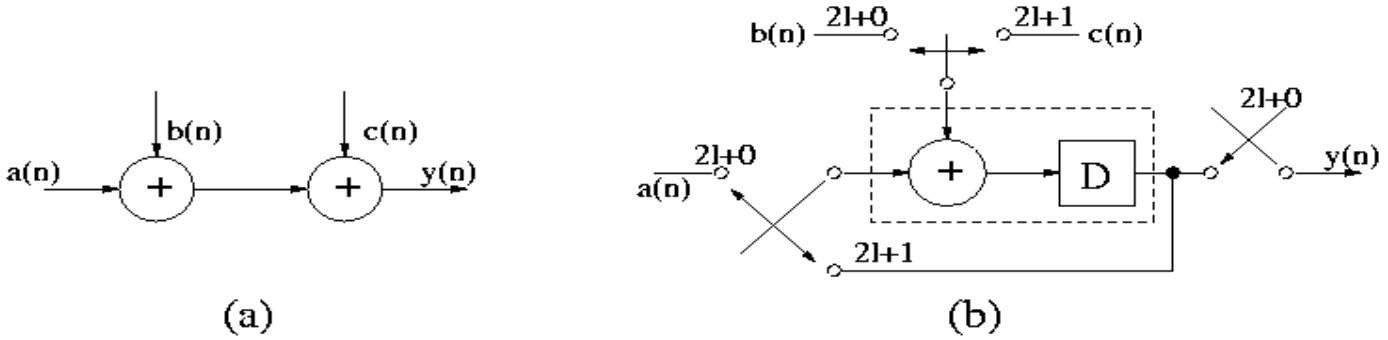
Chapter 6: Folding

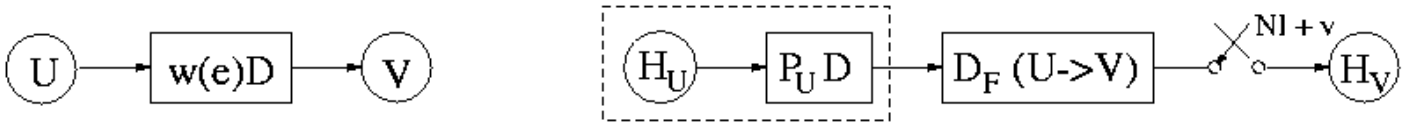
Keshab K. Parhi

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



- Fig(a) shows a DSP program : y(n) = a(n) + b(n) + c(n) .
- Fig(b) shows a folded architecture where 2 additions are folded or time-multiplexed to a single pipelined adder
  One output sample is produced every 2 clock cycles $\Rightarrow$ input should be valid for 2 clock cycles.
- In general, the data on the input of a folded realization is assumed to be valid for N cycles before changing, where N is the number of algorithm operations executed on a single functional unit in hardware.

Folding Transformation :



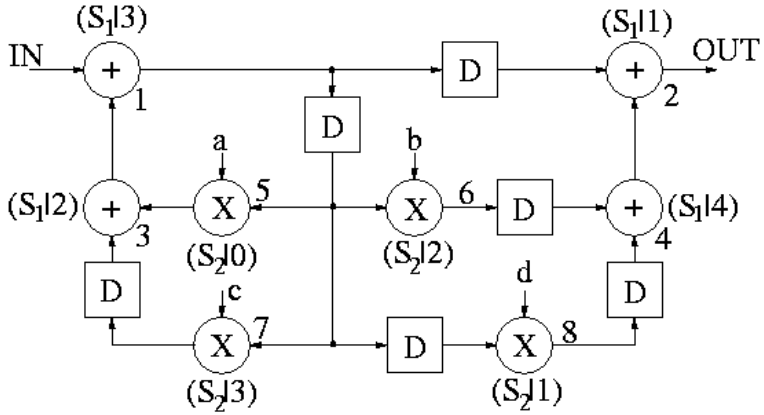
(a) (b)

- N*l* + u and N*l* + v are respectively the time units at which l-th iteration of the nodes U and V are scheduled.
- u and v are called folding orders (time partition at which the node is scheduled to be executed) and satisfy $0 \le u,v \le N-1$.
- N is the folding factor i.e., the number of operations folded to a single functional unit.
- H_u and H_v are functional units that execute u and v respectively.
- H_u is pipelined by P_u stages and its output is available at N*l* + u + P_u.
- Edge U$\rightarrow$V has w(e) delays $\Rightarrow$ the *l*-th iteration of U is used by (*l* + w(e)) th iteration of node V, which is executed at N(*l* + w(e)) + v. So, the result should be stored for :

$$D_F(U \rightarrow V) = [N(l + w(e)) + v] - [Nl + P_u + u]$$

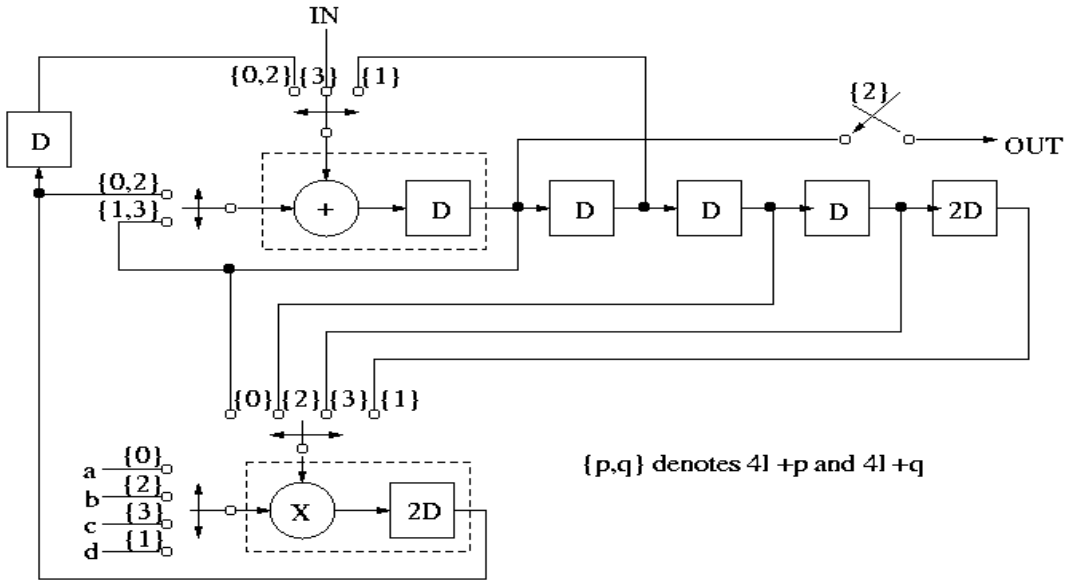
$$\Rightarrow D_F(U \rightarrow V) = Nw(e) - P_u + v - u \quad \textbf{(independent of } l \textbf{)}$$

- Folding Set : An ordered set of N operations executed by the same functional unit. The operations are ordered from 0 to N-1. Some of the operations may be null. For example, Folding set $S_1=\{A_1,0,A_2\}$ 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 $(S_2|2)$.
- Example: Folding a retimed biquad filter by N = 4.



Addition time = 1u.t., Multiplication time = 2u.t., 1 stage pipelined adder and 2 stage pipelined multiplier(i.e., $P_A=1$ and $P_M=2$)

The folding sets are $S_1 = \{4, 2, 3, 1\}$ and $S_2 = \{5, 8, 6, 7\}$



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

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

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

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

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

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

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

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

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

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

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

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

- Retiming for Folding :
  - For a folded system to be realizable $D_F(U \rightarrow V) \geq 0$ for all edges.
  - If $D'_F(U \rightarrow V)$ is the folded delays in the edge $U \rightarrow V$ for the retimed graph then $D'_F(U \rightarrow V) \geq 0$.

  So,

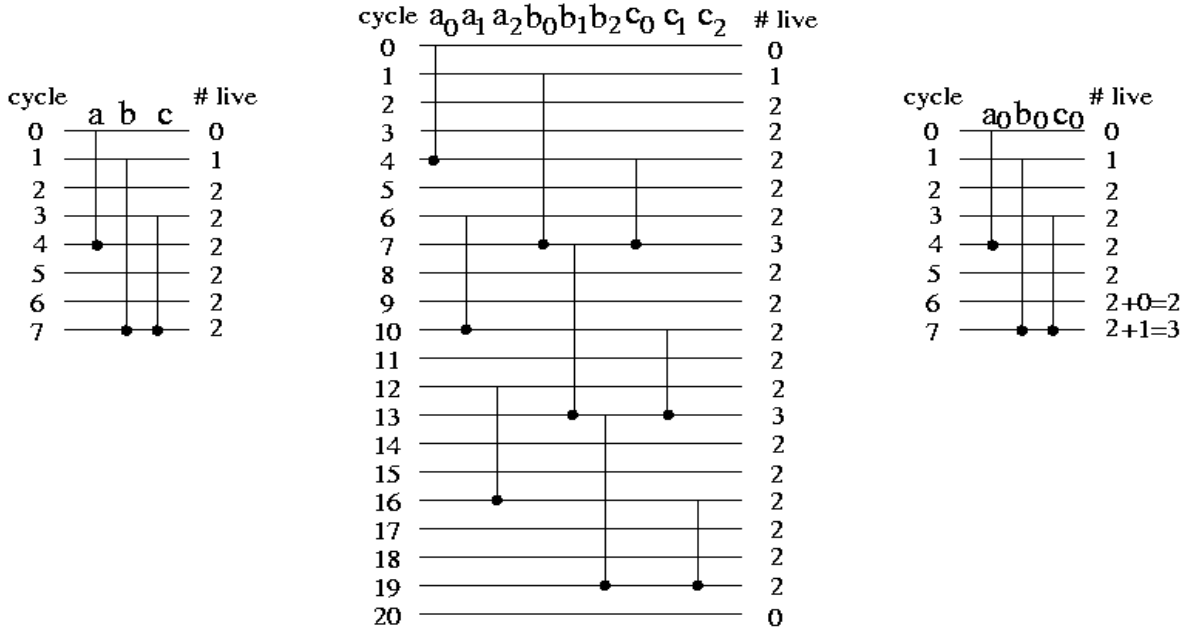
  $$Nw_r(e) - P_U + v - u \geq 0 \quad \text{... where } w_r(e) = w(e) + r(V) - r(U)$$

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

  $$\Rightarrow r(U) - r(V) \leq D_F(U \rightarrow V) / N$$

  $$\Rightarrow r(U) - r(V) \leq \lfloor D_F(U \rightarrow V) / N \rfloor$$ (since retiming values are integers)

- Register Minimization Technique : Lifetime analysis is used for register minimization techniques in a DSP hardware.
- A ‘data sample or variable’ is live from the time it is produced through the time it is consumed. After that it is dead.
- Linear lifetime chart : Represents the lifetime of the variables in a linear fashion.
- Example

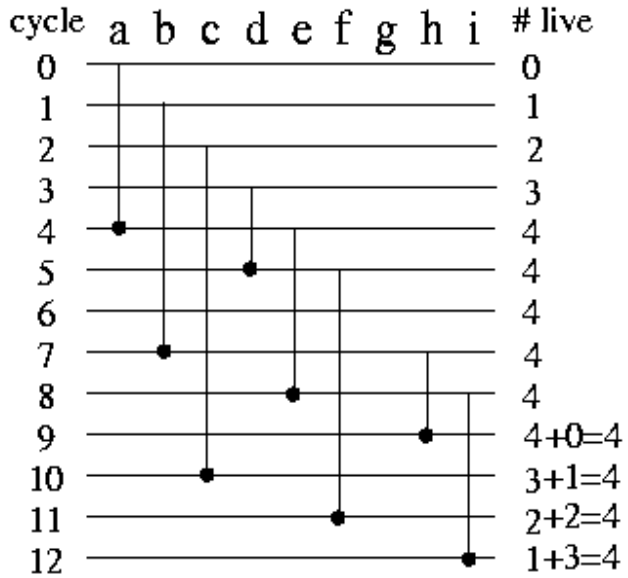


Note : Linear lifetime chart uses the convention that the variable is not live during the clock cycle when it is produced but live during the clock cycle when it is consumed.

- Due to the periodic nature of DSP programs the lifetime chart can be drawn for only one iteration to give an indication of the # of registers that are needed. This is done as follows :
  - Let N be the iteration period
  - Let the # of live variables at time partitions $n \geq N$ be the # of live variables due to 0-th iteration at cycles n-kN for $k \geq 0$. In the example, # of live variables at cycle $7 \geq N$ (=6) is the sum of the # of live variables due to the 0-th iteration at cycles 7 and $(7 - 1 \times 6) = 1$, which is 2 + 1 = 3.
- Matrix transpose example :

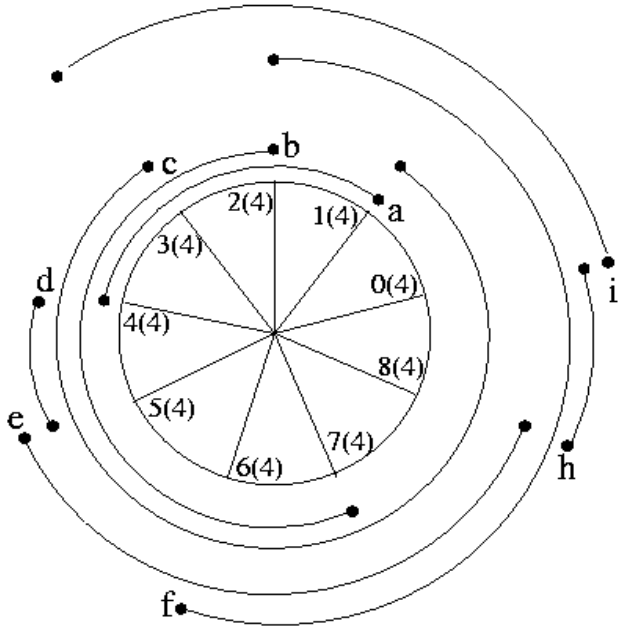
Sample	T_{in}	T_{zlout}	T_{diff}	T_{out}	Life
a	0	0	0	4	0→4
b	1	3	2	7	1→7
c	2	6	4	10	2→10
d	3	1	-2	5	3→5
e	4	4	0	8	4→8
f	5	7	2	11	5→11
g	6	2	-4	6	6→6
h	7	5	-2	9	7→9
i	8	8	0	12	8→12

- To make the system causal a latency of 4 is added to the difference so that T_{out} is the actual output time.

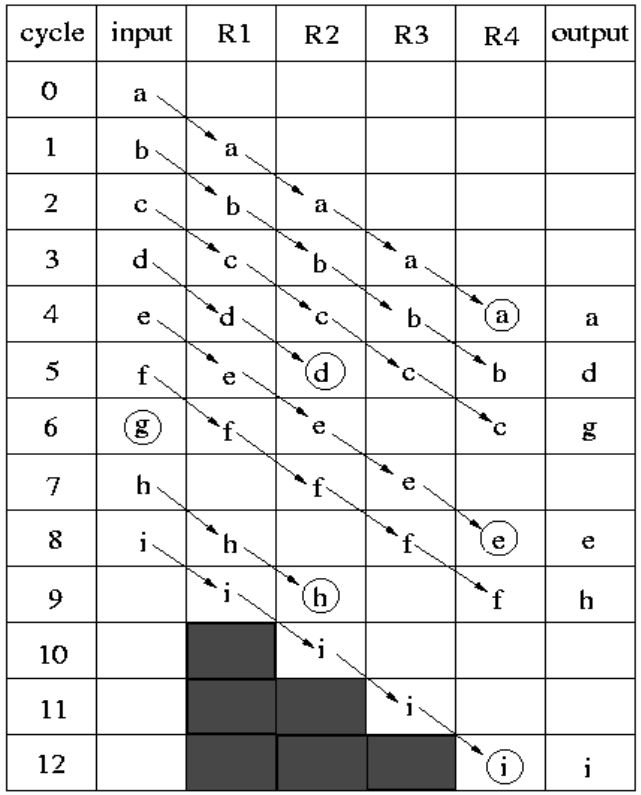


- Circular lifetime chart : Useful to represent the periodic nature of the DSP programs.
- In a circular lifetime chart of periodicity N, the point marked i ($0 \leq i \leq N - 1$) represents the time partition i and all time instances {(N*l* + i)} where *l* is any non-negative integer.
- For example : If N = 8, then time partition i = 3 represents time instances {3, 11, 19, ...}.

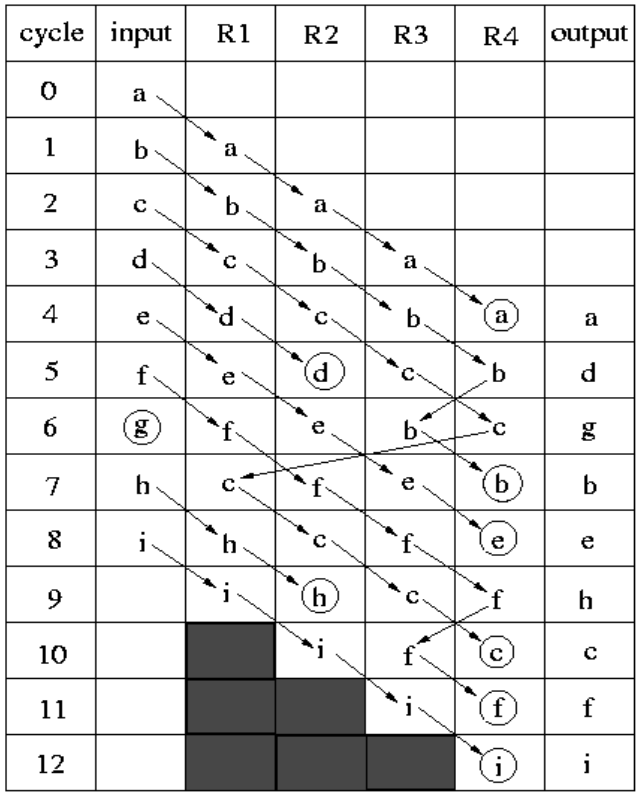
- Note : Variable produced during time unit j and consumed during time unit k is shown to be alive from 'j + 1' to 'k'.
- The numbers in the bracket in the adjacent figure correspond to the # of live variables at each time partition.



Forward Backward Register Allocation Technique :



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

Note : Hashing is done to avoid conflict during backward allocation.

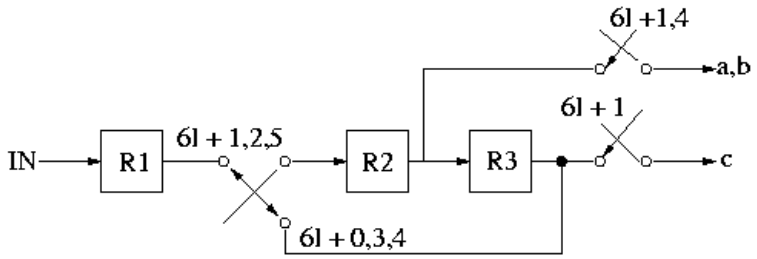
Steps for Forward-Backward Register allocation :

- Determine the minimum number of registers using lifetime analysis.
- 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.
- 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.
- 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.
- 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.
- Repeat steps 4 and 5 until the allocation is complete.

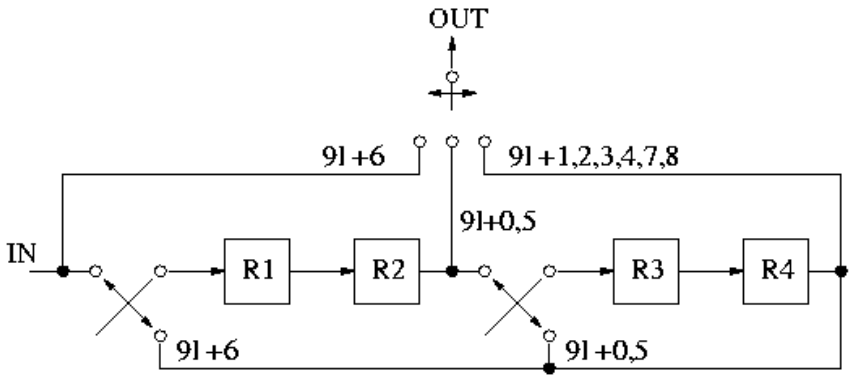
- Example : Forward backward Register Allocation

cycle	input	R1	R2	R3	output
0	a				
1	b	a			
2		b	a		
3			b	a	
4	c			b	
5		c			
6			c		
7				(c)	c

cycle	input	R1	R2	R3	output
0	a				
1	b	a			
2		b	a		
3			b	a	
4	c		(a)	b	a
5		c	b		
6			c	b	
7			(b)	(c)	b, c



- Folded architecture for matrix tranposer :



- 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.

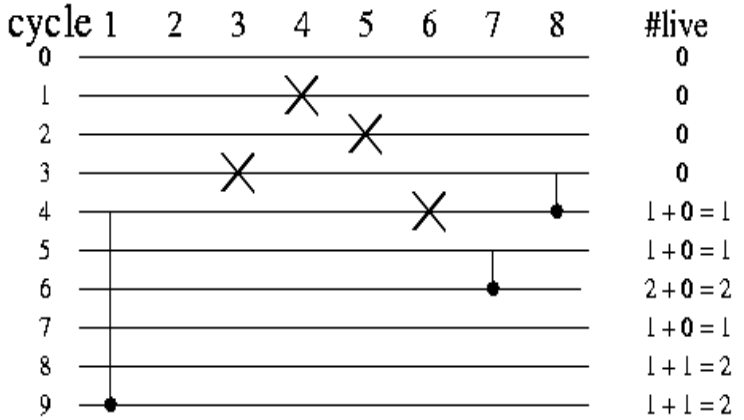
•Example : Biquad Filter

- 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

- 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_8		n_8
5	n_7	n_1		
6		n	n_1	n_7
7			n_1	
8			n_1	
9			n	n_1

➢Folded architecture is drawn with minimum # of registers.