Unfolding & Folding Transformations

Chapter 8

Dr. Shoab A. Khan

Folding Architecture

- Folding design are time-multiplexed architectures
- Mathematical Transformations are also used to fold a DFG
- These transformations take DFG and systematically generate a folded architecture
- The techniques also generate a schedule
  - Clockwise mapping of multiple operations of the DFG on fewer HW computational units of the folded architecture

Folding Design Decision

- The design decision is made based on the ratio of sampling f_s and circuit clock f_c
  - f_s is specific to an application
    - f_s is constraint by Nyquist sampling criteria or bandpass sampling techniques
  - f_c depends upon several factors
    - Target technology
    - Digital design options (Adders, Multipliers)
    - Power considerations etc

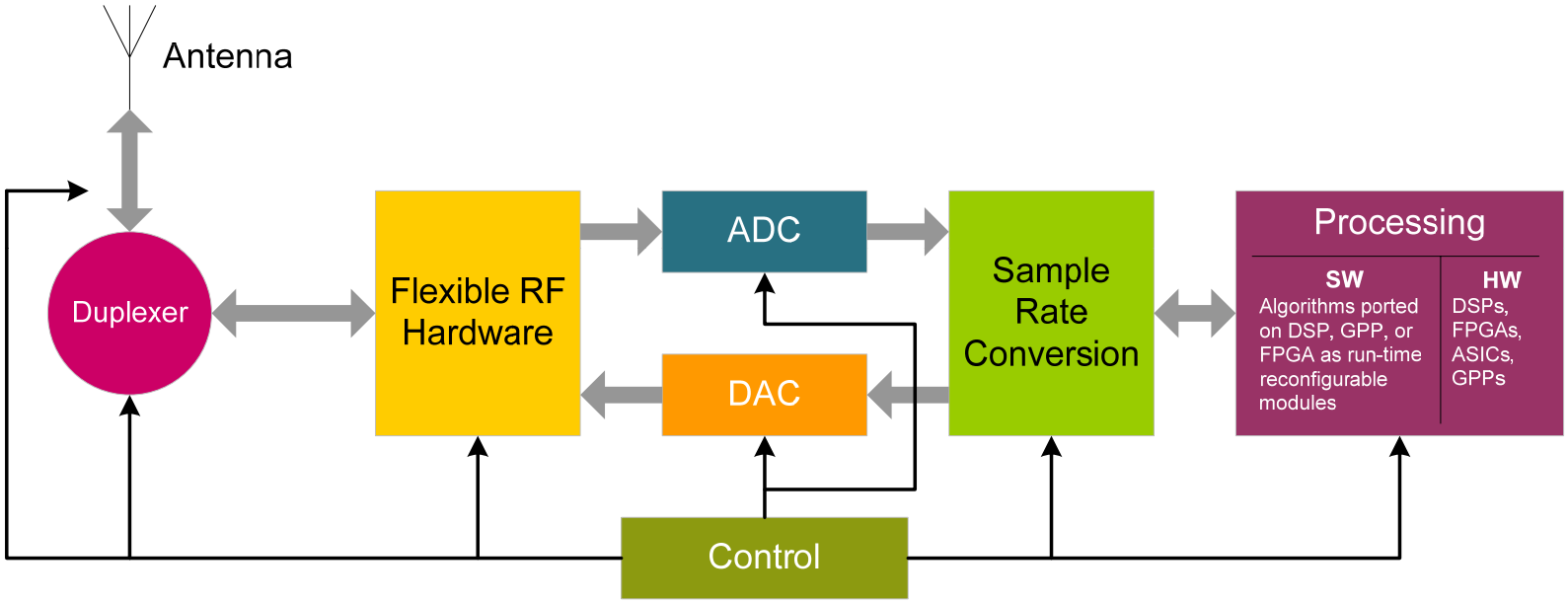
Nyquist Theorem and Bandpass Sampling

- For perfect reconstruction, minimum sampling rate has to be greater than or equal to twice the maximum frequency content in the signal
- For bandpass signals, bandpass sampling can be performed

$$f_s \geq 2 f_N$$

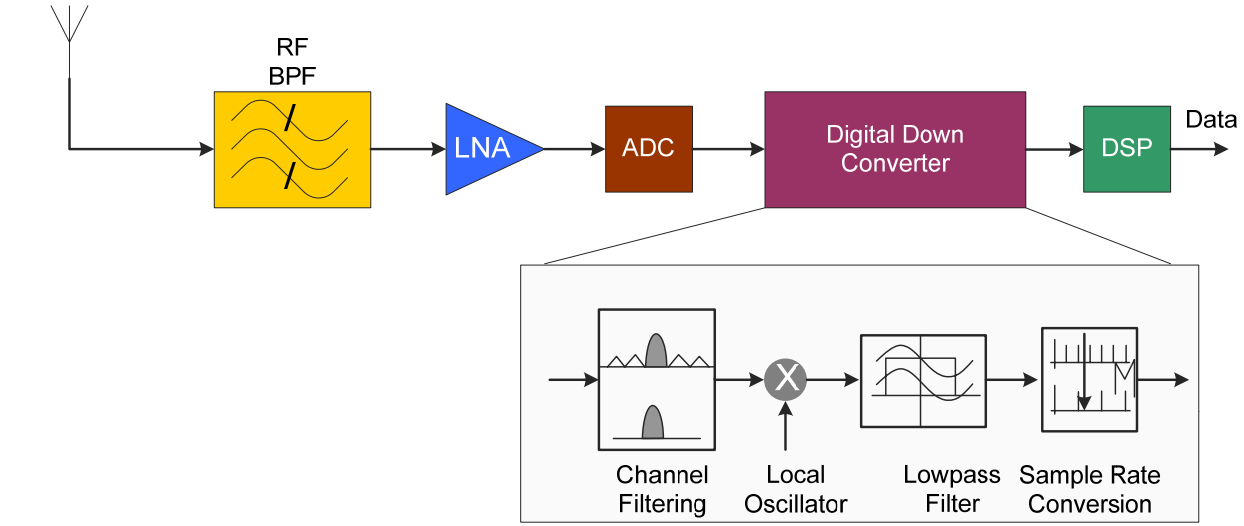
  - Where the sampling rate is set to lower values
  - The spectrum is intentionally aliased to fall on unoccupied lower digital frequencies

Software Defined Radio Architecture and Band-pass Sampling

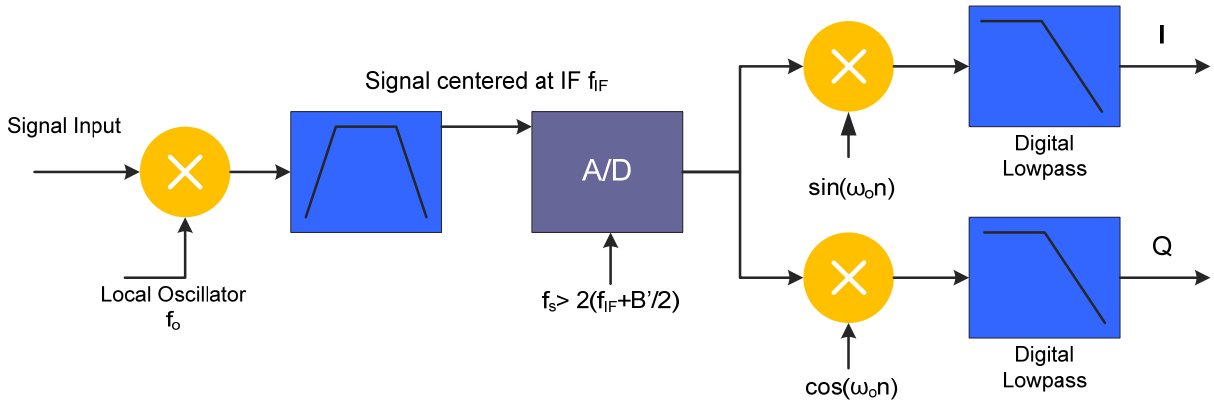


Typical Software Defined Radio Architecture

Software Defined Radio Architecture and Band-pass Sampling



(a)



(b)

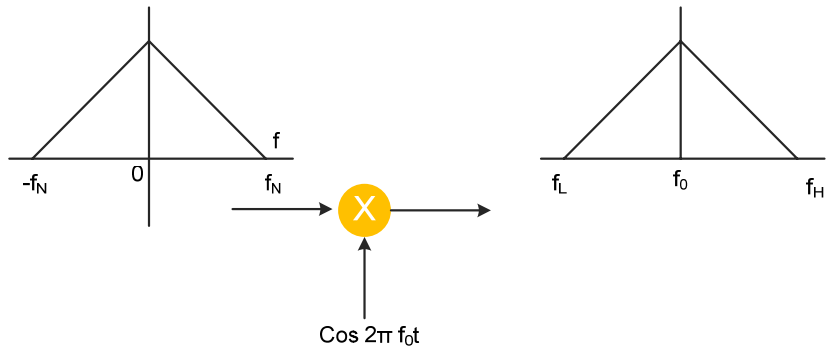
Digital Communication Receiver (a). Direct conversion receiver (b). IF conversion receiver

Software Defined Radio Architecture and Band-pass Sampling

- The signal is voice, video, image or data that is modulated and mixed with a carrier to take alloted limited bandwidth in air

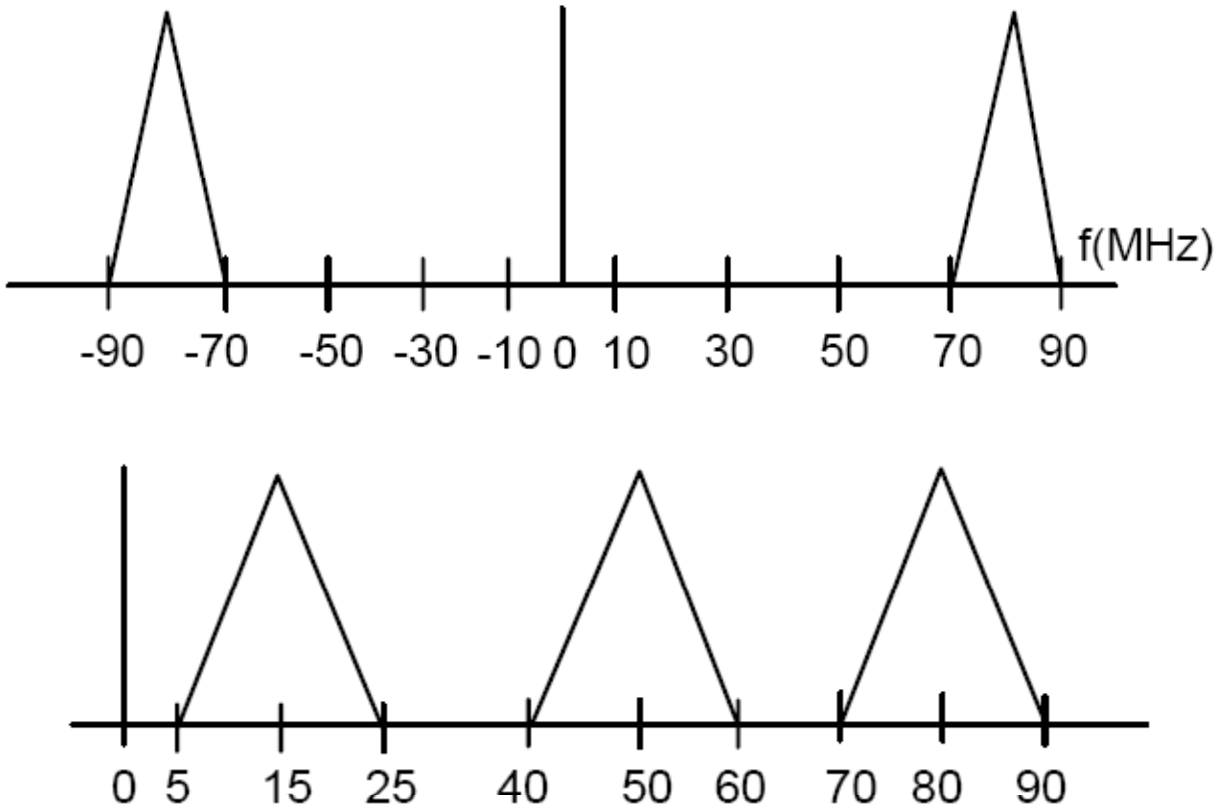
$$B = 2f_N$$

- f_N is the highest frequency content of the baseband signal.
- The signal can be sampled at lower rate using band-pass sampling technique



Baseband signal mixed with a carrier for signal translation in frequency band

Band-pass Sampling



(a)

(b)

(a) The spectrum of a 20 MHz signal centered at IF of 80 MHz
(b) The spectrum of band-pass sampled signal, f_s = 65 MHz

ADC Bandwidth and Band-pass Sampling

- ADC bandwidth (BW) is also an important consideration
- This bandwidth corresponds to the highest frequency at which the internal electronics of ADC can pass the signal without any attenuation

Unfolding Techniques

- SW:
  - unfolding transformation is unrolling a loop to execute several iterations in one unrolled-iteration
- HW:
  - unfolding corresponds to mathematical transformation on a DFG to replicate its functionality for computing multiple output samples for multiple input samples

SW: Loop Unrolling

- Example: dot product of two arrays of size N

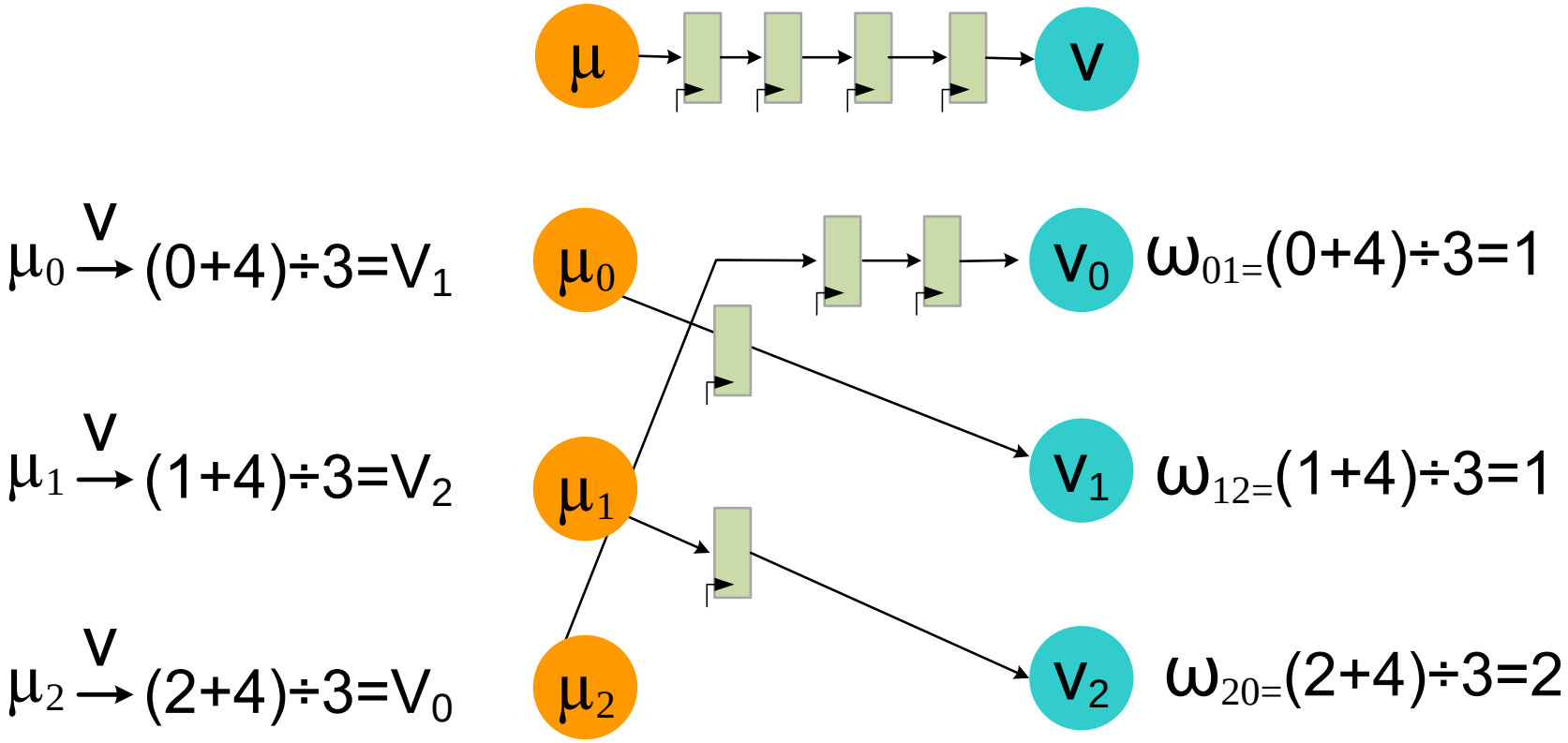
```
sum=0;
for (i=0; i<N; i++)
        sum += a[i]*b[i];
```

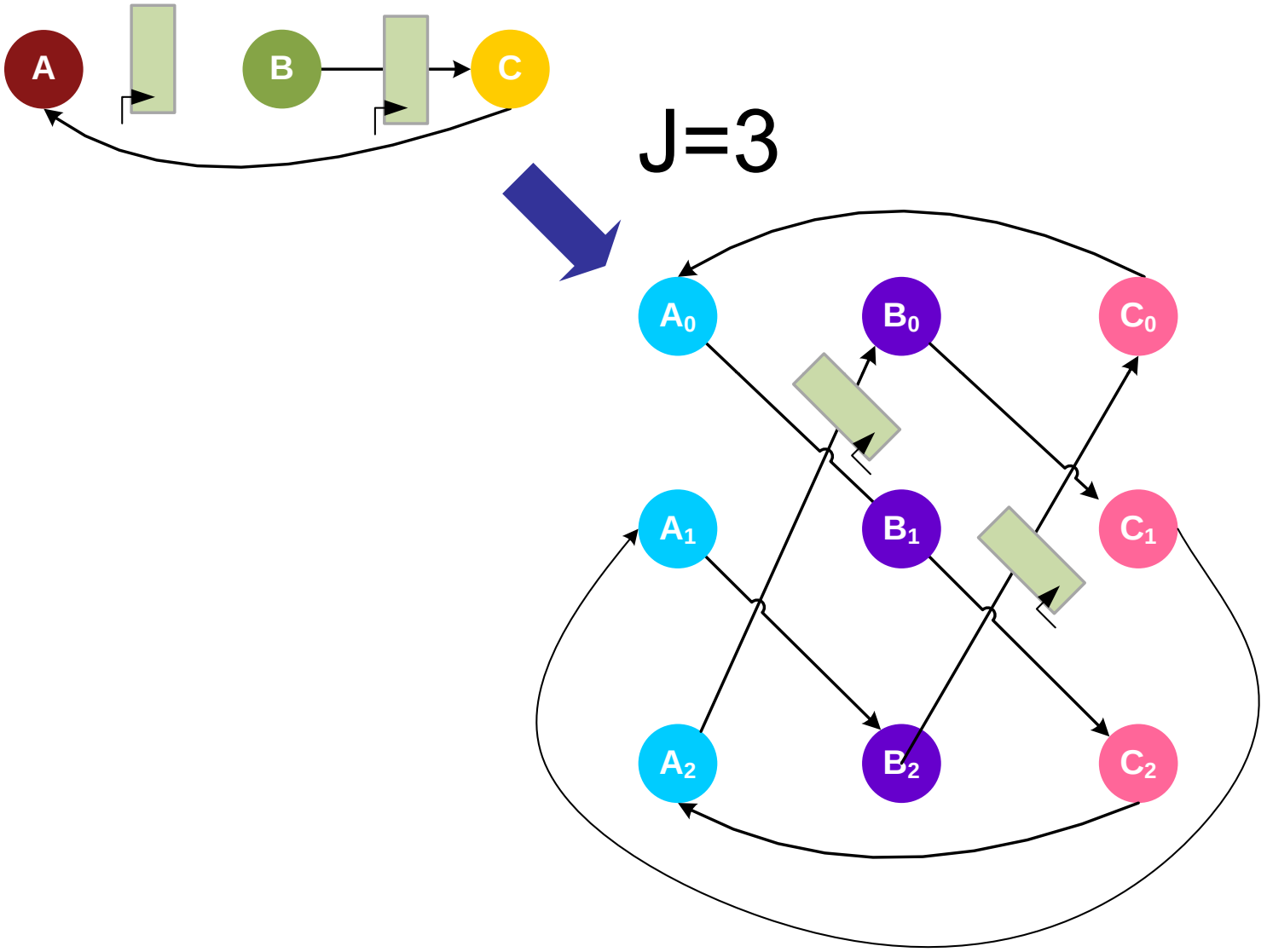
SW: Loop Unrolling

```
sum1=0;
sum2=0;
sum3=0;
sum4=0;

for(i=0; i<N/L; i++)
{
      sum1 += a[i]*b[i];
      sum2 += a[i+1]*b[i+1];
      sum3 += a[i+2]*b[i+2];
      sum4 += a[i+3]*b[i+3];

}

sum = sum0+sum1+sum2+sum3;
```

HW: Unfolding Transformation

- Unfolding by a factor J
- **S0:** Each node *U* of the original DFG is replicated *J* times as $U_0,\ldots,U_{J-1}$
- **S1:** For two connected nodes *U* and *V* in the original DFG with *w* number of delays, draw *J* edges such that each edge *j* for *j=0,1,…,J-1* connects node U_j to node *V(j+w)%J* with $\left\lfloor \frac{j+w}{J} \right\rfloor$ delays
  - % and $\lfloor\ \rfloor$ are remainder and floor operator


μ
v
μ0 v (0+4)÷3=V1
μ1 v (1+4)÷3=V2
μ2 v (2+4)÷3=V0
μ0
μ1
μ2
v0
v1
v2
ω01=(0+4)÷3=1
ω12=(1+4)÷3=1
ω20=(2+4)÷3=2


A
B
C
J=3
A0
B0
C0
A1
B1
C1
A2
B2
C2

Example 8-2

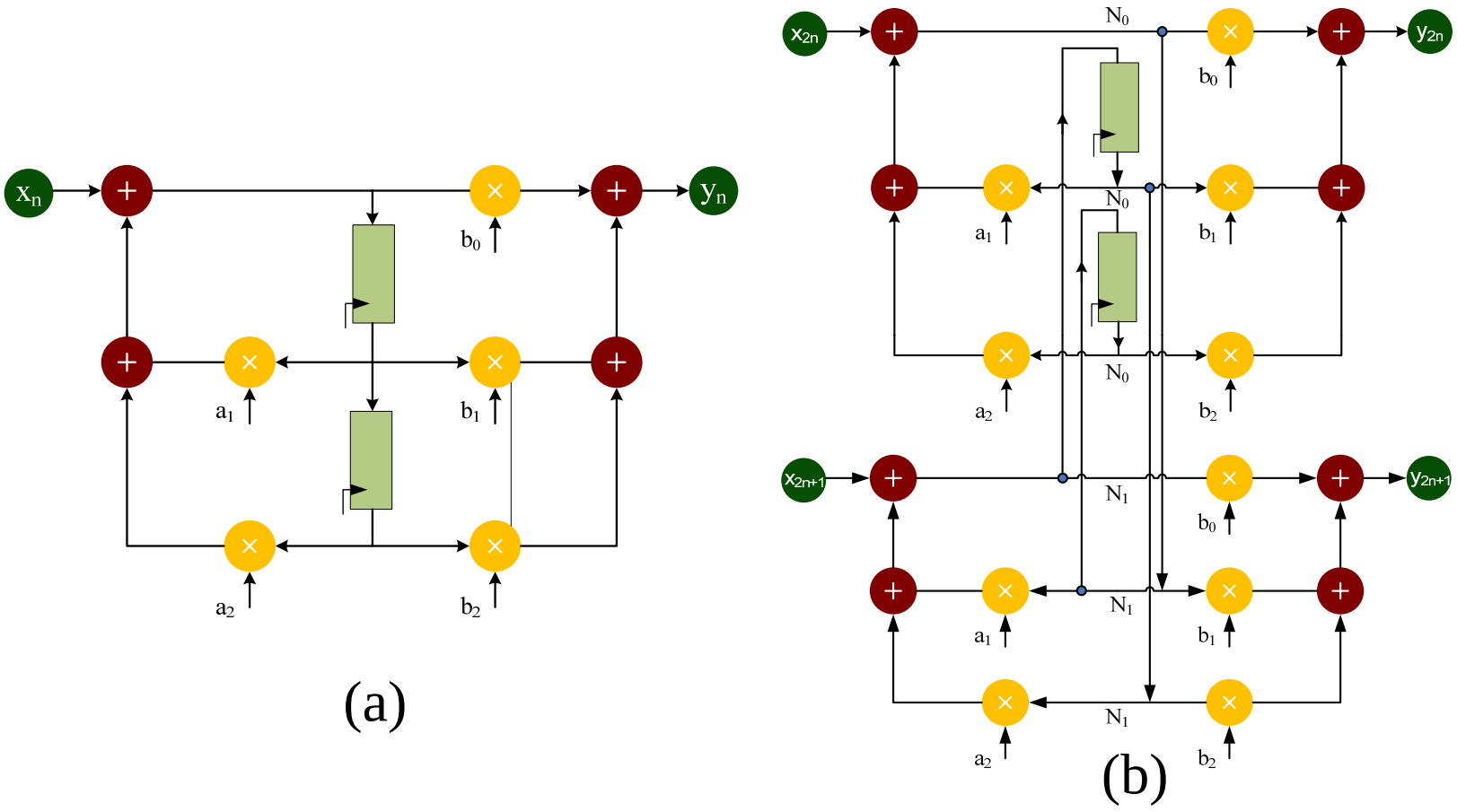


Figure 8-5 (a) A 2nd order TDF structure
(b) Folding of 2nd order TDF structure by folding factor of 2

Loop Unrolling for SW to HW Mapping

```
short xbuf[N+L-1];
short hfb[L];
short ybf[N];

short *xptr, *hptr;
int sum;
int m,n;

xptr = &xbf[L-1];
hptr = hbf;
for (n=0; n<N; n++)
{
        sum = 0;
        for(k=0, m=n; k<L; k++, m--)
                        sum += xptr[m]*hptr[k];
        ybf[n] = sum;
}
```

The unrolled loop for effective HW mapping is given here.

```
xptr = &xbf[L-1];
hptr = hbf;
for (n=4-1; n<N; n+=4)
{
        sum_n = 0;
        sum_n_1 = 0;
        sum_n_2 = 0;
        sum_n_3 = 0;

for (k=0, m=n; k<L; k++, m--)
{
                        sum_n += hptr[k] * xptr[m];
                        sum_n_1 += hptr[k] * xptr[m-1];
                        sum_n_2 += hptr[k] * xptr[m-2];
                        sum_n_3 += hptr[k] * xptr[m-3];
        }
        ybf[n] = sum_n;
        ybf[n-1] = sum_np1;
        ybf[n-2] = sum_np2;
        ybf[n-3] = sum_np3;
}
```

Loop Unrolling for SW to HW Mapping

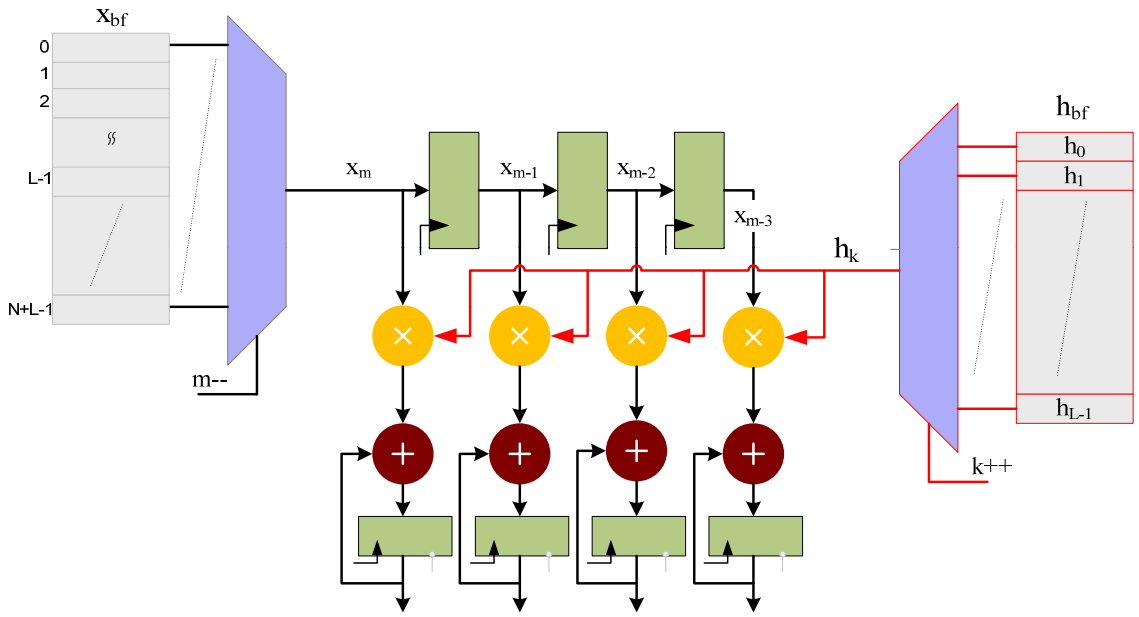


Figure 8-7: HW mapping of code listed in Figure 8-6

Unfolding to Maximize use of Compression Tree

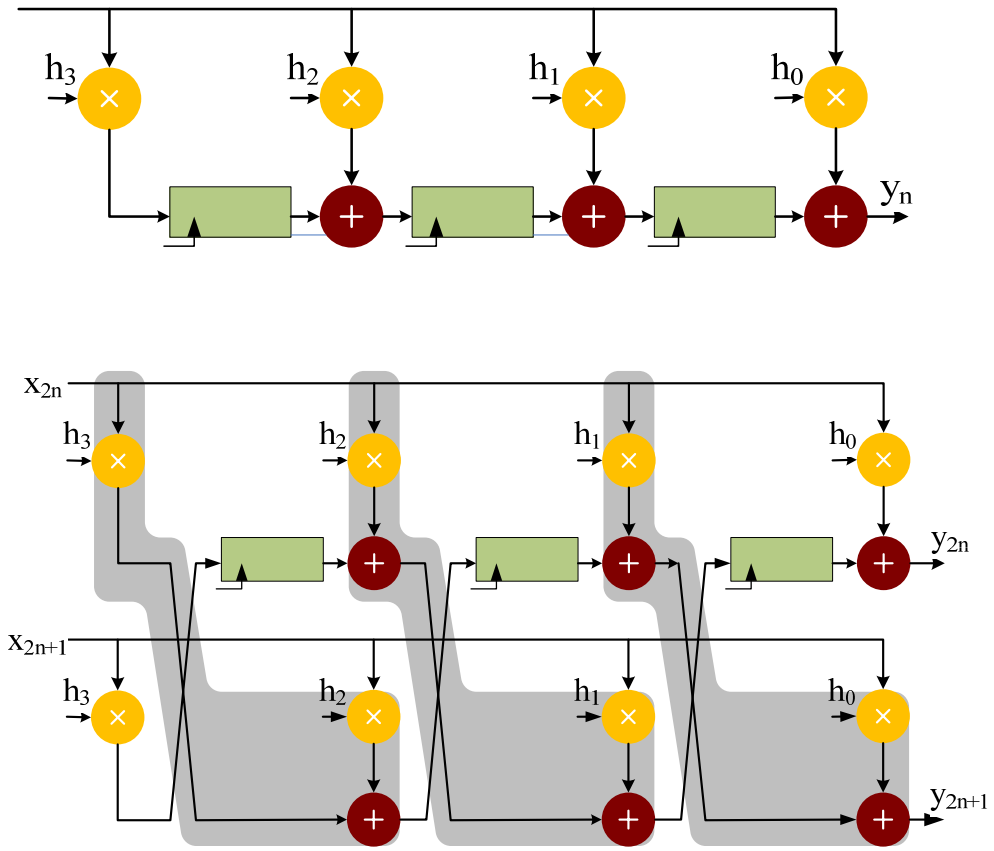


Figure 8-8: Unrolling an FIR filter (a) A 4-Coefficient FIR Filter (b) The filter is unrolled by a factor of 2

Unfolding for Effective Use of FPGA Resources

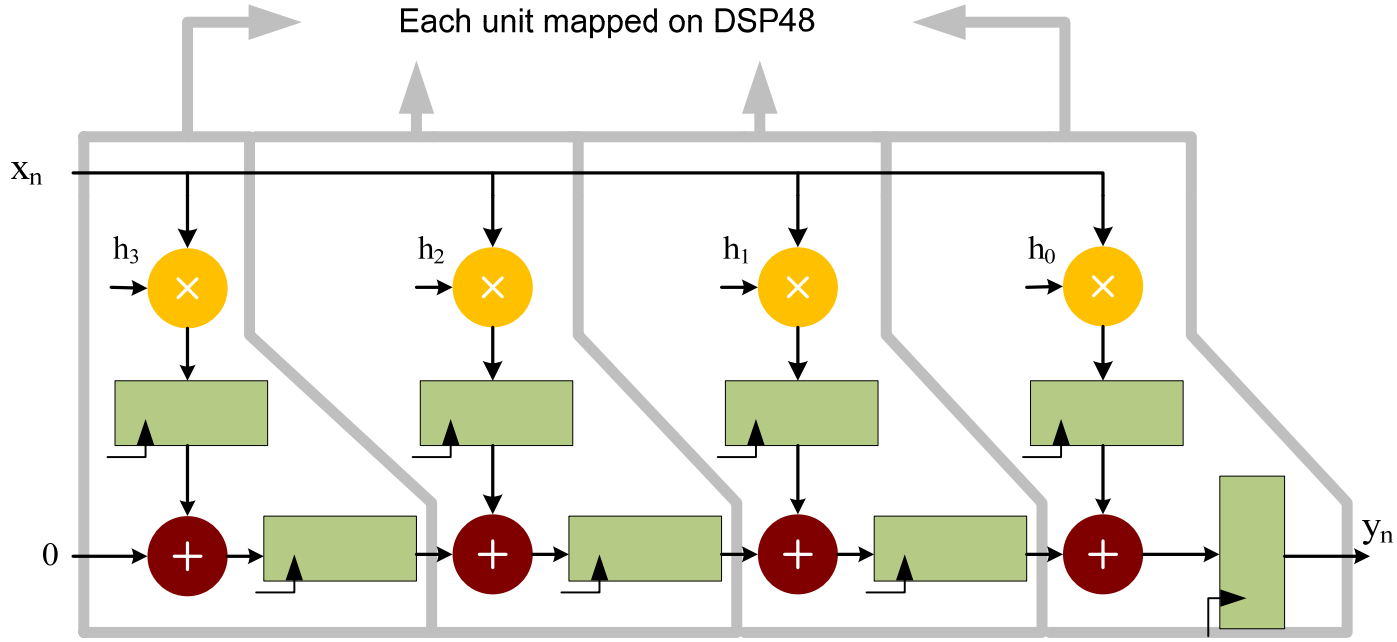
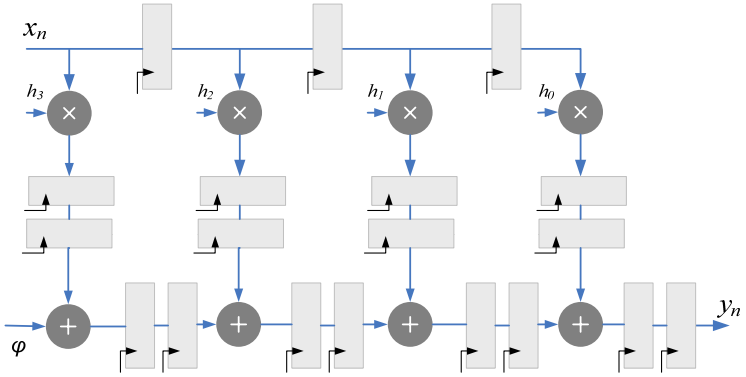
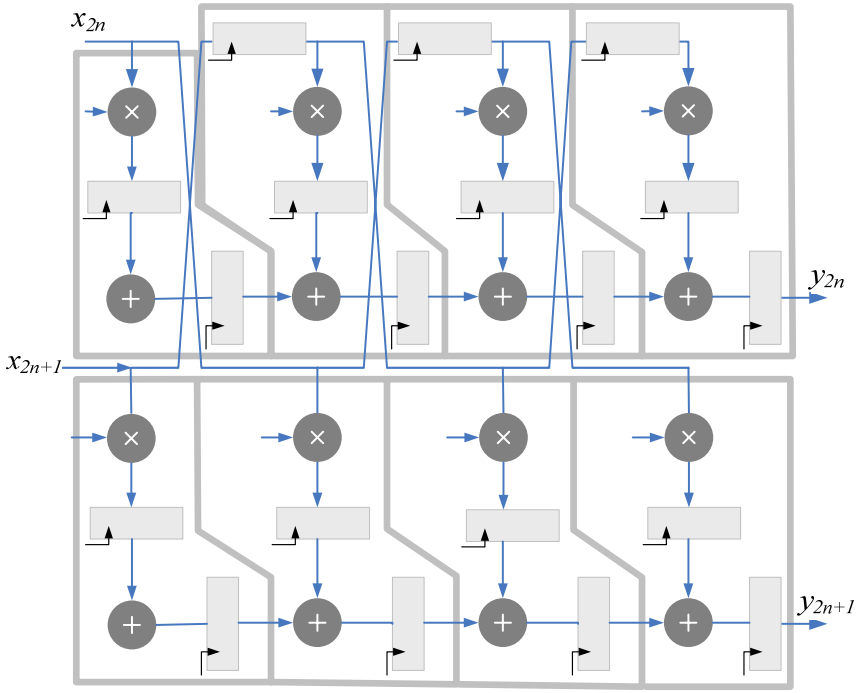


Figure 8-9: Pipelined FIR filter for effective mapping on FPGAs with DSP48 blocks.

Parallel Processing with Embedded Blocks in FIR



(a)



(b)

Figure 8-10: Unfolding for mapping on embedded pipeline MAC unit
(a) The DFG is appropriately pipeline
(b) The pipeline DFG is unfolded for effective mapping on embedded MAC units

Unfolding and Retiming in Feedback Designs

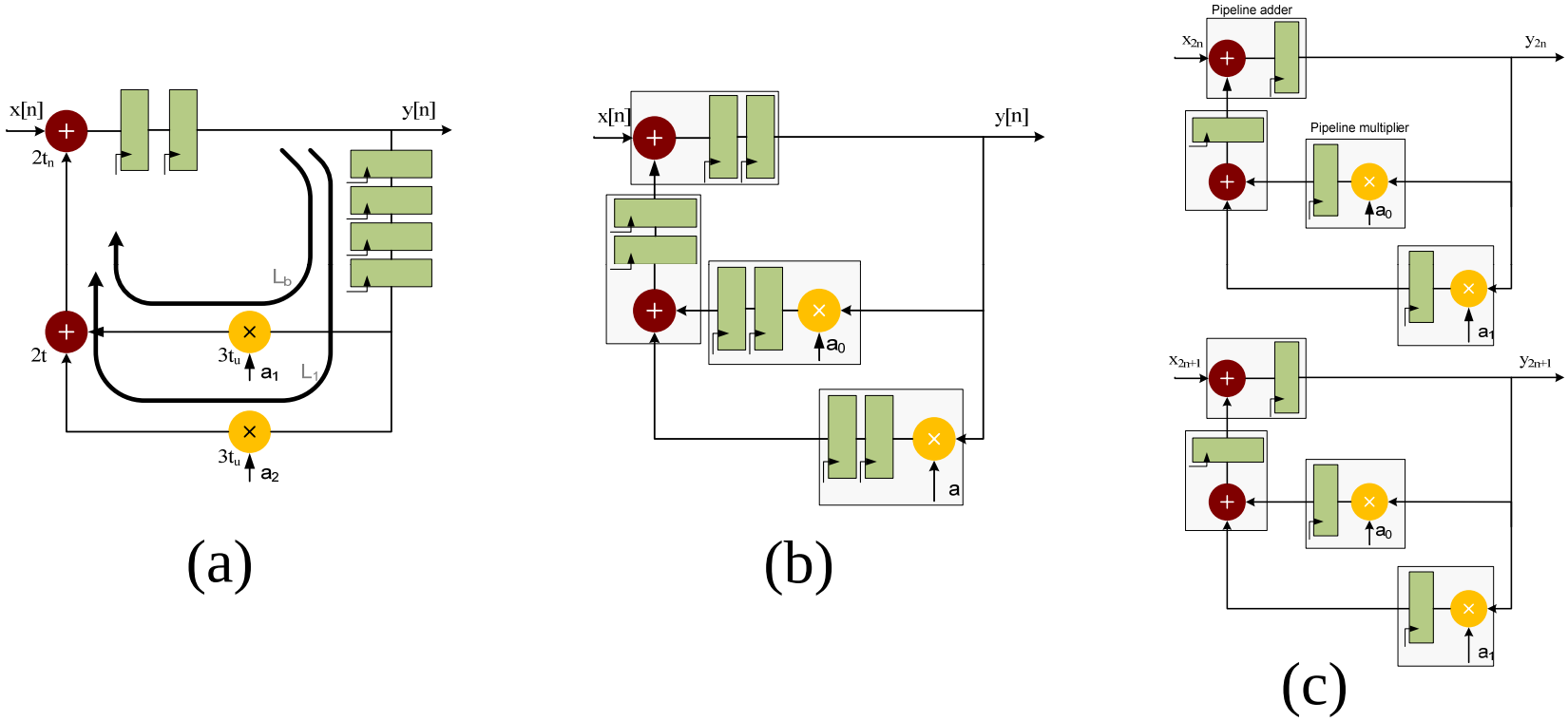


Figure 8-11: Unfolding and Retiming of Feedback DFG
(a) A recursive DFG with 6 algorithmic registers
(b) Retiming of resisters for associating algorithmic registers with computational nodes for effective mapping
(c) An unfolded design for effective utilization of algorithmic registers

Unfolding and Area-Power-throughput tradeoff

- IPB increases
- Design process more number of samples
- Throughput same with a slower circuit clock.
  - The circuit clock gets slower as the critical path is increased
- Effective use of extra registers in feedback design
- Increasing the performance of feed-forward designs with embedded units
  - Parallel processing with DSP48

Folding Techniques

- Time-shared architectures are designed once the circuit clock is at least twice greater than the sampling clock

  $f_c > 2\, f_s$

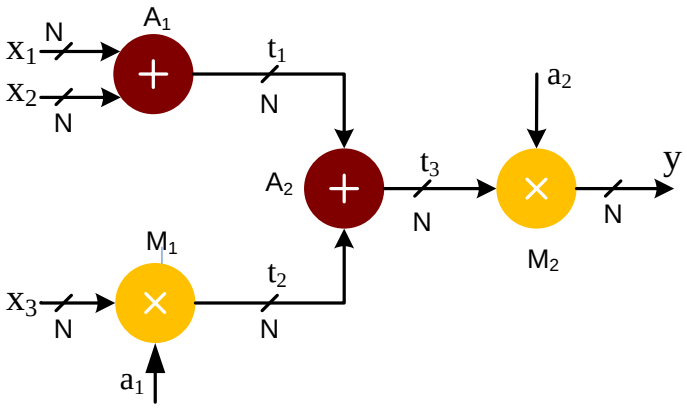
- The design can reuse its hardware resources
  - Input data valid for multiple of f_c

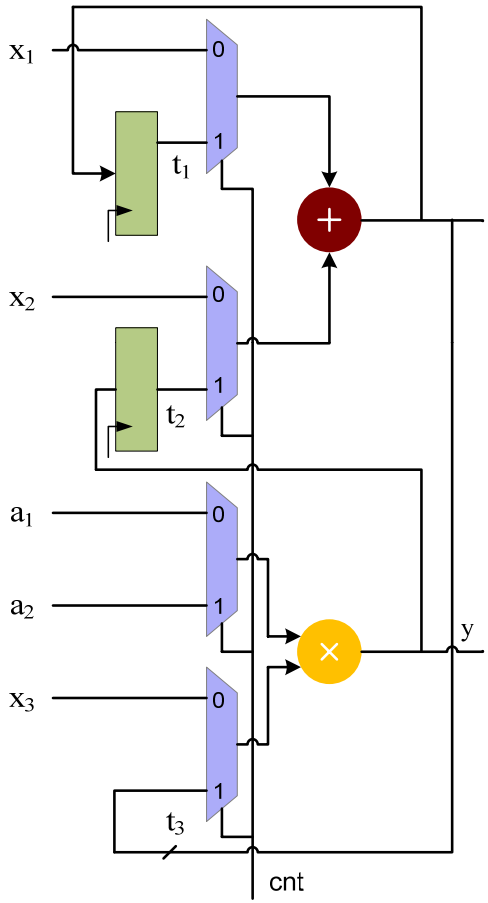
Basic Definition

- **Folding** is a mathematical technique of finding a time-multiplexed architecture and a schedule of mapping multiple operations of a DFG on fewer hardware computational units.
- **Folding Factor** is defined as maximum number of operations in a DFG mapped on a shared computational unit.
- **Folding Set or Folding Scheduler** is the sequence of operations of a DFG mapped on a single computational unit.

Example 8-3

A DFG implementing $out=(x_1+x_2+a_1x_3)a_2$





Folded architecture for the DFG

Folding Regular Structure DFGs

- The algorithms that exhibit regular structures can be easily folded.
  - Implementation of an FIR filter is a good example
- For any regular structure algorithm the folding factor is simply computed as

$$N = \left\lfloor \frac{f_c}{f_s} \right\rfloor$$

Example 8-4

(a) (b)

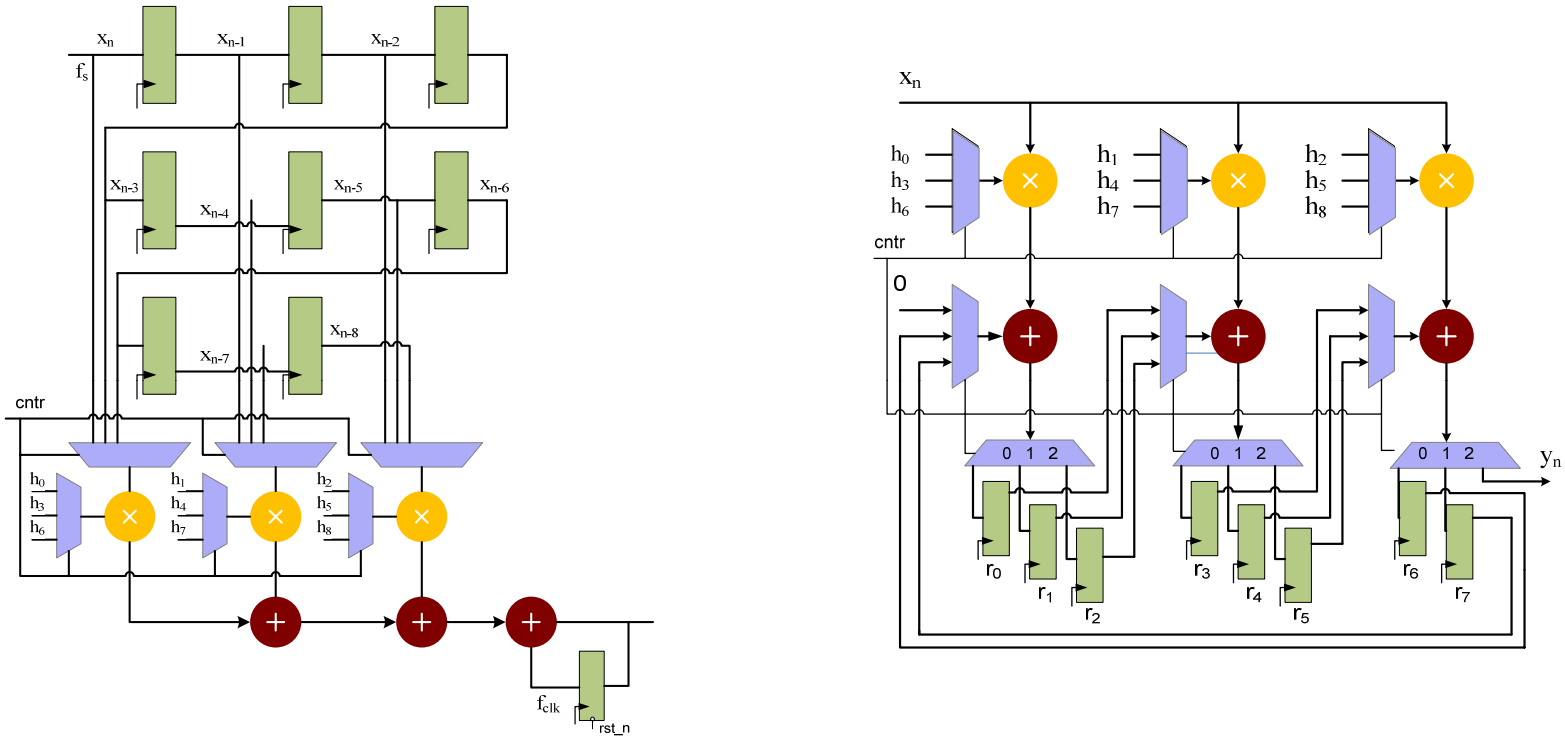
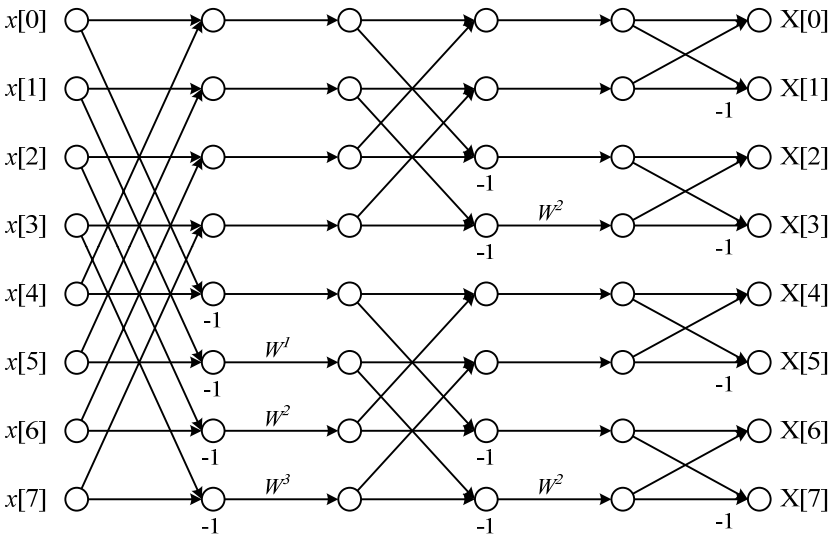
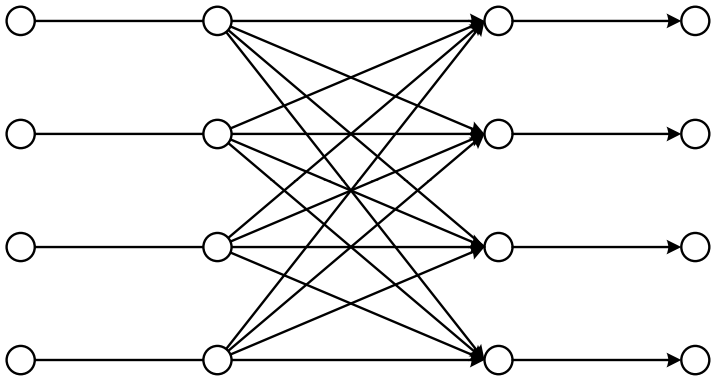


Figure 8-13: A Folded by 3 architecture for a 9 coefficients FIR filter
(a) Folded DF architecture
(b) (b) Folded TDF architecture

Example 8-5



(a)



(b)

Figure 8-14: Flow graphs realizing 8-point FFT algorithms using
(a) Radix-2 butterflies
(b) Radix-4 butterflies

Memory Based Folded FFT Processor

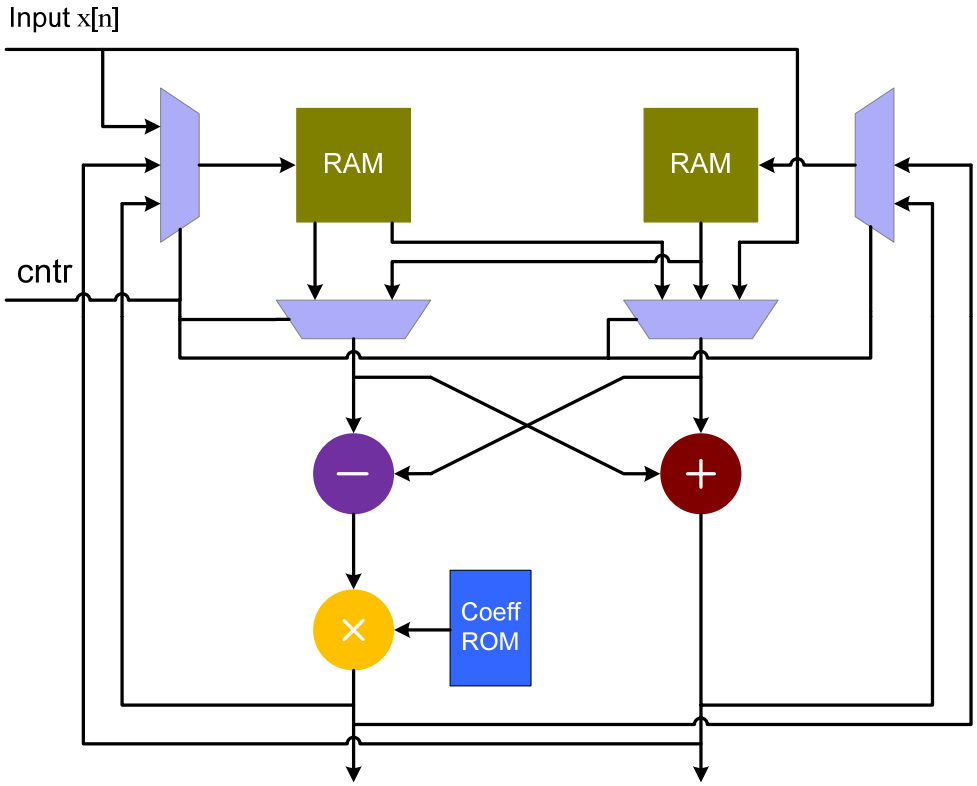


Figure 8-15: Two dual port memories based
FFT processor with one radix-2 butterfly in the datapath

Systolic Folded Architecture

- All the butterflies in one stage of the FFT are folded and mapped on a single butterfly Processing Element (PE)
- To ensure systolic operation, intermediate registers and multiplexers are placed such that the design keep processing data for computation of FFT and right inputs are always available to every PE in every clock cycle for full utilization of the HW.

Example 8-7

- The design for the systolic 8-point FFT implementation.

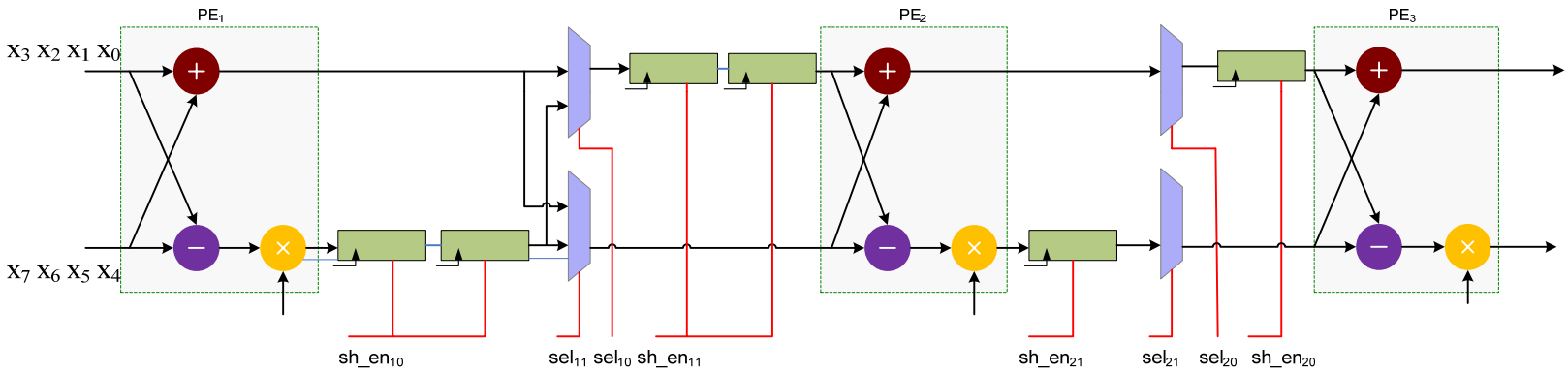


Figure 8-16: A systolic MDC architecture implementing an 8-point decimation in frequency FFT algorithm

Mathematical Transformation for Folding

- In many signal processing application the algorithm is transformed for easy folding.
- Many feed-forward algorithms can be transformed to use recursive formulation for serial computation of output samples.

Mathematical Transformation

- First compute the folding order and a folding set for a DFG,
- Computes the number of delays on each edge in the folded graph using folding transformation
- The folded architecture periodically executes operations of the DFG according to the folding set

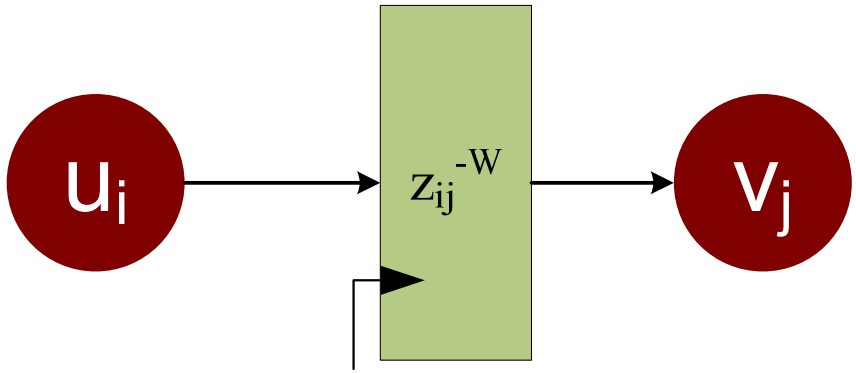
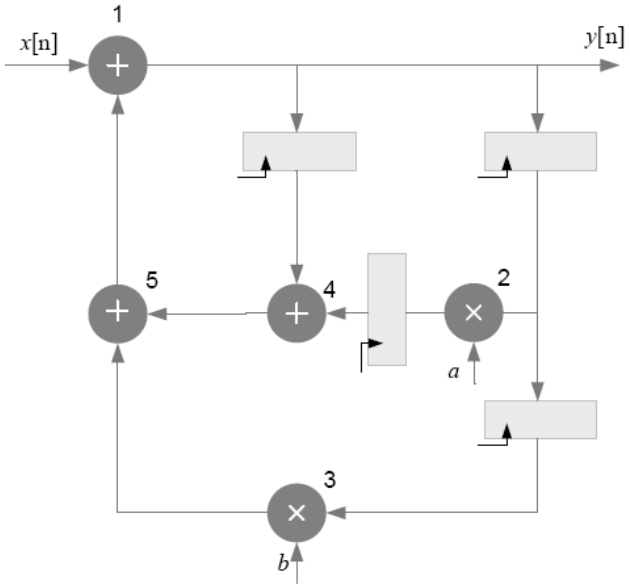
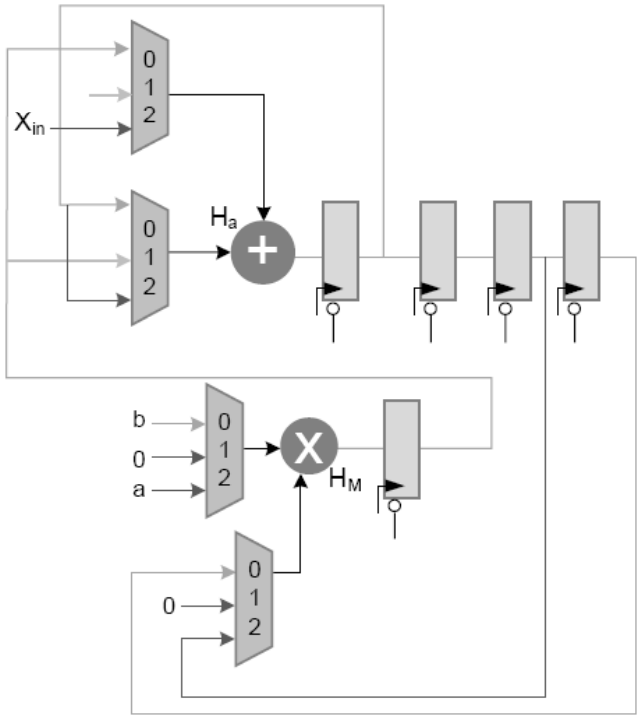


Figure 8-18: An edge in a DFG connecting nodes i and j with w delays

Example: Folding Transformation



$$F_{ij} = N \times W_{ij} + v_j - u_i$$
$$F_{12} = 3 + 2 - 2 = 3$$
$$F_{13} = 3 \times 2 + 0 - 2 = 4$$
$$F_{24} = 3 + 0 - 2 = 1$$
$$F_{14} = 3 + 0 - 2 = 1$$
$$F_{45} = 0 + 1 - 0 = 1$$
$$F_{35} = 0 + 1 - 0 = 1$$
$$F_{51} = 0 + 2 - 1 = 1$$



$$S_a = \{4, 5, 1\} \qquad S_m = \{3, \ ,2\}$$

Algorithmic Transformation

- In many applications a more convenient HW mapping can be conceived by performing an algorithmic transformation of the design.
- For FFT computation Goetzl algorithm is a good example where the FFT computation is implemented as an IIR filter.
- This formulation is effective if only few coefficients of the frequency spectrum are to be computed.

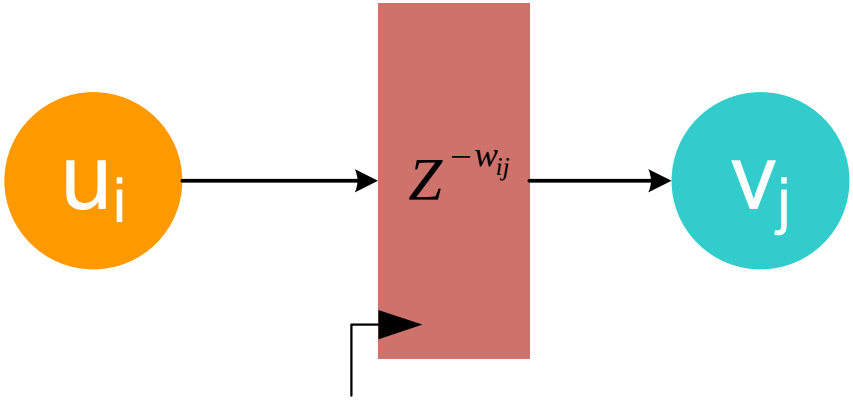
FFT as Circular Convolution

- A more interesting transformation is derived in [] that converts an FFT computation to circular convolution summation.
- The transformation helps in developing an effective architecture for FFT computation.
- Circular convolution equation

$$X[k] = W_N^{\frac{k^2}{2}} \sum_{n=0}^{N-1} \left(x[n] W_N^{\frac{n^2}{2}} \right) W_N^{-(k-n)^2}$$

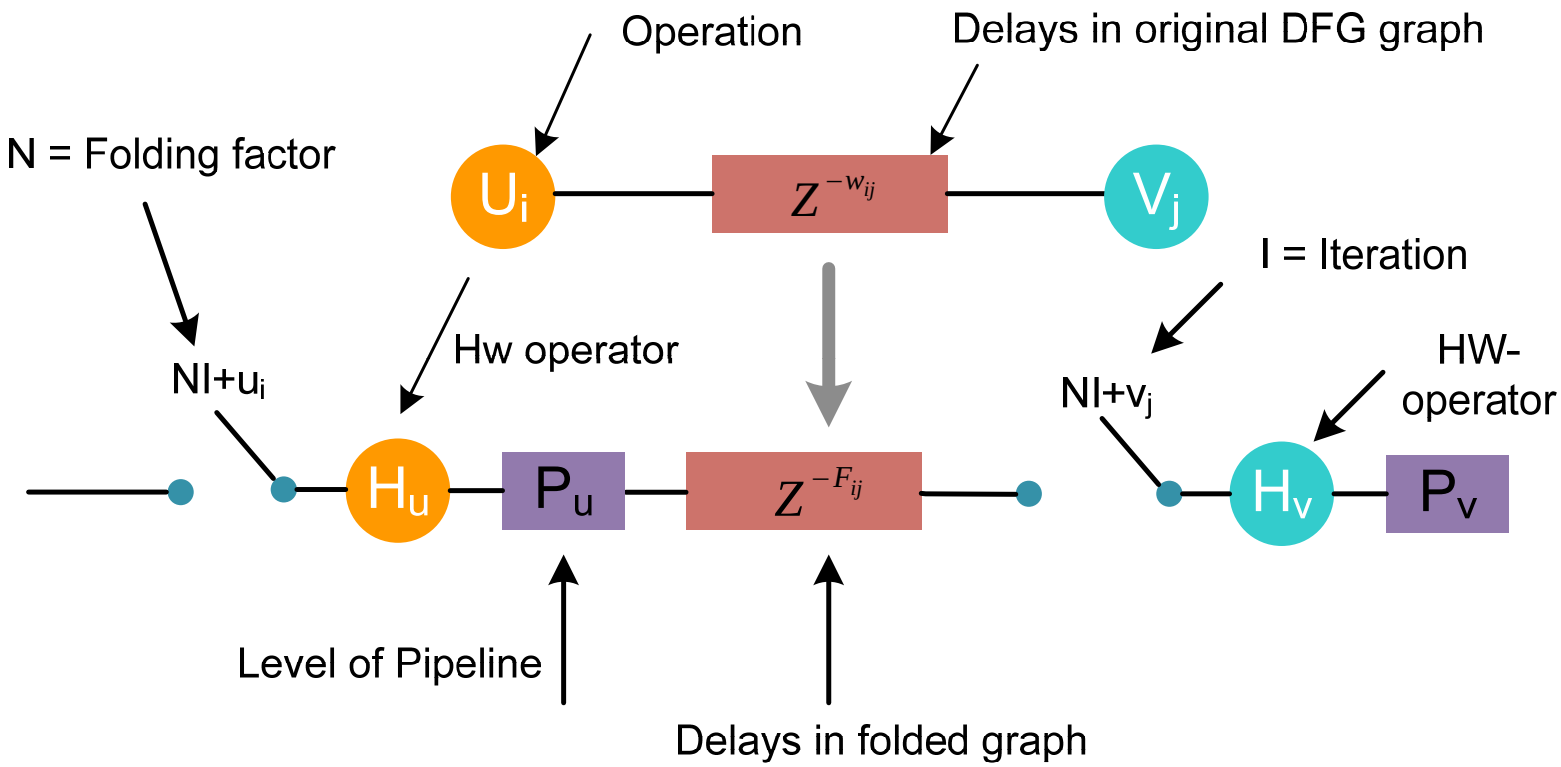
Mathematical Transformation

- Compute the folding order N and a folding set for a DFG
- Computes the number of delays on each edge in the folded graph using folding transformation
- The folded architecture periodically executes operations of the DFG according to the folding set in N circuit clock



An edge in a DFG connecting nodes i and j with w_{ij} delays

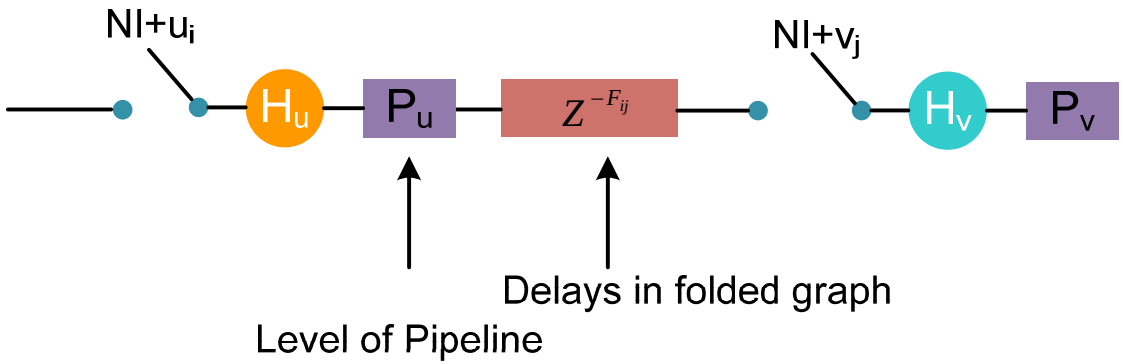
Folding Transformation



u_i and v_j are folding order i.e. scheduled time for Operation U_i and V_j

$0 \leq u_i, v_j \leq N\text{-}1$

Folding Transformation



H_u is pipelined by P_u stages and its output is available at $NI+u+P_u$.

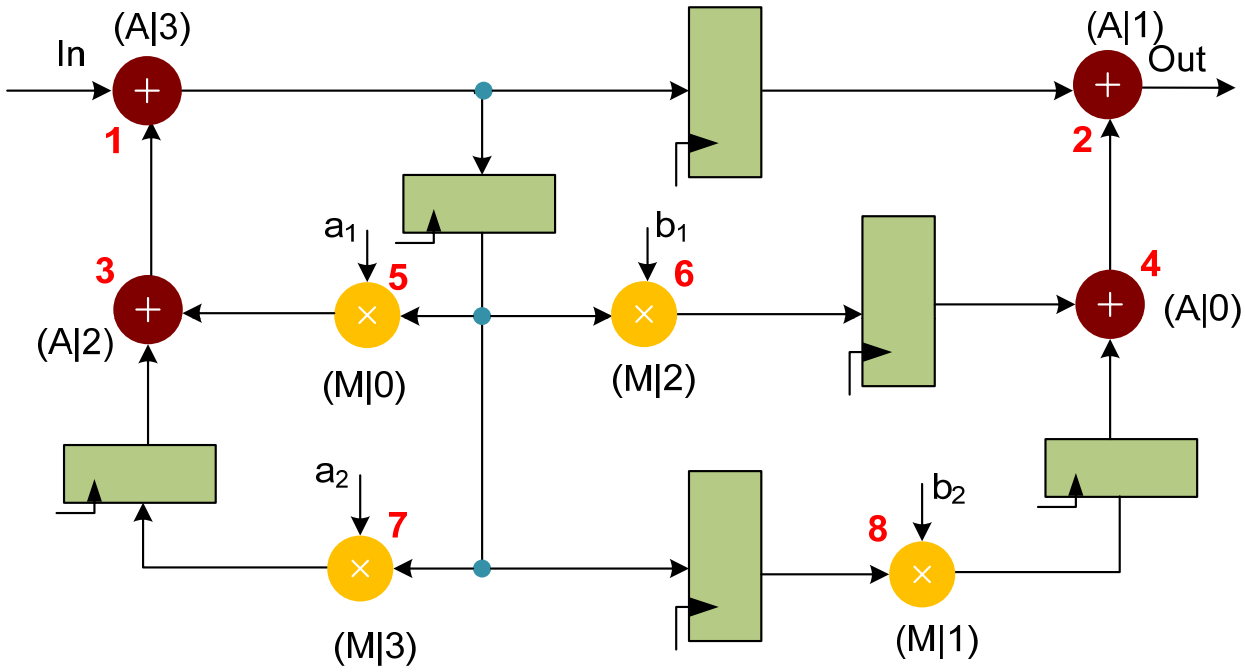
Edge $U_i \longrightarrow V_j$ has w_{ij} delays ➡ the I-th iteration of U_i is used by $(I+w_{ij})$ th iteration of node V_j which is executed at $N(I+w_{ij})+v$.

So the result should be stored for the number of circuit clocks:

$$F_{ij} = [N(I + w_{ij}) + v_j] - [NI + P_u + u_j]$$

➡ $$F_{ij} = Nw_{ij} - P_u + v_j + u_i \text{ (independent of I)}$$

Folding or Order N=4



Additions
A = {4,2,3,1}

Multiplication
M = {5,8,6,7}

Pipeline stages
$P_A = 1$
$P_M = 2$

Folding IIR Filter with N=4

receive send

$$F_{ij} = Nw_{ij} - P_u + v_j - u_i$$

$$F_{12} = 4(1) - 1 + 1 - 3 = 1$$

$$F_{15} = 4(1) - 1 + 0 - 3 = 0$$

$$F_{16} = 4(1) - 1 + 2 - 3 = 2$$

$$F_{17} = 4(1) - 1 + 3 - 3 = 3$$

$$F_{18} = 4(2) - 1 + 1 - 3 = 5$$

$$F_{31} = 4(0) - 1 + 3 - 2 = 0$$

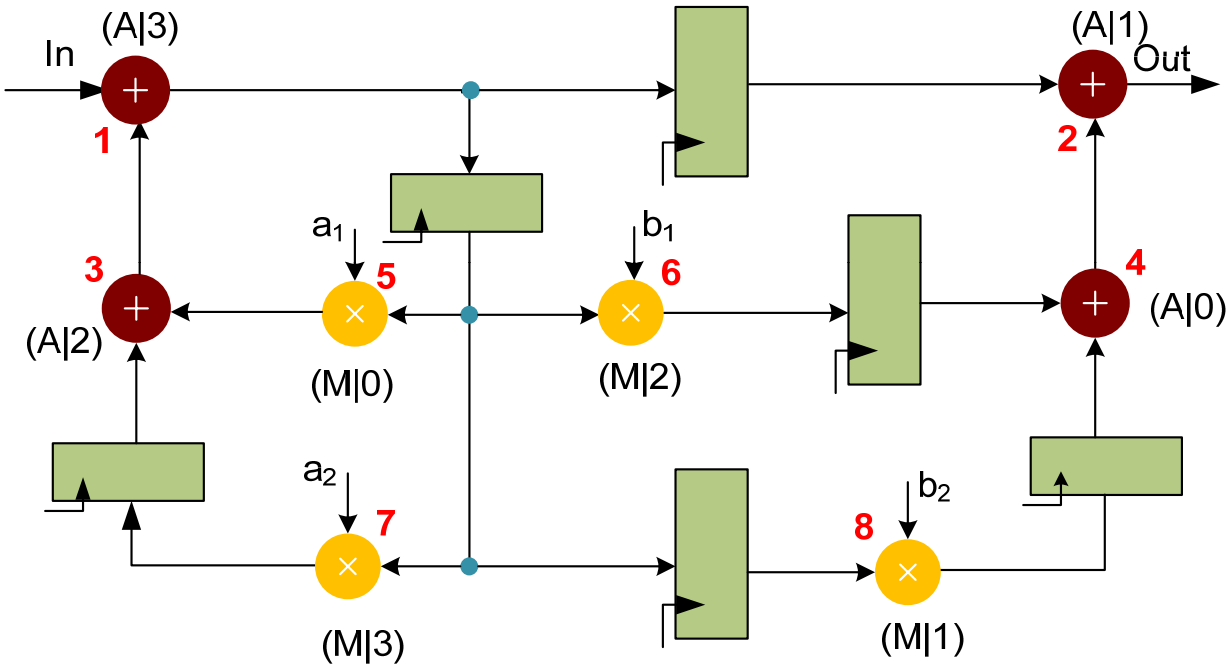
$$F_{42} = 4(0) - 1 + 1 - 0 = 0$$

$$F_{53} = 4(0) - 2 + 2 - 0 = 0$$

$$F_{64} = 4(1) - 2 + 0 - 2 = 0$$

$$F_{73} = 4(1) - 2 + 2 - 3 = 1$$

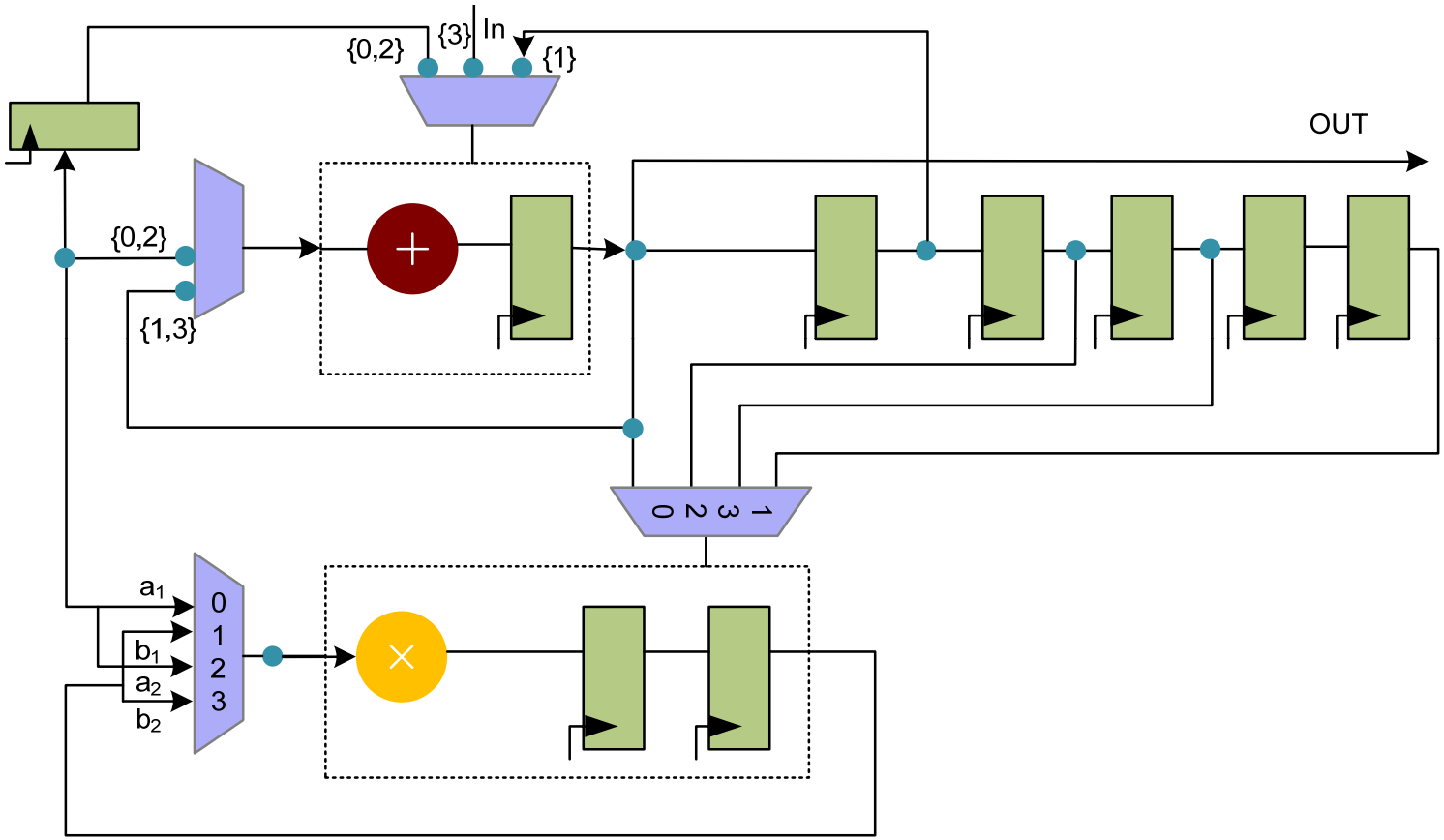
$$F_{84} = 4(1) - 2 + 0 - 1 = 1$$



Valid Folding with pipelining $F_{ij} \geq 0$

Valid Folding without pipelining $F_{ij} \geq 1$

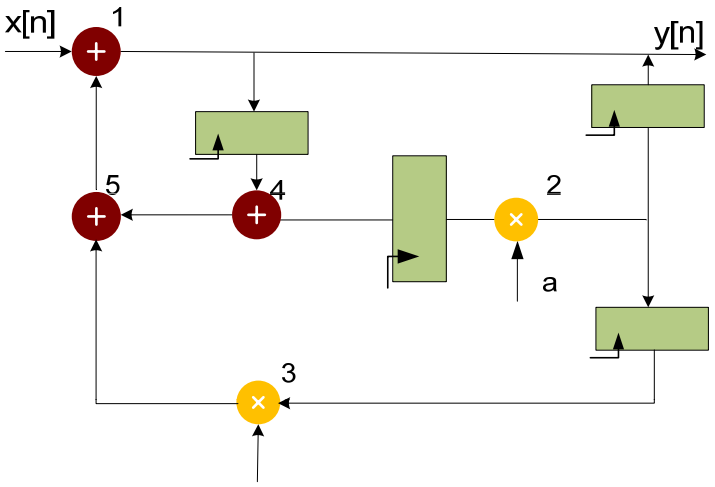
Folding of Biquad Filter N=4

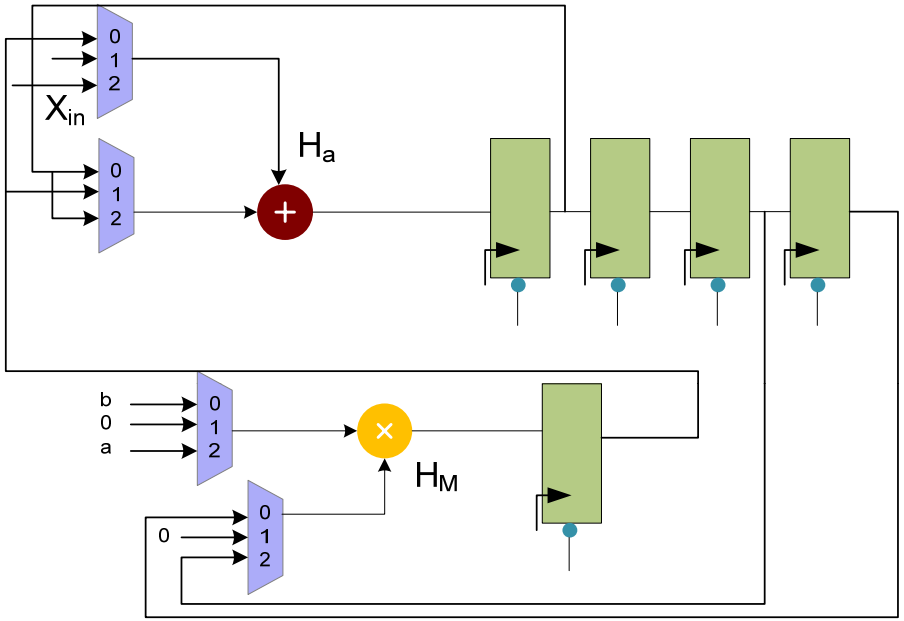


Additions
A = {4,2,3,1}

Multiplication
M = {5,8,6,7}

Example: Folding Transformation





$$F_{ij} = N \times W_{ij} + P_u + v_j - u_i$$

$$F_{12} = 3 + 2 - 2 = 3$$

$$F_{13} = 3 \times 2 + 0 - 2 = 4$$

$$F_{24} = 3 + 0 - 2 = 1$$

$$F_{14} = 3 + 0 - 2 = 1$$

$$F_{45} = 0 + 1 - 0 = 1$$

$$F_{35} = 0 + 1 - 0 = 1$$

$$F_{51} = 0 + 2 - 1 = 1$$

$$P_u = 0 \qquad P_v = 0$$

$$S_a = \{4,5,1\} \qquad S_m = \{3,2\}$$