HW6 Retiming, Unfolding and Folding

VLSI DSP HW6

Shun-Liang Yeh, NCHU Lab612

6/09/2023

INDEX

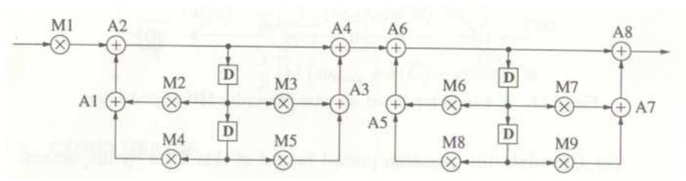
I. Problem & Solutions

Q1

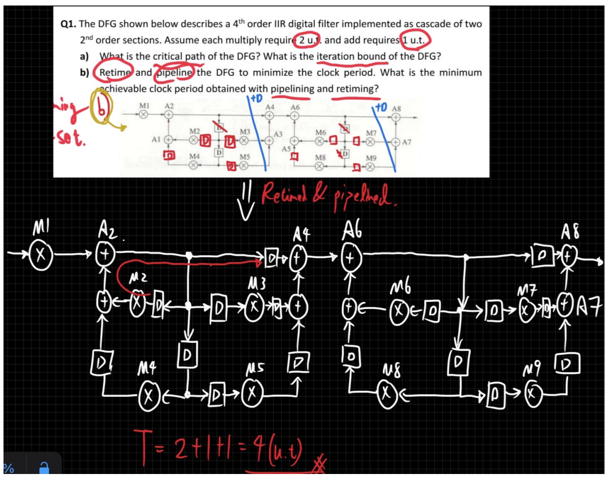
Q1. The DFG shown below describes a 4th order IIR digital filter implemented as cascade of two 2nd order sections. Assume each multiply require 2 u.t. and add requires 1 u.t.

a) What is the critical path of the DFG? What is the iteration bound of the DFG?

b) Retime and pipeline the DFG to minimize the clock period. What is the minimum achievable clock period obtained with pipelining and retiming?



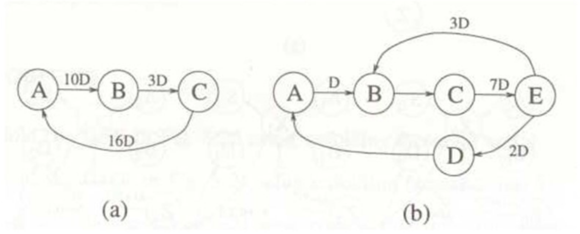
- Remember that you can do retiming and pipelining, not just simply perform piplining. The concept of pipelining feedforward cutsets and retiming feedback cutset is important.
- The art of retiming means moving the delays around the circuit, remember to draw out the retiming contour to ensure that you are doing the correct retiming.

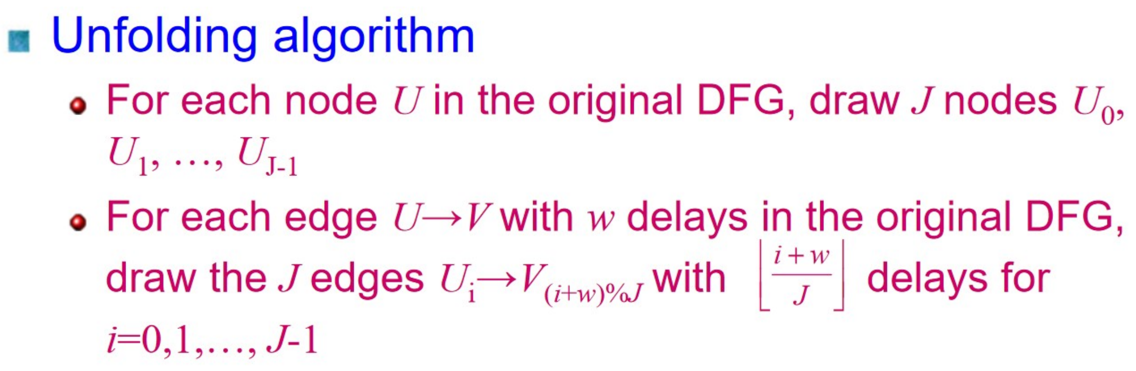


$T = 2+1+1 = 4(u.t)$ #

Q2

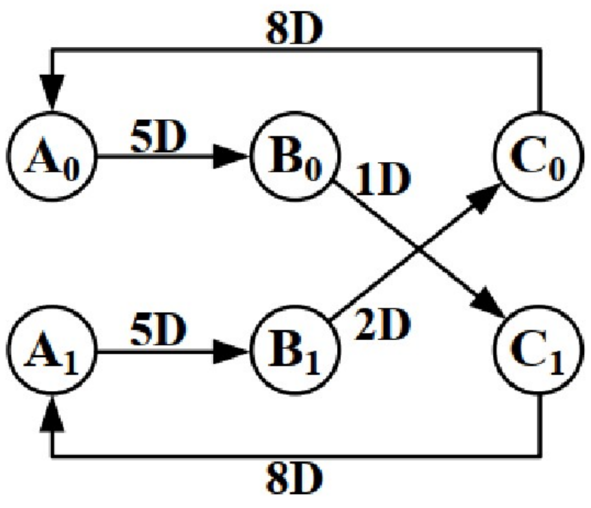
Q2. Unfold the DFG shown below using unfolding factors 2 and 5



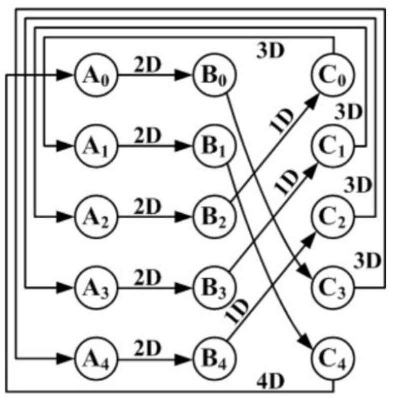


- Unfolding algorithm
  - For each node U in the original DFG, draw J nodes U_0, U_1, …, U_{J-1}
  - For each edge $U \rightarrow V$ with w delays in the original DFG, draw the J edges $U_i \rightarrow V_{(i+w)\%J}$ with $\left\lfloor \frac{i+w}{J} \right\rfloor$ delays for i=0,1,…, J-1

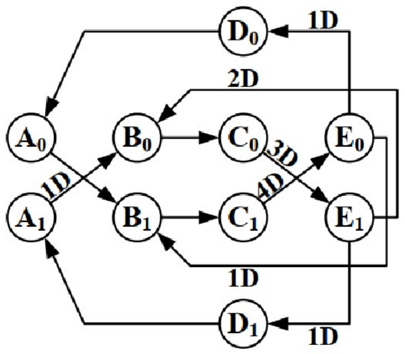
a) J=2



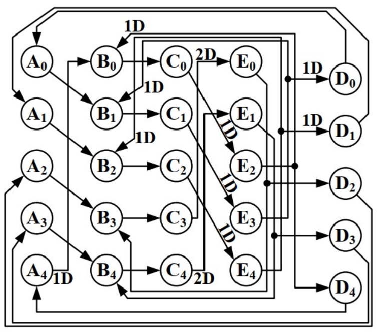
a) J=5



b) J=2



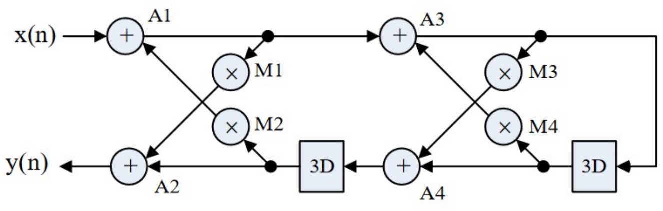
b) J=5



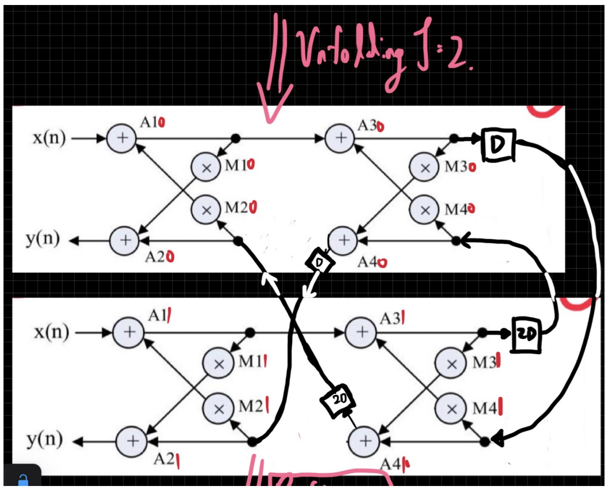
- Unfolding algorithm can be used to achieve the minimal Sampling period, the iteration bound however cannot be changed using unfolding.
- If the iteration bound of the circuit is not an integer, unfolding the circuit might help achieve the minimal sampling period at the cost of more HWs.
- Dummy nodes can be added to simplify the process of Unfolding a circuit.

Q3

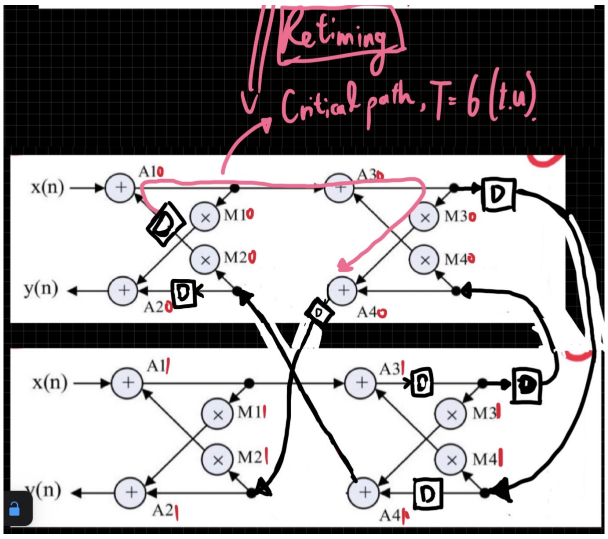
Q3. For the DFG shown below, assume the computing times of multiplier and adder are 3 u.t and 1 u.t., respectively. Unfold the design by a factor of 2 and apply retiming to achieve a critical path delay of 6 u.t..



First perform Unfolding with J = 2

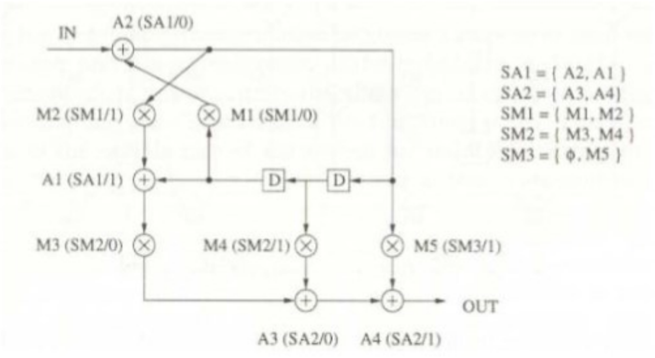


Perform the Retiming to achieve critical path of T = 6



Q4

Q4. For the design shown below.



a) For a folded design with 5 folding sets (SA1, SA2, SM1, SM2, SM3) and a folding factor equal to 2, verify if this is a valid folding

b) If it is not, find a new valid folding design by modifying the folding orders in each set. **Note:** Each folding set has only two possible folding orders. In total, there are 32 possible combinations of folding orders. You may write a program to enumerate all 32 possible combinations and check if any of them are valid.

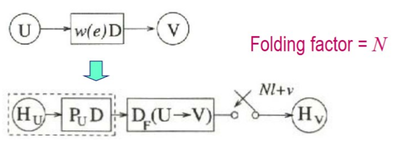
c) In case, you cannot identify any valid combinations of folding orders in (b), perform retiming first followed by folding

d) Derive your folded design without register minimization based on the folding obtained in (b) or (c). You need to draw the folded circuit design.

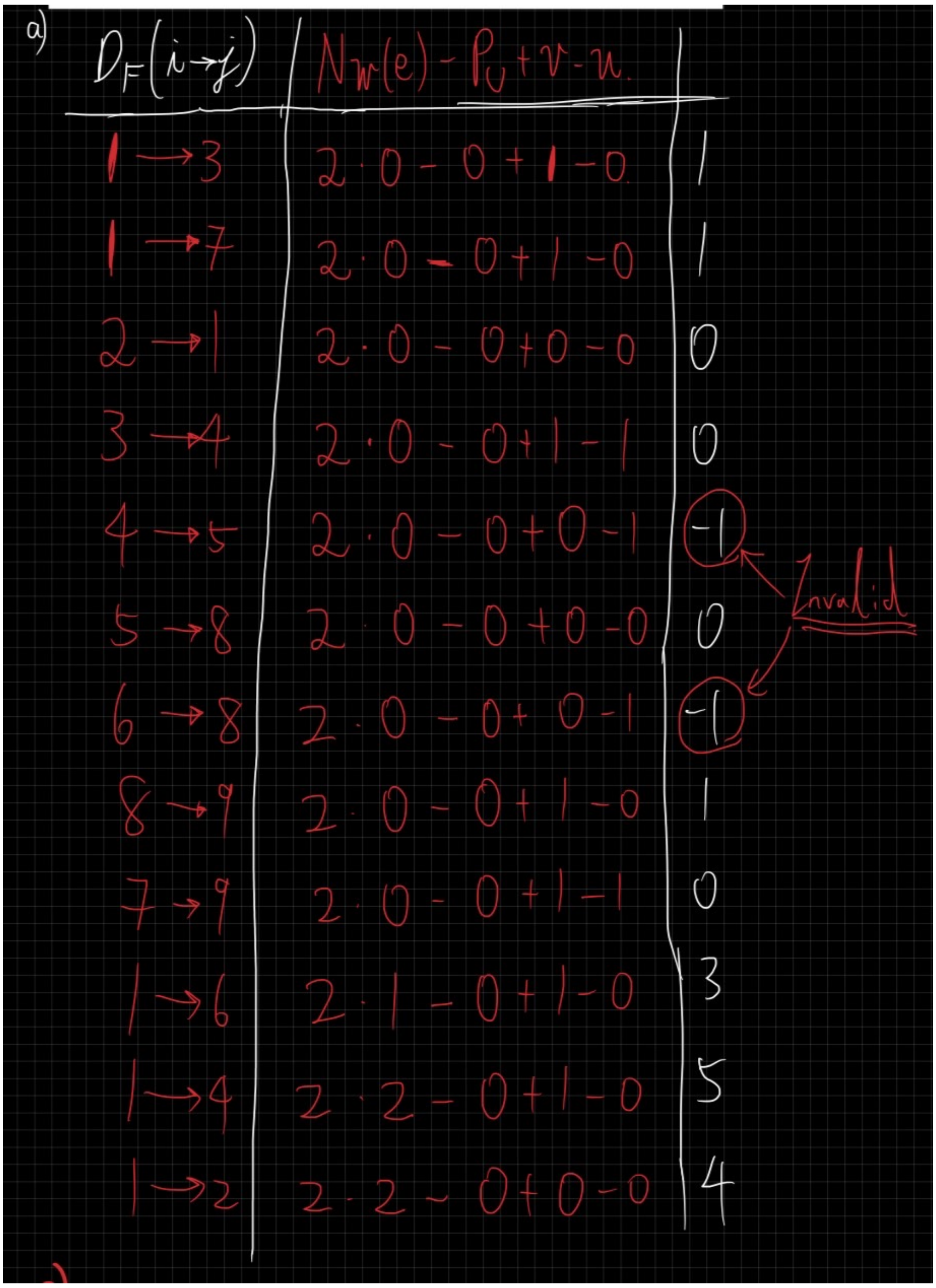
e) Repeat (d) using the forward backward register allocation scheme to minimize the register usage. **Note:** you need to show the derivation details, e.g. life time chart, register allocation table, and so on.

- Preliminary
  - Consider a DFG
  - An edge e connecting nodes U and V with $w(e)$ delays
  - Executions of the l-th iterations of U and V at time units $Nl+u$ and $Nl+v$
  - u and v: folding orders and $0 \le u,v \le N\text{-}1$
  - N: folding factor, the number of operations folded to a single function unit
  - H_U and H_V: function units to execute nodes U and V
  - H_U is pipelined by P_U stages

- Folding an edge
  - $U \xrightarrow{e} V$ has $w(e)$ delays
  - l-th iteration of node U is available at time $Nl+u+P_U$
  - Generated data is used by the $(l+w(e))$-th iteration of V
  - The result must be stored for
  - $D_F(U \xrightarrow{e} V) = [N(l+w(e))+v] - [Nl+P_U+u] = Nw(e) - P_U + v - u$



a)



a)

$D_F(i \to j)$	$Nw(e) - P_U + v - u$	
$1 \to 3$	$2 \cdot 0 - 0 + 1 - 0$	1
$1 \to 7$	$2 \cdot 0 - 0 + 1 - 0$	1
$2 \to 1$	$2 \cdot 0 - 0 + 0 - 0$	0
$3 \to 4$	$2 \cdot 0 - 0 + 1 - 1$	0
$4 \to 5$	$2 \cdot 0 - 0 + 0 - 1$	$\boxed{-1}$
$5 \to 8$	$2 \cdot 0 - 0 + 0 - 0$	0
$6 \to 8$	$2 \cdot 0 - 0 + 0 - 1$	$\boxed{-1}$
$8 \to 9$	$2 \cdot 0 - 0 + 1 - 0$	1
$7 \to 9$	$2 \cdot 0 - 0 + 1 - 1$	0
$1 \to 6$	$2 \cdot 1 - 0 + 1 - 0$	3
$1 \to 4$	$2 \cdot 2 - 0 + 1 - 0$	5
$1 \to 2$	$2 \cdot 2 - 0 + 0 - 0$	4

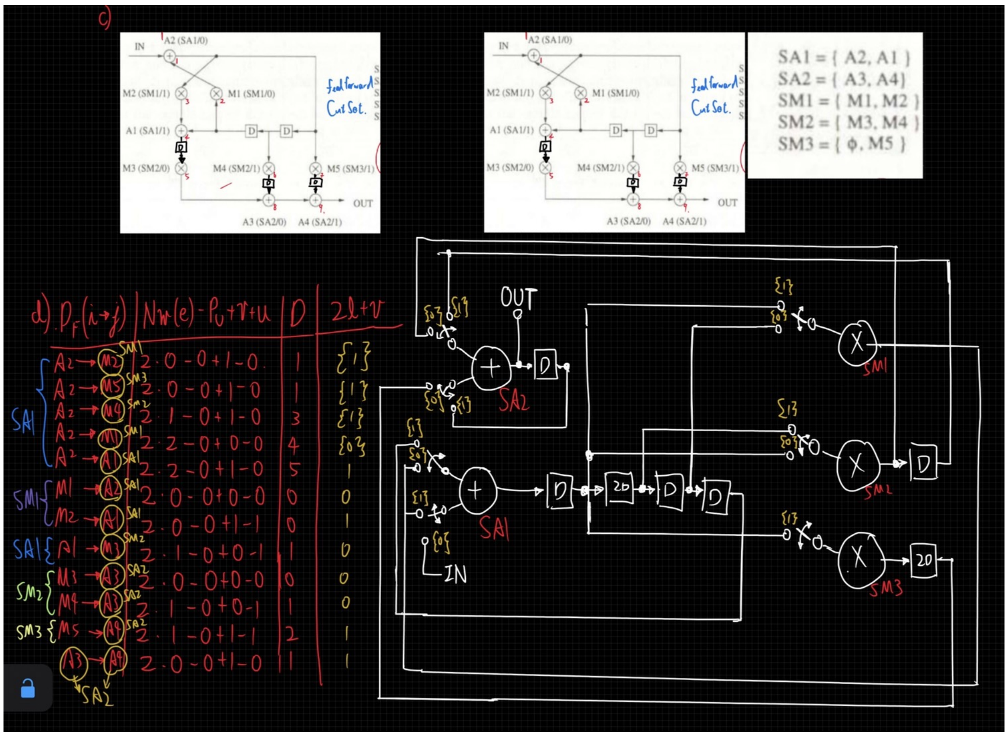
Invalid

b)

- By high level programming language, we can verify that all the permutations of the folding set is not valid.

c) d)

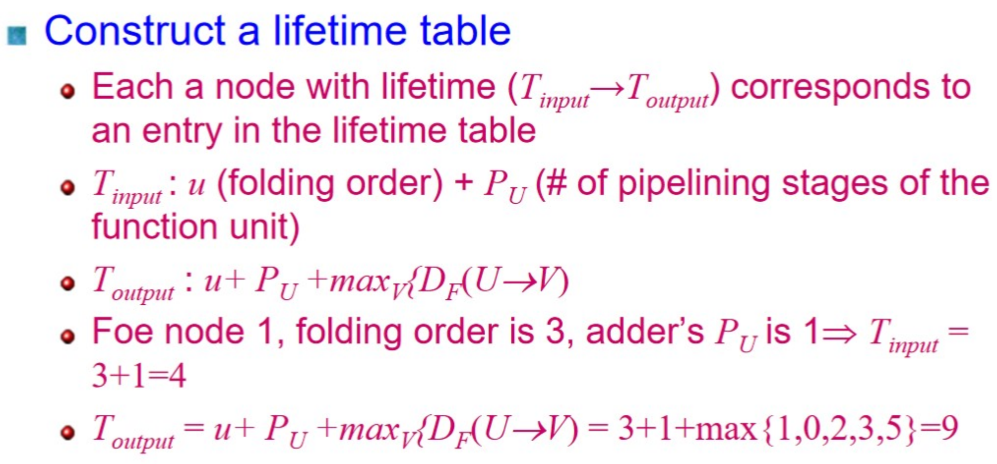
- **Goal**
  - To synthesize control circuits in folded architectures with minimum number of registers
- **Procedures**
  - Perform retiming for folding
  - Write folding equations
  - Use the folding equations to construct a lifetime table
  - Draw the lifetime chart and determine the required number of registers
  - Perform forward-backward register allocation
  - Draw the folded architecture that uses the minimum number of registers



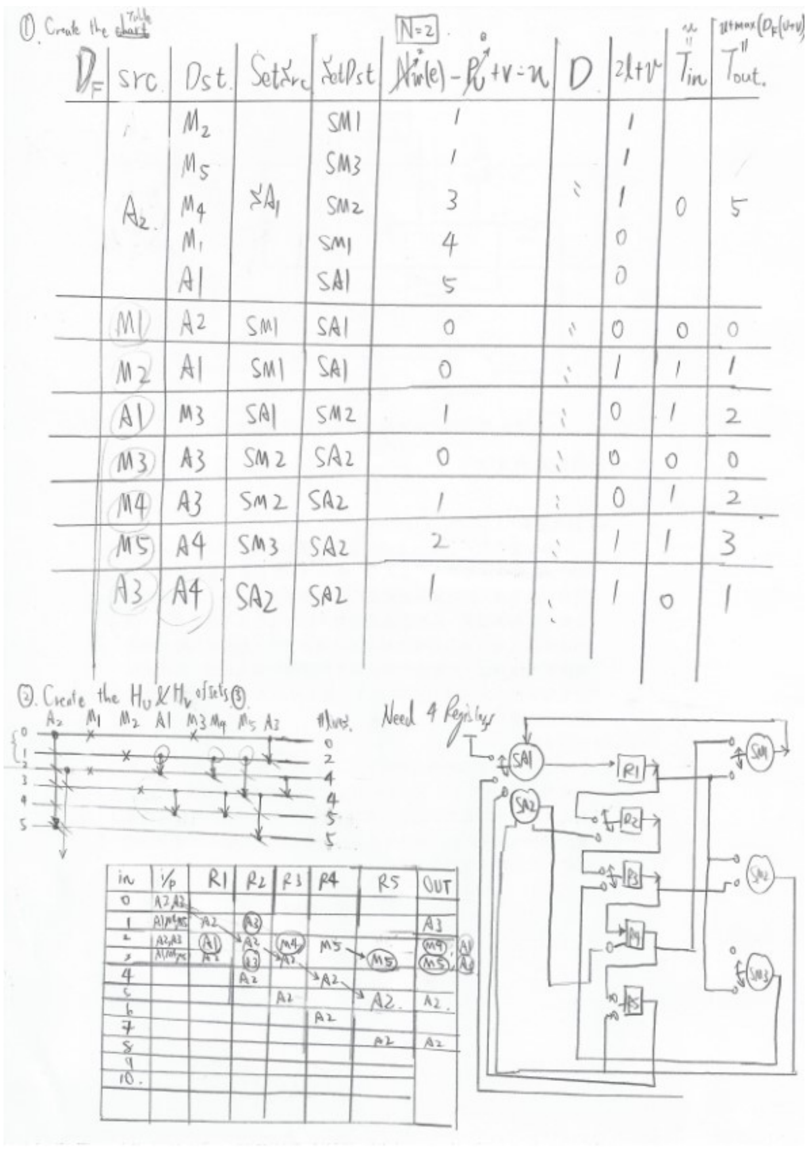
SA1 = { A2, A1 }
SA2 = { A3, A4 }
SM1 = { M1, M2 }
SM2 = { M3, M4 }
SM3 = { ϕ, M5 }

d) $D_F(i \to j)$	$N w(e) - P_U + v + u$	D	$2l + v$
SA1 A2 → M2 SM1	2·0 − 0 + 1 − 0	1	{1}
A2 → M5 SM3	2·0 − 0 + 1 − 0	1	{1}
A2 → M4 SM2	2·1 − 0 + 1 − 0	3	{1}
A2 → M1 SM1	2·2 − 0 + 0 − 0	4	{0}
A2 → A1 SA1	2·2 − 0 + 1 − 0	5	1
SM1 M1 → A2 SA1	2·0 − 0 + 0 − 0	0	0
M2 → A1 SA1	2·0 − 0 + 1 − 1	0	1
SA1 A1 → M3 SM2	2·1 − 0 + 0 − 1	1	0
SM2 M3 → A3 SA2	2·0 − 0 + 0 − 0	0	0
M4 → A3 SA2	2·1 − 0 + 0 − 1	1	0
SM3 M5 → A4 SA2	2·1 − 0 + 1 − 1	2	1
A3 → A4 SA2	2·0 − 0 + 1 − 0	1	1

e)



- **Construct a lifetime table**
  - Each a node with lifetime ($T_{input} \rightarrow T_{output}$) corresponds to an entry in the lifetime table
  - T_{input} : u (folding order) + P_U (# of pipelining stages of the function unit)
  - T_{output} : $u + P_U + max_V\{D_F(U \rightarrow V)$
  - Foe node 1, folding order is 3, adder's P_U is 1 $\Rightarrow T_{input} = 3+1=4$
  - $T_{output} = u + P_U + max_V\{D_F(U \rightarrow V) = 3+1+\max\{1,0,2,3,5\}=9$



① Create the ~~chart~~ Table

$\boxed{N=2}$

D_F	src	Dst	Set src	Set Dst	$Nw(e) - P_U + v - u$	D	$2l+v$	T_{in}	T_{out}
	A_2	M_2	SA_1	SM1	1		1	0	5
		M_5		SM3	1		1		
		M_4		SM2	3		1		
		M_1		SM1	4		0		
		A1		SA1	5		0		
	M1	A2	SM1	SA1	0		0	0	0
	M2	A1	SM1	SA1	0		1	1	1
	A1	M3	SA1	SM2	1		0	1	2
	M3	A3	SM2	SA2	0		0	0	0
	M4	A3	SM2	SA2	1		0	1	2
	M5	A4	SM3	SA2	2		1	1	3
	A3	A4	SA2	SA2	1		1	0	1

$u + \max(D_F(U \rightarrow V))$

② Create the H_U & H_V of sets ③

Need 4 Registers

in	I/P	R1	R2	R3	R4	R5	OUT
0	A2,A3						
1	A1,M4,M5	A2	A3				A3
2	A2,A3	A1	A2	M4	M5		M4; A1
3	A1,M4,M5	A1	A3	A2		M5	M5; A3
4			A2		A2		
5				A2		A2.	A2.
6					A2		
7							
8						A2	A2
9							
10.							

Note

1. While doing register minimization, remember to mark the Src Node, Destination node and their correspondent sets to simplify the operations of mapping.
2. Remember where the input and output ports are.
3. Switches are switche in time instance Nl+v
4. Tin and Tout might varies in different architecture.
5. When drawing the linear lifetime table, remember to draw until the next iteration starts to overlap with the past data.

II. References

[1] UMN EE-5329 VLSI Signal Processing Lecture-11 (Spring 2019),Folding Transformation of Data flow Graph , Prof. Keshab Parhi

[2] UMN EE-5329 VLSI Signal Processing Lecture-10 (Spring 2019), , Unfolding of Data-Flow Graphs with Switches and Digit-Serial Design , Prof. Keshab Parhi

[3] VLSI DSP 2023, Lecture Handouts, Ch8 Folding 8-5 ~ 8-38, Y.T Hwang

[4] VLSI DSP 2023, Lecture Handouts, Ch6 Unfolding 7-4, Y.T Hwang