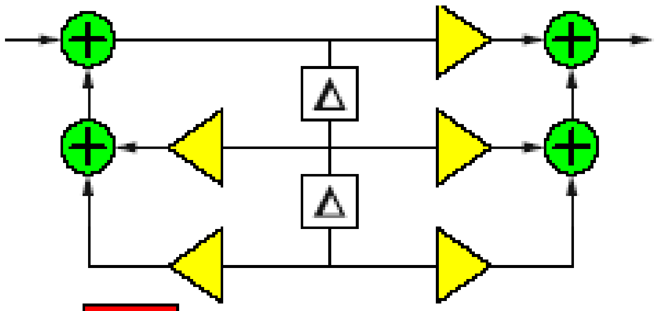
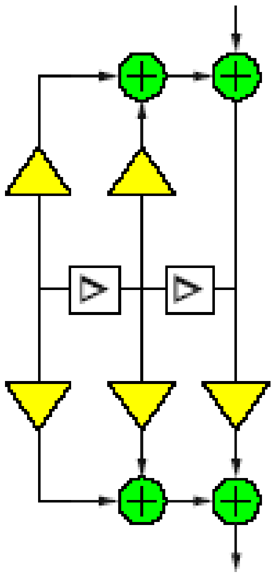
VLSI Signal Processing

Lecture 3 Folding Transformation

Trading Time for Area in DSP



Biquad Filter

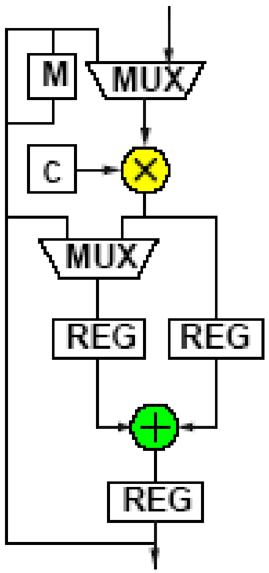


Hardware mapped

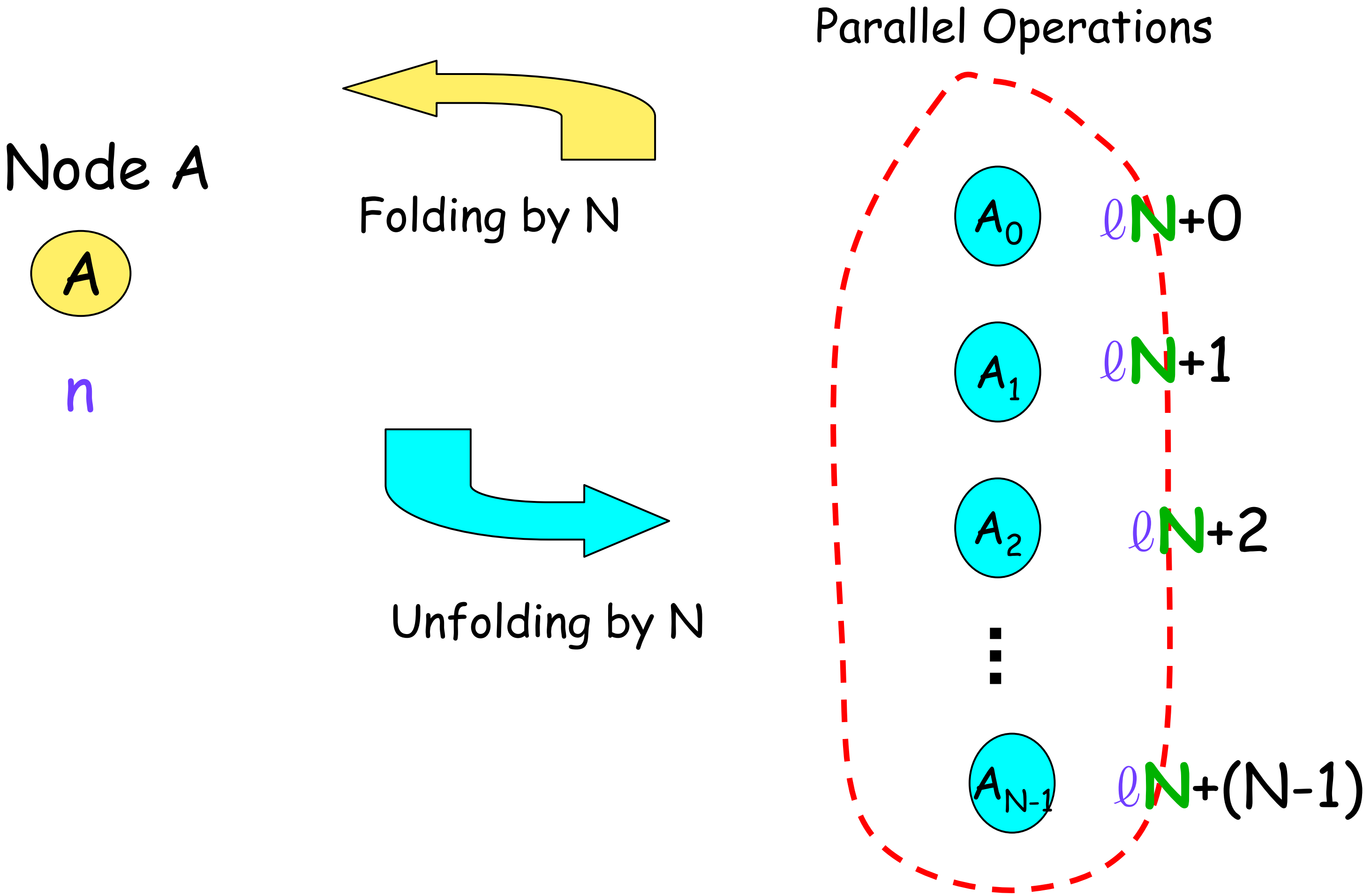
- 5 mult with fixed coeffecients
- 4 adders
- 2 delays
- Latency=1cc

Microcoded

- 1 mult
- 1 adder
- Latency=5cc
- Coeff Memory
- 3 Registers
- Controller

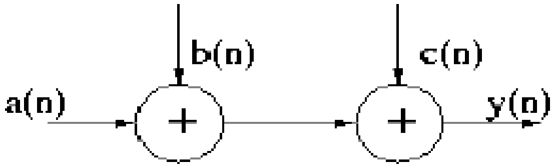


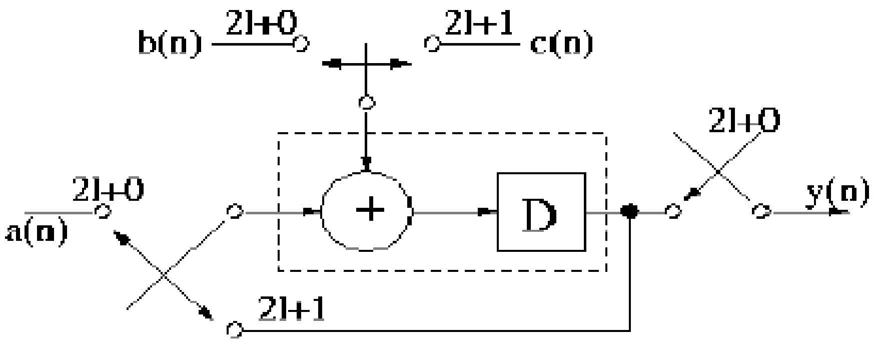
Folding is “Inverse” of Unfolding



Folding is Time-Shared Architecture

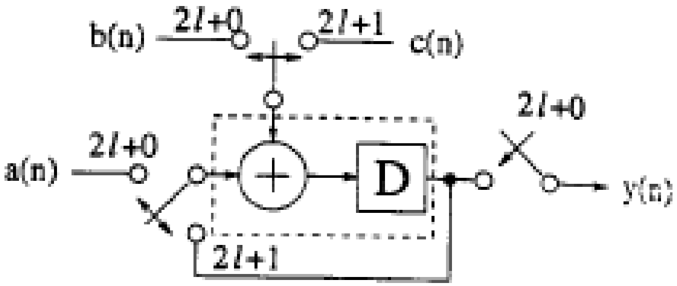
- Folding is a technique to reduce the silicon area by time-multiplexing many operations into single function units
- Folding introduces registers
- Computation time increased





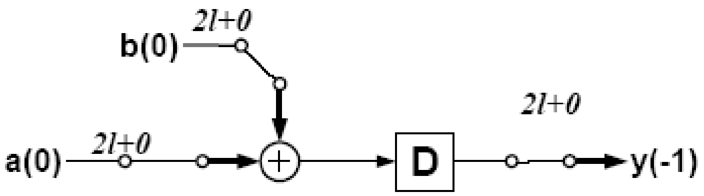
Cycle	Adder Input (Left)	Adder Input (top)	System output
0	a(0)	b(0)	-
1	a(0)+b(0)	c(0)	-
2	a(1)	b(1)	a(0)+b(0)+c(0)
3	a(1)+b(1)	c(1)	-
4	a(2)	b(2)	a(1)+b(1)+c(1)
5	a(2)+b(2)	c(2)	-

Operations in Folding Hardware

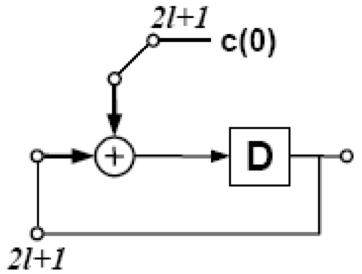


y(n)=a(n)+b(n)+c(n)

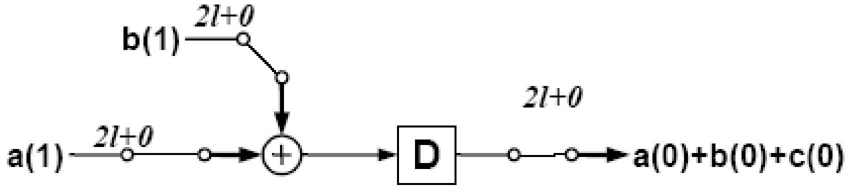
Cycle 0



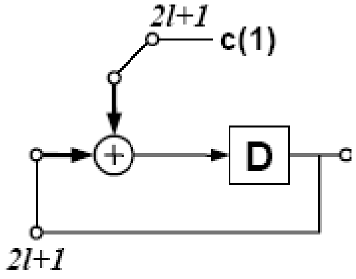
Cycle 1



Cycle 2



Cycle 3

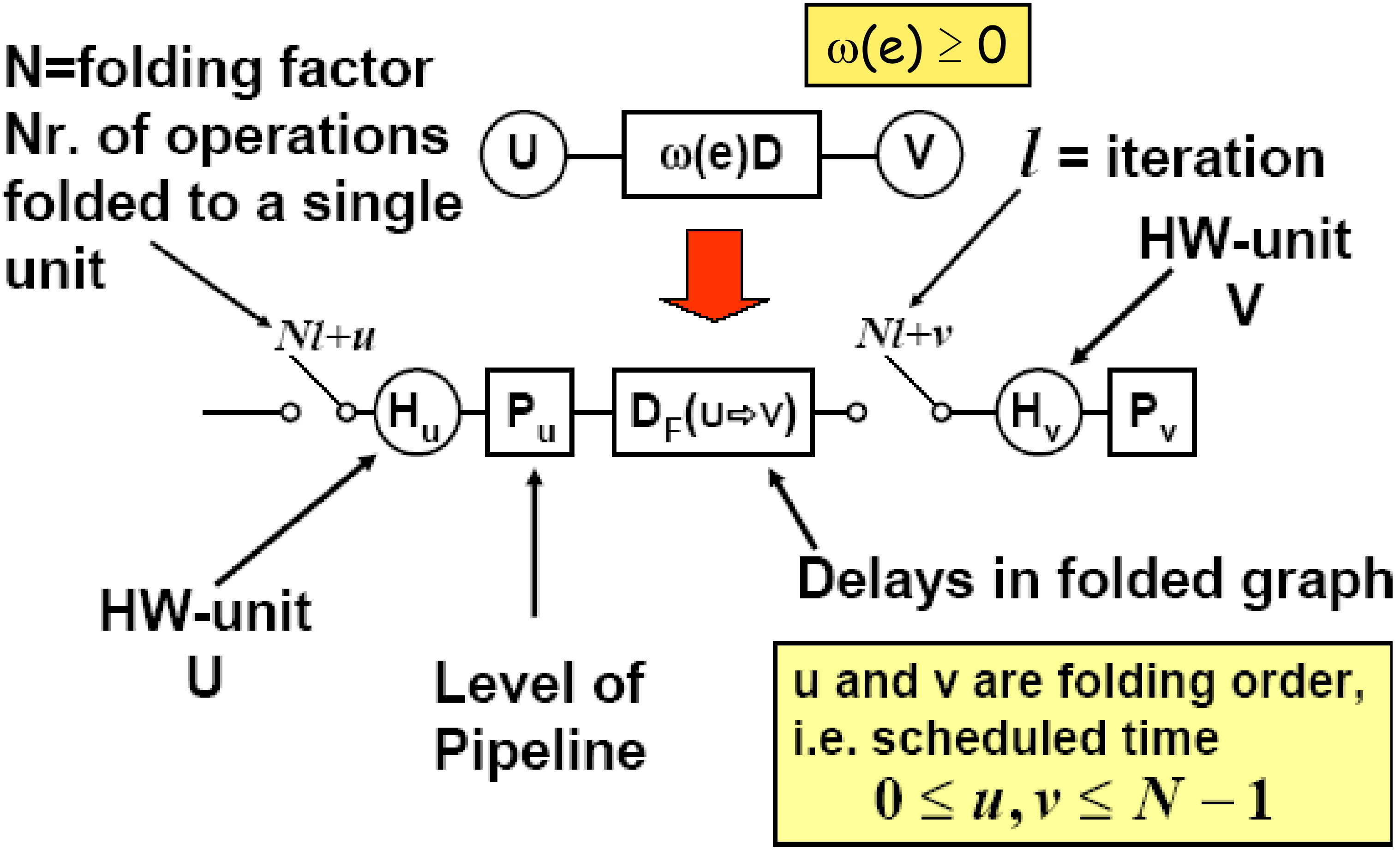


one output/2u.t.

What's Related to Folding

- Reduce hardware by folding factor N
- $T_{computation}$ increased by N times
- Extremes
  - Fully parallel v.s. 1 function unit (or node) per algorithm operation
- Extra registers
- Control unit
- Latency

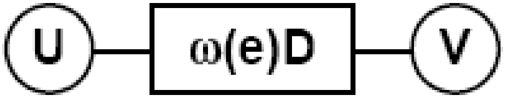
Folding Transformation

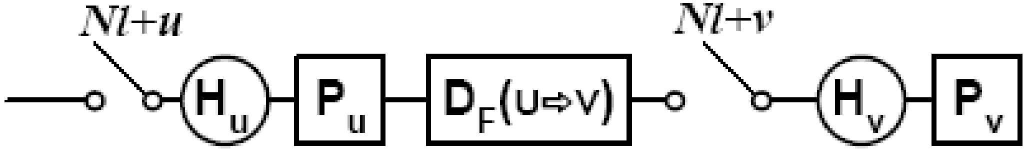


u and v are folding order, i.e. scheduled time

$$0 \leq u, v \leq N-1$$

Delay Calculation





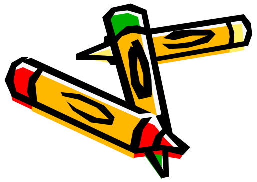
- N is the folding factor (i.e. the number of cycles to perform an iteration)
- $N\ell+u$ and $N\ell+v$ are the time units at which ℓ-th iteration of the nodes U and V are scheduled respectively, and u and v are the time partitions at which the nodes are scheduled to execute and satisfy $0 \le u,v < N$
- H_u is pipelined into P_u stages with output available at $N\ell+u+P_u$
- The ℓ-th iteration of U is used by $(\ell+w(e))$-th iteration of node V, which is executed at $N(l+w(e))+v$. So, the result should be stored for:

$$D_F(UV) = [N(\ell+w(e))+v]-[N\ell+P_u+u]= N\,w(e) -P_u +v-u$$

Independent of ℓ

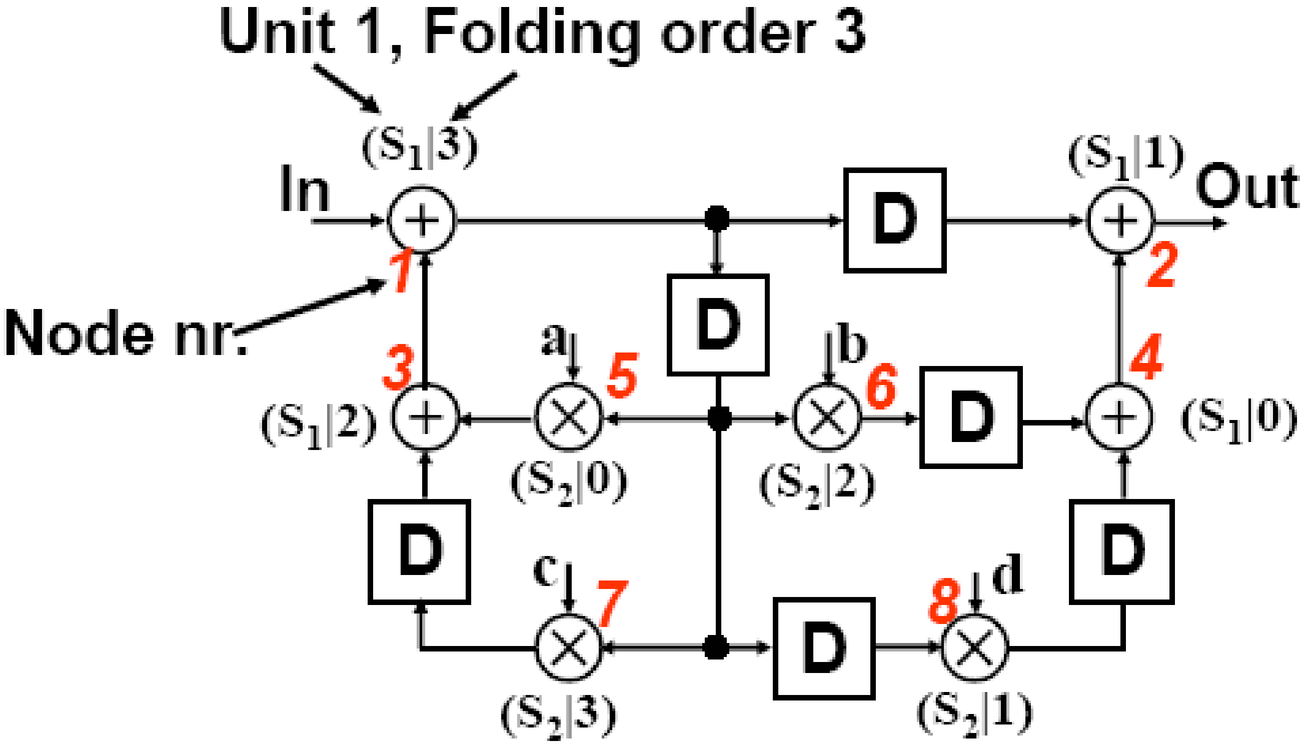
Folding Set Related to Node

- $N\ell+v$, $0\leq v\leq N-1$, related to S → $(S|v)$
- A folding set is an ordered set of operations to be executed on the same node
- Folding set contains N entries
- Example: $S_1=\{A_1,\varnothing,A_2\}$
  - Operation A_1 is executed at time instances 3ℓ, denoted by $(S_1|0)$
  - Operation A_2 is executed at time instances $3\ell+2$, denoted by $(S_1|2\}$
  - $\varnothing$ is null operation



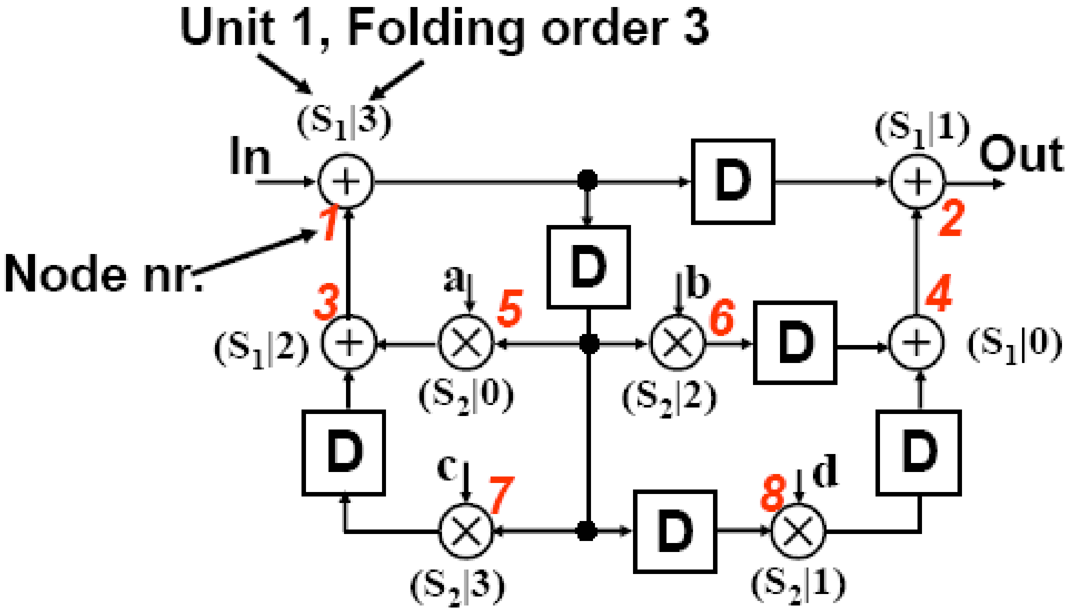
Folding Example

- Biquad Filter



Example

- Fold the biquad filter by N=4



- Assume T_A=1 u.t. and T_M=2 u.t. with 1 and 2 stages of pipelining, respectively (i.e. P_A=1, P_M=2)
- The folding set is S_A={4,2,3,1} and S_M={5,8,6,7}

Folding Equations

$$D_F(U\to V) = Nw(e) - P_u + \overset{\text{receive}}{v} - \overset{\text{send}}{u}$$

$D_F(1\to 2) = 4(1) - 1 + 1 - 3 = 1$

$D_F(1\to 5) = 4(1) - 1 + 0 - 3 = 0$

$D_F(1\to 6) = 4(1) - 1 + 2 - 3 = 2$

$D_F(1\to 7) = 4(1) - 1 + 3 - 3 = 3$

$\boxed{D_F(1\to 8) = 4(2) - 1 + 1 - 3 = 5}$

$D_F(3\to 1) = 4(0) - 1 + 3 - 2 = 0$

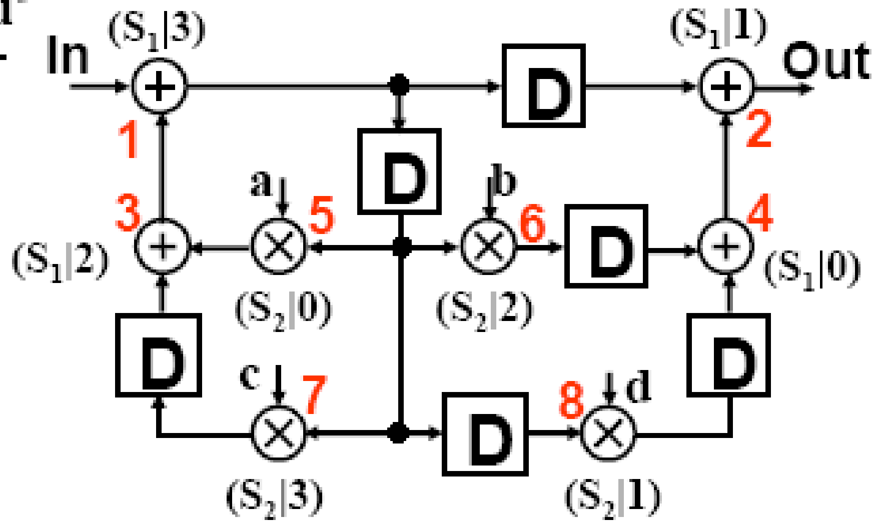
$D_F(4\to 2) = 4(0) - 1 + 1 - 0 = 0$

$D_F(5\to 3) = 4(0) - 2 + 2 - 0 = 0$

$D_F(6\to 4) = 4(1) - 2 + 0 - 2 = 0$

$D_F(7\to 3) = 4(1) - 2 + 2 - 3 = 1$

$D_F(8\to 4) = 4(1) - 2 + 0 - 1 = 1$



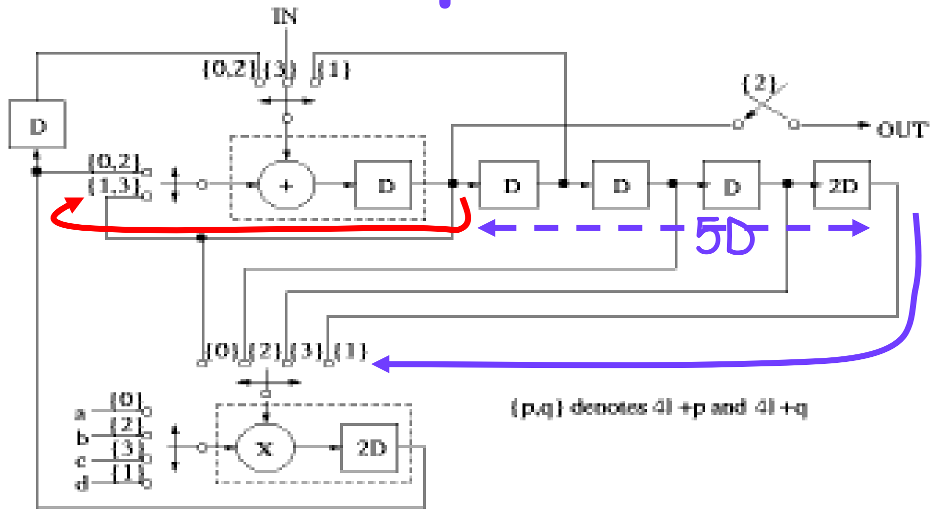
Valid folding

$$D_F(U\to V) \geq 0$$

Folded Biquad Filter

Additions
$S_1=\{4,2,3,1\}$

Multiplications
$S_1=\{5,8,6,7\}$



Folding equations for each of the 11 edges are as follows:

$D_F(1\rightarrow2) = 4(1) - 1 + 1 - 3 = 1$

$D_F(1\rightarrow6) = 4(1) - 1 + 2 - 3 = 2$

$\boxed{D_F(1\rightarrow8) = 4(2) - 1 + 1 - 3 = 5}$

$D_F(4\rightarrow2) = 4(0) - 1 + 1 - 0 = 0$

$D_F(6\rightarrow4) = 4(1) - 2 + 0 - 2 = 0$

$D_F(8\rightarrow4) = 4(1) - 2 + 0 - 1 = 1$

$D_F(1\rightarrow5) = 4(1) - 1 + 0 - 3 = 0$

$D_F(1\rightarrow7) = 4(1) - 1 + 3 - 3 = 3$

$\boxed{D_F(3\rightarrow1) = 4(0) - 1 + 3 - 2 = 0}$

$D_F(5\rightarrow3) = 4(0) - 2 + 2 - 0 = 0$

$D_F(7\rightarrow3) = 4(1) - 2 + 2 - 3 = 1$

Example

$$D_F(U\rightarrow V) = Nw(e) - P_u + \overset{\text{receive}}{v} - \overset{\text{send}}{u}$$

$D_F(1\rightarrow2) = 4(1) - 1 + 1 - 3 = 1$

$D_F(1\rightarrow5) = 4(1) - 1 + 0 - 3 = 0$

$D_F(1\rightarrow6) = 4(1) - 1 + 2 - 3 = 2$

$D_F(1\rightarrow7) = 4(2) - 1 + 3 - 3 = 7$

$D_F(1\rightarrow8) = 4(2) - 1 + 1 - 3 = 5$

$D_F(3\rightarrow1) = 4(0) - 1 + 3 - 2 = 0$

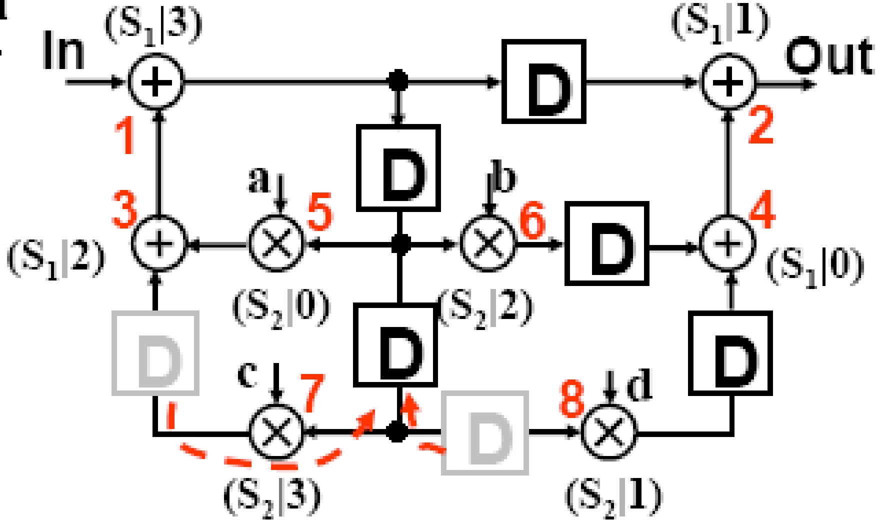
$D_F(4\rightarrow2) = 4(0) - 1 + 1 - 0 = 0$

$D_F(5\rightarrow3) = 4(0) - 2 + 2 - 0 = 0$

$D_F(6\rightarrow4) = 4(1) - 2 + 0 - 2 = 0$

$D_F(7\rightarrow3) = 4(0) - 2 + 2 - 3 = -3$

$D_F(8\rightarrow4) = 4(1) - 2 + 0 - 1 = 1$



Not valid folding

$$D_F(U\rightarrow V)<0$$

Folding of Biquad Filter

receive send

$$D_F(U \to V) = Nw(e) - P_u + v - u$$

$D_F(1\to2) = -3$

$D_F(1\to5) = 0$

$D_F(1\to6) = 2$

$D_F(1\to7) = 7$

$D_F(1\to8) = 5$

$D_F(3\to1) = 0$

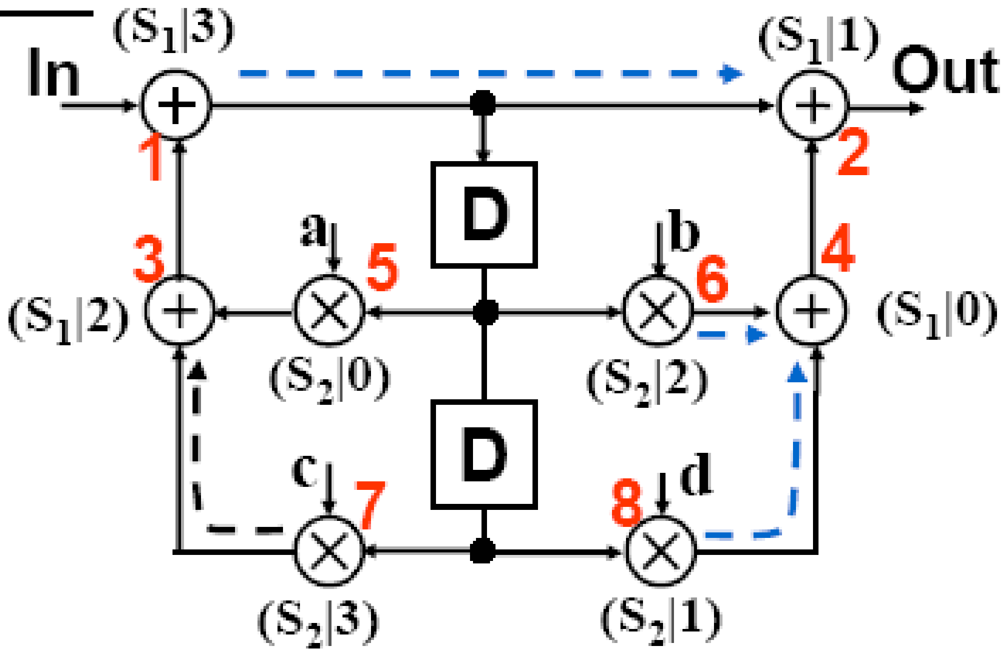
$D_F(4\to2) = 0$

$D_F(5\to3) = 0$

$D_F(6\to4) = -4$

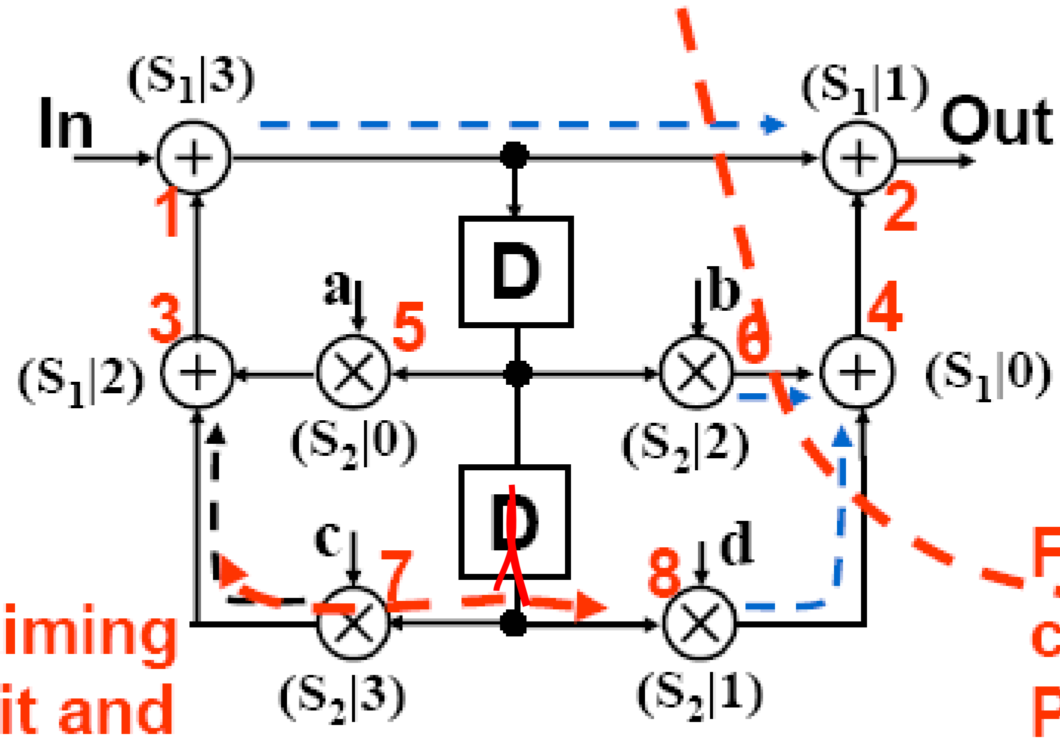
$D_F(7\to3) = -3$

$D_F(8\to4) = -3$



$D_F(U \to V) < 0$ → Not Valid folding

Retiming



Retiming
Split and
move delay

Feedforward
cutset ⇨
Pipelining

Folding Retimed Biquad Filter

$$D_F(U\rightarrow V) = Nw(e) - P_u + \overset{\text{receive}}{v} - \overset{\text{send}}{u}$$

$D_F(1\rightarrow 2) = 4(1) - 1 + 1 - 3 = \boxed{1}$

$D_F(1\rightarrow 5) = 4(1) - 1 + 0 - 3 = 0$

$D_F(1\rightarrow 6) = 4(1) - 1 + 2 - 3 = 2$

$D_F(1\rightarrow 7) = 4(1) - 1 + 3 - 3 = 3$

$D_F(1\rightarrow 8) = 4(2) - 1 + 1 - 3 = 5$

$D_F(3\rightarrow 1) = 4(0) - 1 + 3 - 2 = 0$

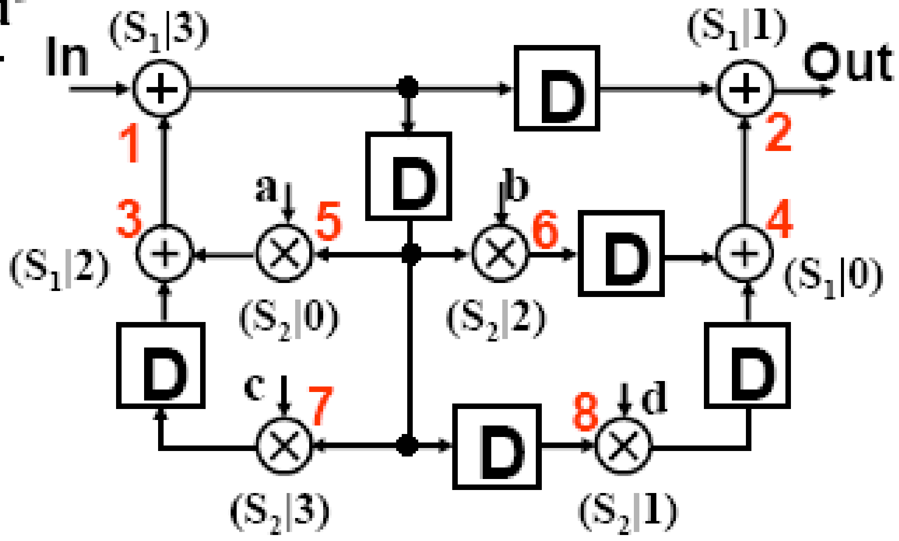
$D_F(4\rightarrow 2) = 4(0) - 1 + 1 - 0 = 0$

$D_F(5\rightarrow 3) = 4(0) - 2 + 2 - 0 = 0$

$D_F(6\rightarrow 4) = 4(1) - 2 + 0 - 2 = \boxed{0}$

$D_F(7\rightarrow 3) = 4(1) - 2 + 2 - 3 = \boxed{1}$

$D_F(8\rightarrow 4) = 4(1) - 2 + 0 - 1 = \boxed{1}$



Valid folding

$$D_F(U\rightarrow V)\geq 0$$

Causalization: Retiming for Folding

- A folded architecture is realizable if and only if all delays $D_F(UV)$ are non-negative
- Retiming for folding
  - All D'_F (UV) of the retimed graph G' are non-negative
  - $N\ w_r(e)–P_U+v–u \geq 0$ … where $w_r(e) = w(e)+r(V)-r(U)$

    $N(w(e)+r(V)–r(U))-P_U+v–u \geq 0$

    $r(U)–r(V) \leq D_F\ (UV)\ /\ N \leq \lfloor D_F\ (UV)\ /\ N \rfloor$

Retiming values are integers

Register Minimization

- Folding may insert registers.
- Lifetime analysis is used for register minimization techniques in a DSP hardware
  - A variable is live from the time it is produced until the time it is consumed. After then, it is dead.
- Linear lifetime chart: represents the lifetime of the variables in a linear fashion.
- Max. number of live variables in linear lifetime chart ➔ Min. number of registers in implementation

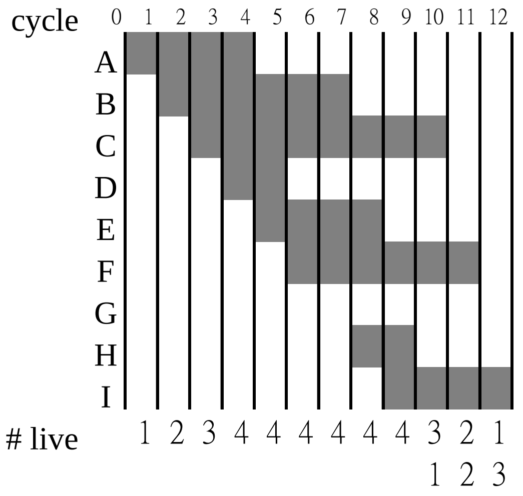
Data Format Converter

- e.g. 3-by-3 Matrix transposition
  - input sequence: ABCDEFGHI
  - output sequence: ADGBEHCFI

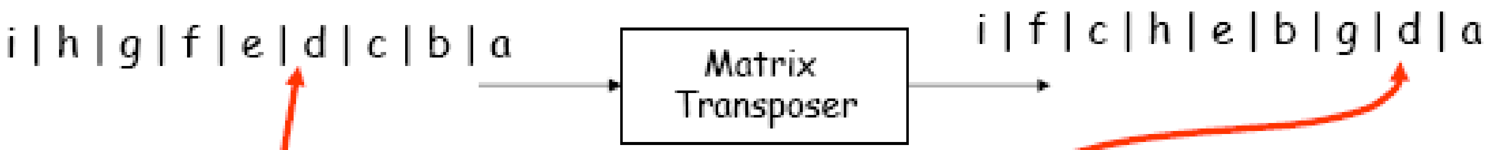
- **Step 1**: lifetime analysis

Sample	Tinput	Tzlout	Tdiff	Toutput	Life Period
A	0	0	0	4	0 ~ 4
B	1	3	2	7	1 ~ 7
C	2	6	4	10	2 ~ 10
D	3	1	-2	5	3 ~ 5
E	4	4	0	8	4 ~ 8
F	5	7	2	11	5 ~ 11
G	6	2	-4	6	6 ~ 6
H	7	5	-2	9	7 ~ 9
I	8	8	0	12	8 ~ 12

zero-latency output



Lifetime Table


i | h | g | f | e | d | c | b | a
Matrix Transposer
i | f | c | h | e | b | g | d | a

Sample	T_{in}	T_{zlout}	T_{diff}	T_{out}	Life
a	0	0	0	4	0→4
b	1	[illegible]	[illegible]	7	1→7
c	2	[illegible]	[illegible]	10	2→10
d	3	1	-2	5	3→5
e	4	4	0	8	4→8
f	5	7	2	11	5→11
g	6	2	-4	6	6→6
h	7	5	-2	9	7→9
i	8	8	0	12	8→12

Out before In

$T_{diff} = T_{zlout} - T_{input}$, T_{zlout} = zero latency

Forward Backward Register Allocation

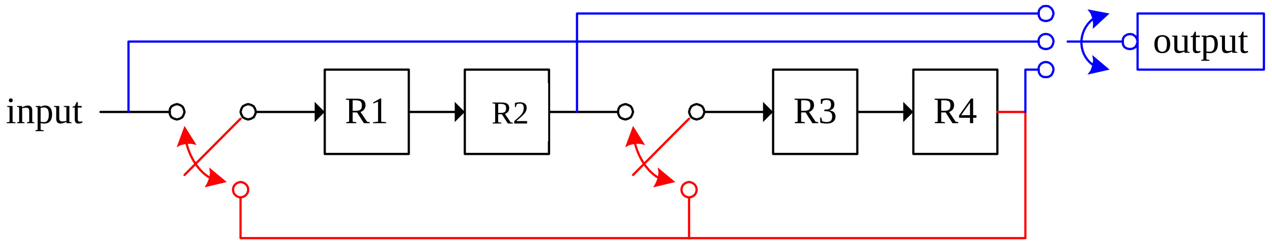
Steps for Forward-Backward Register allocation :

1. Determine the minimum number of registers using lifetime analysis.
2. Input each variable at the time step corresponding to the beginning of its lifetime. If multiple variables are input in a given cycle, these are allocated to multiple registers with preference given to the variable with the longest lifetime.
3. Each variable is allocated in a forward manner until it is dead or it reaches the last register. In forward allocation, if the register i holds the variable in the current cycle, then register i + 1 holds the same variable in the next cycle. If (i + 1)-th register is not free then use the first available forward register.
4. Being periodic the allocation repeats in each iteration. So hash out the register R_j for the cycle l + N if it holds a variable during cycle l.
5. For variables that reach the last register and are still alive, they are allocated in a backward manner on a first come first serve basis.
6. Repeat steps 4 and 5 until the allocation is complete.

Step 2: forward-backward register allocation

cycle	input	R1	R2	R3	R4	output
0	a					
1	b	a				
2	c	b	a			
3	d	c	b	a		
4	e	d	c	b	a	a
5	f	e	d	c	b	d
6	g	f	e		c	g
7	h		f	e		
8	i	h		f	e	e
9		i	h		f	h
10		▨	i			
11		▨	▨	i		
12		▨	▨	▨	i	i

cycle	input	R1	R2	R3	R4	output
0	a					
1	b	a				
2	c	b	a			
3	d	c	b	a		
4	e	d	c	b	a	a
5	f	e	d	c	b	d
6	g	f	e	b	c	g
7	h	c	f	e	b	b
8	i	h	c	f	e	e
9		i	h	c	f	h
10		▨	i	f	c	c
11		▨	▨	i	f	f
12		▨	▨	▨	i	i



Register Minimization

- Register minimization in folded architectures :
  - Perform retiming for folding
  - Write the folding equations
  - Use the folding equations to construct a lifetime table
  - Draw the lifetime chart and determine the required number of registers
  - Perform forward-backward register allocation
  - Draw the folded architecture that uses the minimum number of registers.

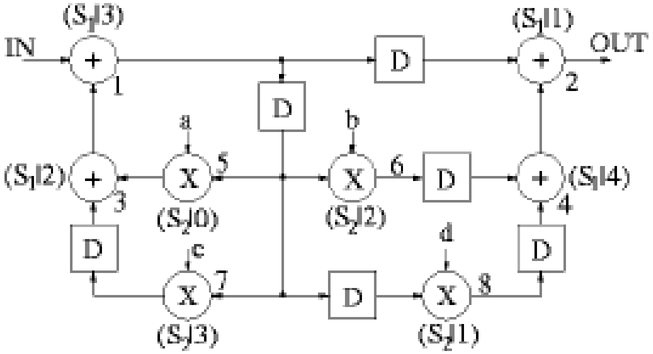
- Example : Biquad Filter
  - Steps 1 & 2 have already been done.
  - Step 3:The lifetime table is then constructed. The 2nd row is empty as $D_F(2 \rightarrow U)$ is not present.

Note : As retiming for folding ensures causality, we need not add any latency.

Node	$T_{in} \rightarrow T_{out}$
1	4→9
2	--
3	3→3
4	1→1
5	2→2
6	4→4
7	5→6
8	3→4

Register Minimization of Biquad Filter

Node	$T_{in} \rightarrow T_{out}$
1	4→9
2	--
3	3→3
4	1→1
5	2→2
6	4→4
7	5→6
8	3→4

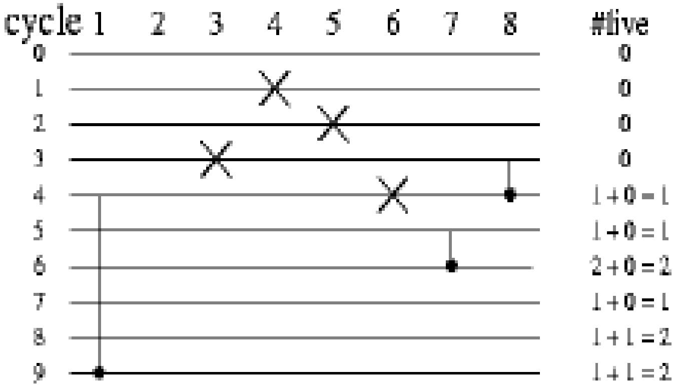


$D_F(1\rightarrow2) = 4(1) - 1 + 1 - 3 = 1$
$D_F(1\rightarrow5) = 4(1) - 1 + 0 - 3 = 0$
$D_F(1\rightarrow6) = 4(1) - 1 + 2 - 3 = 2$
$D_F(1\rightarrow7) = 4(1) - 1 + 3 - 3 = 3$
$D_F(1\rightarrow8) = 4(2) - 1 + 1 - 3 = 5$
$D_F(3\rightarrow1) = 4(0) - 1 + 3 - 2 = 0$
$D_F(4\rightarrow2) = 4(0) - 1 + 1 - 0 = 0$
$D_F(5\rightarrow3) = 4(0) - 2 + 2 - 0 = 0$
$D_F(6\rightarrow4) = 4(1) - 2 + 0 - 2 = 0$
$D_F(7\rightarrow3) = 4(1) - 2 + 2 - 3 = 1$
$D_F(8\rightarrow4) = 4(1) - 2 + 0 - 1 = 1$

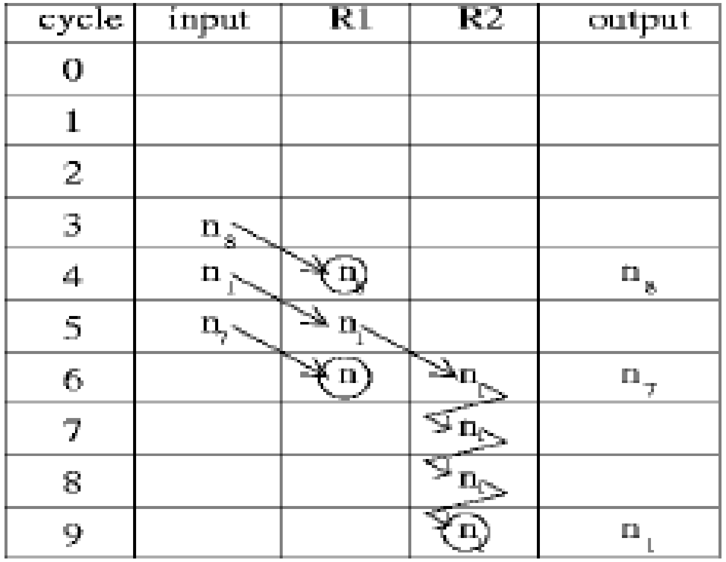
One entry for each node:

- $T_{input} = u + P_u$, u=folding order, P_u=pipeline time unit data is produced
- $T_{output} = u + P_u + \max_V\{D_F(U \rightarrow V)\}$,
  $\max_V\{D_F(U \rightarrow V)\}$ = (longest folded path)

- Step 4 : Lifetime chart is constructed and registers determined.

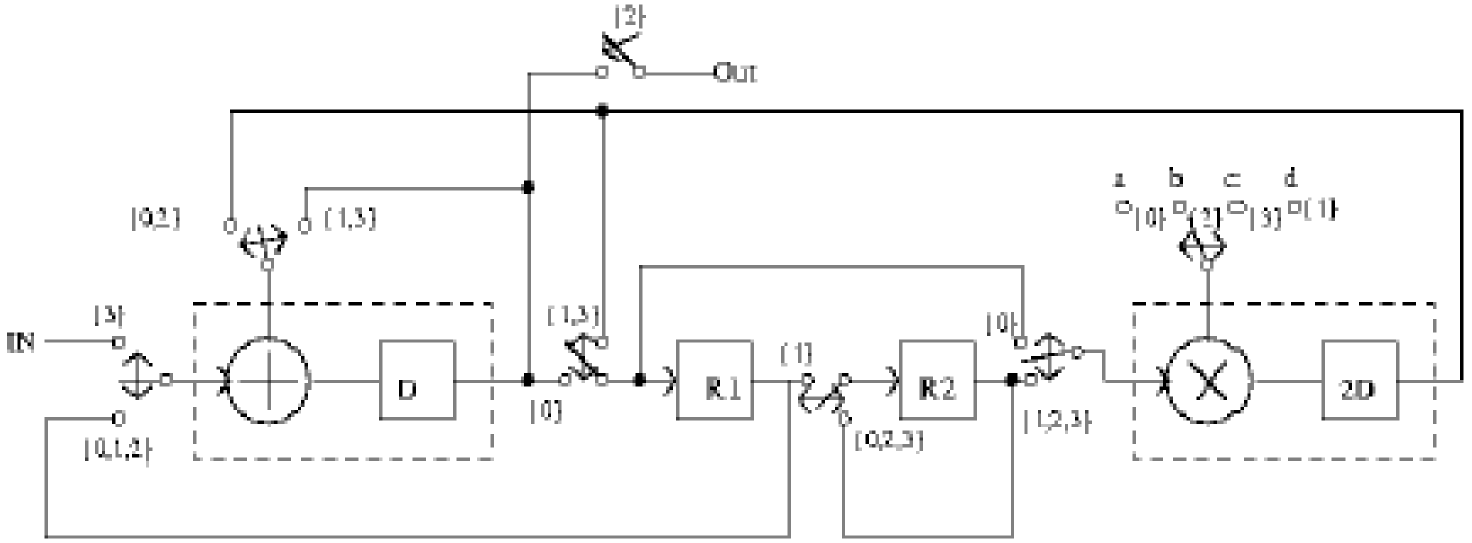


- Step 5 : Forward-backward register allocation

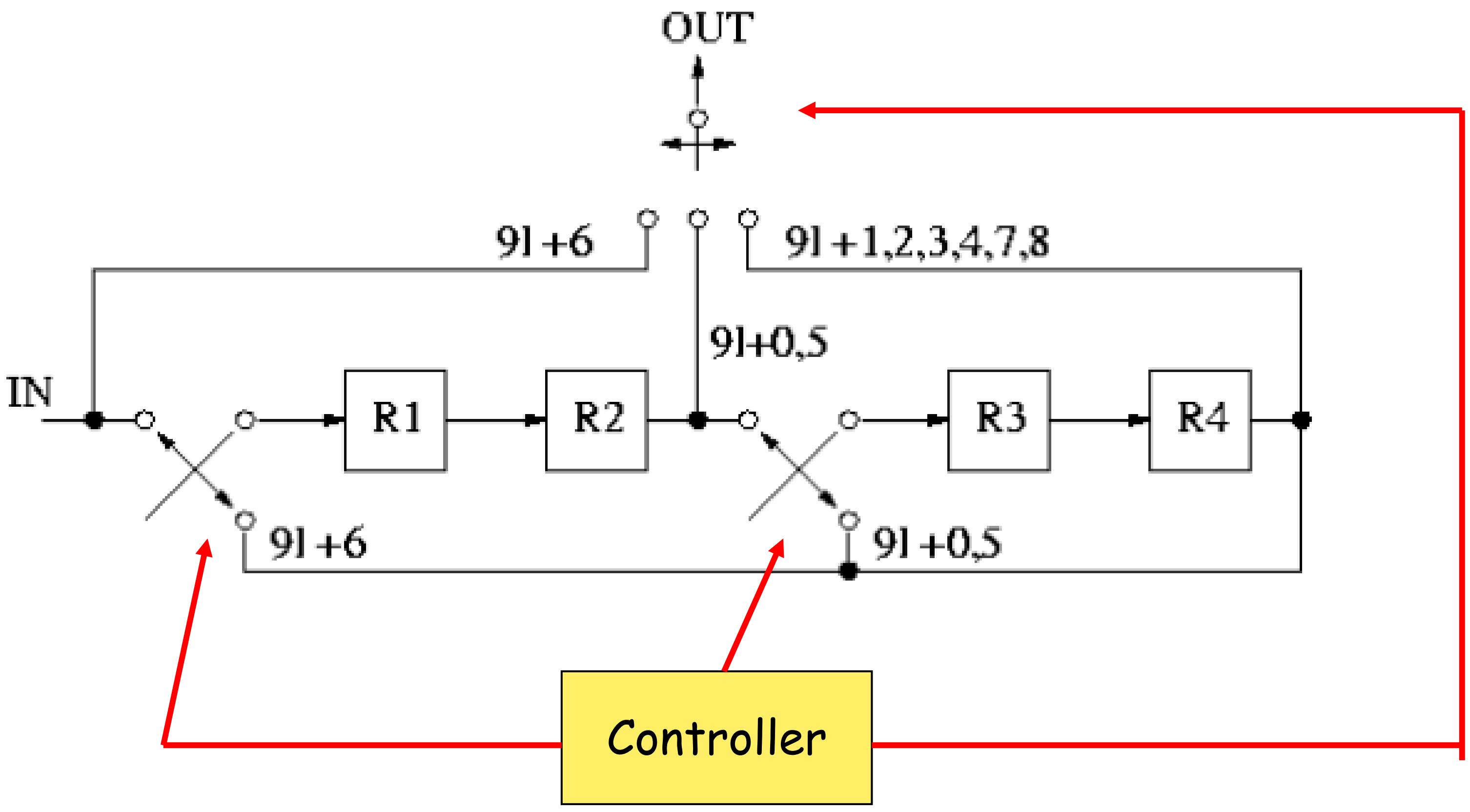


cycle	input	R1	R2	output
0				
1				
2				
3	n_8			
4	n_1	n		n_8
5	n_7	n_1		
6		n	n	n_7
7			n	
8			n	
9			n	n_1

Register Minimization for Biquad Filter

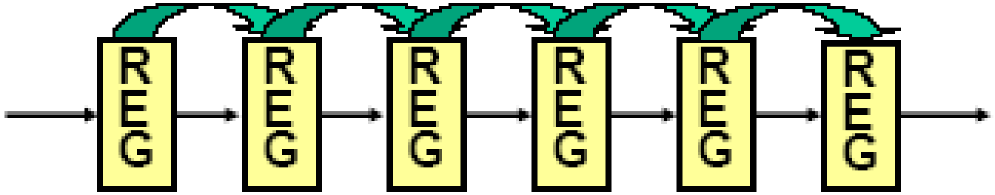


Controller for Folded Architecture



Remarks

- Shift Registers (or FIFO)

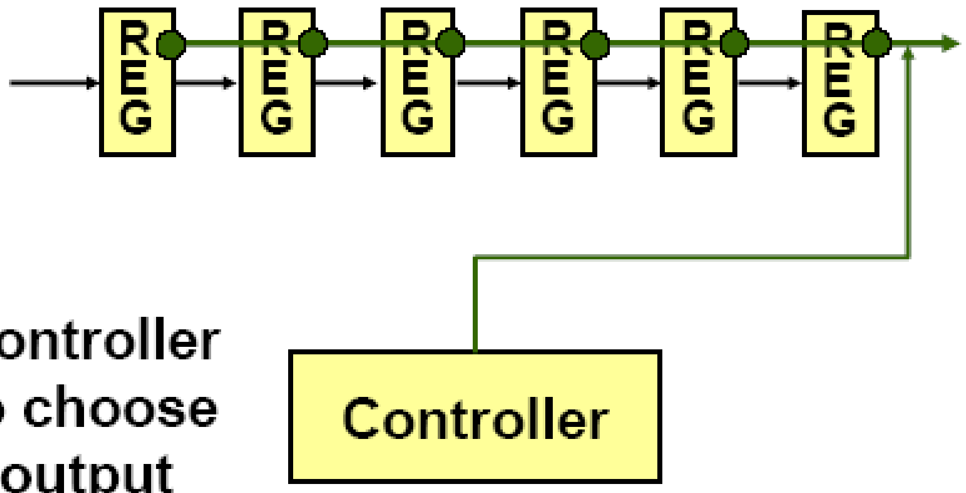


Latency = (# of Regs) clock cycles

Moving data consumes power

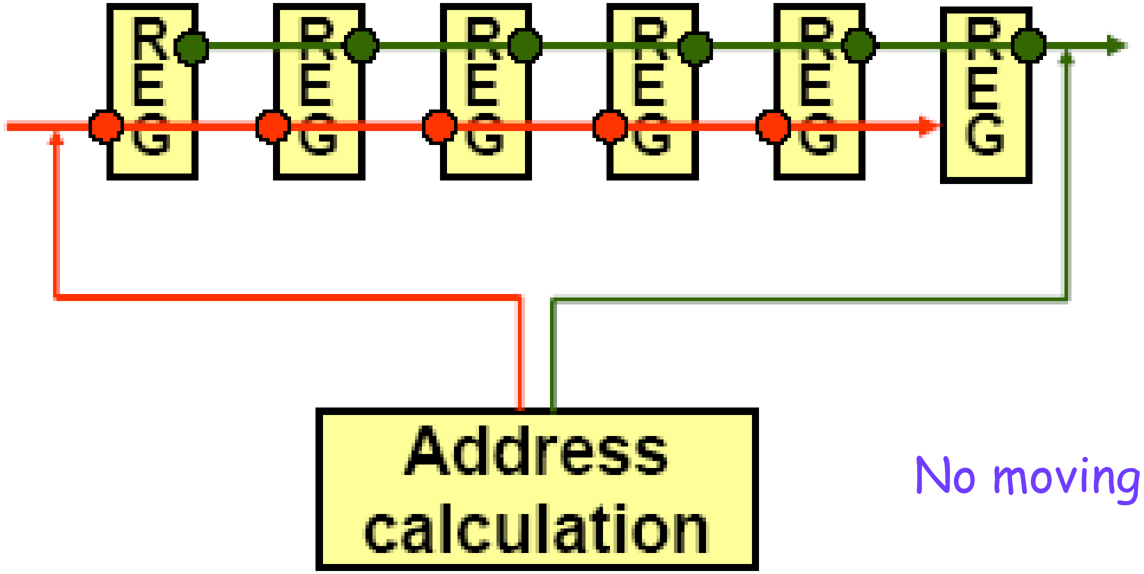
Connecting Outputs FIFO

- Connecting outputs to prevent from moving data (also to reduce latency)



Controller to choose output

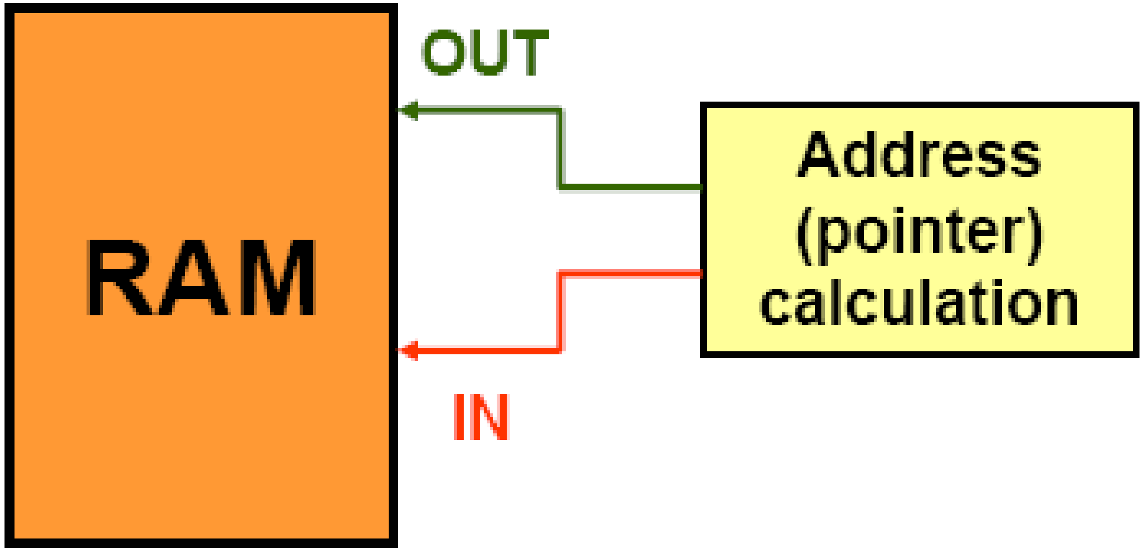
Connecting Inputs/Outputs FIFO



No moving of data

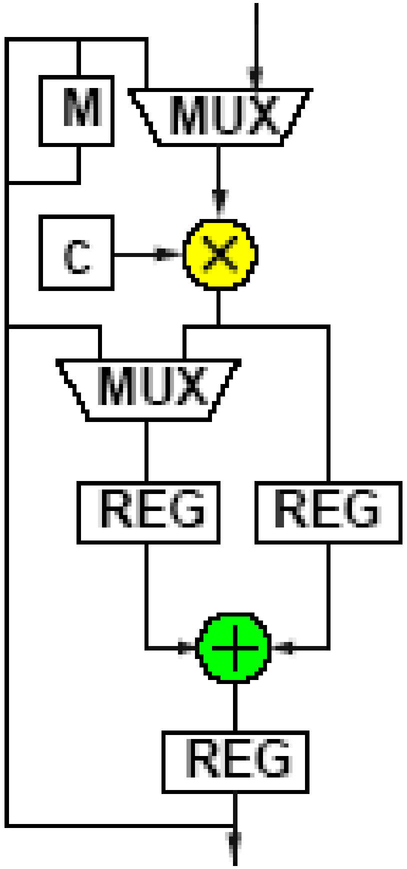
RAM

FIFO with Pointer



No moving of data
but complexity in address calculation

Implementation Using Memory



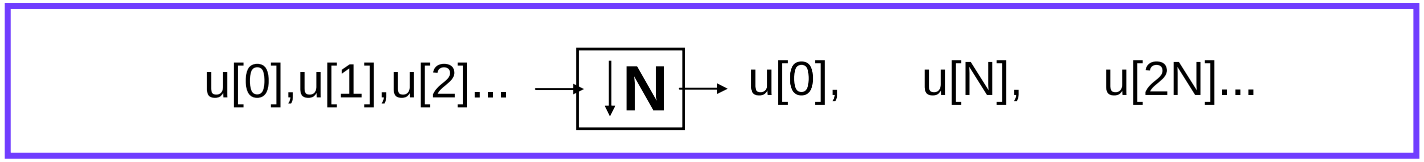
- Data Generation ?

Hardware Slowdown

- N active clock edges lead one sample ahead (one iteration), which is implemented using N-cascaded registers.
- N independent data streams can be interleaved into the N-slowdown hardware. (e.g. 2-channel stereo audio can share the same pre-filter hardware by hardware slowdown)
- The slowdown operations can be viewed as folding N identical hardware into a single one; i.e. the w(e) is multiplied by N.

Review of Multi-rate Systems

- Decimation : decimator (downsampler)



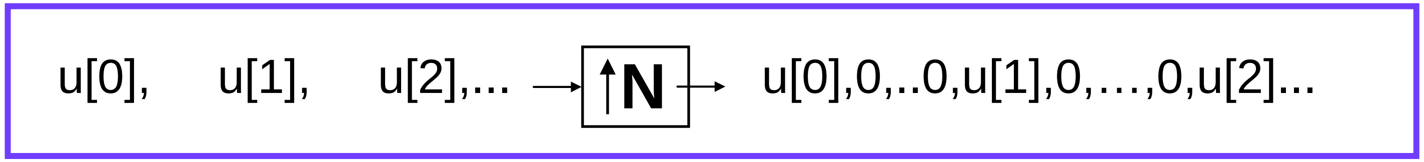
$u[0],u[1],u[2]\ldots \rightarrow \boxed{\downarrow N} \rightarrow u[0],\quad u[N],\quad u[2N]\ldots$

$y_D[n]=u[Nn]$

example : u[k]: 1,2,3,4,5,6,7,8,9,…

2-fold downsampling: 1,3,5,7,9,...

- Interpolation : expander (upsampler)



$u[0],\quad u[1],\quad u[2],\ldots \rightarrow \boxed{\uparrow N} \rightarrow u[0],0,..0,u[1],0,\ldots,0,u[2]\ldots$

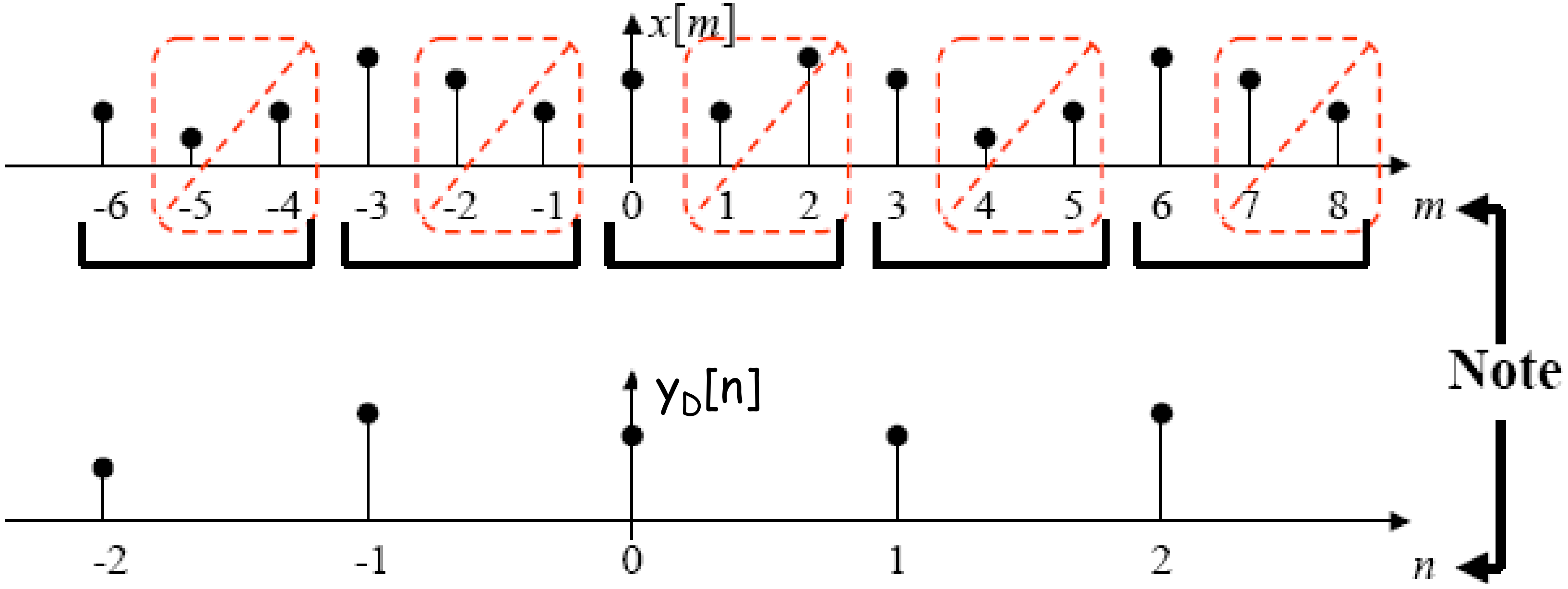
example : u[k]: 1,2,3,4,5,6,7,8,9,…

2-fold upsampling: 1,0,2,0,3,0,4,0,5,0...

$y_E[n]=u[n/N]$, if $N|n$

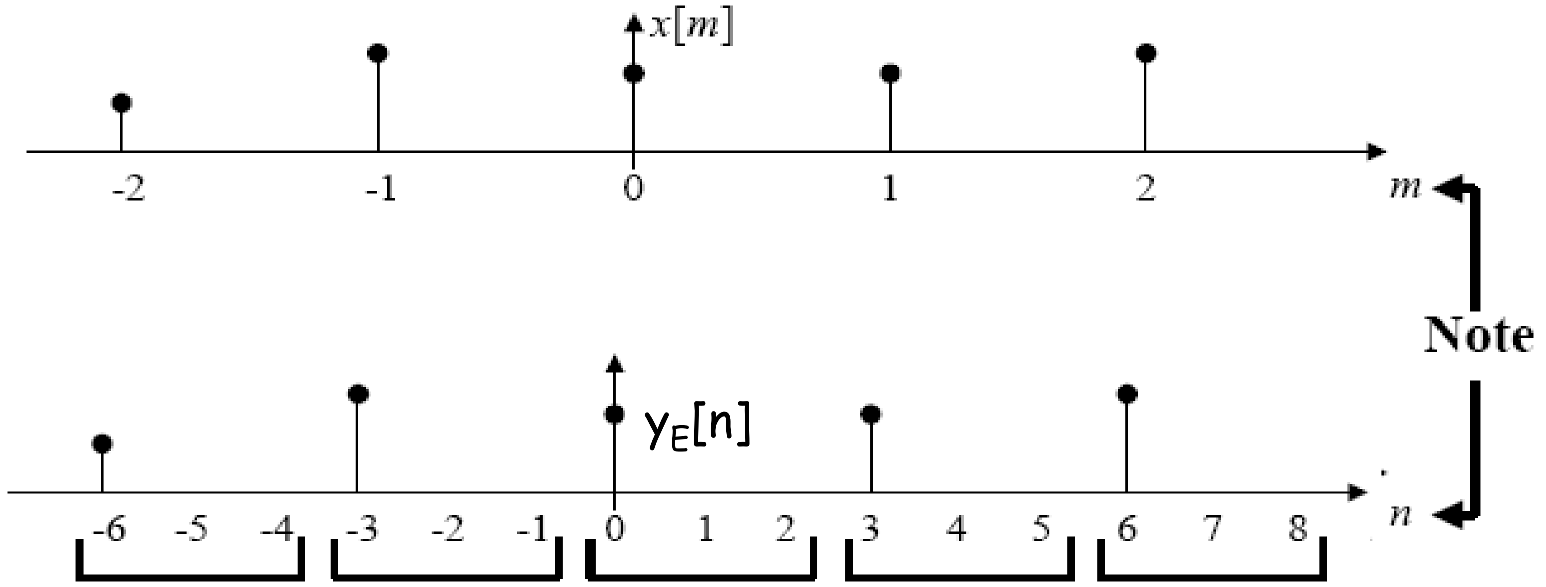
$y_E[n]=0$, otherwise

Decimation by M



Time domain representation $y_D[n]=u[Nn]$

Interpolation by L



Time domain representation

$y_E[n]=u[n/N]$, if N|n
$y_E[n]=0$, otherwise

Filter Banks Introduction

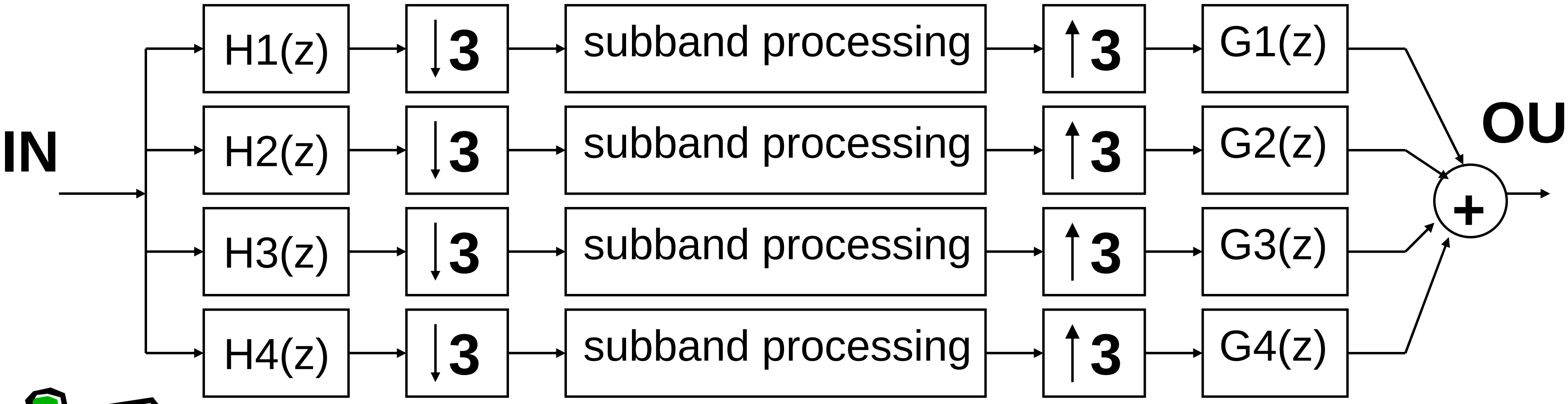
General `subband processing' set-up/overview:

-Signals split into frequency channels/subbands (`analysis bank')

-Per-channel/subband processing

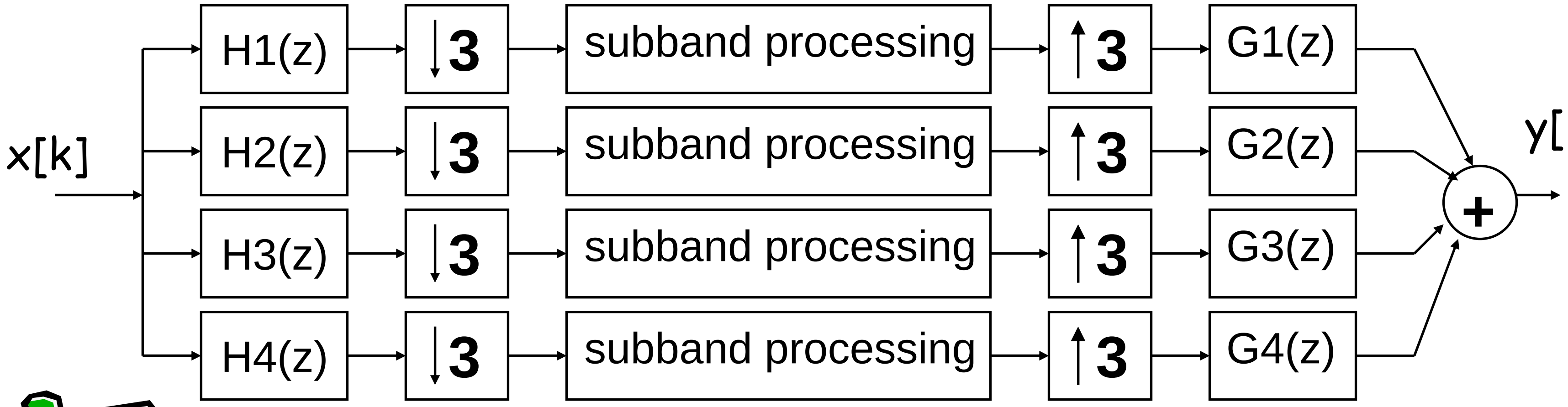
-Reconstruction (`synthesis bank')

-Multi-rate structure: down-sampling / up-sampling



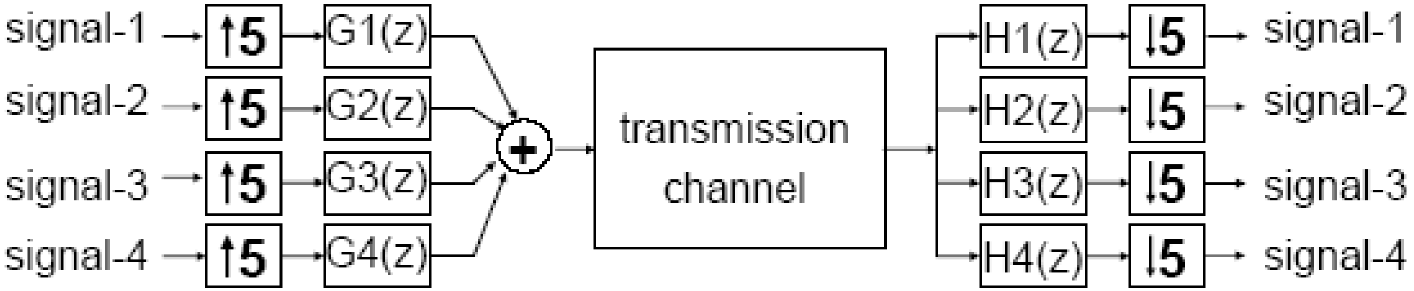
Perfect Reconstruction

- Assume subband processing does not modify subband signals (lossless coding/decoding)
- The overall aim could be to have y[k]=x[k-d], i.e. the output signal is just the input signal up to a certain delay
- But downsampling may introduce aliasing



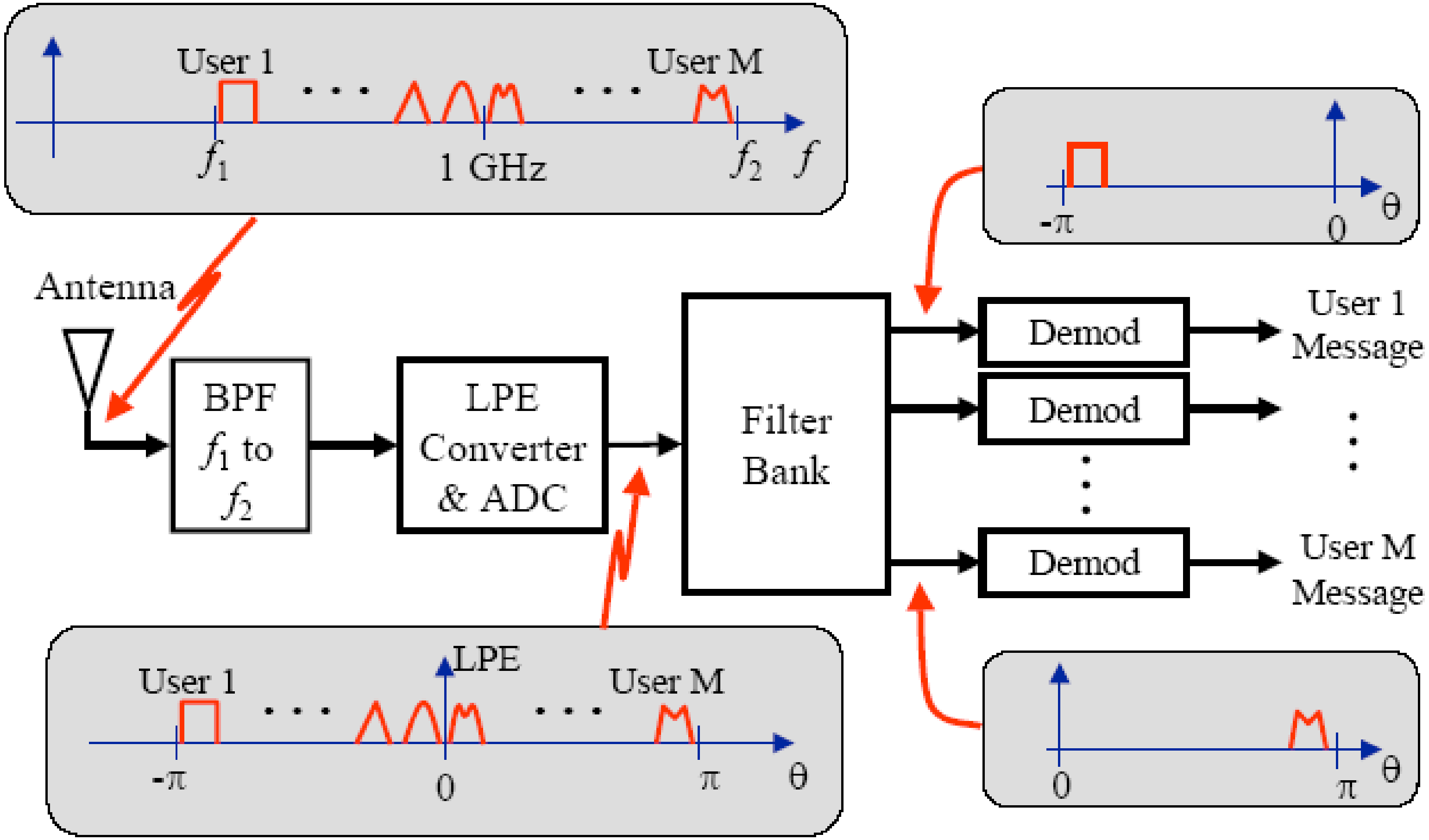
Example: FDM

- Frequency division multiplexing
  - M different source signals multiplexed into 1 transmit signal by expanders & synthesis filters
  - Received signal decomposed into M source signals by analysis filters & decimators (N ≥ M)

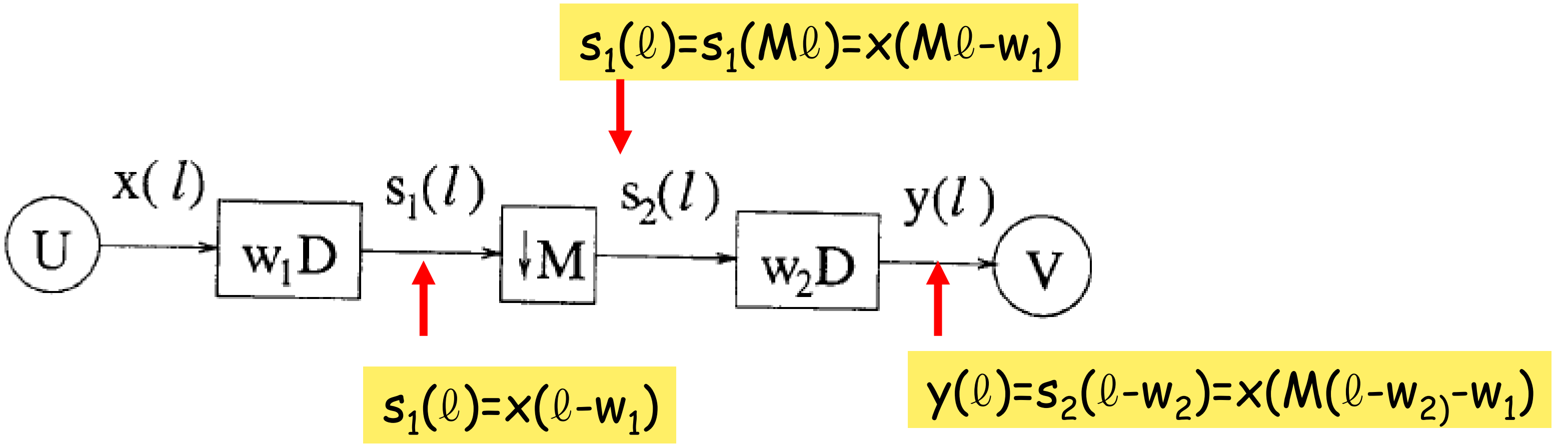


  - Non-ideal synthesis & analysis filters results in cross-talk between channels.

FDM


User 1
User M
f_1
1 GHz
f_2
f
Antenna
BPF f_1 to f_2
LPE Converter & ADC
Filter Bank
Demod
Demod
Demod
User 1 Message
User M Message
-π
0
θ
LPE
π

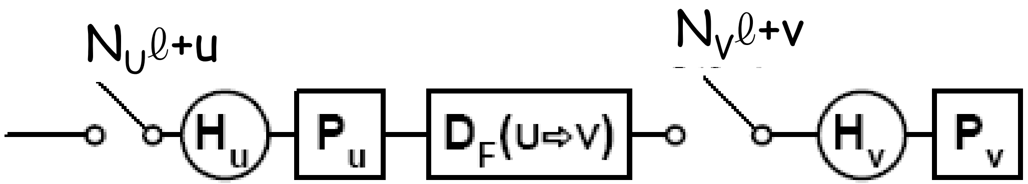
Folded Multi-rate Systems



$s_1(\ell)=s_1(M\ell)=x(M\ell-w_1)$

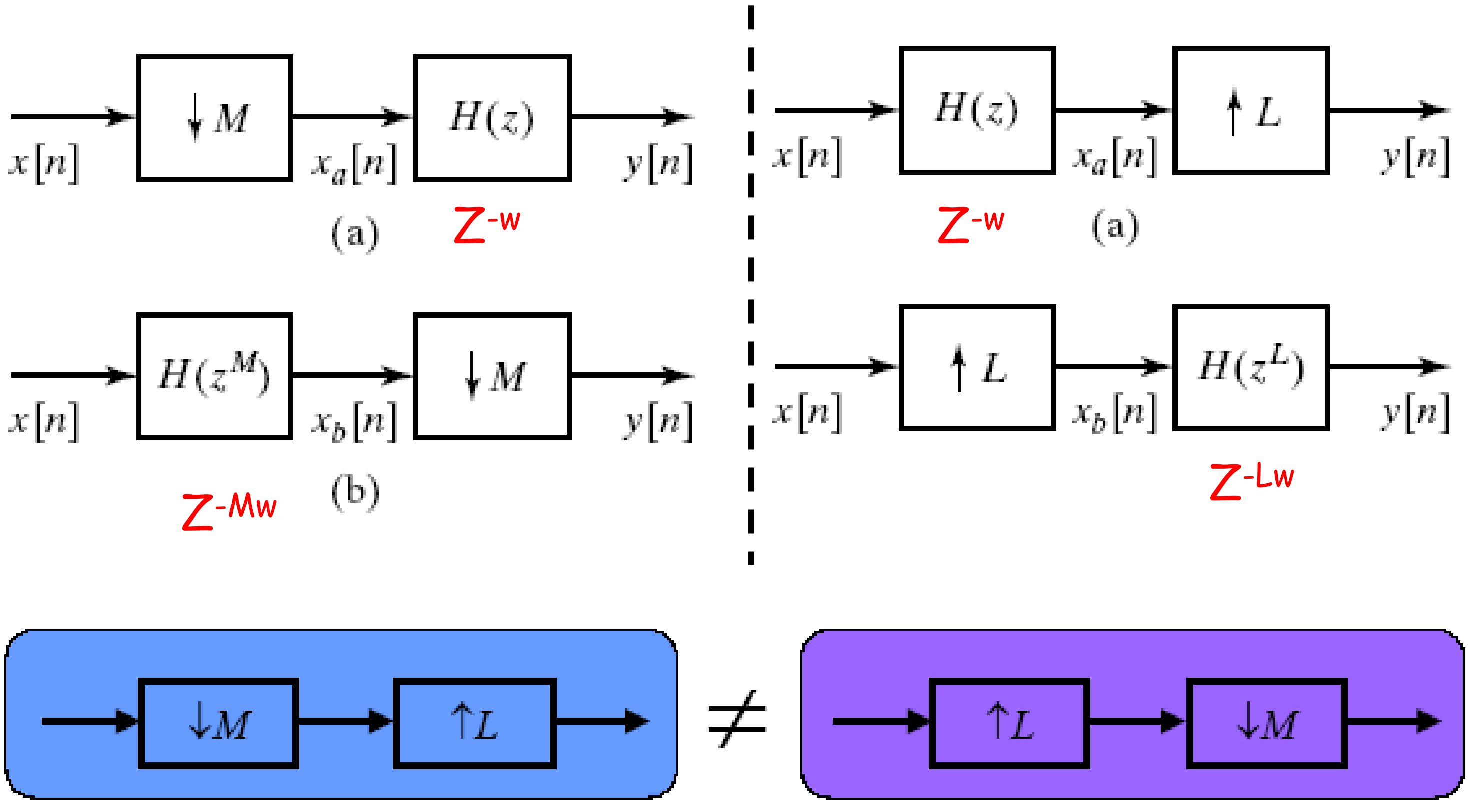
$s_1(\ell)=x(\ell-w_1)$

$y(\ell)=s_2(\ell-w_2)=x(M(\ell-w_{2)}-w_1)$

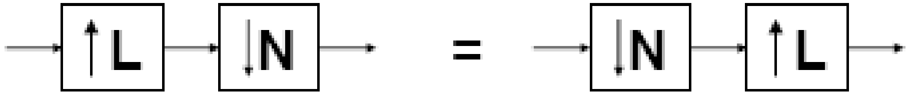


$$\begin{aligned} D_F(U\to V) &= [N_V l+v]-[N_U(M(l-w_2)-w_1)+u+P_u] \\ &= (N_V-MN_U)l+N_U(Mw_2+w_1)-P_U+v-u \\ &= N_U(Mw_2+w_1)-P_U+v-u\geq 0 \end{aligned}$$

Nobel Identities for Multirate Systems



Identities

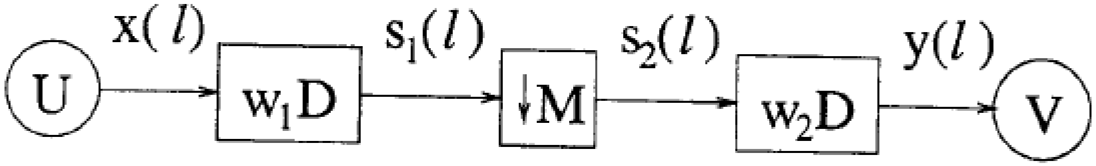


if and only if L and N are ***coprime*** !!!!!

Example 1: u[k]=1,2,3,4,5,6,7,8,9,… (L=2,N=3)
a) 2-fold up: 1,0,2,0,3,0,4,0,… | a) 3-fold down: 1,4,7,...
b) 3-fold down:1,0,4,0,7,0,… | b) 2-fold up: 1,0,4,0,7,...

Example 2: u[k]=1,2,3,4,5,6,7,8,9,… (L=2,N=4)
a) 2-fold up: 1,0,2,0,3,0,4,0,… | a) 4-fold down: 1,5,9,...
b) 4-fold down:1,3,5,7,9,… | b) 2-fold up: 1,0,5,0,9,...

Retiming MR-DFG



$r(D_{uv})$: # of delays, moved from each of its output arcs to each of its input arcs

$w_1' = w_1 + M r(D_{uv}) - r(U)$ $\qquad$ $w_2' = w_2 + r(V) - r(D_{UV})$

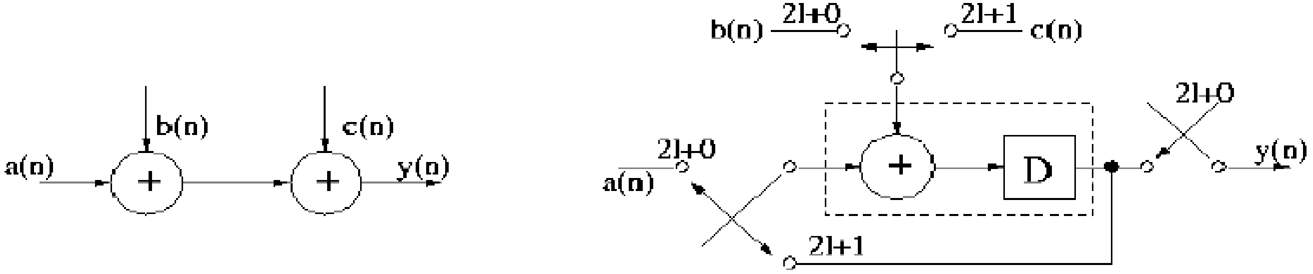
$$D_F'(U \to V) = N_U(Mw_2' + w_1') - P_U + v - u$$
$$= D_F(U \to V) + N_U(Mr(V) - r(U)) \geq 0$$

$$r(U) - Mr(V) \leq \left\lfloor \frac{D_F(U \to V)}{N_U} \right\rfloor$$

Folding Transformation

- Folding transformation time-multiplexes several algorithmic operations (e.g. multiply & add) into reduced functional units to save silicon area

Example



- Procedures
  1. Operation scheduling & binding
  2. Delay calculation
  3. Causalization
  4. Data generation with SIU

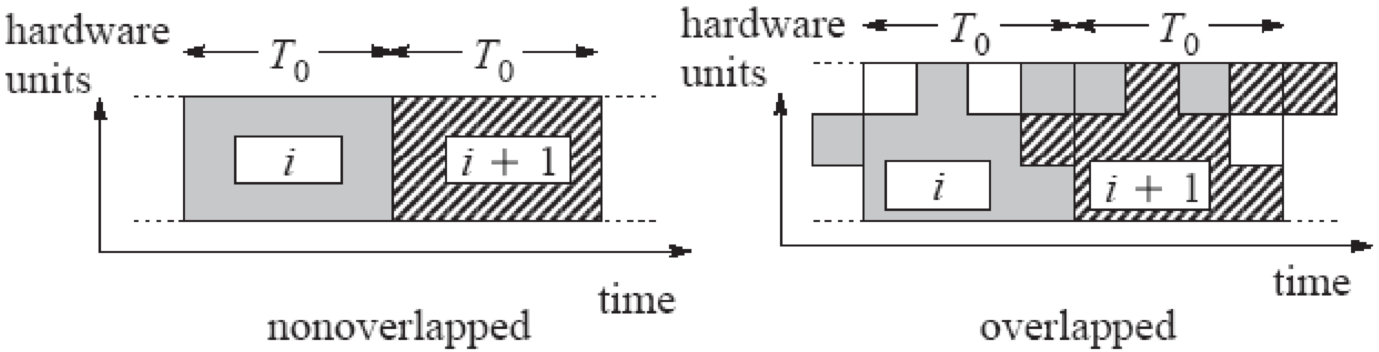
Terminology

- Scheduling: Determine for each operation the time at which it should be performed such that no precedence constraint is violated.
- Allocation: Specify the hardware resources that will be necessary
- Assignment: Provide a mapping from each operation to a specific functional unit and from each variable to a register

- Scheduling (except for a few versions) is NP-complete ➔ heuristics have to be used

Static Scheduling

- Static scheduling means mapping to time and processor (functional unit, register, etc.) is identical in all iterations
- A static schedule is either overlapped (expositing inter-iteration parallelism) or no-overlapped
- An overlapped schedule is also called loop folding, software pipelining

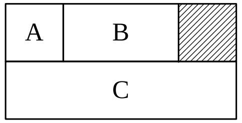


Scheduling

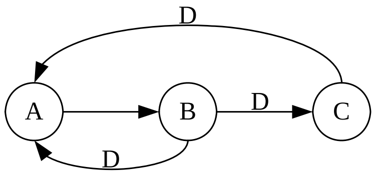
- Acyclic Precedence Graph

  a graph by removing all edges with delay elements from an SDFG
- Intra-iteration scheduling
  - obviously, the schedule can be improved by retiming

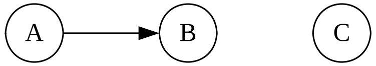
periodic nonoverlapping schedule



retimed DFG

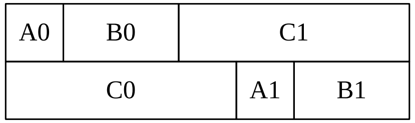


acyclic precedence graph

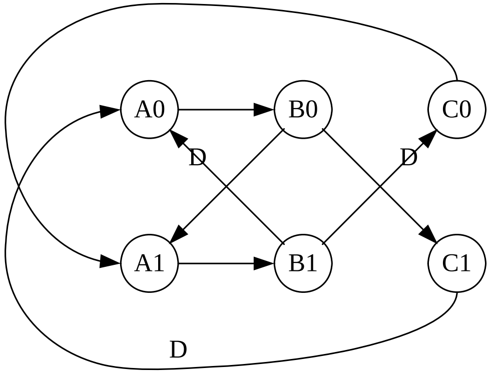


- Scheduling with multiple iterations (by unfolding)

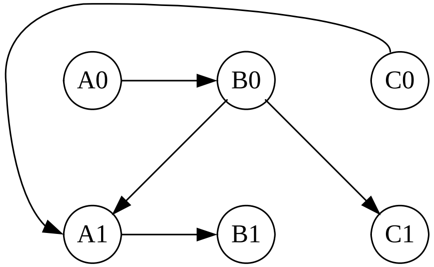
periodic nonoverlapping schedule



2-unfolded DFG



acyclic precedence graph



Basic Scheduling Algorithms

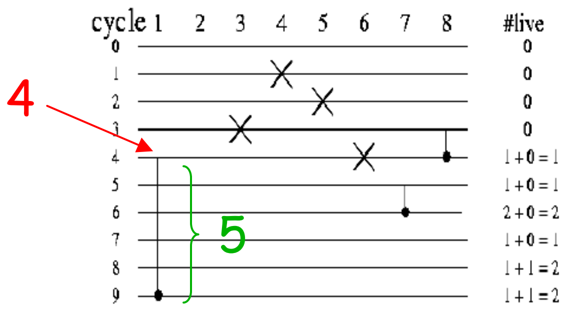
- As soon as possible (ASAP)
- As late as possible (ALAP)
- Force-directed scheduling
  - time-constrained scheduler
- List scheduling
  - resource constrained scheduler

SIU Optimization

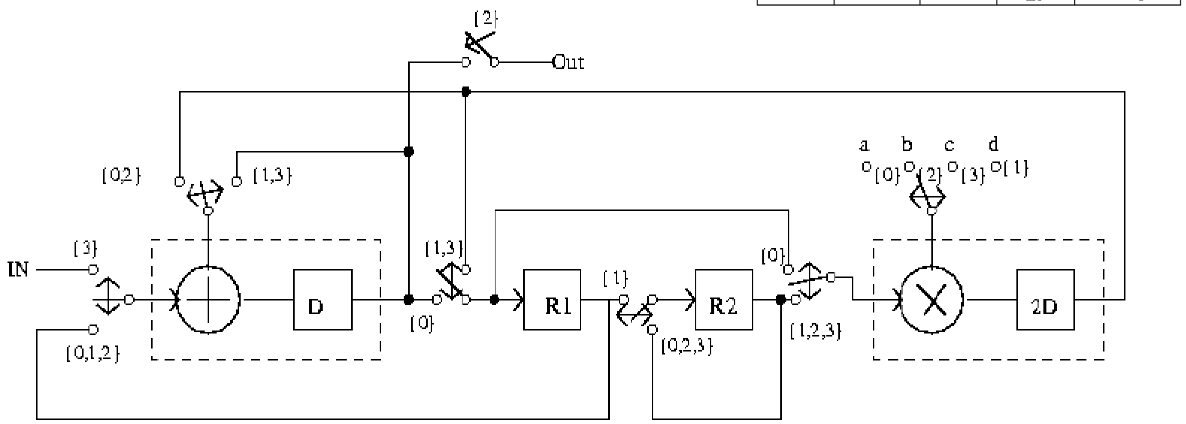
$D_F(1\rightarrow2) = 4(1) - 1 + 1 - 3 = 1$

$D_F(1\rightarrow6) = 4(1) - 1 + 2 - 3 = 2$

$D_F(1\rightarrow8) = 4(2) - 1 + 1 - 3 = 5$



cycle	input	R1	R2	output
0				
1				
2				
3	n_8			
4	n_1	n_8		n_8
5	n_7	n_1		
6		n	n_1	n_7
7			n_1	
8			n_1	
9			n_1	n_1



Remark: single MUL/ADD can perform various linear DSP kernels with different SIU architectures.