# ELE617E
# Lectures

Prof. Dr. Müştak E. Yalçın

Istanbul Technical University

mustak.yalcin@itu.edu.tr

# Folding

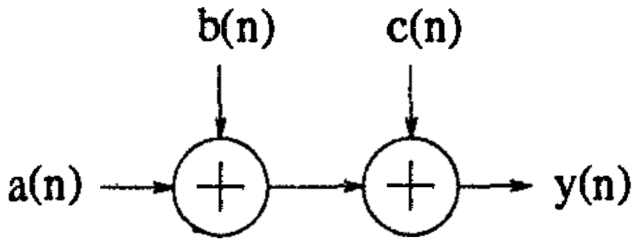
Folding is a technique to reduce the silicon area by time-multiplexing many algorithm operations into single functional units (such as adders and multipliers).

reduce hardware resources (# multipliers, # adders), at the expense of reduced throughput, by mapping multiple operations on the same HW resource.

- Folding is not: from MIMO back to SISO.
- Typically: given required throughput (latency) of a function (application), apply folding or unfolding (along with other transformations) to achieve the required throughput at minimal hardware costs.

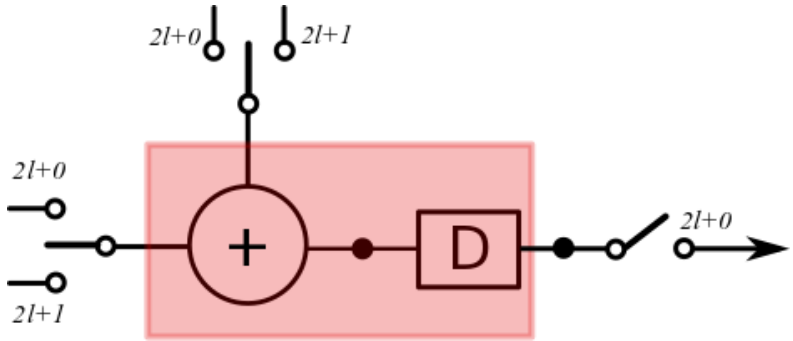
# Folding

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

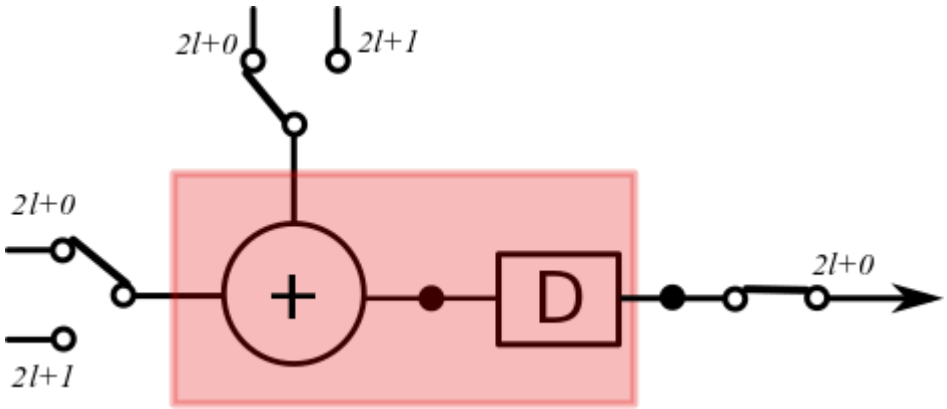



$$T_{\text{clk}} \geq 2T_{\text{add}}$$

Lets time-multiplexed on a single pipelined adder !

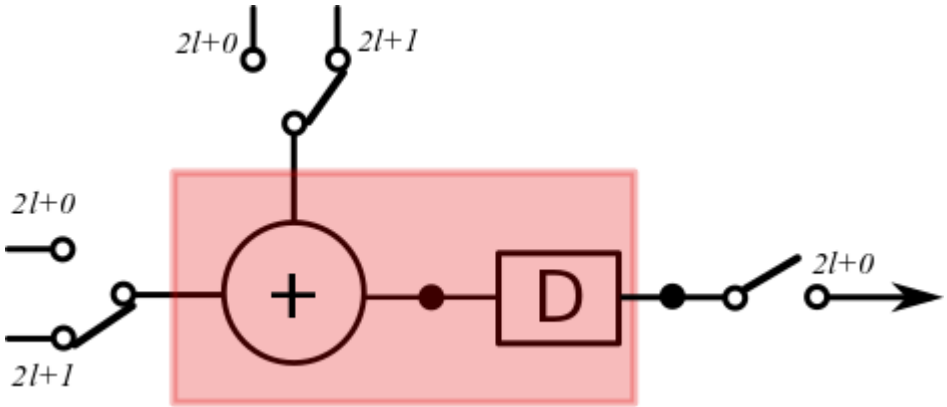


# Folding




| 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 | $a(2)+b(2)$ | $c(2)$ | – |

# Folding




| 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 | $a(2)+b(2)$ | $c(2)$ | – |

One output sample is produced every 2 clock cycles !

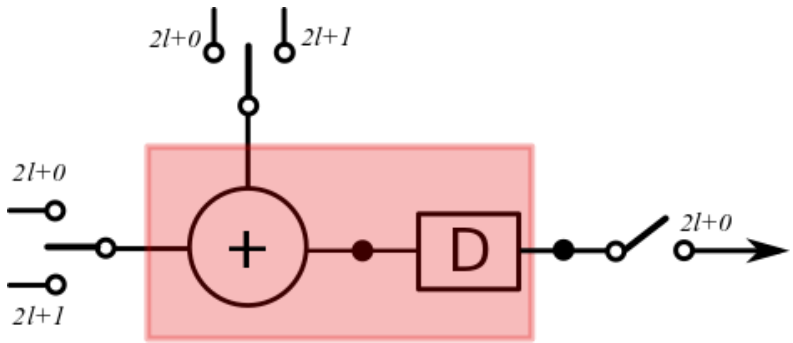
# Folding Transformation

A systematic technique for designing control circuits for HW where alg. operations are time-multiplexed on a single functional unit.

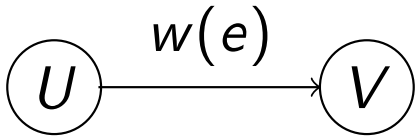
- $N$ *folding factor*: the number of operations folded to a single functional unit.
- $I$: iteration cycle
- $u, v$ *folding orders*: $0 \leq u, v \leq N-1$
- $H_u$: the functional unit that executes the node $U$.
- $P_U$: pipelining stages of the functional unit of $H_U$
- $w(e)$: delays of edge $U \xrightarrow{e} V$
- $NI + v$: is the time units at which $I - th$ iteration of the $v$ is scheduled.
- *Folding Set*: An ordered set of $N$ operations executed by the same functional unit. $S_1 = \{A_1, 0, A_2\}$ is for folding order $N = 3$. $A_1$ has a folding order of 0 and $A_2$ of 2 and are respectively denoted by $(S_1|0)$ and $(S_1|2)$.

# Folding Transformation

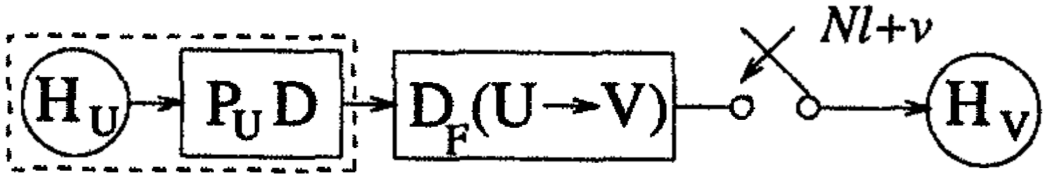
- $N$ *folding factor*: the number of op. folded to a single func. unit.
- $I$: iteration cycle
- $u, v$ *folding orders*: $0 \leq u, v \leq N-1$
- $H_u$: the functional unit that executes the node $U$.
- $P_U$: pipelining stages of the functional unit of $H_U$
- $w(e)$: delays of edge $U \xrightarrow{e} U$
- $NI + v$: is the time units at which $I$th iteration of the $V$ is scheduled.
- *Folding Set*: An ordered set of $N$ operations executed by the same functional unit.



# Folding Transformation




the $I$-th iteration of $U$ is used by $(I + w(e))$-th iteration of node $V$



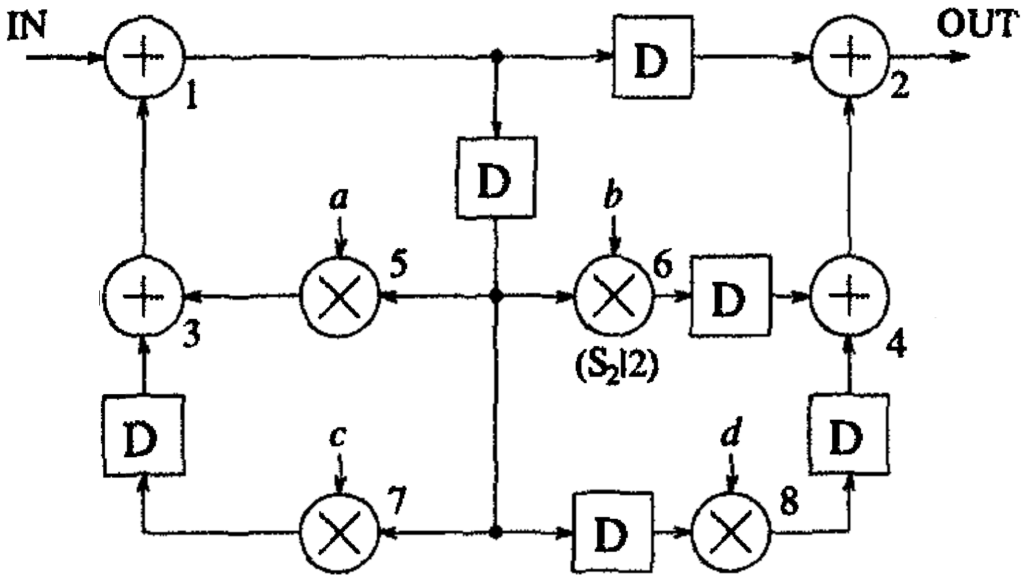

$H_U$'s output is available at $NI + u + P_U$ and The node $V$ is executed at $N(I + w(e)) + v$.
So, the result should be stored for

$$\begin{aligned} D_F(U \longrightarrow V) &= [N(I + w(e)) + v] - [NI + P_u + u] \\ &= Nw(e) - P_u + v - u \end{aligned}$$

independent of $I$.

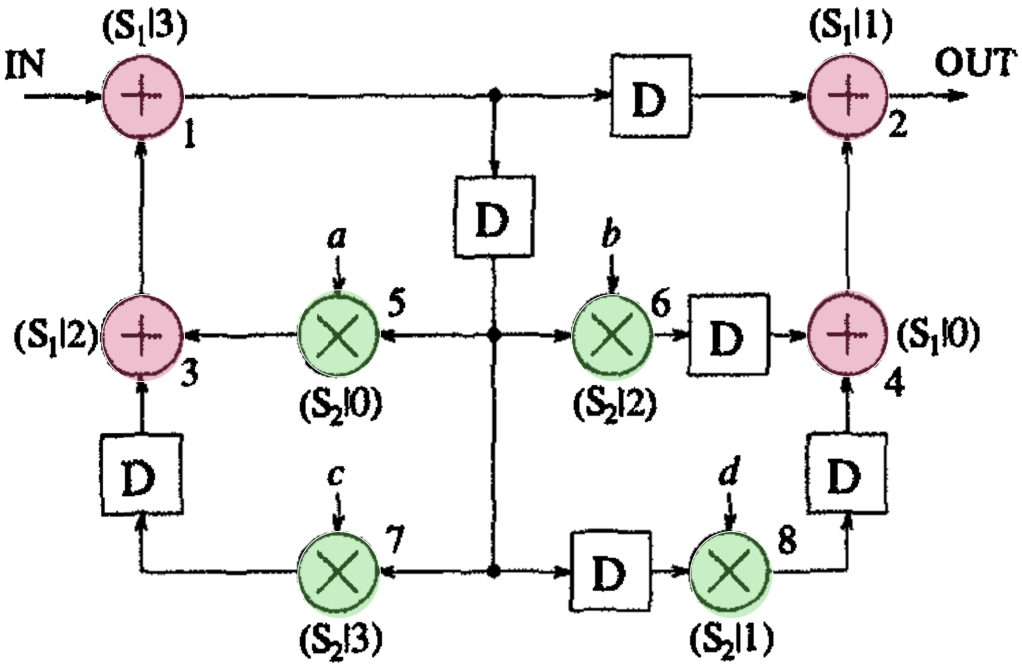
# Folding Transformation

Folding a retimed biquad filter by N = 4.

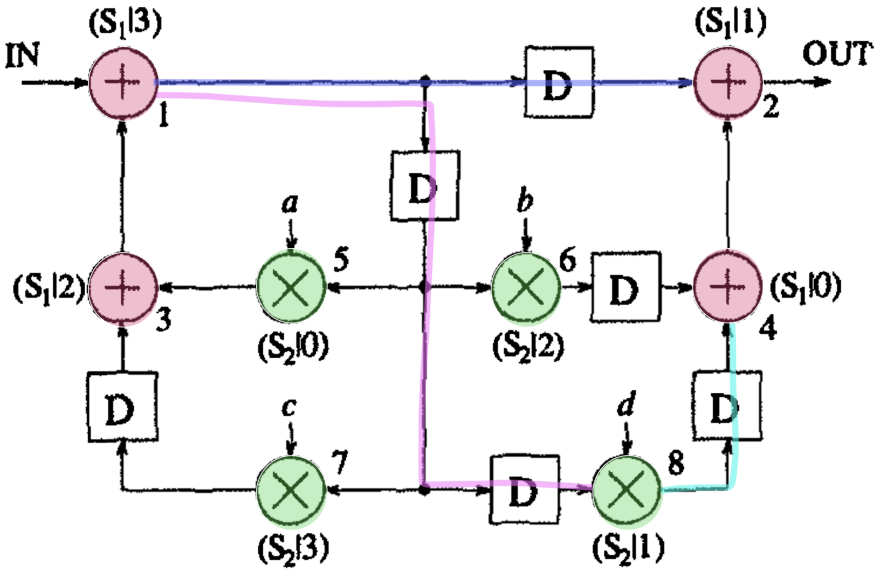



Folding Sets : $S_1 = \{4, 2, 3, 1\}$ and $S_2 = \{5, 8, 6, 7\}$, Addition time = 1u.t., Multiplication time = 2 u.t., 1 stage pipelined adder and 2 stage pipelined multiplier(i.e., $P_A = 1$ and $P_M = 2$)

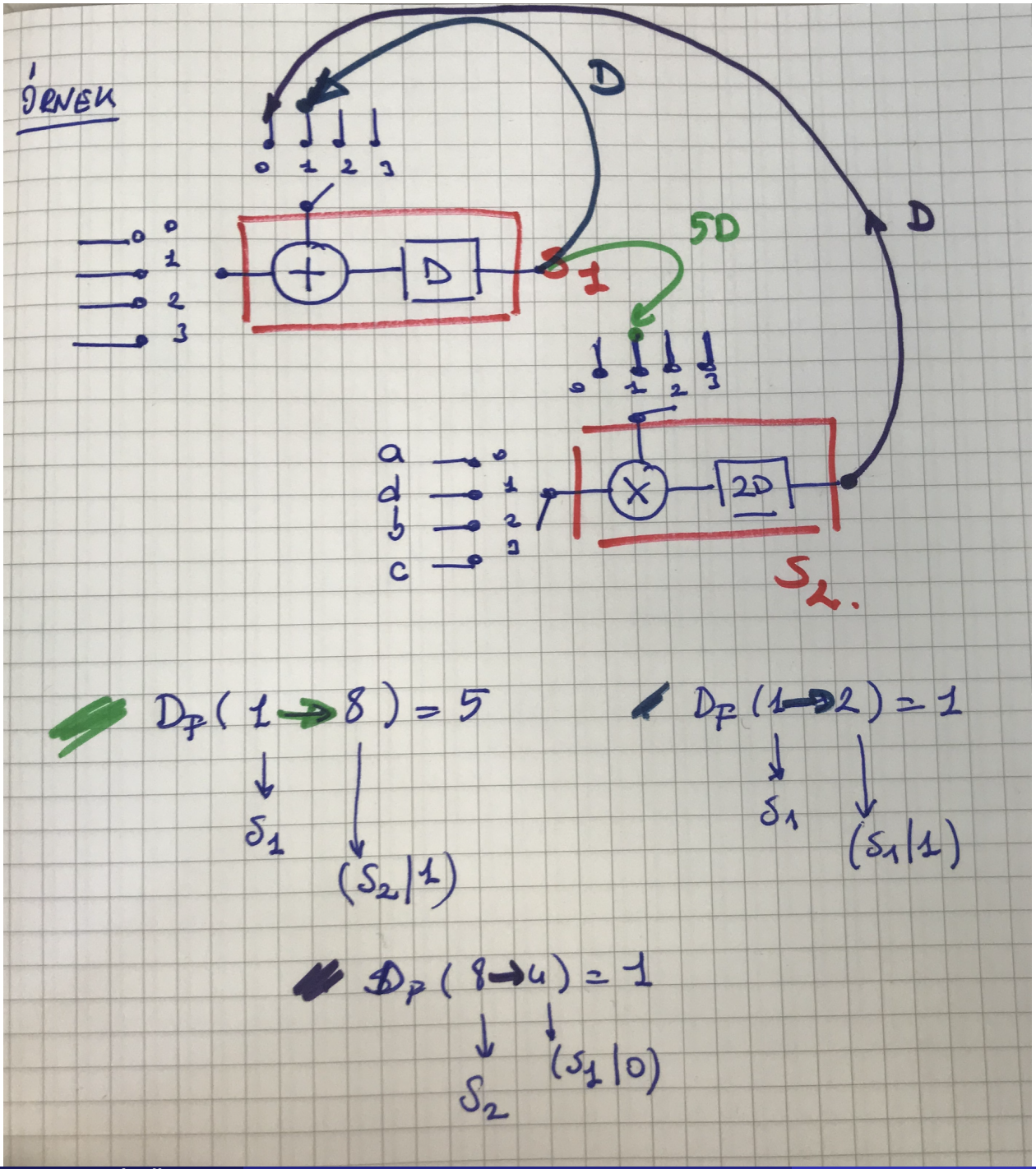
# Folding Transformation



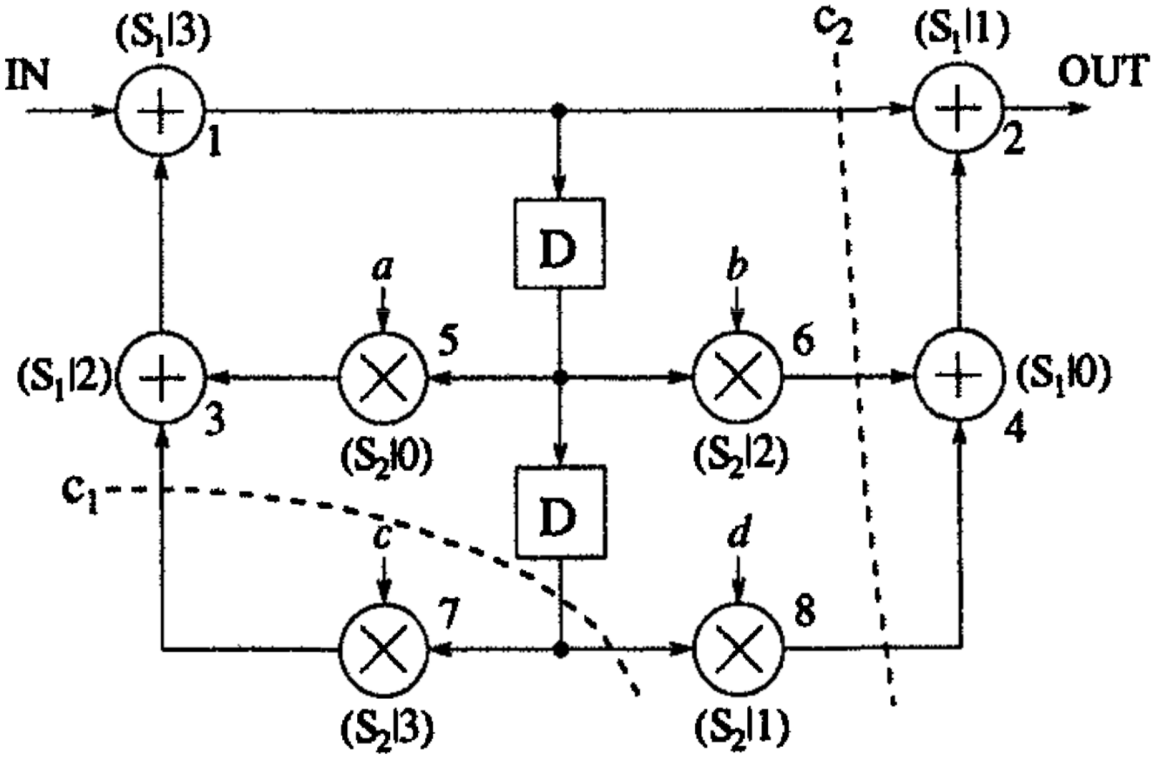
# Folding Transformation




$$D_F(1 \rightarrow 2) = Nw(e) - P_u + v{-}u$$
$$= Nw(e) - P_u + v{-}u$$
$$= 4(1) - 1 + 1{-}3 = 1$$
$$D_F(1 \rightarrow 8) = 4(2) - 1 + 1 - 3 = 5$$
$$D_F(8 \rightarrow 4) = 4(1) - 2 + 0 - 1 = 1$$


İRNEK
D
5D
D
0 1 2 3
+
D
$S_1$
a
d
b
c
X
2D
$S_2$.
$D_p(1 \to 8) = 5$
$S_1$
$(S_2|1)$
$D_p(1 \to 2) = 1$
$S_1$
$(S_1|1)$
$D_p(8 \to 4) = 1$
$S_2$
$(S_1|0)$

Obtain $D_F$s for $N = 4$




$$D_F(6 \to 4) = Nw(e) - P_u + v–u$$
$$= 4(0) - 2 + 0–2 = -4$$
$$D_F(7 \to 3) = 4(0) - 2 + 2 - 3 = -3$$

A folded architecture is realizable if and only if all delays $D_F(U \rightarrow V)$ are non-negative

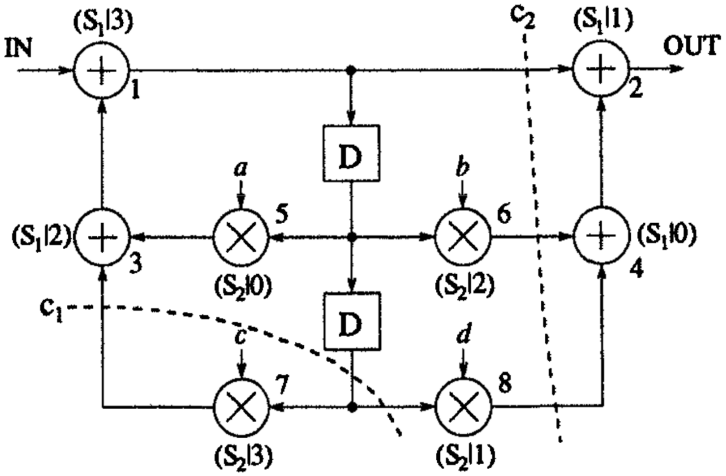
Retiming for folding

- All $D'_F(U \rightarrow V)$ of the retimed graph $G'$ are non-negative
- $Nw_r(e)–P_U + v–u \geq 0$ where $w_r(e) = w(e) + r(V) - r(U)$
  So

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

$$r(U)–r(V) \leq \left\lfloor \frac{D_F(U \rightarrow V)}{N} \right\rfloor$$

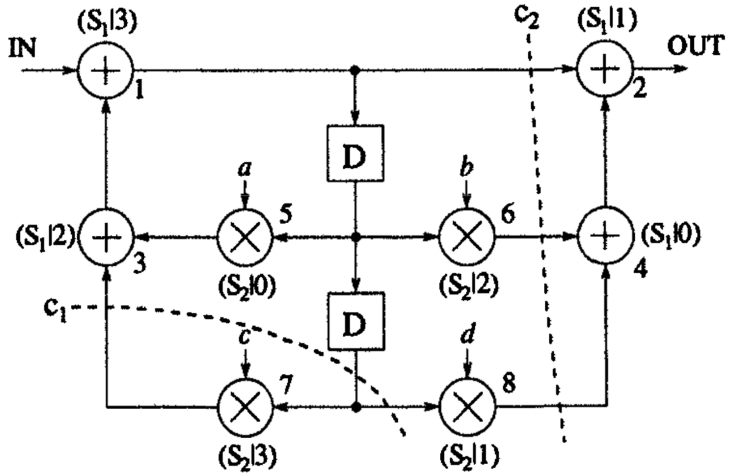
to determine if a solution exists and to a solution if one indeed exists (like before :( ).




| Edge | Folding Equation | Retiming for Folding Constraint |
|---|---|---|
| $1 \to 2$ | $D_F(1 \to 2) = -3$ | $r(1) - r(2) \leq -1$ |
| $1 \to 5$ | $D_F(1 \to 5) = 0$ | $r(1) - r(5) \leq 0$ |
| $1 \to 6$ | $D_F(1 \to 6) = 2$ | $r(1) - r(6) \leq 0$ |
| $1 \to 7$ | $D_F(1 \to 7) = 7$ | $r(1) - r(7) \leq 1$ |
| $1 \to 8$ | $D_F(1 \to 8) = 5$ | $r(1) - r(8) \leq 1$ |
| $3 \to 1$ | $D_F(3 \to 1) = 0$ | $r(3) - r(1) \leq 0$ |
| $4 \to 2$ | $D_F(4 \to 2) = 0$ | $r(4) - r(2) \leq 0$ |
| $5 \to 3$ | $D_F(5 \to 3) = 0$ | $r(5) - r(3) \leq 0$ |
| $6 \to 4$ | $D_F(6 \to 4) = -4$ | $r(6) - r(4) \leq -1$ |
| $7 \to 3$ | $D_F(7 \to 3) = -3$ | $r(7) - r(3) \leq -1$ |
| $8 \to 4$ | $D_F(8 \to 4) = -3$ | $r(8) - r(4) \leq -1$ |

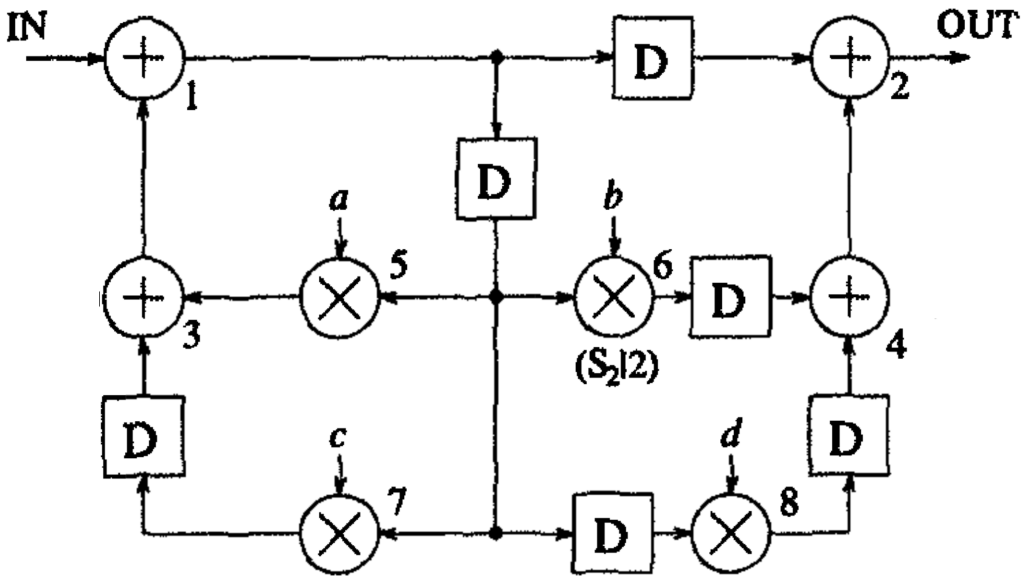
$D_F(7 \to 3) = -3$ then $r(7) - r(3) \leq \lfloor \frac{-3}{4} \rfloor = -1$ ▸ Folyd_Warshall.m

```
w =

   Inf   Inf     0   Inf   Inf   Inf   Inf   Inf   Inf
    -1   Inf   Inf     0   Inf   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf     0   Inf    -1   Inf   Inf
   Inf   Inf   Inf   Inf   Inf    -1   Inf    -1   Inf
     0   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
     0   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
     1   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
     1   Inf   Inf   Inf   Inf   Inf   Inf   Inf   Inf
     0     0     0     0     0     0     0     0     0

>> Folyd_Warshall(w)

ans =

     0   Inf     0   Inf     0   Inf    -1   Inf   Inf   Inf
    -1   Inf    -1     0    -1    -1    -2    -1   Inf   Inf
     0   Inf     0   Inf     0   Inf    -1   Inf   Inf   Inf
    -1   Inf    -1   Inf    -1    -1    -2    -1   Inf   Inf
     0   Inf     0   Inf     0   Inf    -1   Inf   Inf   Inf
     0   Inf     0   Inf     0   Inf    -1   Inf   Inf   Inf
     1   Inf     1   Inf     1   Inf     0   Inf   Inf   Inf
     1   Inf     1   Inf     1   Inf     0   Inf   Inf   Inf
    -1     0    -1     0    -1    -1    -2    -1     0   Inf
    -1     0    -1     0    -1    -1    -2    -1     0     0
```




| Edge | Folding Equation | Retiming for Folding Constraint |
|---|---|---|
| $1 \to 2$ | $D_F(1 \to 2) = -3$ | $r(1) - r(2) \leq -1$ |
| $1 \to 5$ | $D_F(1 \to 5) = 0$ | $r(1) - r(5) \leq 0$ |
| $1 \to 6$ | $D_F(1 \to 6) = 2$ | $r(1) - r(6) \leq 0$ |
| $1 \to 7$ | $D_F(1 \to 7) = 7$ | $r(1) - r(7) \leq 1$ |
| $1 \to 8$ | $D_F(1 \to 8) = 5$ | $r(1) - r(8) \leq 1$ |
| $3 \to 1$ | $D_F(3 \to 1) = 0$ | $r(3) - r(1) \leq 0$ |
| $4 \to 2$ | $D_F(4 \to 2) = 0$ | $r(4) - r(2) \leq 0$ |
| $5 \to 3$ | $D_F(5 \to 3) = 0$ | $r(5) - r(3) \leq 0$ |
| $6 \to 4$ | $D_F(6 \to 4) = -4$ | $r(6) - r(4) \leq -1$ |
| $7 \to 3$ | $D_F(7 \to 3) = -3$ | $r(7) - r(3) \leq -1$ |
| $8 \to 4$ | $D_F(8 \to 4) = -3$ | $r(8) - r(4) \leq -1$ |

$$w_r(7 \to 3) = 0 + r(3) - r(7) = 0 - 1 - (-2) = 1$$



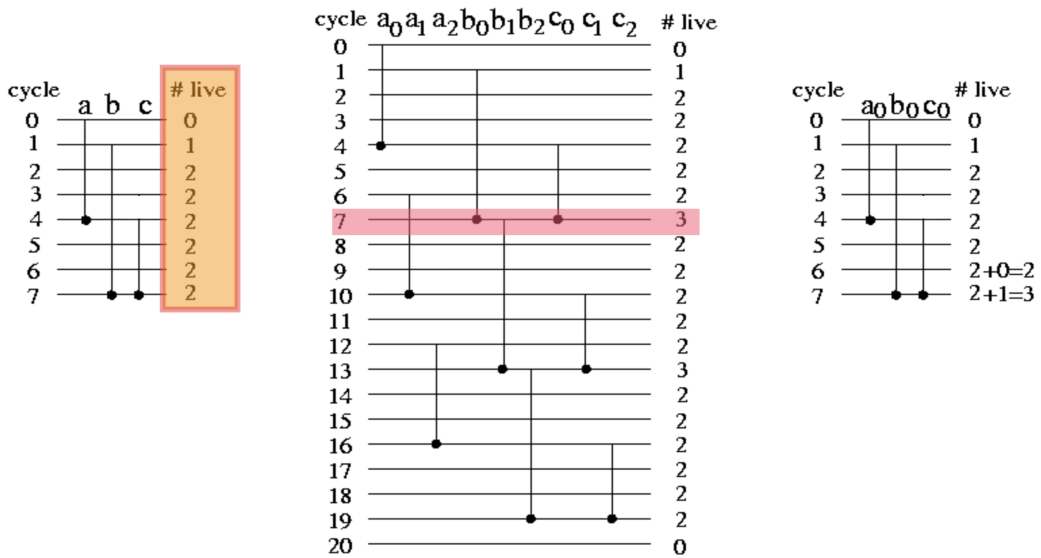
# Register Minimization

- Folding may insert registers.
- Life time 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. During that time,the variable is stored in a register.
- Linear life time 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

# Life Time Analysis

a procedure used to compute the minimum number of registers required to implement a DSP algorithm in hardware!
A variable is live from the time it is produced through the time it is consumed (dead).
A variable occupies one register while it is live. In lifetime analysis, the number of live variables at each time unit is computed, and the maximum number of live variables at any time unit is determined. Life time chart:

PERIODIC NATURE OF THE DSP ALG. MUST ALWAYS BE TAKEN INTO ACCOUNT!



# Life Time Analysis

Life Time Analysis begins with lifetime table:
Ex: The transpose operation:

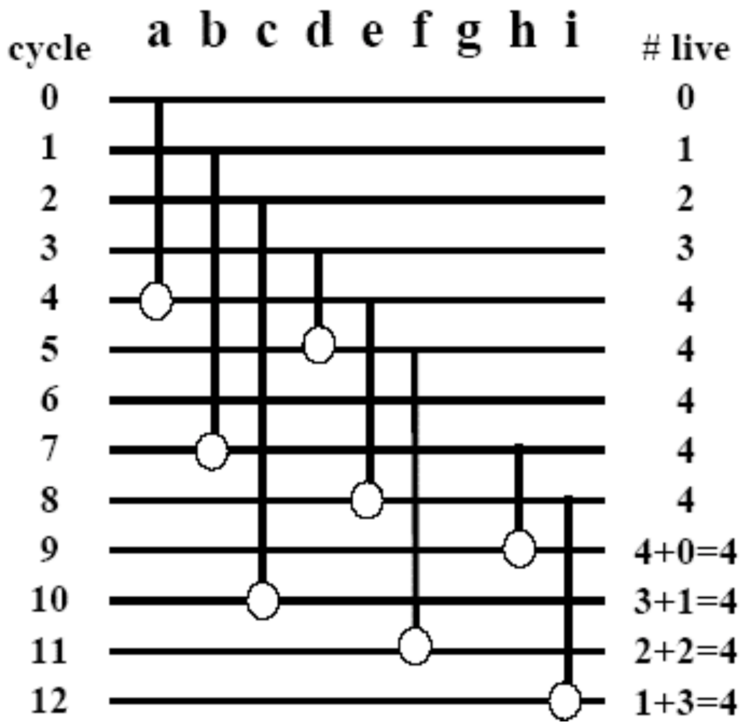
$$ABCDEFGHI \rightarrow ADGBEHCFI$$

| Sample | $T_{\mathbf{input}}$ | $T_{\mathbf{zlout}}$ | $T_{\mathbf{diff}}$ | $T_{\mathbf{output}}$ | **Life Period** |
|---|---|---|---|---|---|
| a | 0 | 0 | 0 | 4 | $0 \rightarrow 4$ |
| b | 1 | 3 | 2 | 7 | $1 \rightarrow 7$ |
| c | 2 | 6 | 4 | 10 | $2 \rightarrow 10$ |
| d | 3 | 1 | -2 | 5 | $3 \rightarrow 5$ |
| e | 4 | 4 | 0 | 8 | $4 \rightarrow 8$ |
| f | 5 | 7 | 2 | 11 | $5 \rightarrow 11$ |
| g | 6 | 2 | -4 | 6 | $6 \rightarrow 6$ |
| h | 7 | 5 | -2 | 9 | $7 \rightarrow 9$ |
| i | 8 | 8 | 0 | 12 | $8 \rightarrow 12$ |

$T_{\mathbf{latency}} = |\min\{T_{\mathbf{diff}}\}|$ $T_{\mathbf{output}} = T_{\mathbf{zlout}} + T_{\mathbf{latency}}$

# Life Time Analysis

| Sample | **Life Period** |
|---|---|
| a | $0 \rightarrow 4$ |
| b | $1 \rightarrow 7$ |
| c | $2 \rightarrow 10$ |
| d | $3 \rightarrow 5$ |
| e | $4 \rightarrow 8$ |
| f | $5 \rightarrow 11$ |
| g | $6 \rightarrow 6$ |
| h | $7 \rightarrow 9$ |
| i | $8 \rightarrow 12$ |

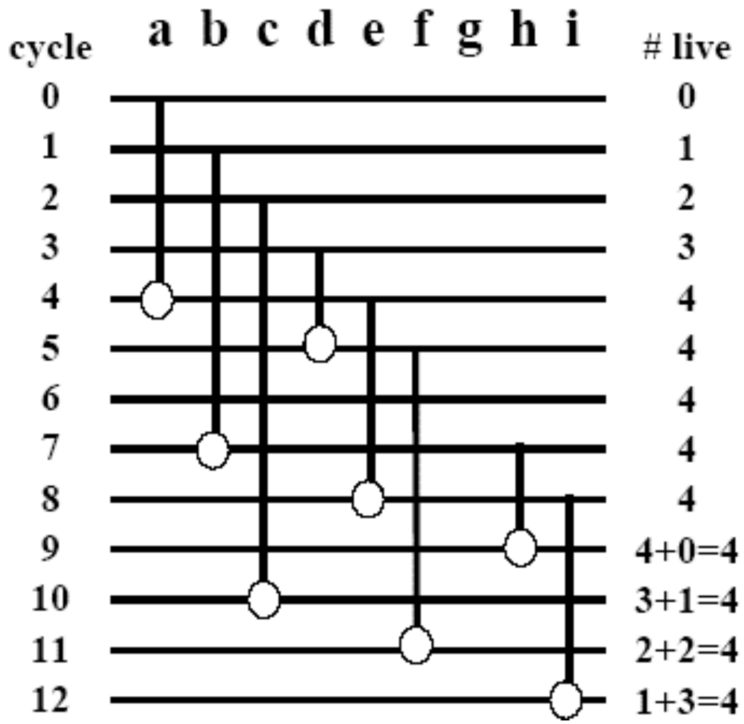


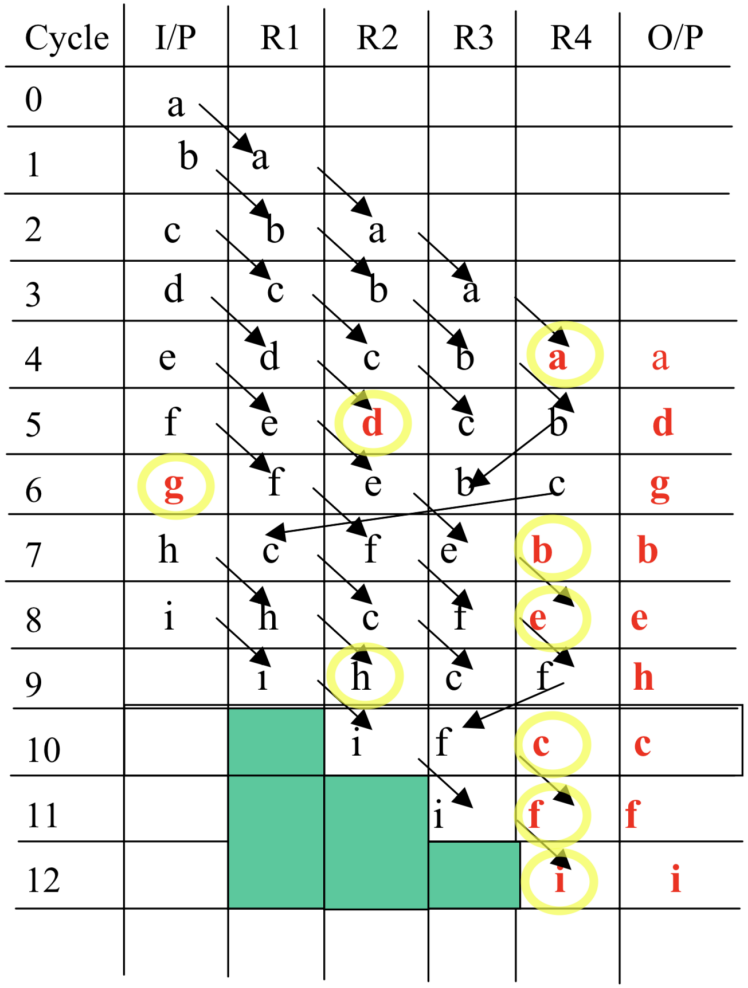
# Forward Backward Register Allocation

Determine the min. number of registers

- Input each variable at the time its life starts. If more than one use multiple registers such that the longest lifetime is allocated to the initial register.
- Each variable is allocated in a forward manner until it is dead or reaches the last register
- Allocation is periodic, all allocation to current iteration repeats itself after after N
- If reaches the last register and not dead allocate backward ((if more than one, choose one that has been allocated backward before), then forward again and so on.

# Forward Backward Register Allocation

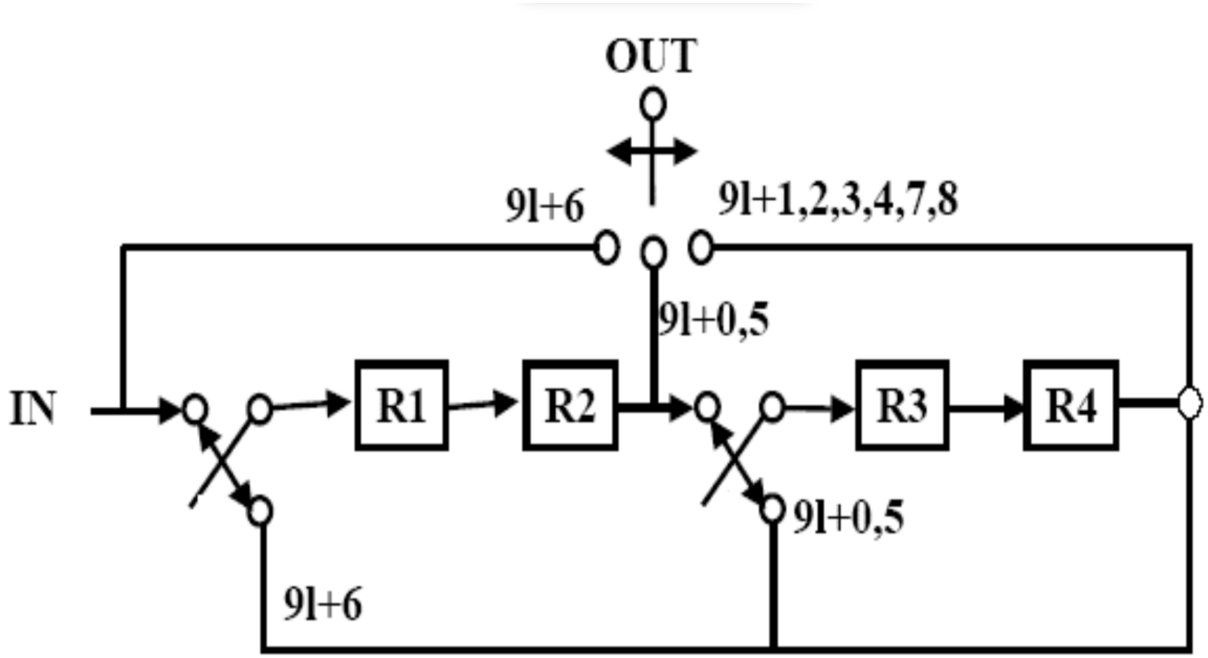



allocation table

# Synthesis

| Cycle | I/P | R1 | R2 | R3 | R4 | O/P |
|---|---|---|---|---|---|---|
| 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** |

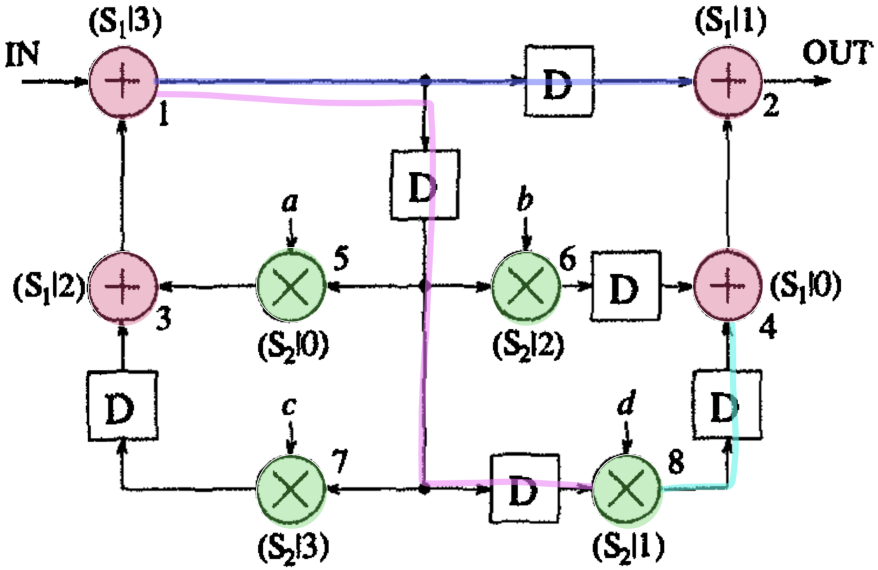


# Register Minimization for Folded Architecture

- Perform retiming for folding
- Write the folding equations
- Use the folding equations to construct a lifetime table
- Draw the lifetime chart and determine the minimum number of registers
- Perform forward-backward register allocation
- Draw the folded architecture

# Register Minimization for Folded Architecture

- Perform retiming for folding
- Write the folding equations




$$D_F(1 \rightarrow 2) = Nw(e) - P_u + v - u$$
$$= 4(1) - 1 + 1 - 3 = 1$$
$$D_F(1 \rightarrow 8) = 4(2) - 1 + 1 - 3 = 5$$
$$D_F(8 \rightarrow 4) = 4(1) - 2 + 0 - 1 = 1$$

# Register Minimization for Folded Architecture

- Use the folding equations to construct a lifetime table (Node $u$ is created at time $u + P_u$ and Node $u$ is consumed at time $u + P_u + \max_v\{(D_F(U \rightarrow V)\}$ )

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

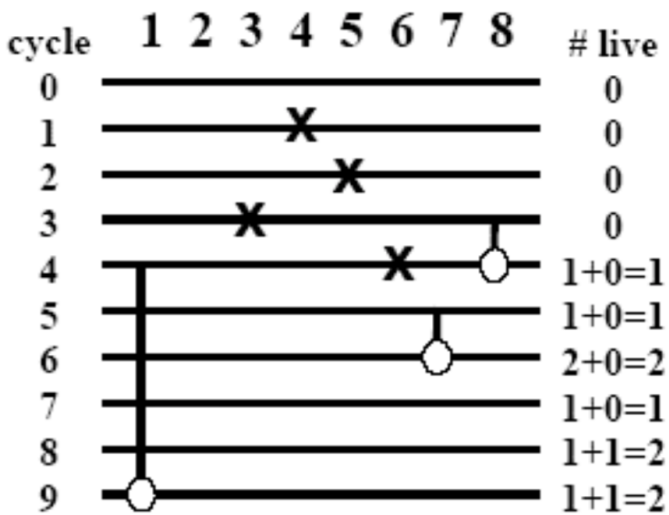
$$
\begin{aligned}
D_F(1 \rightarrow 2) &= 4(1) - 1 + 1 - 3 = 1 \\
D_F(1 \rightarrow 5) &= 4(1) - 1 + 0 - 3 = 0 \\
D_F(1 \rightarrow 6) &= 4(1) - 1 + 2 - 3 = 2 \\
D_F(1 \rightarrow 7) &= 4(1) - 1 + 3 - 3 = 3 \\
D_F(1 \rightarrow 8) &= 4(2) - 1 + 1 - 3 = 5 \\
D_F(3 \rightarrow 1) &= 4(0) - 1 + 3 - 2 = 0 \\
D_F(4 \rightarrow 2) &= 4(0) - 1 + 1 - 0 = 0 \\
D_F(5 \rightarrow 3) &= 4(0) - 2 + 2 - 0 = 0 \\
D_F(6 \rightarrow 4) &= 4(1) - 2 + 0 - 2 = 0 \\
D_F(7 \rightarrow 3) &= 4(1) - 2 + 2 - 3 = 1 \\
D_F(8 \rightarrow 4) &= 4(1) - 2 + 0 - 1 = 1.
\end{aligned}
$$

| Node | $T_{\text{input}} \rightarrow T_{\text{output}}$ |
|---|---|
| 1 | $3+1 \rightarrow 3+1+5$ |
| 2 | – |
| 3 | $3 \rightarrow 3$ |
| 4 | $1 \rightarrow 1$ |
| 5 | $2 \rightarrow 2$ |
| 6 | $4 \rightarrow 4$ |
| 7 | $5 \rightarrow 6$ |
| 8 | $3 \rightarrow 4$ |

# Register Minimization for Folded Architecture

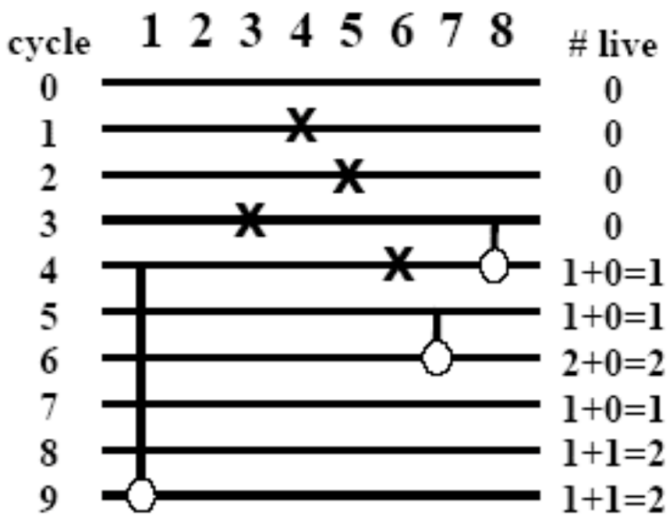
- Draw the lifetime chart and determine the minimum number of registers

| Node | $T_{input} \rightarrow T_{output}$ |
|---|---|
| 1 | $4 \rightarrow 9$ |
| 2 | – |
| 3 | $3 \rightarrow 3$ |
| 4 | $1 \rightarrow 1$ |
| 5 | $2 \rightarrow 2$ |
| 6 | $4 \rightarrow 4$ |
| 7 | $5 \rightarrow 6$ |
| 8 | $3 \rightarrow 4$ |



# Register Minimization for Folded Architecture

- Perform forward-backward register allocation




| cycle | i/p | R1 | R2 | o/p |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | n8 | | | |
| 4 | n1 | (n8) | | n8 |
| 5 | n7 | n1 | | |
| 6 | | (n7) | n1 | n7 |
| 7 | | | n1 | |
| 8 | | | n1 | |
| 9 | | | (n1) | n1 |

# Register Minimization for Folded Architecture

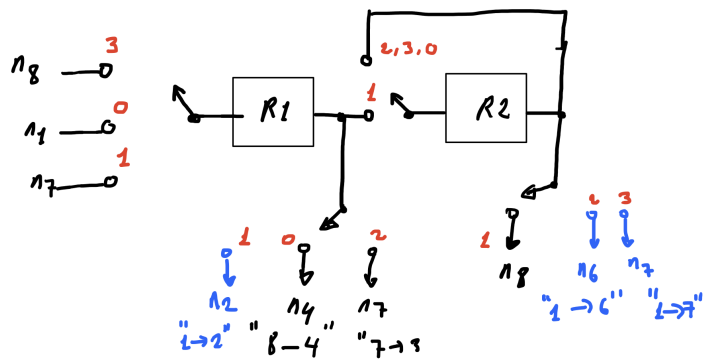
- Draw the folded architecture

| | cycle | i/p | R1 | R2 | o/p |
|---|---|---|---|---|---|
| 0 | 0 | | | | |
| 1 | 1 | | | | |
| 2 | 2 | | | | |
| 3 | 3 | n8 | | | |
| 0 | 4 | n1 | (n8) | | n8 |
| 1 | 5 | n7 | n1 | | |
| 2 | 6 | | (n7) | n1 | n7 |
| 3 | 7 | | | n1 | |
| 0 | 8 | | | n1 | |
| 1 | 9 | | | (n1) | n1 |

8 → 4

7 → 3

1 → 8




$$
\begin{aligned}
D_F(1 \to 2) &= 4(1) - 1 + 1 - 3 = 1 \\
D_F(1 \to 5) &= 4(1) - 1 + 0 - 3 = 0 \\
D_F(1 \to 6) &= 4(1) - 1 + 2 - 3 = 2 \\
D_F(1 \to 7) &= 4(1) - 1 + 3 - 3 = 3 \\
D_F(1 \to 8) &= 4(2) - 1 + 1 - 3 = 5
\end{aligned}
$$

$T_{input} \to T_{out}$

4 → 5

4 → 4

4 → 6

4 → 7

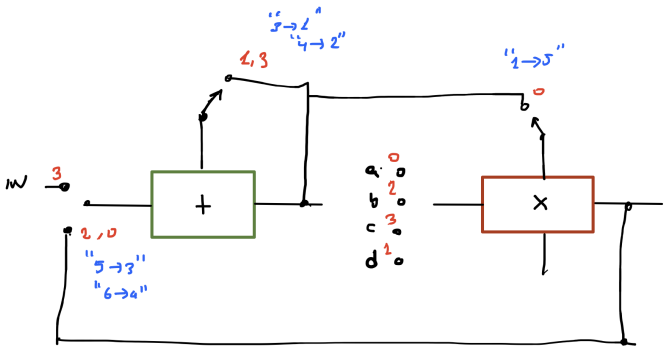
4 → 9

# Register Minimization for Folded Architecture

- Draw the folded architecture

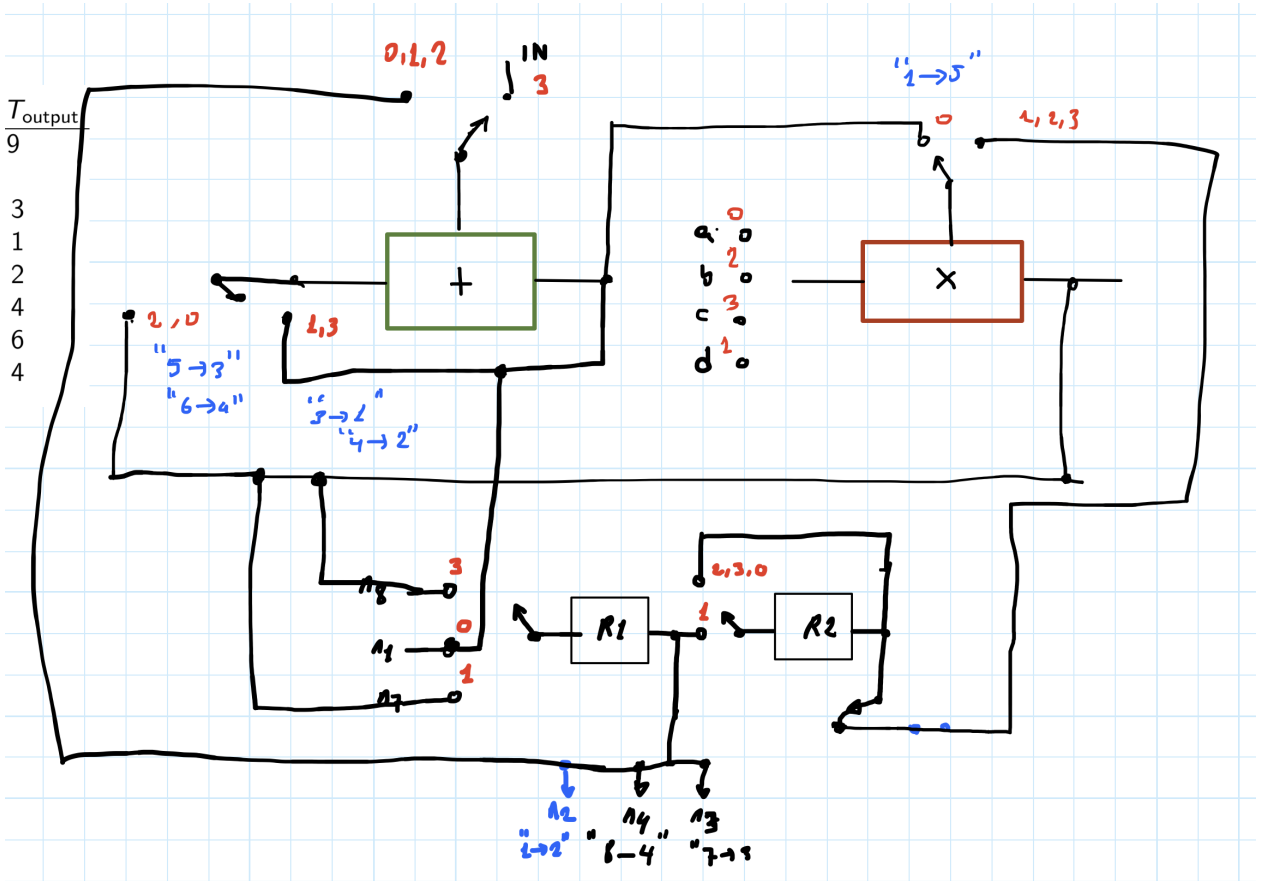
$$
\begin{aligned}
D_F(3 \to 1) &= 4(0) - 1 + 3 - 2 = 0 \\
D_F(4 \to 2) &= 4(0) - 1 + 1 - 0 = 0 \\
D_F(5 \to 3) &= 4(0) - 2 + 2 - 0 = 0 \\
D_F(6 \to 4) &= 4(1) - 2 + 0 - 2 = 0
\end{aligned}
$$

| Node | $T_{input} \to T_{output}$ |
|---|---|
| 1 | $4 \to 9$ |
| 2 | – |
| 3 | $3 \to 3$ |
| 4 | $1 \to 1$ |
| 5 | $2 \to 2$ |
| 6 | $4 \to 4$ |
| 7 | $5 \to 6$ |
| 8 | $3 \to 4$ |



# Register Minimization for Folded Architecture

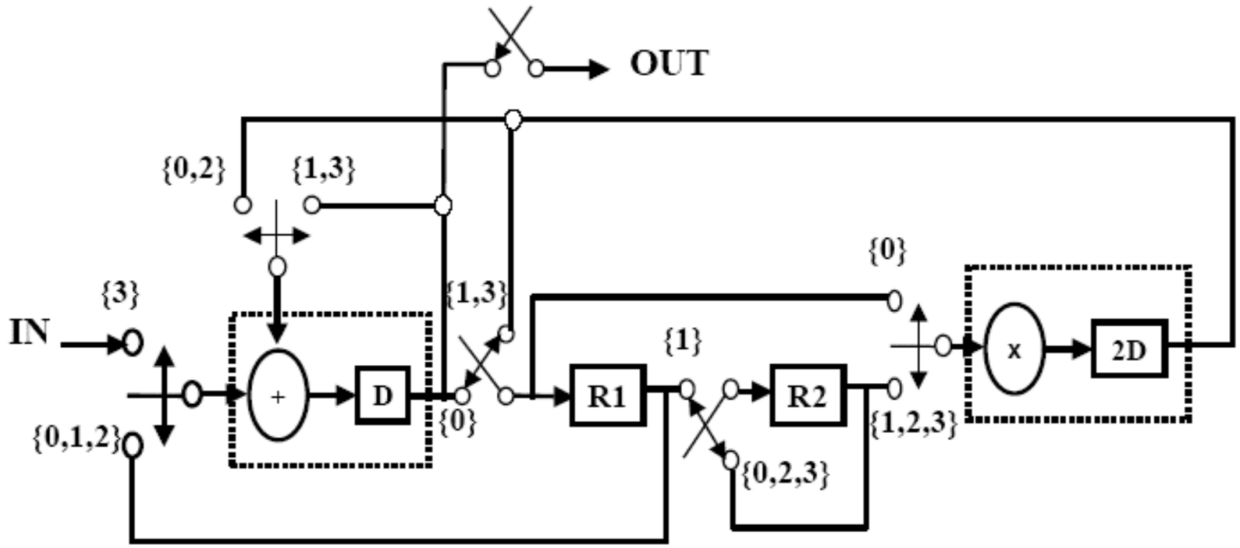
- Draw the folded architecture



# Register Minimization for Folded Architecture

- Draw the folded architecture

| cycle | i/p | R1 | R2 | o/p |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | n8 | | | |
| 4 | n1 | (n8) | | n8 |
| 5 | n7 | n1 | | |
| 6 | | (n7) | n1 | n7 |
| 7 | | | n1 | |
| 8 | | | n1 | |
| 9 | | | (n1) | n1 |




HW: Page 167, IIR Filter Example.