed<8,1> h[4],
 6                 ac_channel< ac_fixed<19,4> > &y){
 7   static shift_class<ac_fixed<8,1>,4> regs;
 8   ac_fixed<19,8> temp = 0;
 9   static ac_int<1,0> cnt;
10
11   regs << x.read();
12   MAC:for (int i = 0; i<4; i++) {
13     temp += h[i]*regs[i];
14   }
15   if(cnt==1)//Phase 1
16     y.write(temp);
17   cnt++;
18 }
PATH: $MGC_HOME/shared/examples/docs/bluebook/filter/dec2_pure.cpp
```

Manual Decimation

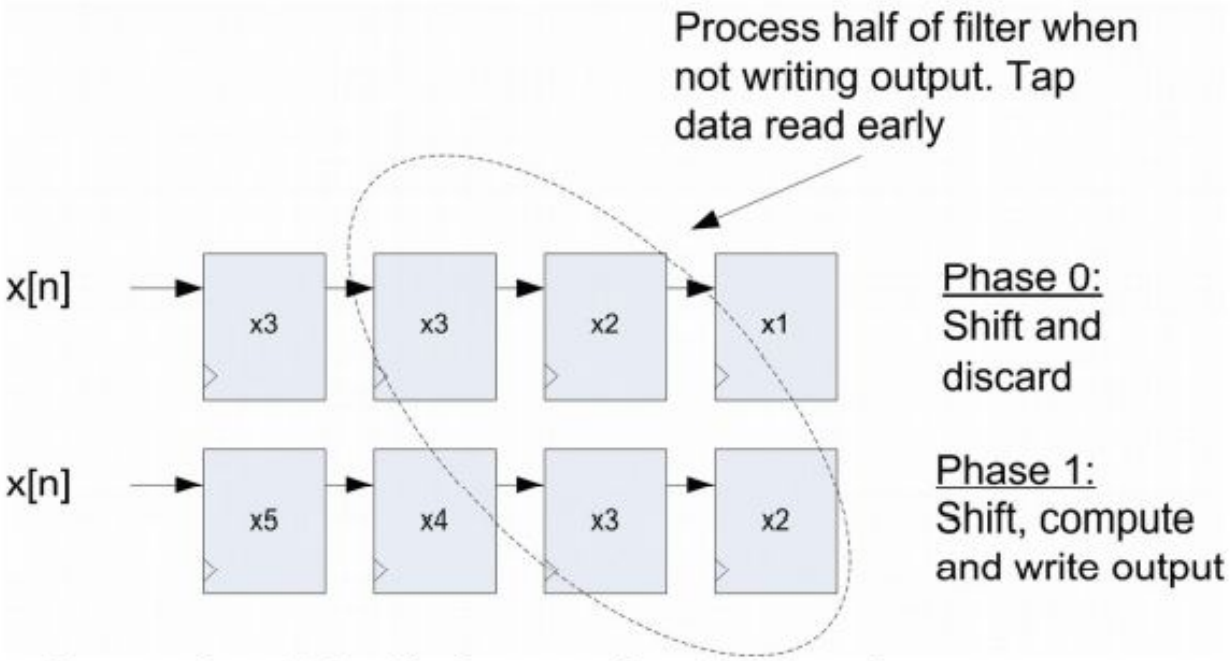


Illustration 146: Understanding Decimation

Phase1: y[5] = h[0]*regs[0] + h[1]*regs[1] + h[2]*regs[2] + h[3]*regs[3]

Phase0: temp = h[2]*regs[1] + h[3]*regs[2]

Phase1: y[5] = h[0]*regs[0] + h[1]*regs[1] + temp

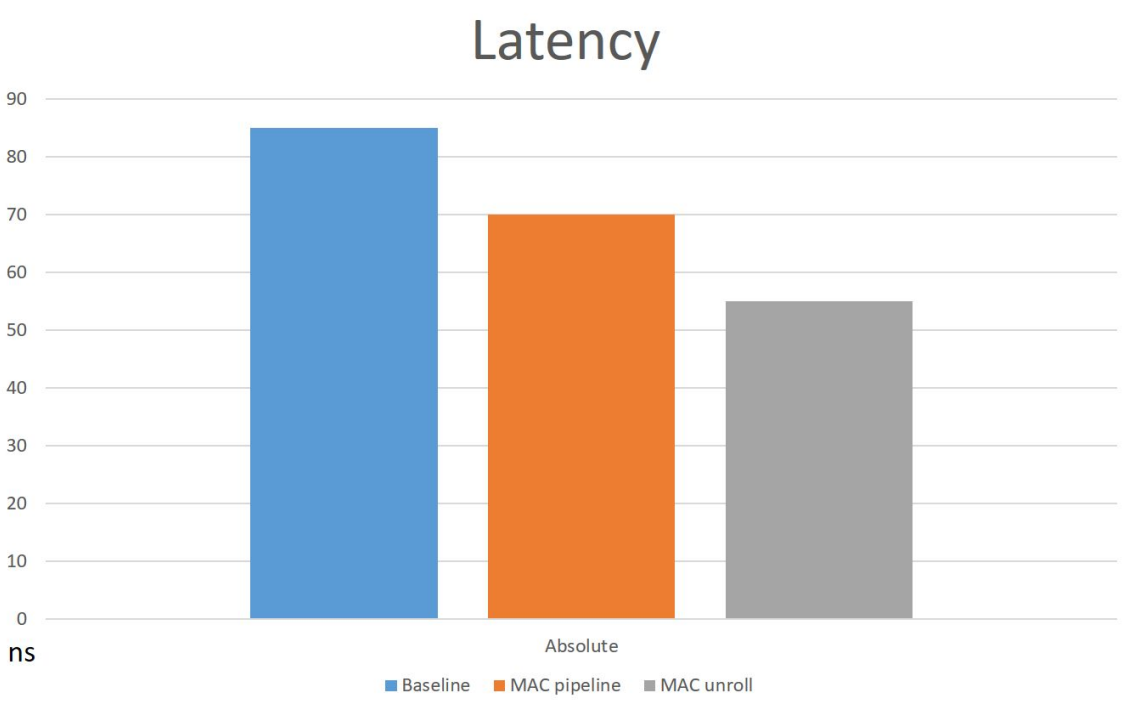
Manual Decimation

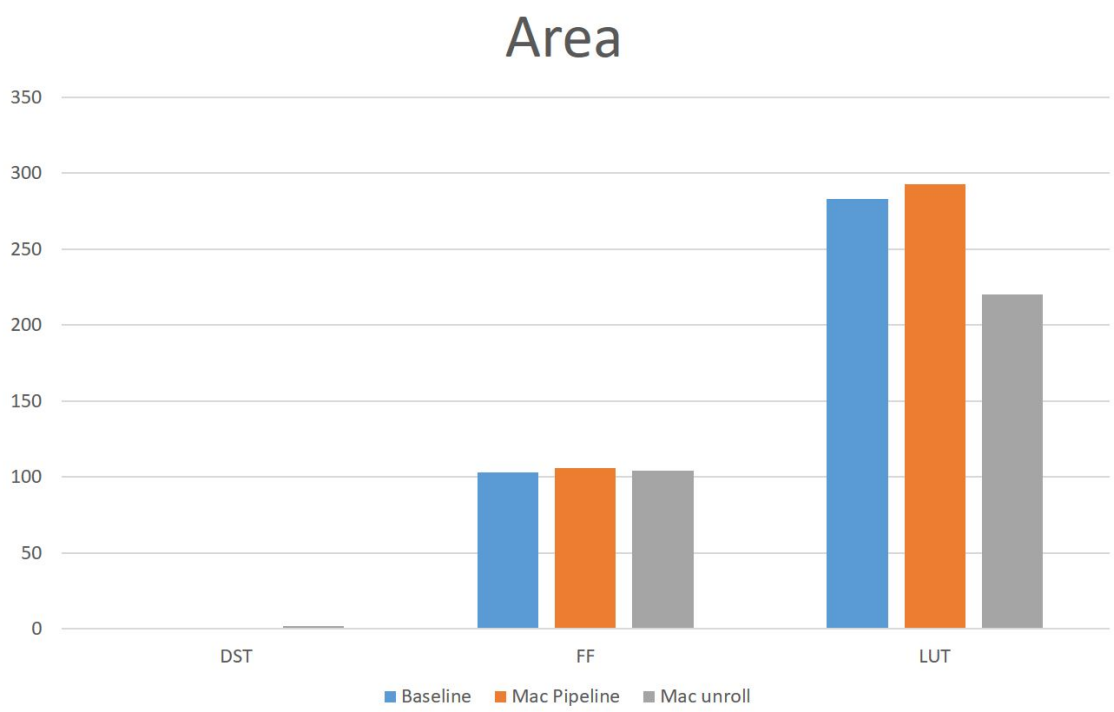
```
regs << x->read();
MAC0:for (int i = 0; i<2; i++) {
    acc += h[i+((1-cnt)<<1)]*regs[i+1-cnt];
}
if(cnt==1){
    y->write(acc);
    acc = 0;
}
cnt++;
```

Shift and Discard

Shift, compute and write output

Manual Decimation





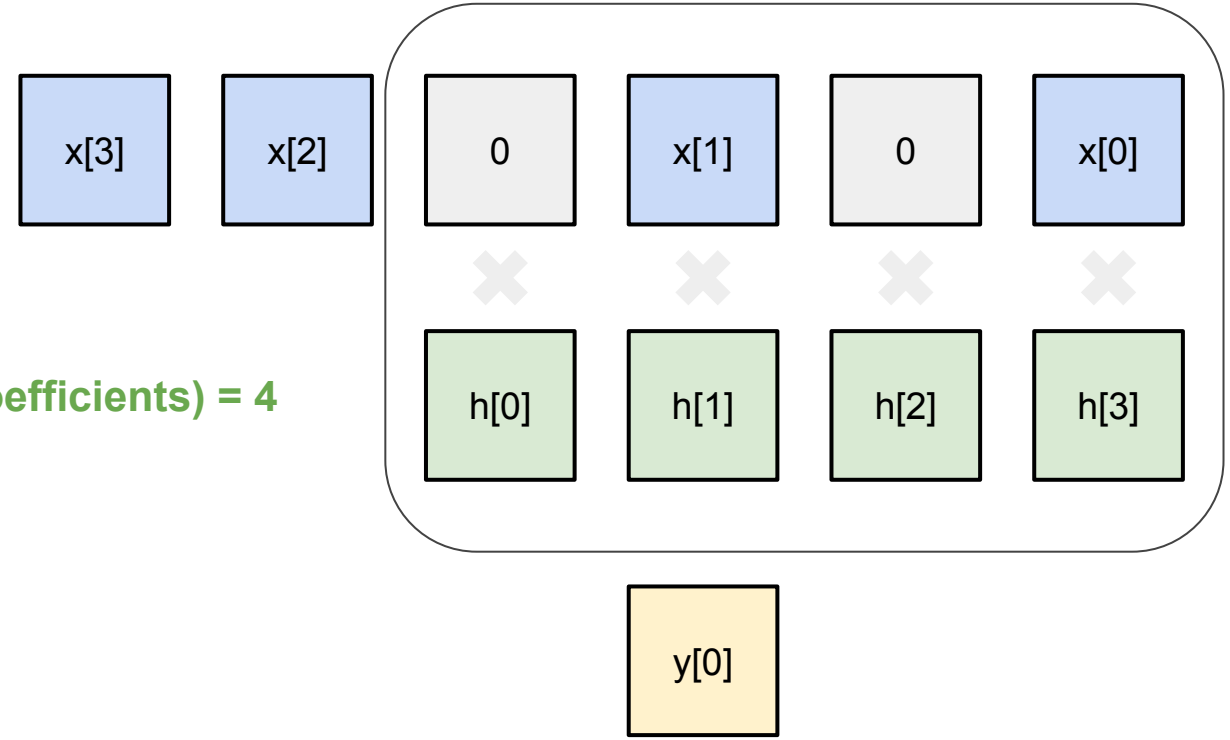
Algorithmic vs. Manual Decimation: Constraints

- All IO mapped to wire enable interfaces
- All arrays mapped to registers
- IO input rate=1 sample/clock (**algorithmic**)
- Main loop pipelined with II=2 (**algorithmic**) / input rate (**manual**)
- All loops (**algorithmic**) / MAC loop (**manual**) fully unrolled

10.5 Using Interpolation in Filters

Using Interpolation in Filters

- **Up** sampling + lowpass FIR filter

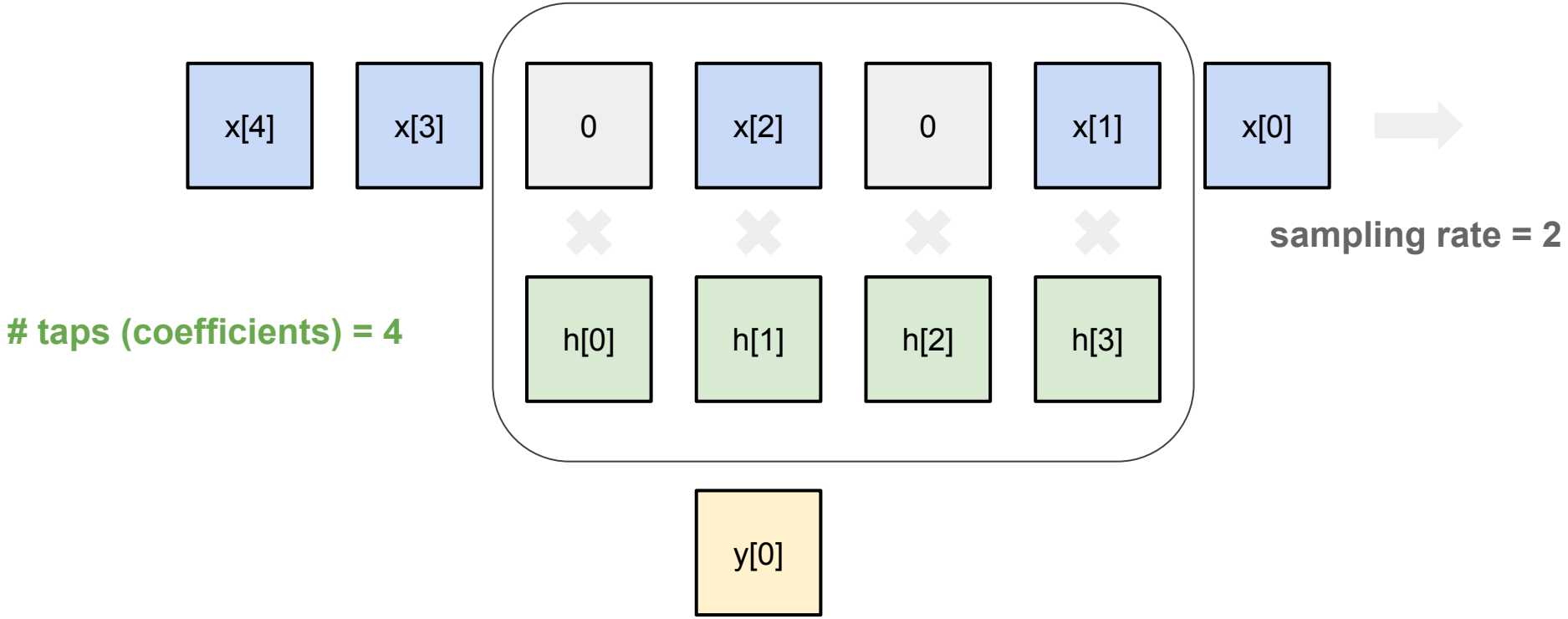


 # taps (coefficients) = 4

sampling rate = 2

Using Interpolation in Filters

- **Up** sampling + lowpass FIR filter



Algorithmic Interpolation

```
void fir_filter(streamIn_t* x,
                fixIn_t h[NUM_TAPS],
                streamOut_t* y){
    inter<0,8,1,8,1,NUM_TAPS,S_RATE>(x,h,y);
}
```

Algorithmic Interpolation

```
14  void inter(hls::stream<ap_fixed<W0,I0> >* x,
15                 ap_fixed<W1,I1> h[N],
16                 hls::stream<ap_fixed<_WN<W0,W1,N>::val,_WN<I0,I1,N>::val> >* y) {
17      static shift_class<ap_fixed<W0,I0>,N> regs;
18      ap_fixed<_WN<W0,W1,N>::val,_WN<I0,I1,N>::val> acc = 0;
19
20      WRITE:for (int i = 0; i < RATE; i++){
21          if( i == 0)
22              regs << x->read();
23          else
24              regs << 0;
25          MAC:for (int j = 0; j<N; j++) {
26              acc += h[j] * regs[j];
27          }
28          y->write(acc);
29          acc = 0;
30      }
31  }
32
```

main loop

MAC loop

Algorithmic Interpolation

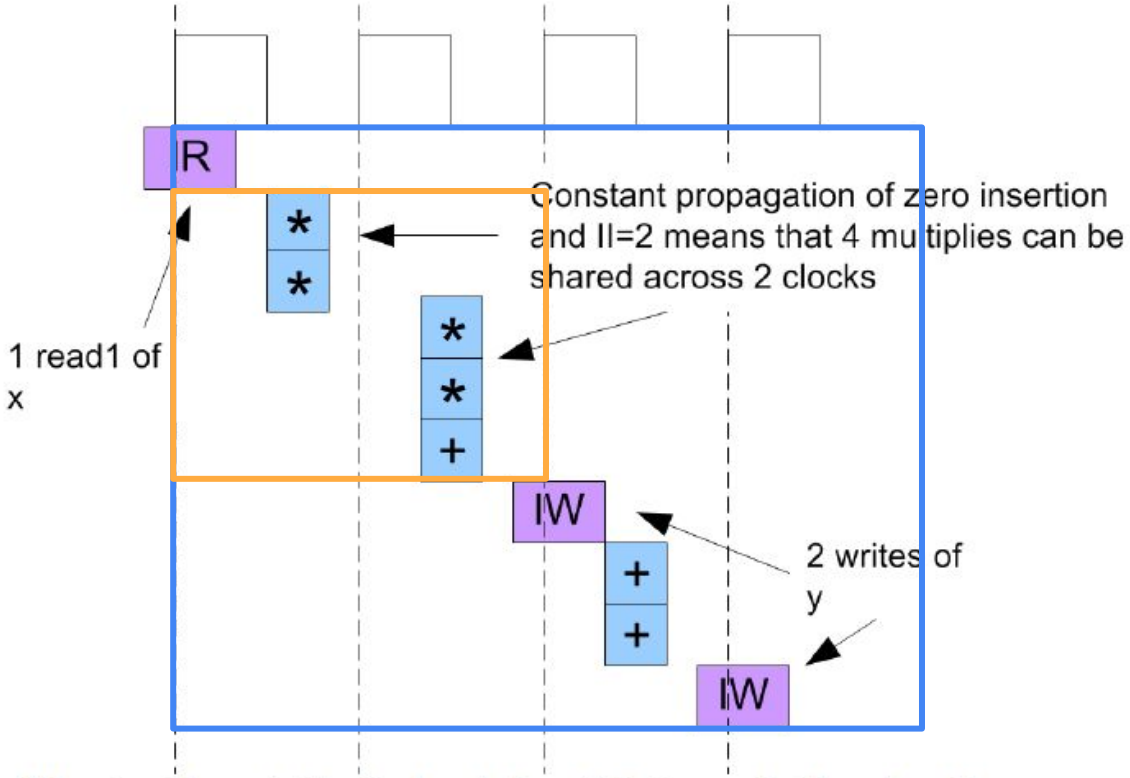


Illustration 147: Schedule of Interpolation by Two

main loop

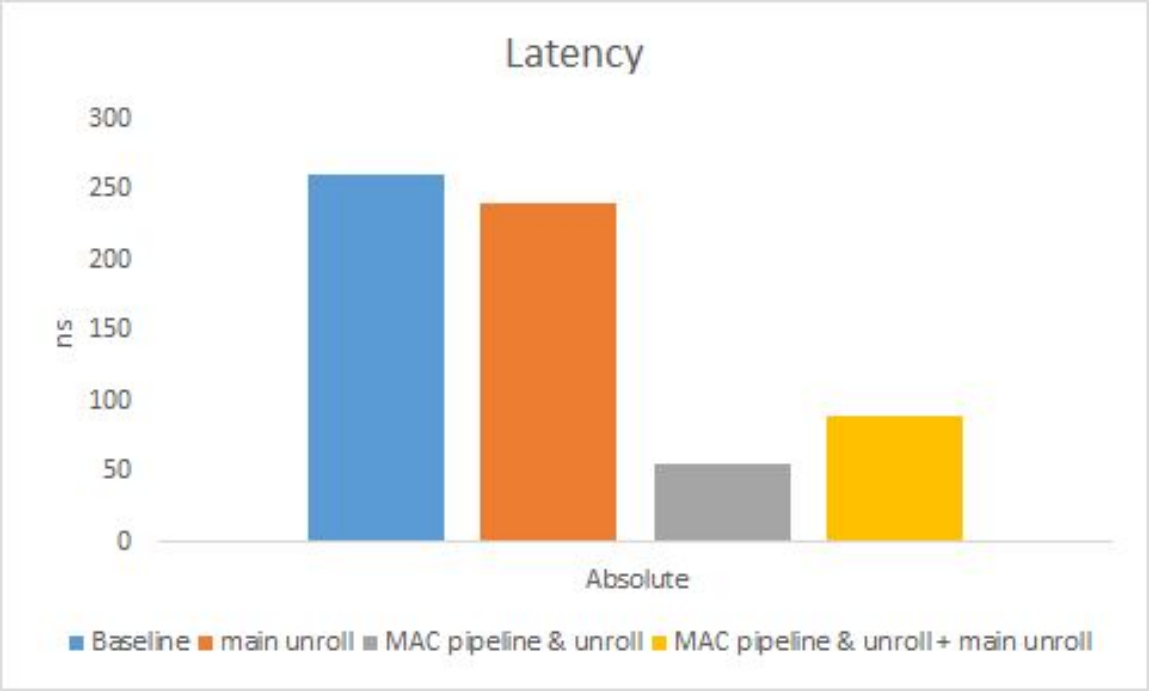
MAC loop

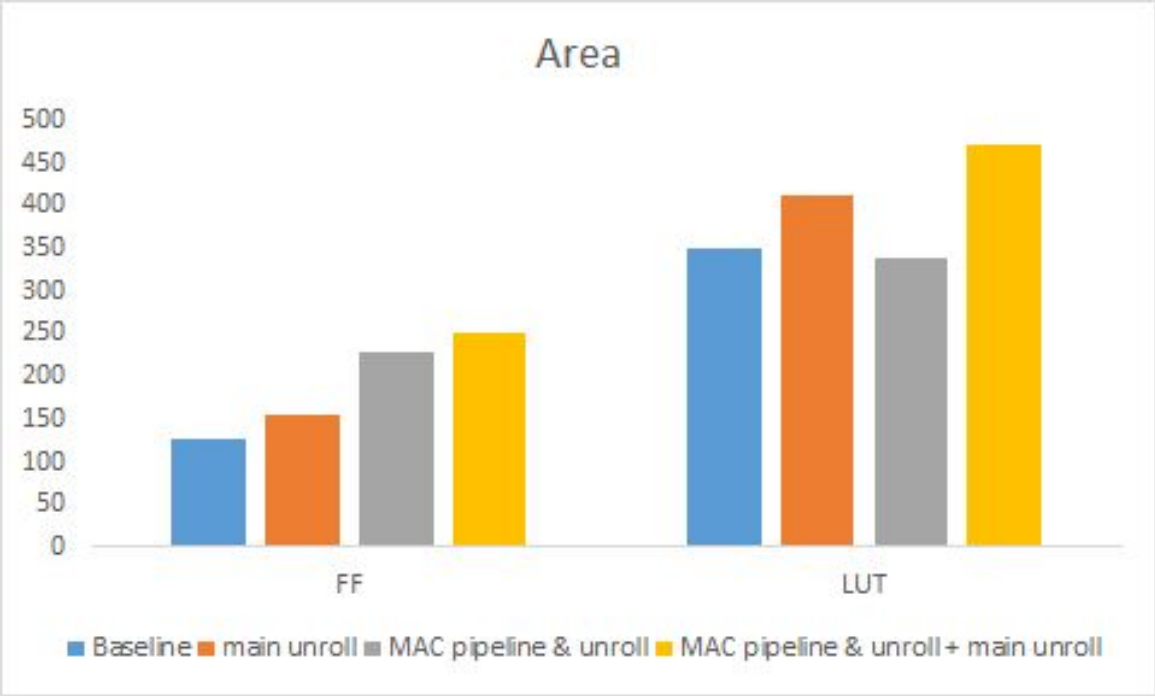
pipeline II=input rate(2)
unroll

minimise # multiplexers

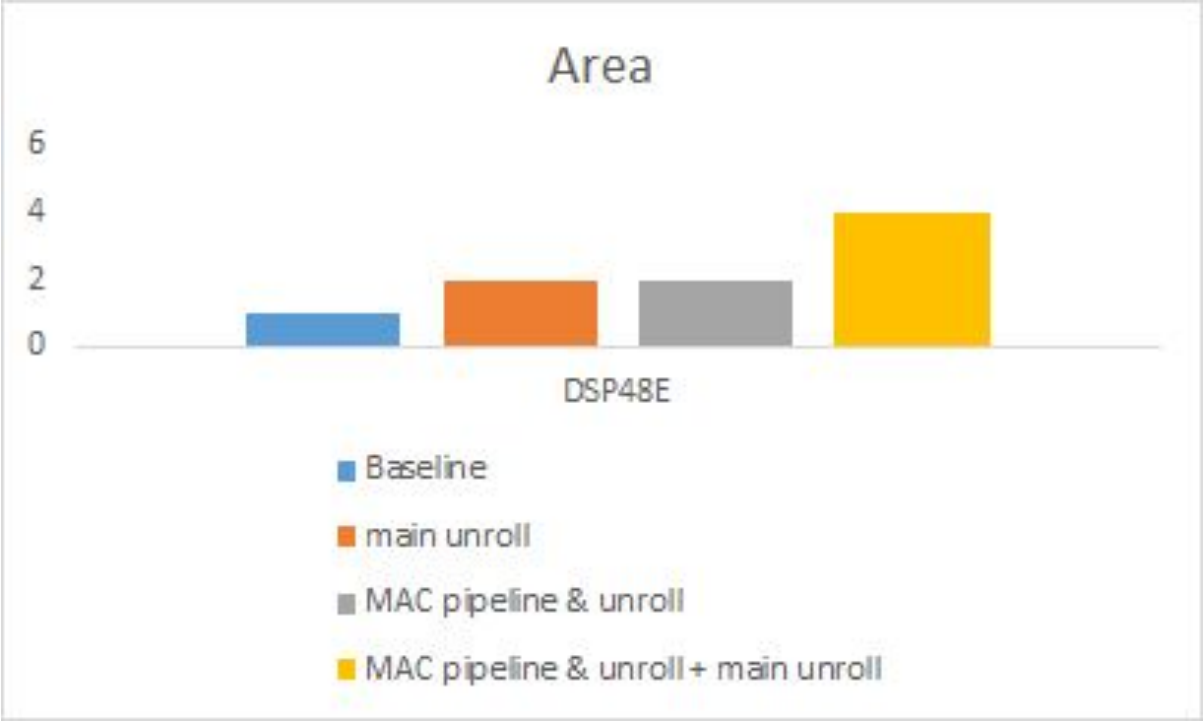
unroll

Algorithmic Interpolation





Algorithmic Interpolation



Algorithmic Interpolation

original

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	3.957 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
52	52	0.260 us	0.260 us	52	52	none

Detail

Instance

Loop

Write pipeline

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	3.957 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
36	36	0.180 us	0.180 us	36	36	none

Detail

Instance

Loop

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	4.294 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
11	11	55.000 ns	55.000 ns	11	11	none

Detail

Instance

Loop

MAC pipeline

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.702 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
16	16	0.160 us	0.160 us	16	16	none

Detail

Instance

Loop

loop unroll

Algorithmic Interpolation

Utilization Estimates

original

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	-	-	-
Instance	-	1	123	317	-
Memory	-	-			-
Multiplexer	-	-	-	33	-
Register	-	-	3	-	-
Total	0	1	126	350	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	~0	0

Utilization Estimates

Write pipeline

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	-	-	-
Instance	0	1	269	426	-
Memory	-	-			-
Multiplexer	-	-	-	33	-
Register	-	-	3	-	-
Total	0	1	272	459	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	~0	0

Utilization Estimates

MAC pipeline

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	-	-	-
Instance	0	2	224	306	-
Memory	-	-		-	-
Multiplexer	-	-	-	33	-
Register	-	-	3	-	-
Total	0	2	227	339	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	~0	0

Utilization Estimates

loop unroll

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	4	-	-	-
Expression	-	0	0	216	-
FIFO	-	-			-
Instance	-	-	5	105	
Memory	-	-	-	-	-
Multiplexer	-	-	-	152	-
Register	-	-	156	-	-
Total	0	4	161	473	0
Available	280	220	106400	53200	0
Utilization (%)	0	1	~0	~0	0

Manual Interpolation

taps N/sample rate RATE

Half the taps are always zero

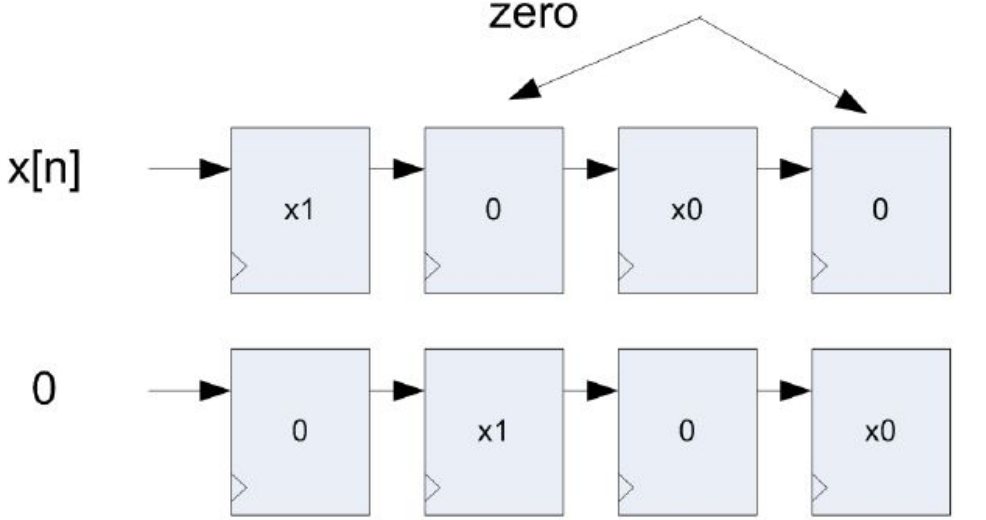


Illustration 148: Manual Interpolation

filter

MAC loop

pipeline II=input rate(2)
unroll

Phase 0:
Shift input

y[n] = h[0] * x1 + h[2] * x0

Phase 1:
Shift zero

y[n] = h[1] * x1 + h[3] * x0

c.f. algorithmic interpolation
Explicit code sharing - slightly better area?

Manual Interpolation

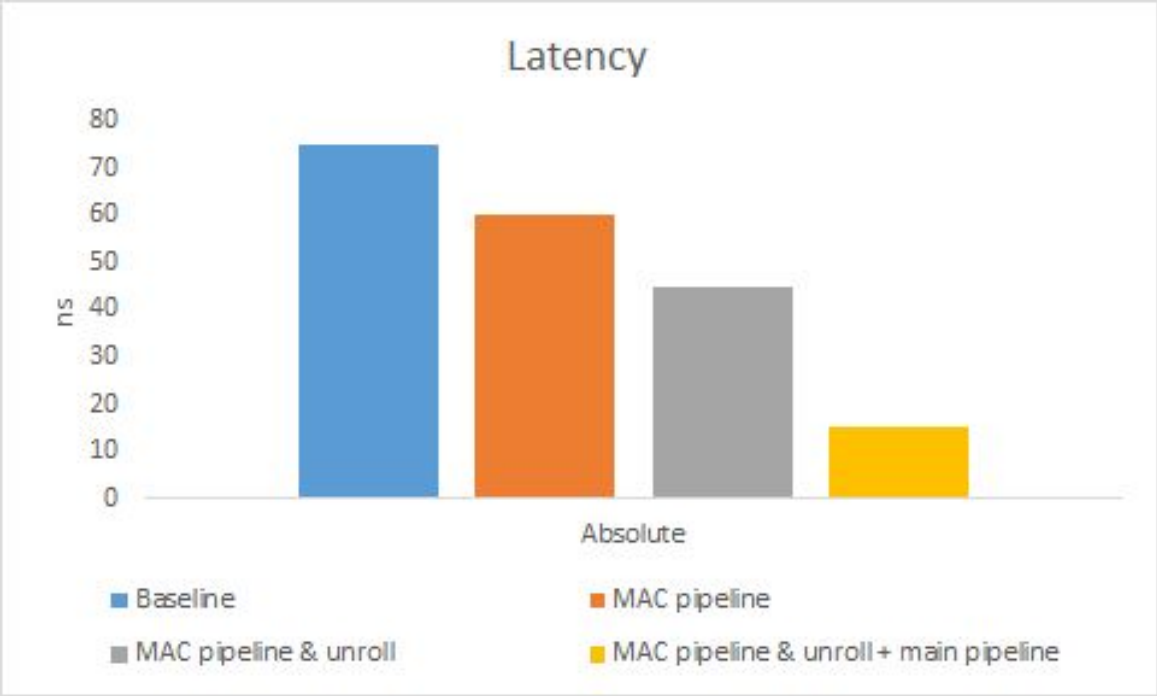
```
3   void fir_filter(streamIn_t* x,
4                   fixIn_t h[NUM_TAPS],
5                   streamOut_t* y) {
6       static shift_class<fixIn_t, NUM_TAPS> regs;
7       static fixOut_t temp;
8       static uint1_t cnt;
9       if (cnt == 0)
10          regs << x->read();
11      else
12          regs << 0;
13      MAC0:for (int i = 0; i < S_RATE; i++) {
14          temp += h[ i * S_RATE + cnt] * regs[ i * S_RATE + cnt];
15      }
16      y->write(temp);
17      temp = 0;
18      cnt++;
19  }
```

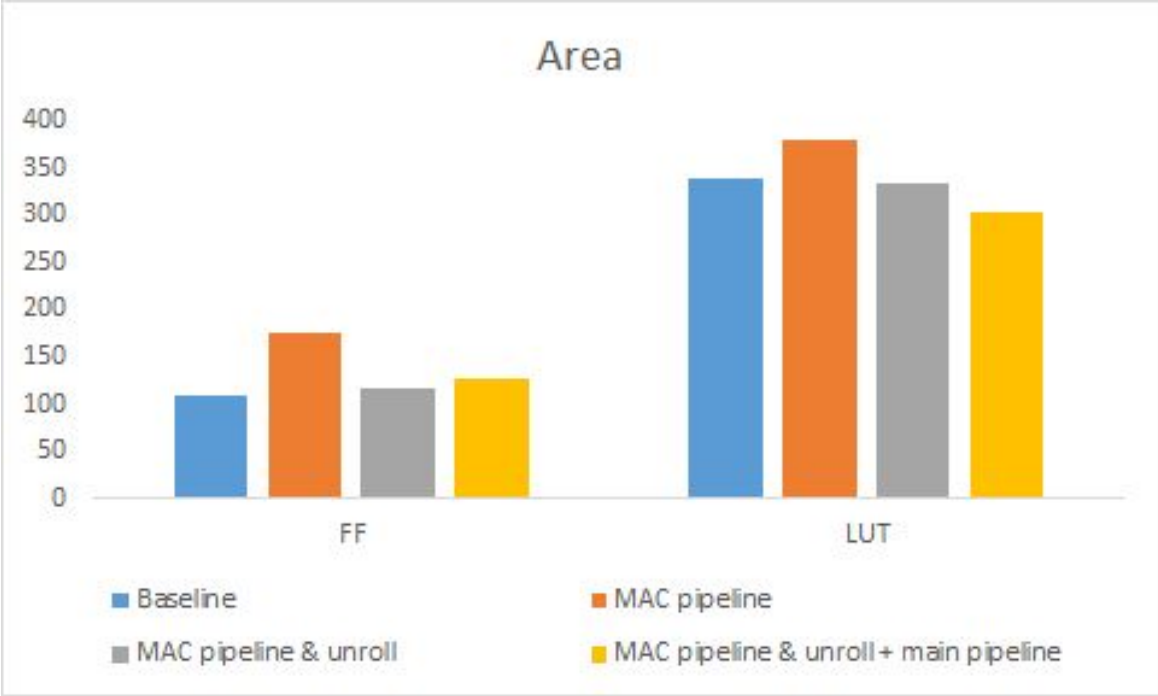
Phase 0 filter

Phase 1

MAC loop

Manual Interpolation





Manual Interpolation

Performance Estimates

original

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	4.170 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
12	12	60.000 ns	60.000 ns	12	12	none

Performance Estimates

loop unroll

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	8.599 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
8	8	80.000 ns	80.000 ns	8	8	none

Performance Estimates

MAC pipeline

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	4.170 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
15	15	75.000 ns	75.000 ns	15	15	none

Manual Interpolation

Utilization Estimates

original

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	145	-
FIFO	-	-	-	-	-
Instance	-	-	5	105	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	89	-
Register	-	-	105	-	-
Total	0	0	110	339	0
Available	280	220	106400	53200	0
Utilization (%)	0	0	~0	~0	0

Utilization Estimates

MAC pipeline

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	147	-
FIFO	-	-	-	-	-
Instance	-	-	5	105	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	96	-
Register	0	-	170	32	-
Total	0	0	175	380	0
Available	280	220	106400	53200	0
Utilization (%)	0	0	~0	~0	0

Utilization Estimates

loop unroll

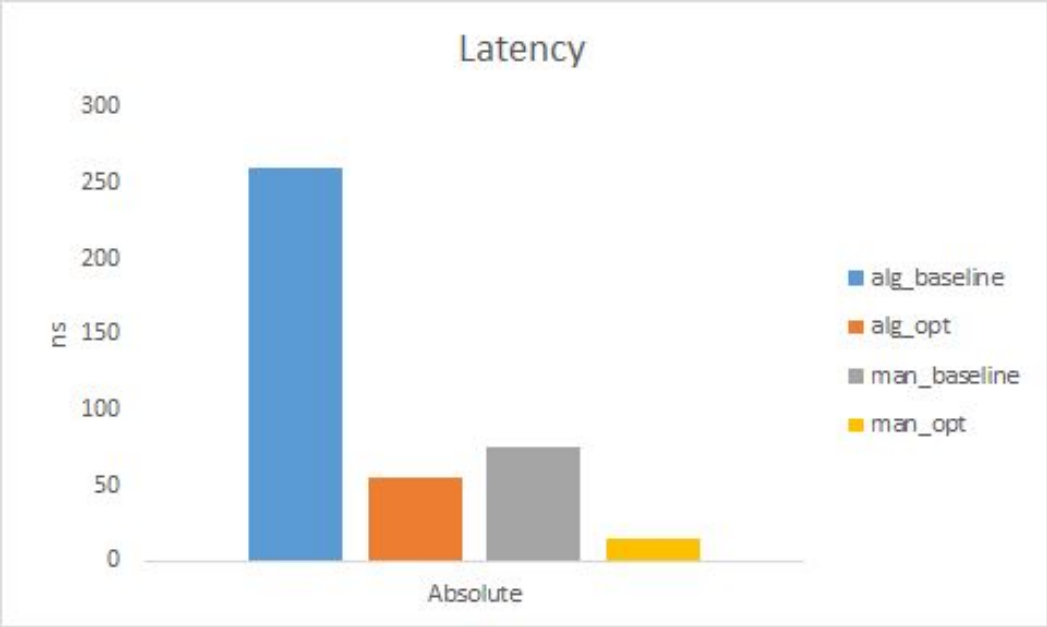
Summary

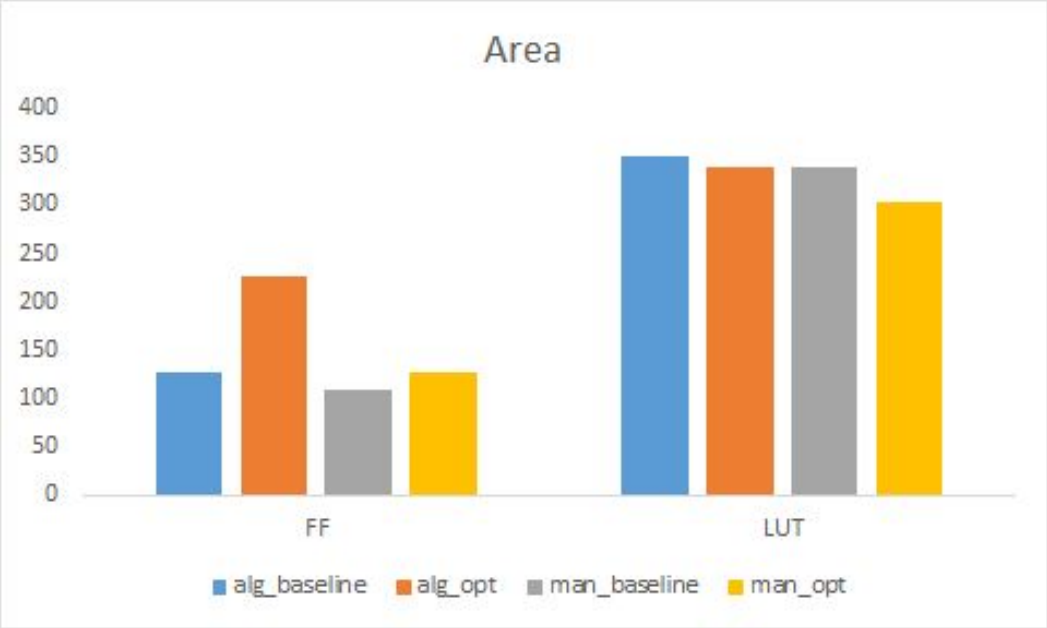
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	162	-
FIFO	-	-	-	-	-
Instance	-	-	5	105	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	60	-
Register	-	-	64	-	-
Total	0	0	69	327	0
Available	280	220	106400	53200	0
Utilization (%)	0	0	~0	~0	0

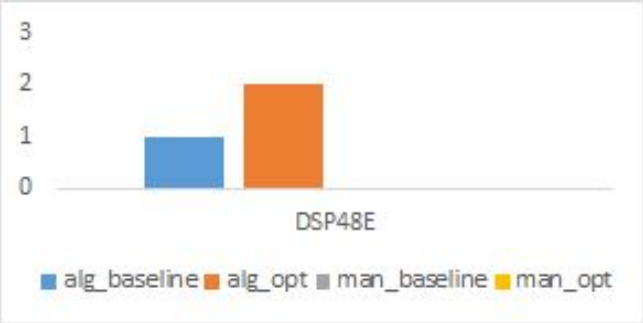
Algorithmic vs. Manual Interpolation: Constraints

- All IO mapped to wire enable interfaces
- All arrays mapped to registers
- Main loop pipelined with II=input rate, input rate = 2 (**algorithmic**)
- All loops (**algorithmic**) / MAC loop (**manual**) fully unrolled

Algorithmic vs. Manual Interpolation







Algorithmic vs. Manual Interpolation

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	4.294 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
11	11	55.000 ns	55.000 ns	11	11	none

Detail

- Instance
- Loop

Performance Estimates

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	4.170 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
12	12	60.000 ns	60.000 ns	12	12	none

Detail

- Instance
- Loop

Algorithmic vs. Manual Interpolation

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	-	-	-
FIFO	-	-	-	-	-
Instance	0	2	224	306	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	33	-
Register	-	-	3	-	-
Total	0	2	227	339	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	~0	0

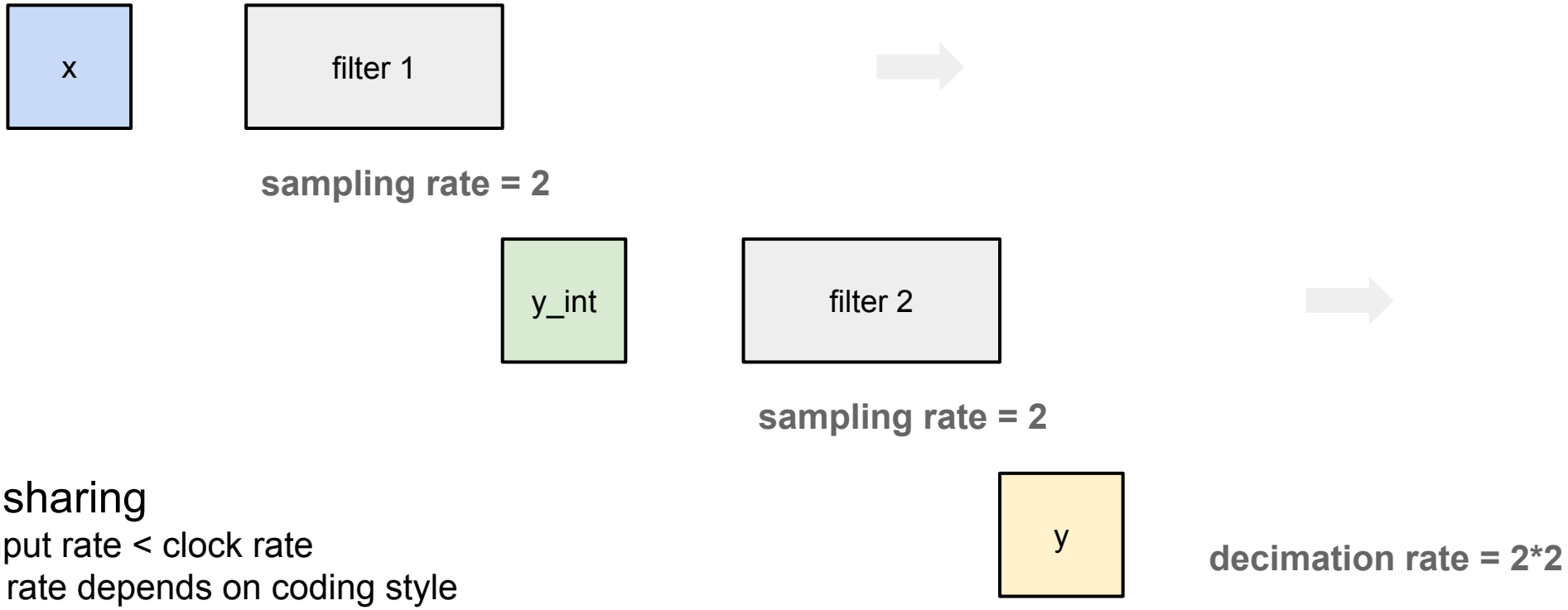
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	0	0	147	-
FIFO	-	-	-	-	-
Instance	-	-	5	105	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	96	-
Register	0	-	170	32	-
Total	0	0	175	380	0
Available	280	220	106400	53200	0
Utilization (%)	0	0	~0	~0	0

10.6 Multi-stage Decimation

Multi-stage Decimation



- Resource sharing
  - when input rate < clock rate
  - sharing rate depends on coding style

Multi-block Decimation

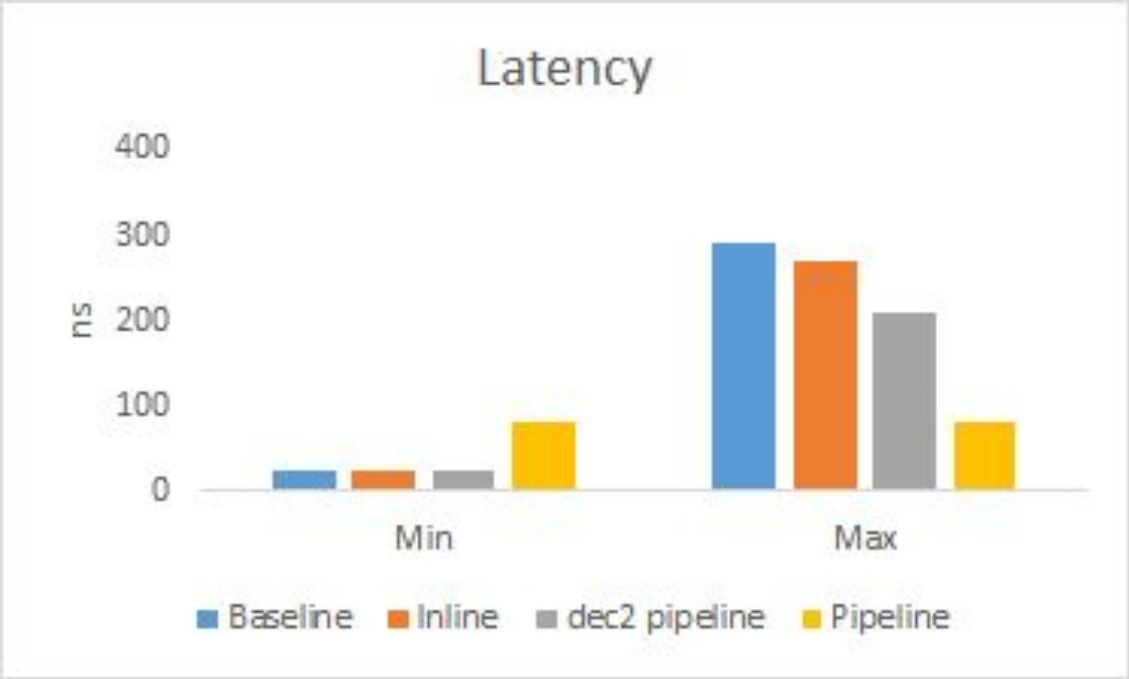
- When no resource sharing between cascading decimation filters is needed

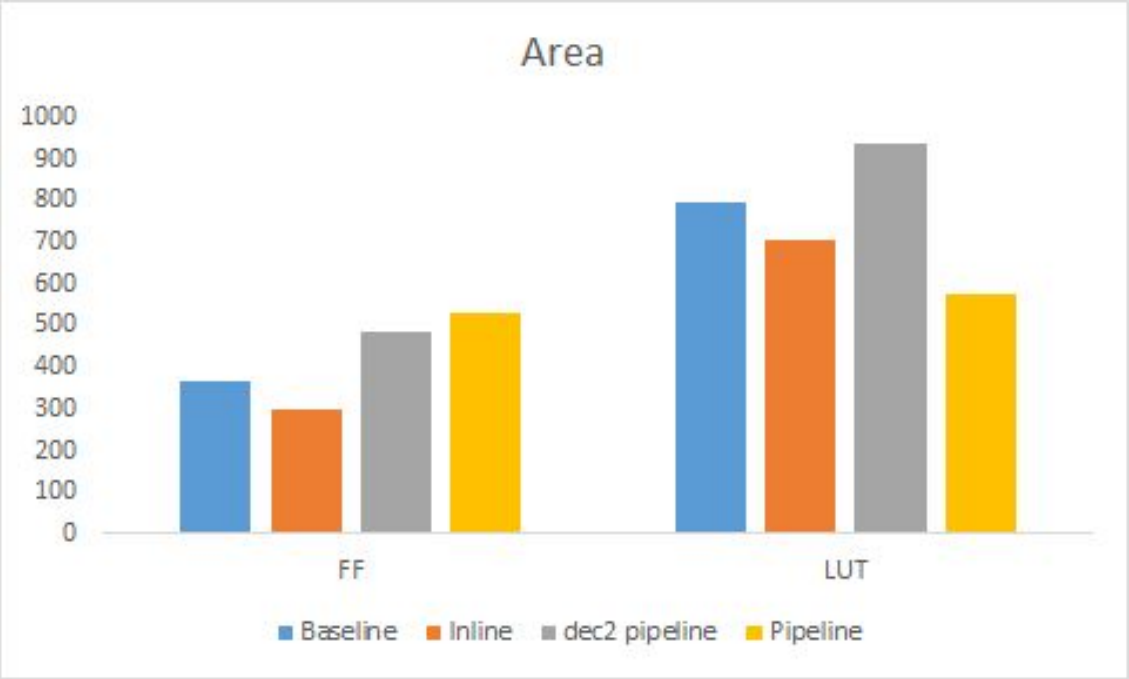
```
void dec2_2stage(streamIn_t &x,
                 fixIn_t h[NUM_TAPS],
                 streamOut_t &y) {
    static streamMid_t y_int;
    //#pragma HLS DATAFLOW
    #pragma HLS stream variable=x depth=2
    #pragma HLS stream variable=y_int depth=2
    #pragma HLS stream variable=y depth=2

    static bool f0_vld_in, f0_vld_out, f1_vld_out;
    f0_vld_in = true;
    BLOCK0:dec<0,8,1,8,1,NUM_TAPS,S_RATE>(x,h,y_int,f0_vld_in,f0_vld_out);
    BLOCK1:dec<1,18,4,8,1,NUM_TAPS,S_RATE>(y_int,h,y,f0_vld_out,f1_vld_out);
```

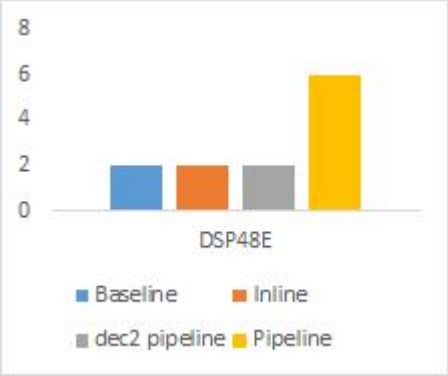
BLOCK0 pipeline II=4

BLOCK1 pipeline II=2

Multi-block Decimation







Single-block Decimation

- Can take advantage of decreasing rate of computation at each stage

```
void dec2_2stage(fixIn_t &x,
                 fixIn_t h[NUM_TAPS],
                 fixOut_t &y) {
    static fixMid_t y0_int = 0;
    static uint2_t cnt;
    uint2_t sel;
    if ( !cnt[0] ) {//sel for cnt==0 and cnt==2
        sel = 0;

    } else if( cnt == 3 ) {
        sel = 1;
    } else {
        sel = 2;
    }
    switch(sel){
        case 0:
            dec2<0>(x,h,y0_int);//read x every 4 clocks with II=2
            break;
        case 1:
            dec2<1>(y0_int,h,y);
            break;
        default:
            break;
    }
    cnt++;
}
```

filter0

filter1

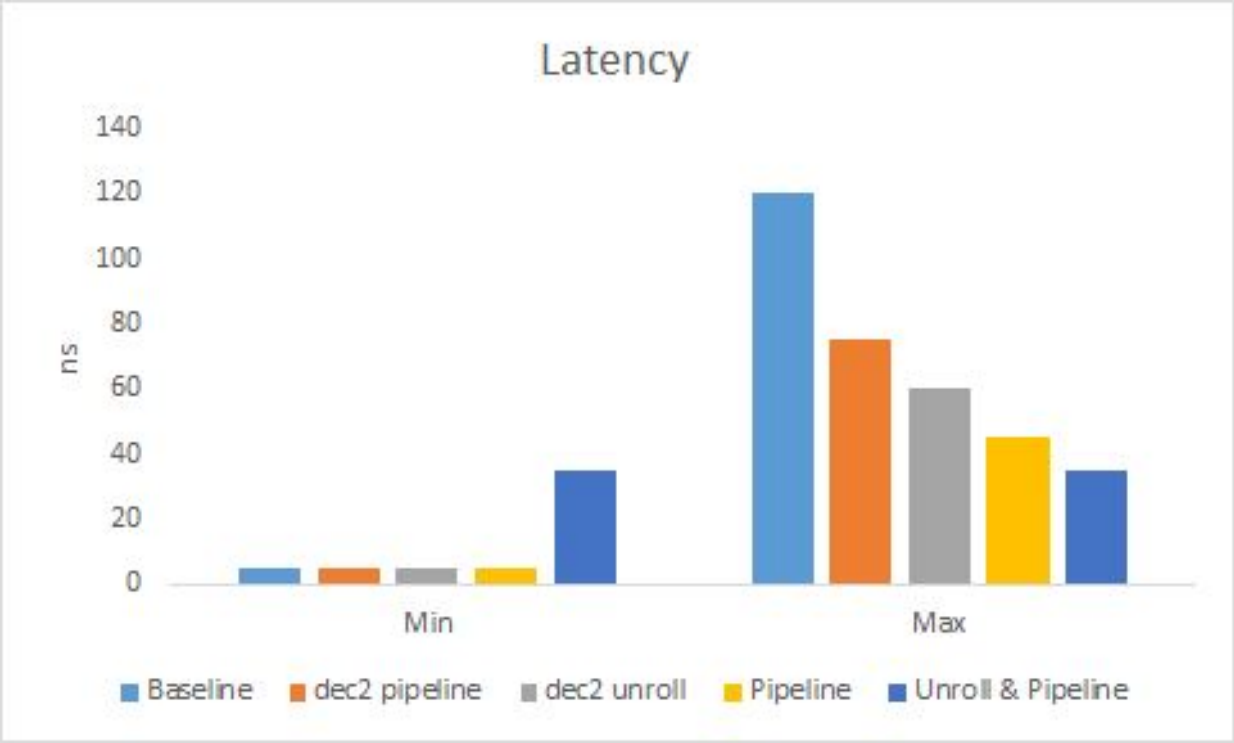
pipeline II=2, read rate = 4

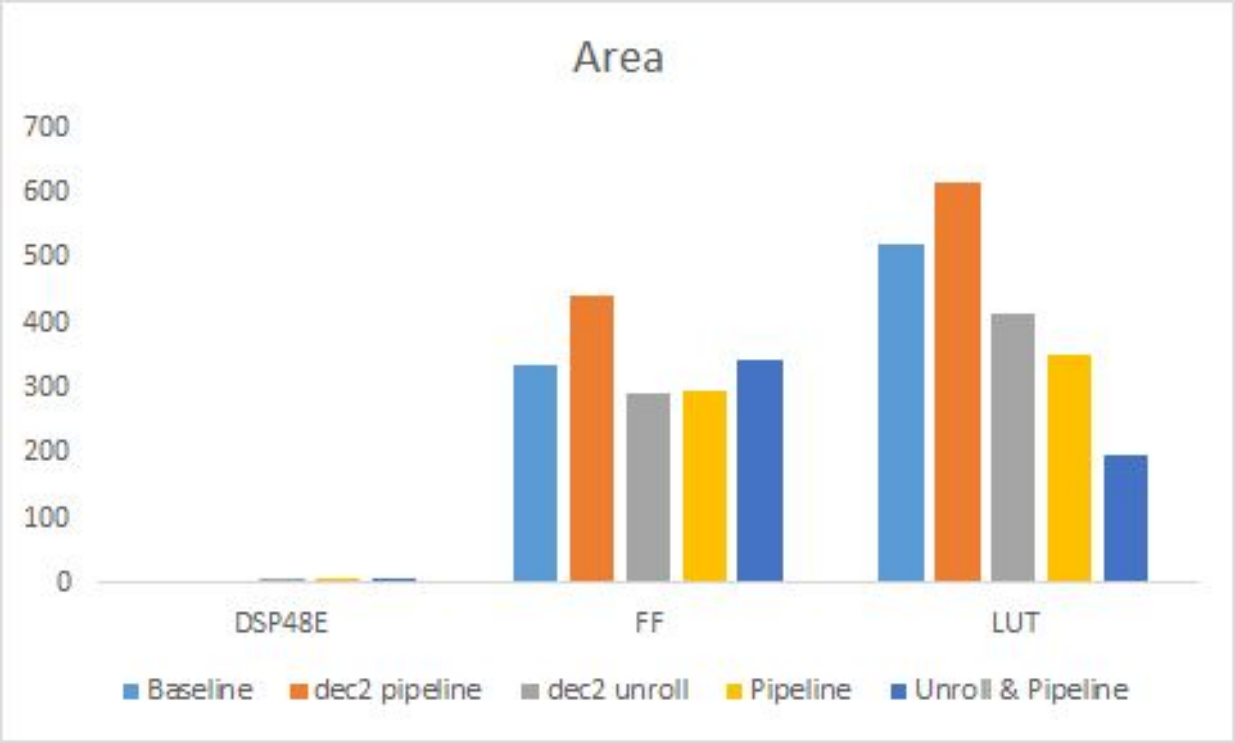
- in mutually exclusive conditions
  - resource can be shared

pipeline II=2, read rate = 8

output rate = 16

Algorithmic Single-block Decimation





Single-block Decimation: Pipeline

Performance Estimates — original

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	3.455 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
24	24	0.120 us	0.120 us	24	24	none

Detail

Instance

Loop

Performance Estimates — MAC pipeline

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	5.00 ns	3.744 ns	0.62 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
15	15	75.000 ns	75.000 ns	15	15	none

Detail

Instance

Loop

Performance Estimates — loop unroll

Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00 ns	9.400 ns	1.25 ns

Latency

Summary

Latency (cycles)		Latency (absolute)		Interval (cycles)		
min	max	min	max	min	max	Type
1	9	10.000 ns	90.000 ns	1	9	none

Detail

Instance

Loop

Single-block Decimation: Pipeline

Utilization Estimates

Summary

original

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	2	-	-	-
Expression	-	-	0	170	-
FIFO	-	-	-	-	-
Instance	-	-	10	230	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	107	-
Register	-	-	325	-	-
Total	0	2	335	507	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	~0	0

Utilization Estimates

Summary

MAC pipeline

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	2	-	-	-
Expression	-	-	0	178	-
FIFO	-	-	-	-	-
Instance	-	-	10	230	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	131	-
Register	0	-	432	64	-
Total	0	2	442	603	0
Available	280	220	106400	53200	0
Utilization (%)	0	~0	~0	1	0

Utilization Estimates

Summary

loop unroll

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	7	-	-	-
Expression	-	0	0	68	-
FIFO	-	-	-	-	-
Instance	-	-	10	230	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	107	-
Register	-	-	210	-	-
Total	0	7	220	405	0
Available	280	220	106400	53200	0
Utilization (%)	0	3	~0	~0	0

Manual Single-block Decimation

- Can take advantage of decreasing rate of computation at each stage

```
18      if(!phase_cnt[0])//if even counts
19          sel_phase = 0;
20      else if (phase_cnt % 4 == 3)//if every 4th odd count
21          sel_phase = 1;
22      else
23          sel_phase = 3;//do nothing
24
25
26      if ( phase_cnt % 4 == 0 ){//read at rate of 4
27          x_int = x->read();
28          f0_vld_in = true;
29      } else
30          f0_vld_in = false;
31      switch(sel_phase){
32          case 0:
33              f0.exec(x_int,h,y0_int,f0_vld_in,f0_vld_out);
34              break;
35          case 1:
36              f1.exec(y0_int,h,y1_int,f0_vld_out, f1_vld_out);
37              if(f1_vld_out) {
38                  y->write(y1_int);
39              }
40              break;
41          default:
42              break;
43      }
44      phase_cnt++;
```

phase_cnt is used to count all phases (16 in total) coarse grain counter

filter0

filter1

- in mutually exclusive conditions
  - resource can be shared

read input at rate 4:
no need be larger than
f0 read rate * sampling rate

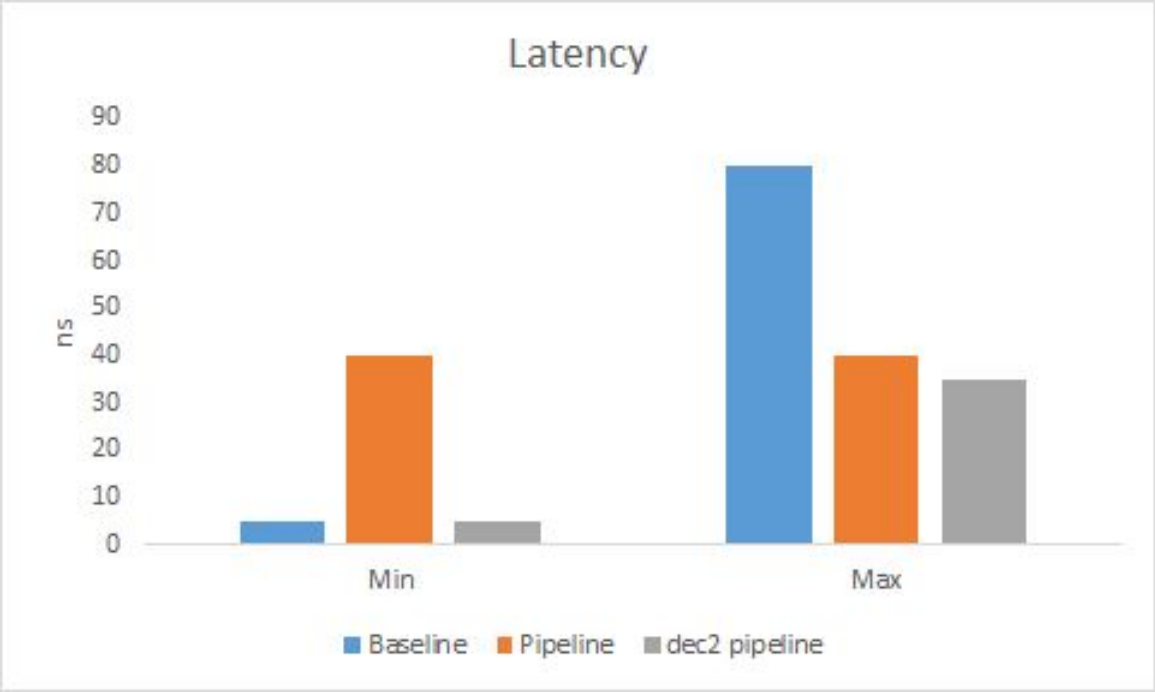
```
18      if(!phase_cnt[0])//if even counts
19          sel_phase = 0;
20      else if (phase_cnt % 4 == 3)//if every 4th odd count
21          sel_phase = 1;
22      else
23          sel_phase = 3;//do nothing
24
25
26      if ( phase_cnt % 4 == 0 ){//read at rate of 4
27          x_int = x->read();
28          f0_vld_in = true;
29      } else
30          f0_vld_in = false;
31      switch(sel_phase){
32          case 0:
33              f0.exec(x_int,h,y0_int,f0_vld_in,f0_vld_out);
34              break;
35          case 1:
36              f1.exec(y0_int,h,y1_int,f0_vld_out, f1_vld_out);
37              if(f1_vld_out) {
38                  y->write(y1_int);
39              }
40              break;
41          default:
42              break;
43      }
44      phase_cnt++;
```

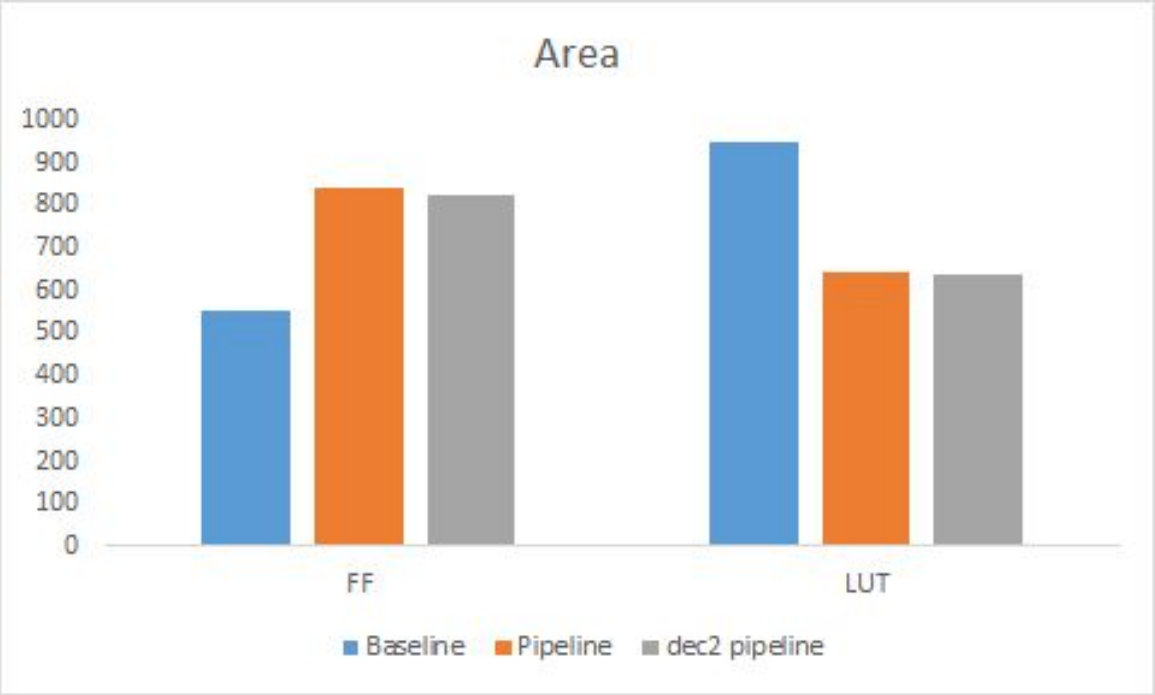
filter0

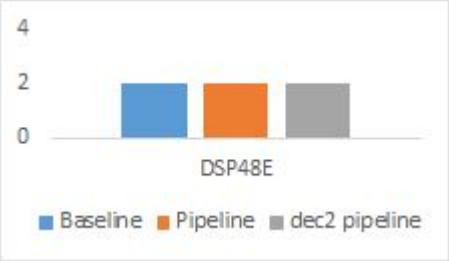
filter1

cnt	read input	f0 call	f0 output	f1 call	f1 output
0000	x	x			
0001	read rate = 2				
0010		x		write rate = 4	
0011				x	x
0100	x	x			
0101		write rate = 4			
0110		x	x		
0111				x	
1000	x	x	read rate = 2		
1001					
1010		x			
1011				x	
1100	x	x			
1101					
1110		x	x		
1111				x	

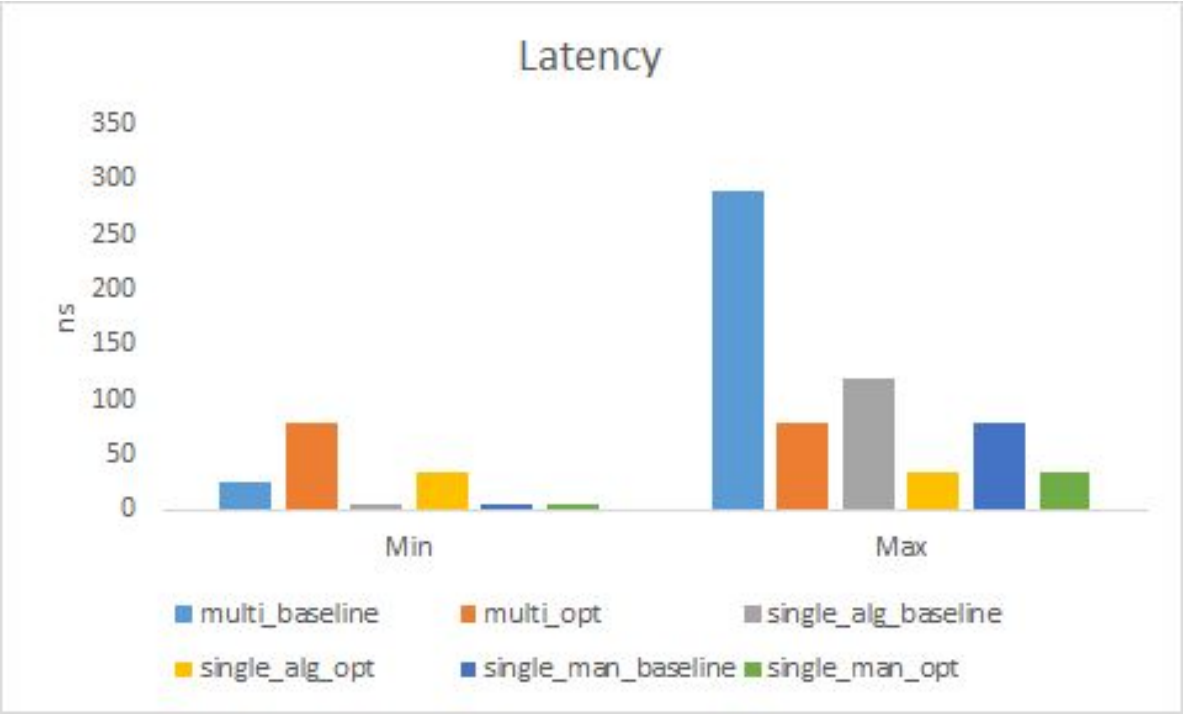
Manual Single-block Decimation

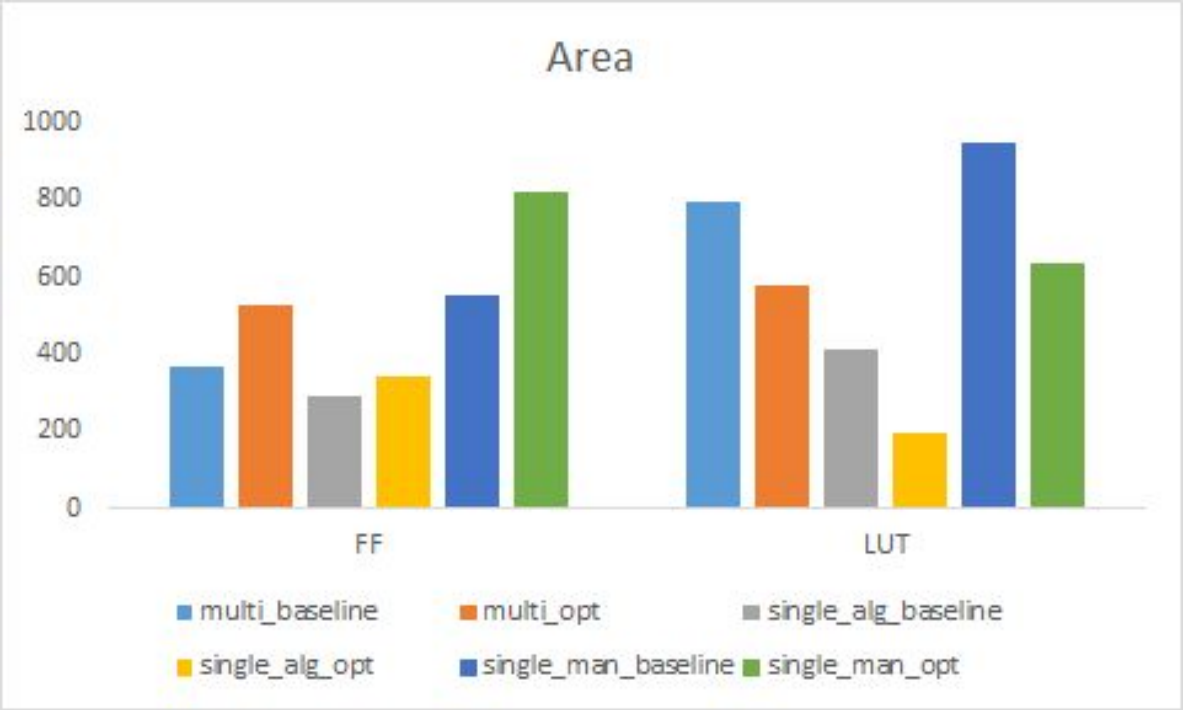


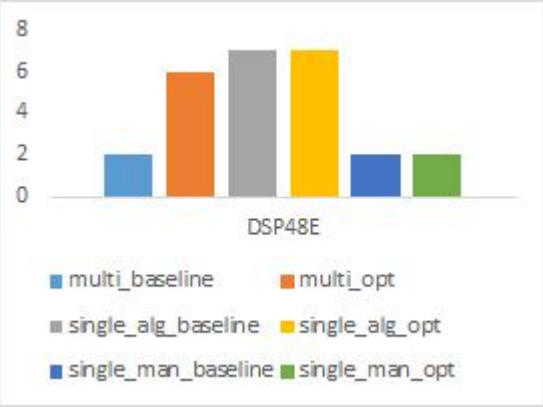




Multi-stage Decimation Comparison







https://github.com/yuweitt/HLS-CodingStyle-FIR_FILTER