GeMerP

Hardware Software Co-Design LU

Gerald Lampl, 0728579
Patrick Knöbel, 0925219
Markus Hennebach, 0926654

January 29, 2013

Outline

FFT partitioning

Denote the signal with s and with h the impulse response. Take a block size N used for the FFT and partition the signal and the impulse response, the resulting sets of partitions are S for the signal and I for the impulse response.

$$(s * h) = \sum_{\substack{i \in I \\ j \in S}} (s_i * h_j) = \sum_{j \in S} \sum_{i \in I} \left(s_{|I| + j\frac{N}{2} - i} * h_i \right) \tag{1}$$

FFT optimization

Let sr be the right channel and sl be the left channel of the signal.

$$s[n] = sr[n] + jsl[n] \tag{2}$$

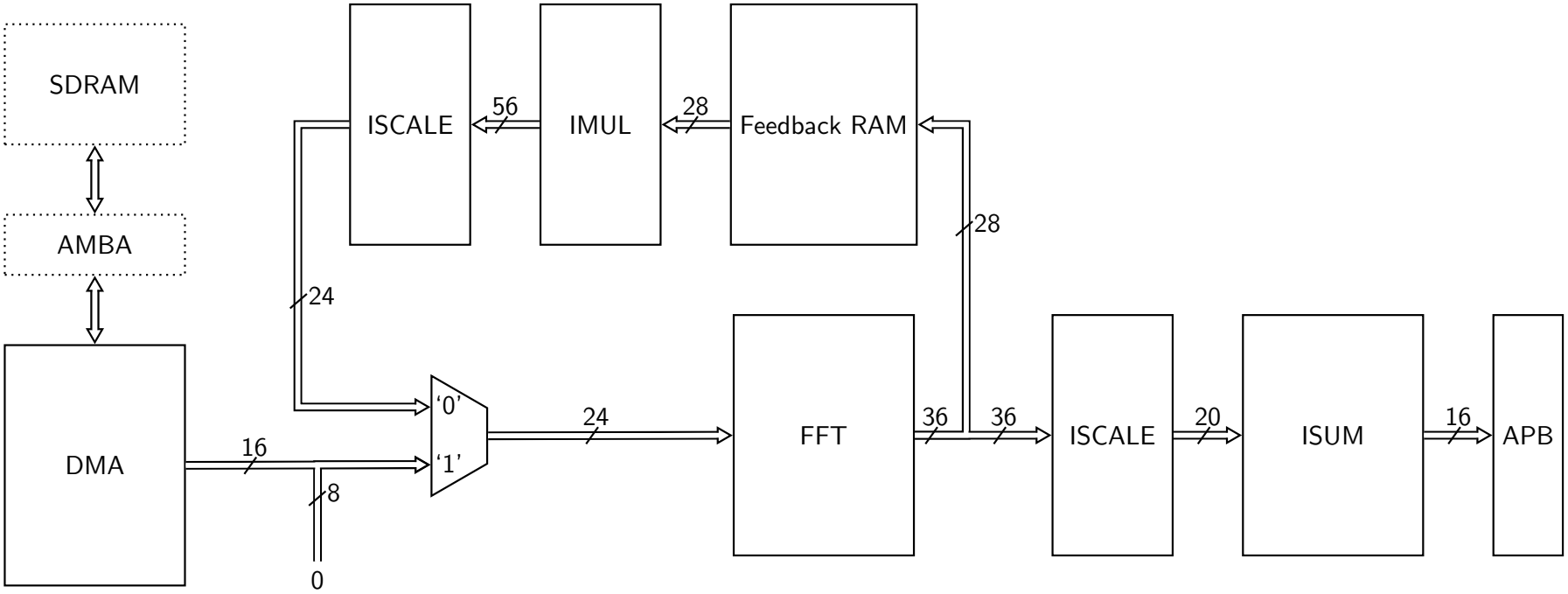
The channels can be obtained with

$$sr[n] = (s[n] + s^*[n])/2 \tag{3}$$
$$sl[n] = (s[n] - s^*[n])/(2j). \tag{4}$$

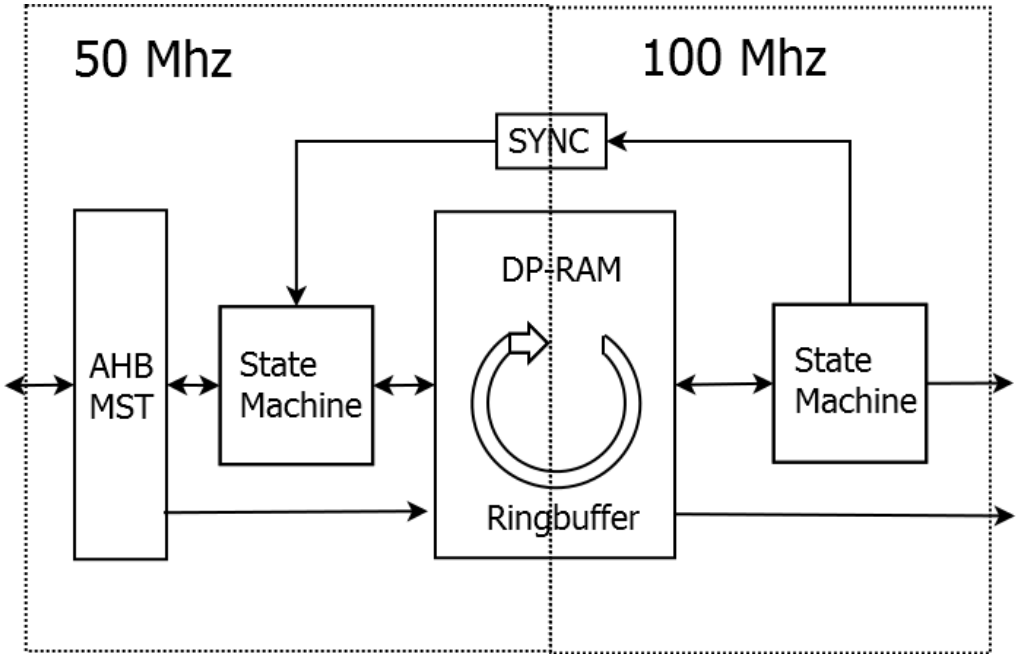
Note the relation $FFT_N(k, z^*) = FFT_N(N-k, z)^*$ which gives

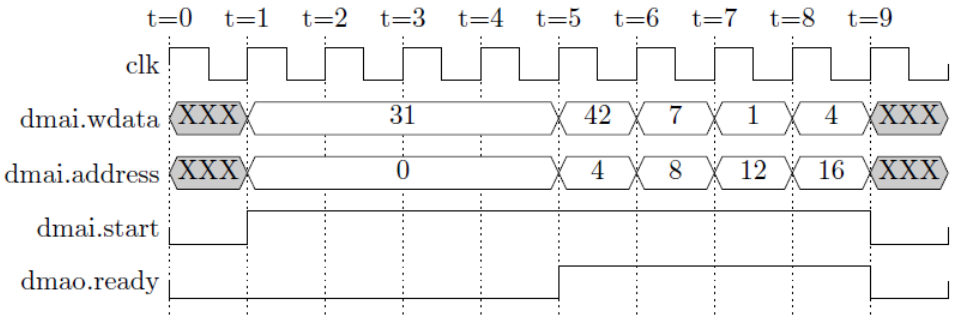
$$FFT_N(k, sr) = (FFT_N(k, s) + FFT_N(N-k, s)^*)/2 \tag{5}$$
$$FFT_N(k, sl) = (FFT_N(k, s) - FFT_N(N-k, s)^*)/(2j). \tag{6}$$

Architectural & Behaviourial Analysis

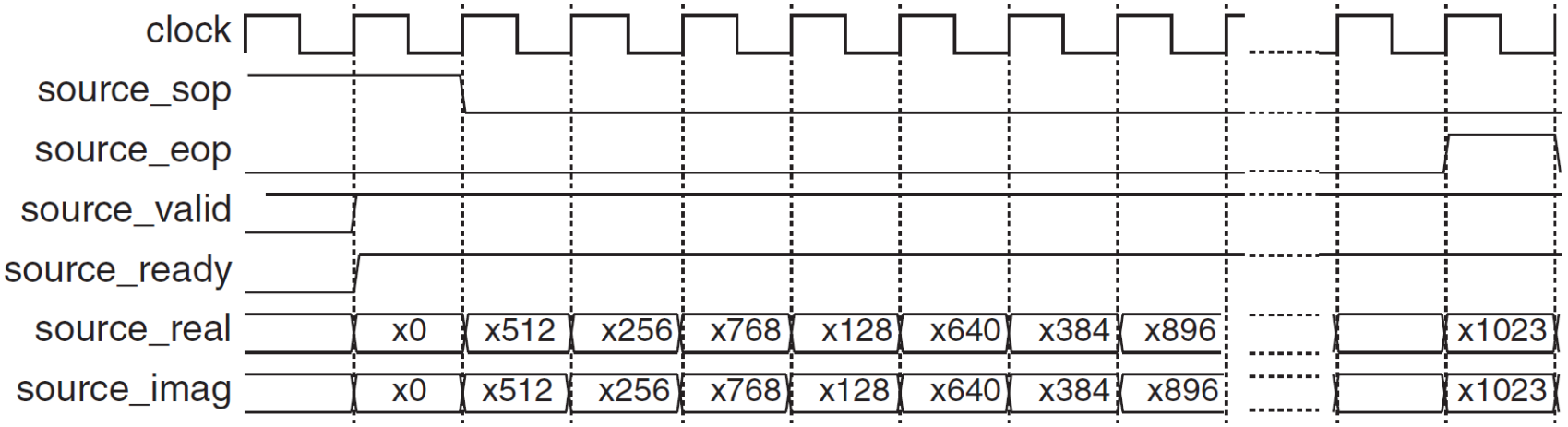


DMA

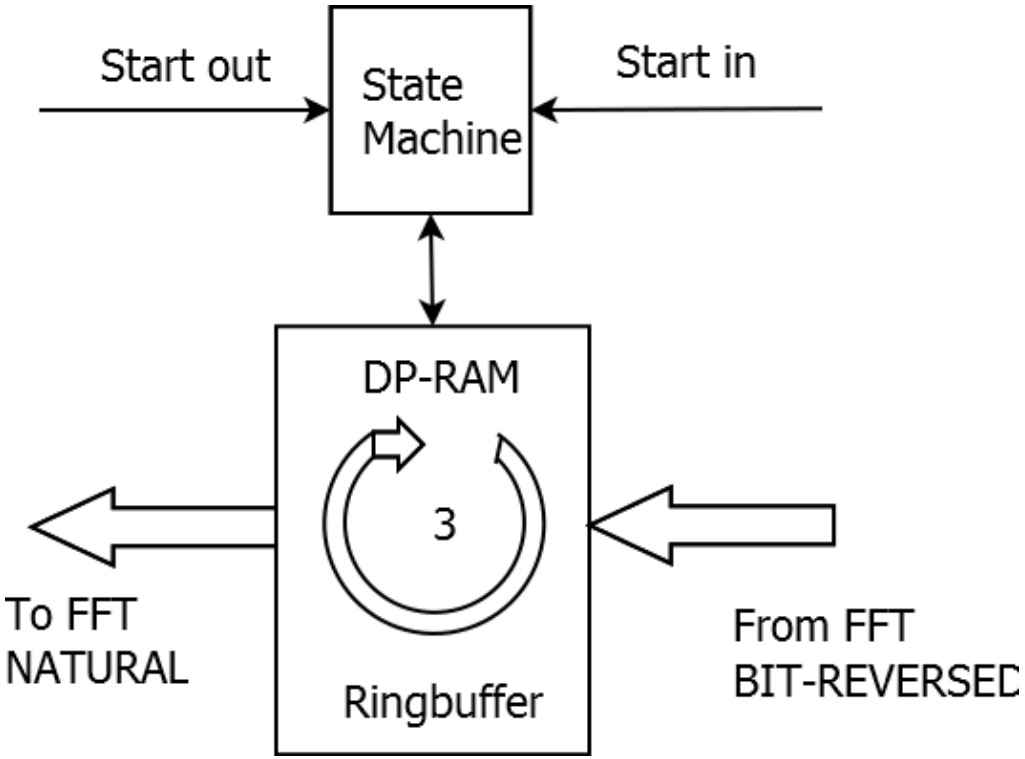

50 Mhz
100 Mhz
SYNC
AHB MST
State Machine
DP-RAM
Ringbuffer
State Machine


t=0 t=1 t=2 t=3 t=4 t=5 t=6 t=7 t=8 t=9
clk
dmai.wdata
XXX 31 42 7 1 4 XXX
dmai.address
XXX 0 4 8 12 16 XXX
dmai.start
dmao.ready

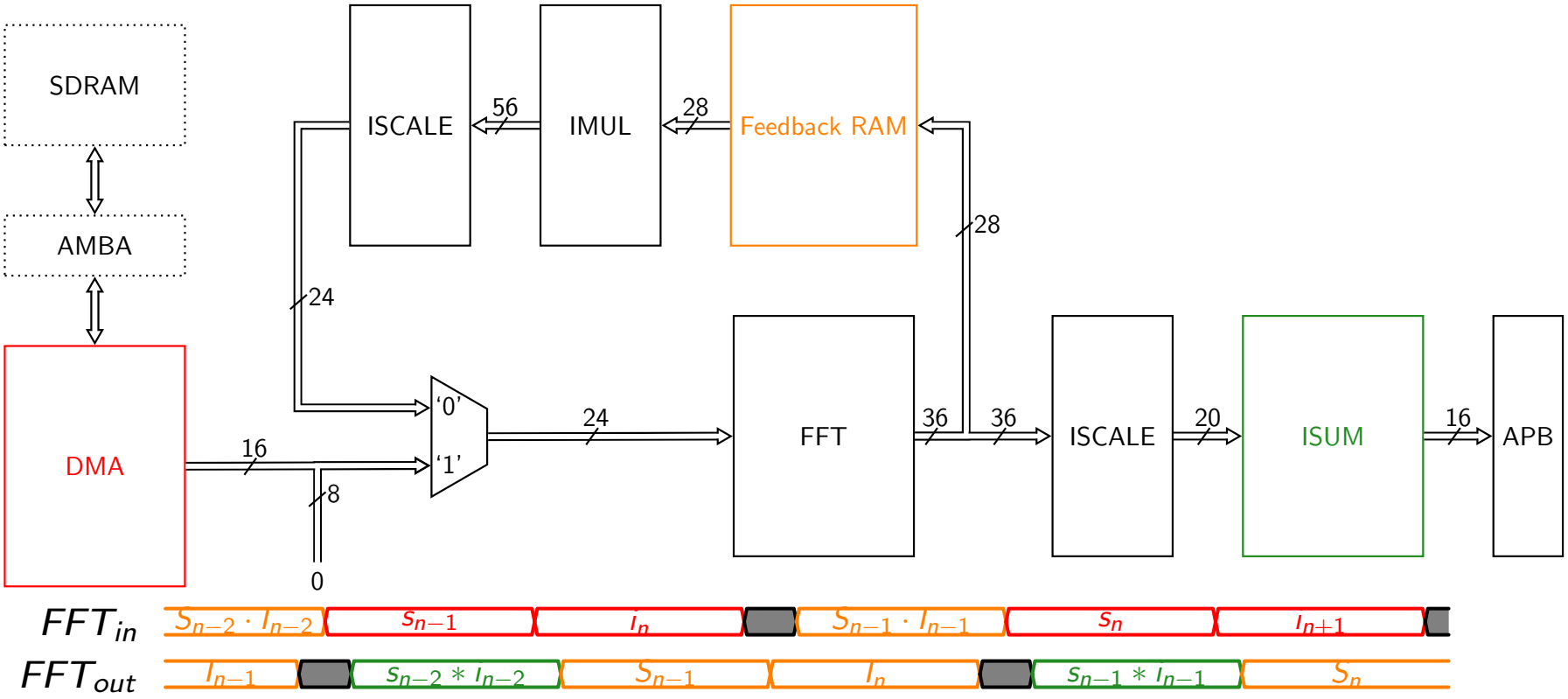
FFT


clock
source_sop
source_eop
source_valid
source_ready
source_real
source_imag
x0
x512
x256
x768
x128
x640
x384
x896
x1023

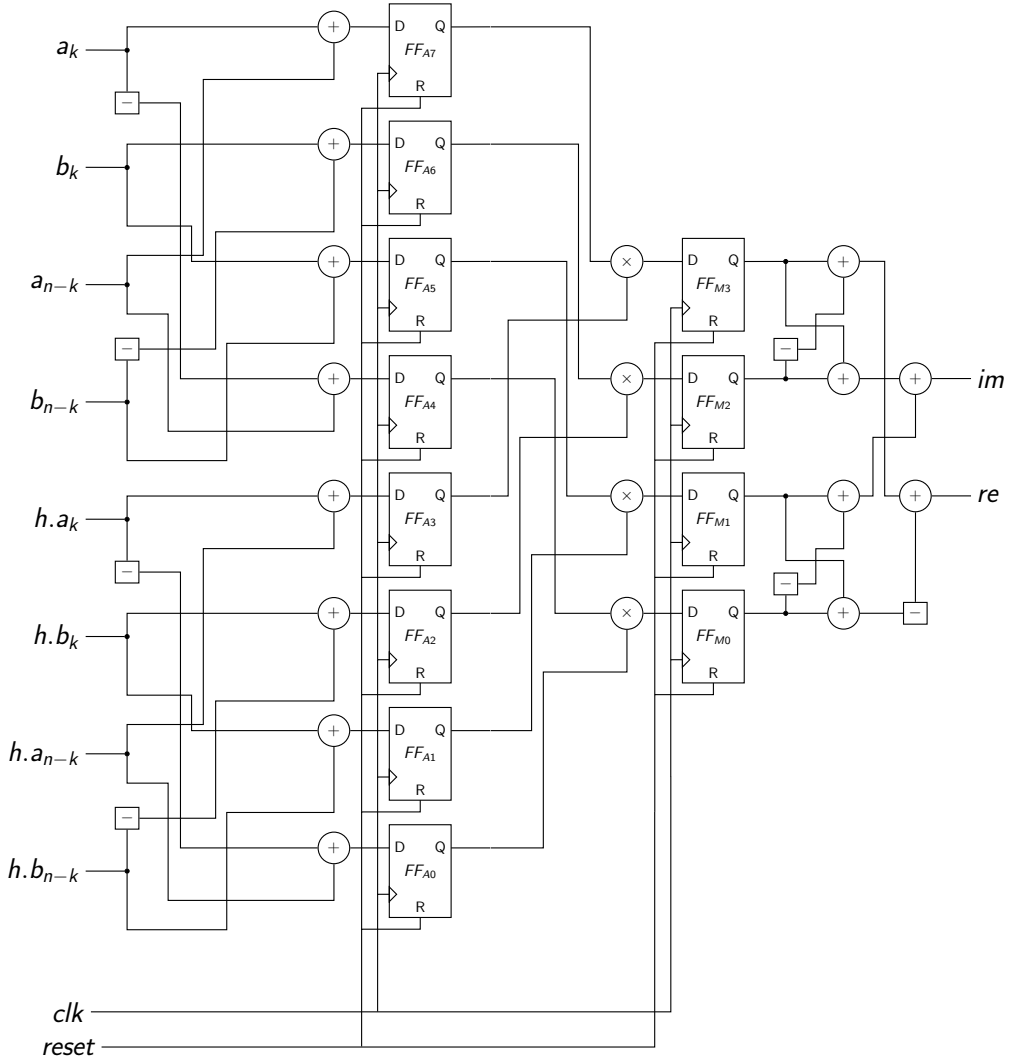
Feedback RAM



Feedback RAM



IMUL



$$A_7 = a_k + a_{n-k}$$
$$A_6 = h.a_k + h.a_{n-k}$$
$$A_5 = b_k + \overline{b_{n-k}}$$
$$A_4 = h.b_k + \overline{h.b_{n-k}}$$
$$A_3 = b_k + b_{n-k}$$
$$A_2 = h.b_k + h.b_{n-k}$$
$$A_1 = \overline{a_k} + a_{n-k}$$
$$A_0 = \overline{a_k} + a_{n-k}$$

$$M_3 = A_7 \cdot A_3$$
$$M_2 = A_6 \cdot A_2$$
$$M_1 = A_5 \cdot A_1$$
$$M_0 = A_4 \cdot A_0$$

$$im = M_3 + M_2 + M_1 + \overline{M_0}$$
$$re = M_3 + \overline{M_2} + M_1 + M_0$$

ISCALE

```
x← resize data_in to 2^(FACTOR_WIDTH)+DATA_WIDTH
for i←DATA_WIDTH →2^(FACTOR_WIDTH)+DATA_WIDTH-1 do
    if i>factor+DATA_WIDTH-2 then
        z(i-DATA_WIDTH)←x(i)
    else
        z(i-DATA_WIDTH)←x(2^(FACTOR_WIDTH)+DATA_WIDTH-1)
    end if
end for
for i←1→2^(FACTOR_WIDTH) − 1 do
    if z(i)!=z(i-1) then
return ←x(2^(FACTOR_WIDTH)+DATA_WIDTH-1)&(0→DATA_WIDTH-
2←not(x(2^(FACTOR_WIDTH)+DATA_WIDTH-1)))
    end if
end for
y(DATA_WIDTH-1)←x(2^(FACTOR_WIDTH)+DATA_WIDTH-1)
for i←0→DATA_WIDTH-2 do
    y(i)←x(i+factor)
end for
return y
```

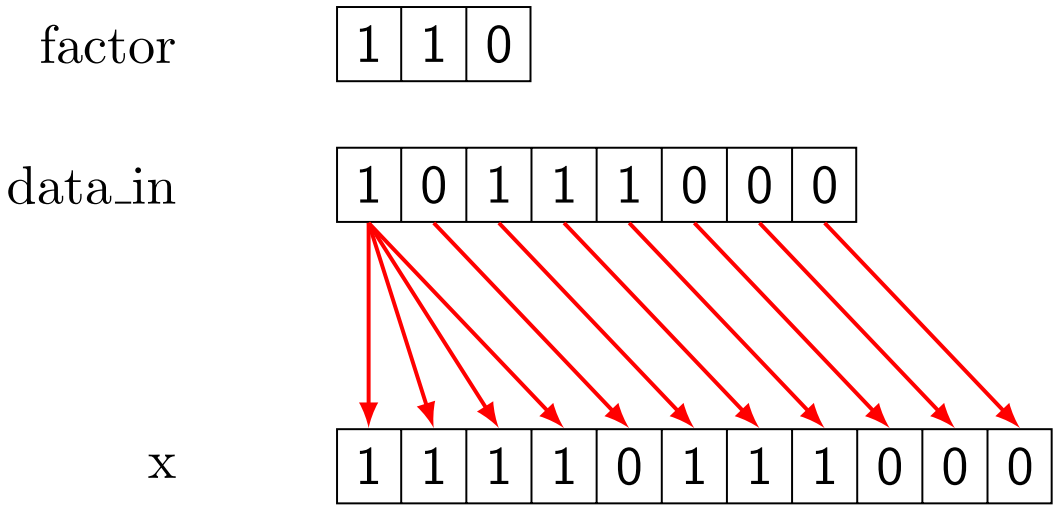
ISCALE

factor	1	1	0					
data_in	1	0	1	1	1	0	0	0

ISCALE

factor	1	1	0								
data_in	1	0	1	1	1	0	0	0			
x											

ISCALE



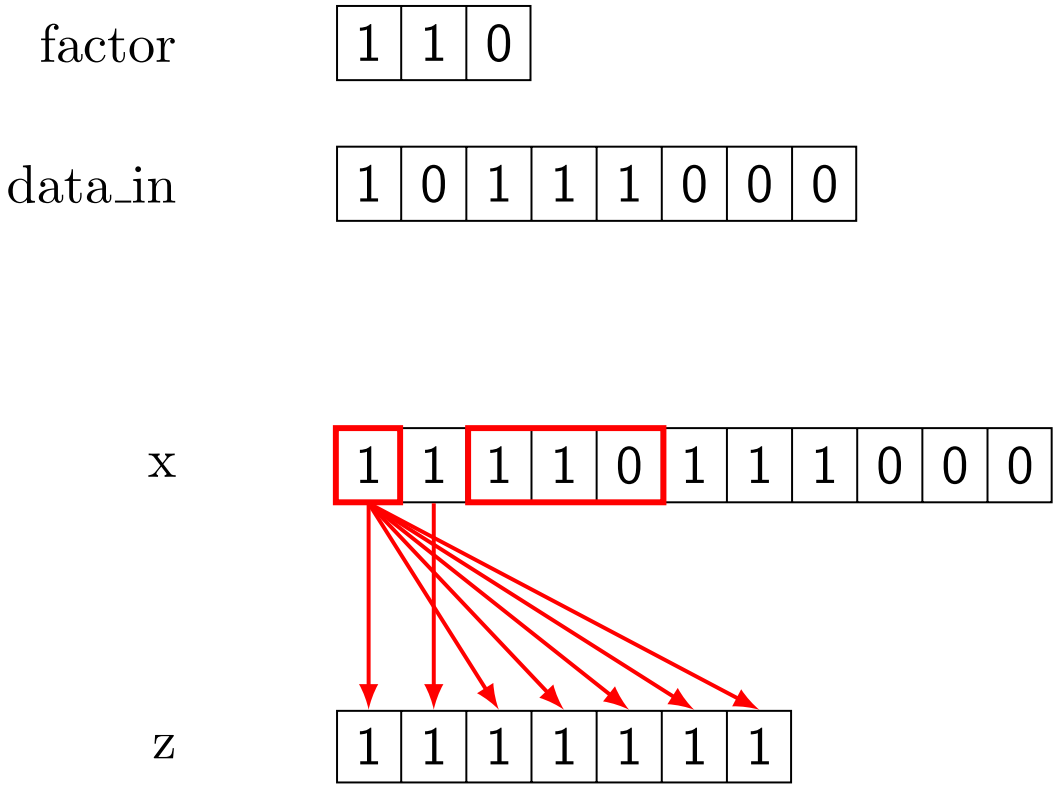
ISCALE

factor	1	1	0								
data_in	1	0	1	1	1	0	0	0			
x	1	1	1	1	0	1	1	1	0	0	0

ISCALE

factor	1	1	0								
data_in	1	0	1	1	1	0	0	0			
x	1	1	1	1	0	1	1	1	0	0	0
z											

ISCALE



ISCALE

factor	1	1	0

data_in	1	0	1	1	1	0	0	0

x	1	1	1	1	0	1	1	1	0	0	0

z	1	1	1	1	1	1	1

ISCALE

factor	1	1	0								
data_in	1	0	1	1	1	0	0	0			
x	1	1	1	1	0	1	1	1	0	0	0
z	1	1	1	1	1	1	1				

ISCALE

factor	1	1	0								
data_in	1	0	1	1	1	0	0	0			
x	1	1	1	1	0	1	1	1	0	0	0
z	1	1	1	1	1	1	1				

ISCALE

factor	1	1	0

data_in	1	0	1	1	1	0	0	0

x	1	1	1	1	0	1	1	1	0	0	0

z	1	1	1	1	1	1	1

ISCALE

factor	1	1	0								
data_in	1	0	1	1	1	0	0	0			
x	1	1	1	1	0	1	1	1	0	0	0
z	1	1	1	1	1	1	1				

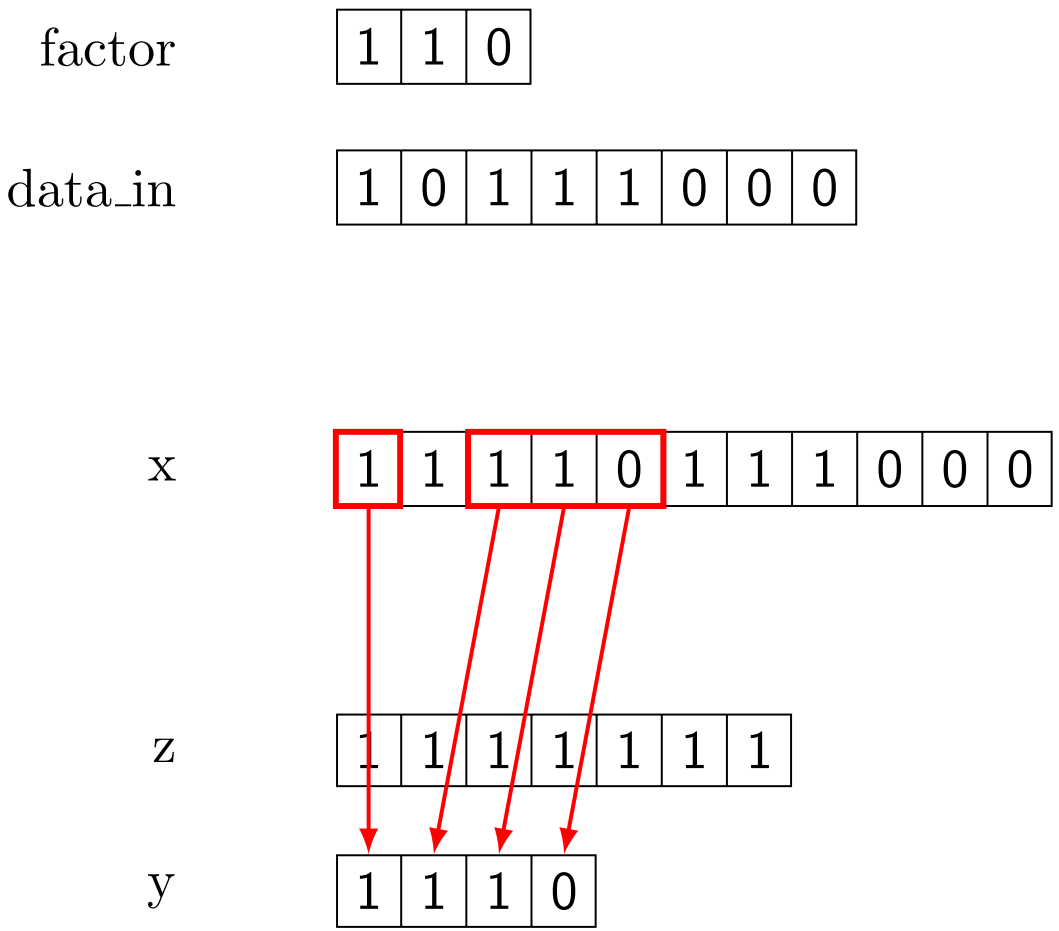
ISCALE

factor	1	1	0								
data_in	1	0	1	1	1	0	0	0			
x	1	1	1	1	0	1	1	1	0	0	0
z	1	1	1	1	1	1	1				

ISCALE

factor	1	1	0								
data_in	1	0	1	1	1	0	0	0			
x	1	1	1	1	0	1	1	1	0	0	0
z	1	1	1	1	1	1	1				
y											

ISCALE



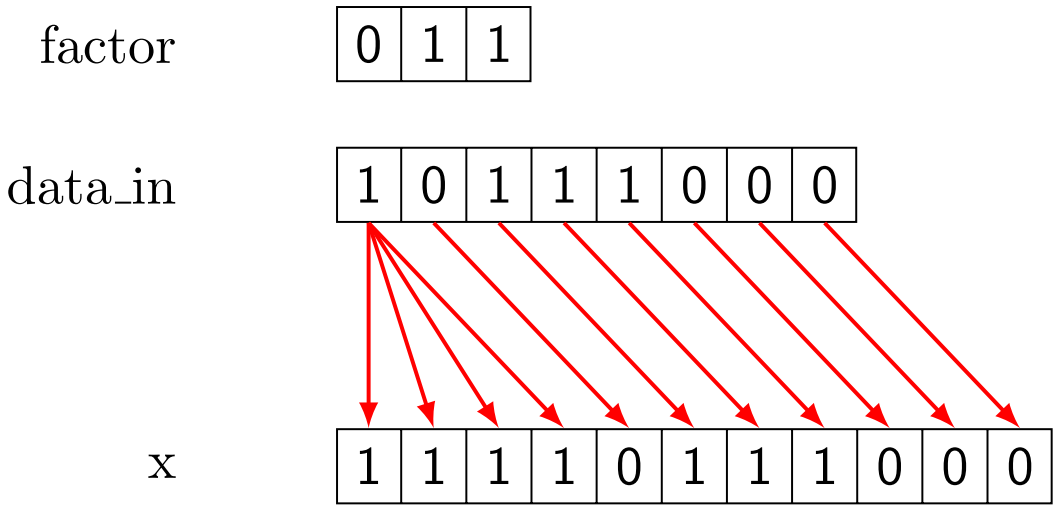
ISCALE

factor	0	1	1					
data_in	1	0	1	1	1	0	0	0

ISCALE

factor	0	1	1								
data_in	1	0	1	1	1	0	0	0			
x											

ISCALE



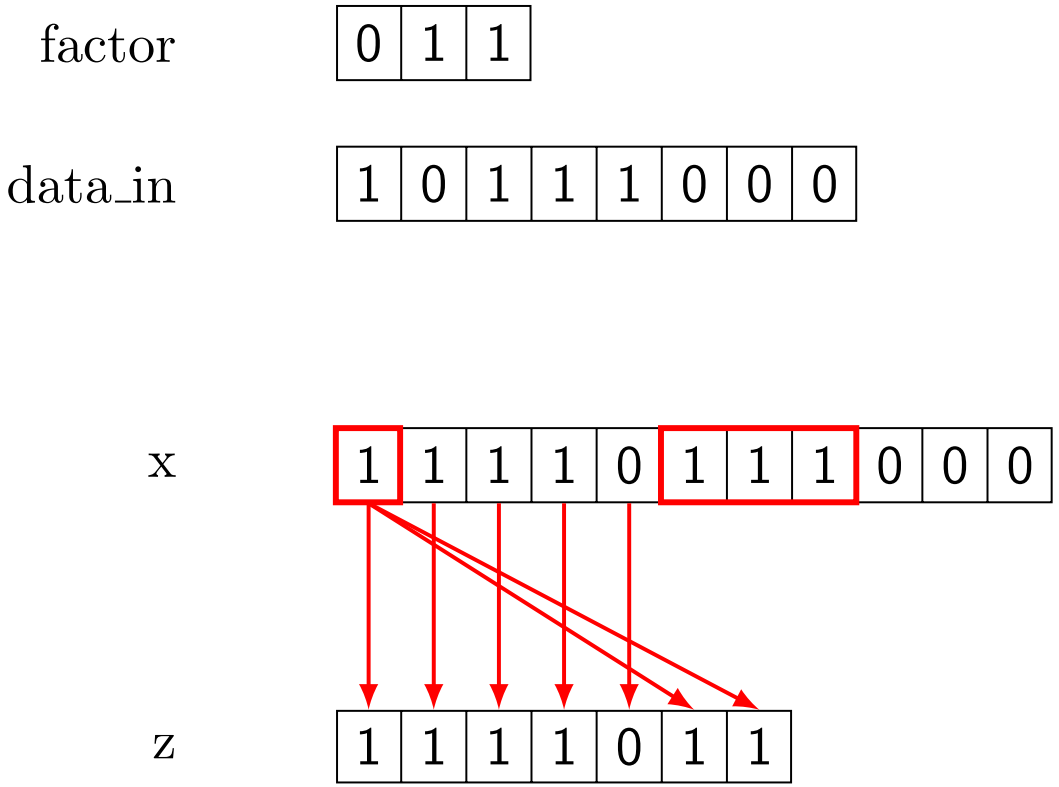
ISCALE

factor	0	1	1								
data_in	1	0	1	1	1	0	0	0			
x	1	1	1	1	0	1	1	1	0	0	0

ISCALE

factor	0	1	1								
data_in	1	0	1	1	1	0	0	0			
x	1	1	1	1	0	1	1	1	0	0	0
z											

ISCALE



ISCALE

factor	0	1	1

data_in	1	0	1	1	1	0	0	0

x	1	1	1	1	0	1	1	1	0	0	0

z	1	1	1	1	0	1	1

ISCALE

factor	0	1	1

data_in	1	0	1	1	1	0	0	0

x	1	1	1	1	0	1	1	1	0	0	0

z	1	1	1	1	0	1	1

ISCALE

factor	0	1	1

data_in	1	0	1	1	1	0	0	0

x	1	1	1	1	0	1	1	1	0	0	0

z	1	1	1	1	0	1	1

ISCALE

factor

0	1	1

data_in

1	0	1	1	1	0	0	0

x

1	1	1	1	0	1	1	1	0	0	0

z

1	1	1	1	0	1	1

ISCALE

factor	0	1	1

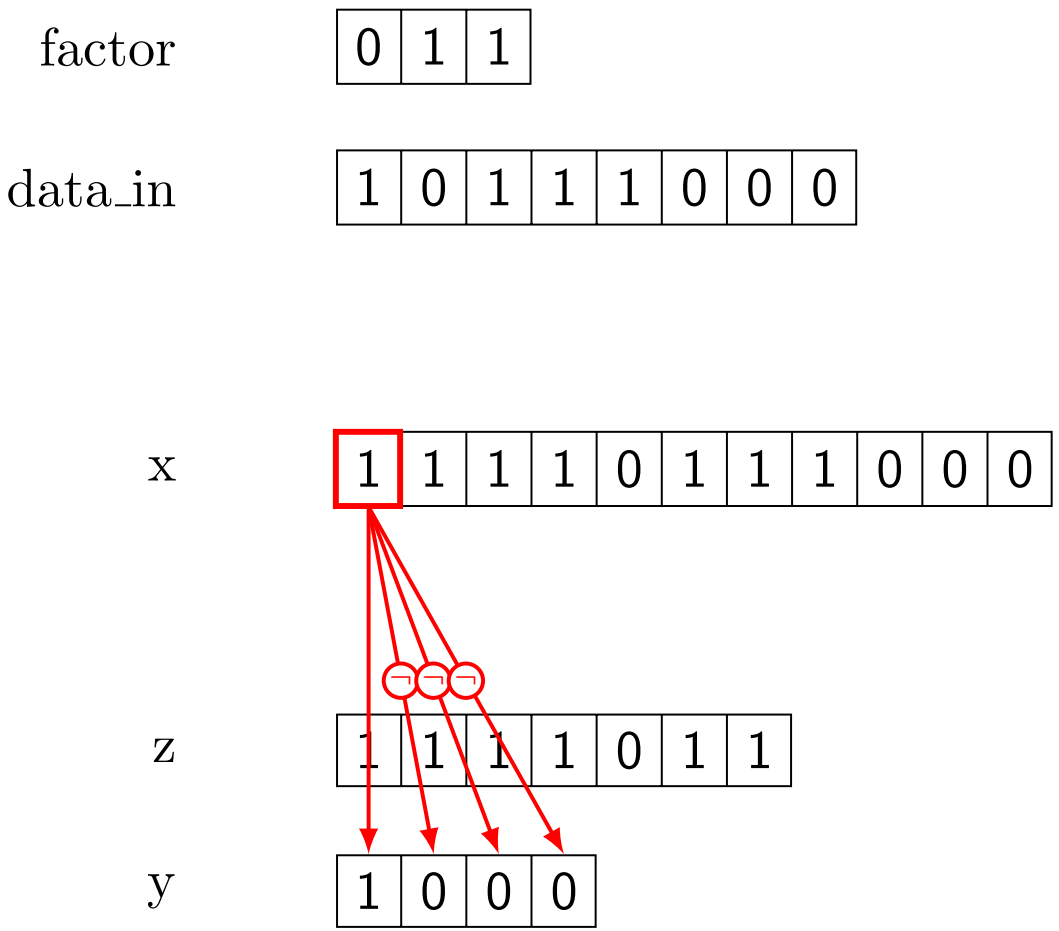
data_in	1	0	1	1	1	0	0	0

x	1	1	1	1	0	1	1	1	0	0	0

z	1	1	1	1	0	1	1

y				

ISCALE



ISUM

- 3 dual-ported RAM
- 345 blocks added (1 block = 512×20 bit)
- Behaviour:
  - Add incoming values to RAM 1 & 2
  - Put outgoing values from RAM 3 on APB
  - After 512 vectors wait for next block
  - Turn RAMs after 345 blocks

Cost Analysis

Criteria	Cost per unit	Raw design count	Raw design cost	Total design count	Total Design cost
LUT	1	9.639	9.639	18.599	18.599
Register (bit)	0.5	9.091	4.545,5	12.901	6.450,5
Internal FPGA Memory (bit)	0.1	178.659	17.865,9	2.444.259	244.425,9
Embedded Multiplier (9-bit)	25	128	3.225	128	3.225
PLL Clock Output	100	1	100	1	100
SDRAM (byte)	0.05	705.600	35.280	1024	35.331,2
SRAM (byte)	0.2	0	0	0	0
Code size (byte)	1	24.346	24.346	24.346	24.346
Input-Output delay (ms)	500	<6.8	3.400	<6.8	3.400
			98.401,4		335.852,6

Traps & Pitfalls

- Wrong design decisions, underestimated memory costs
- Control timing interrupt problems
- Parallel programming, timing requirements not met
- Overflow handling and scaling
- Truncation noise using two's-complement
- SCARTS
  - Malloc-free order
  - Interrupt 0 not working
- Grlib AHB DMA controller – SDRAM refresh problem

Thank you for your attention!