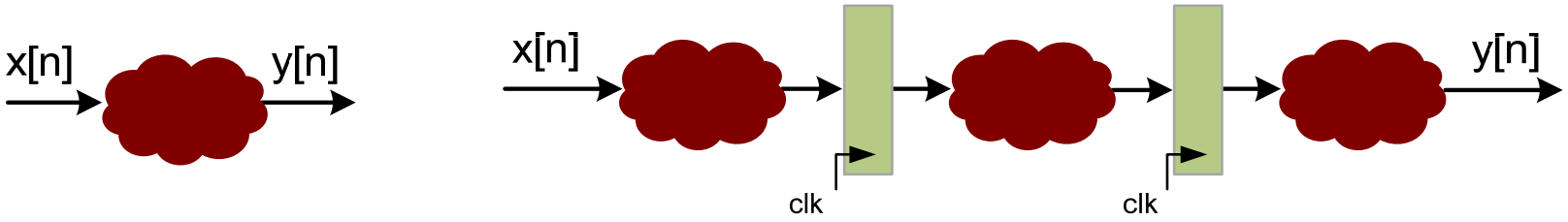
Pipelining & Retiming

Lecture 7

Dr. Shoab A. Khan

Pipelining

- Pipeline registers are added to reduce the critical path delay of a combinational cloud.
- Example
  - A combinational logic is pipelined with three levels of pipelining stages with two sets of pipeline registers



Pipelining & Retiming

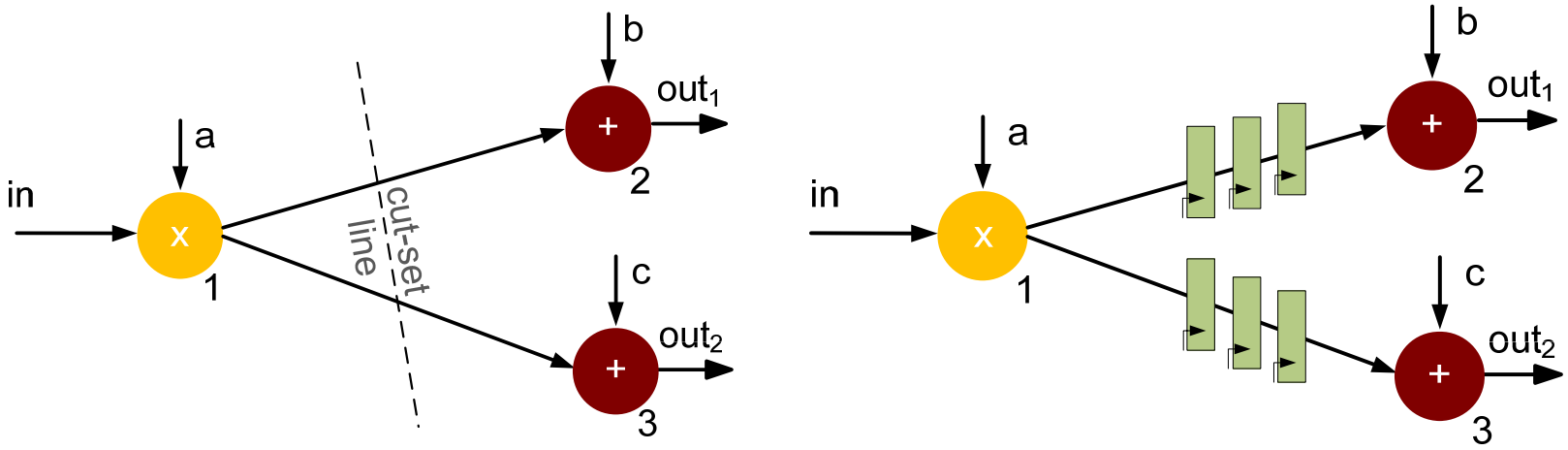
- Pipelining & Retiming

  Retiming is a mapping from a given DFG, G to a retimed DFT, G_r such that the corresponding transfer function of G and G_r only differ by a pure delay z^{-L}

- Purposes
  - To facilitate pipelining
  - To reduce clock cycle time
  - To reduce number of registers needed.

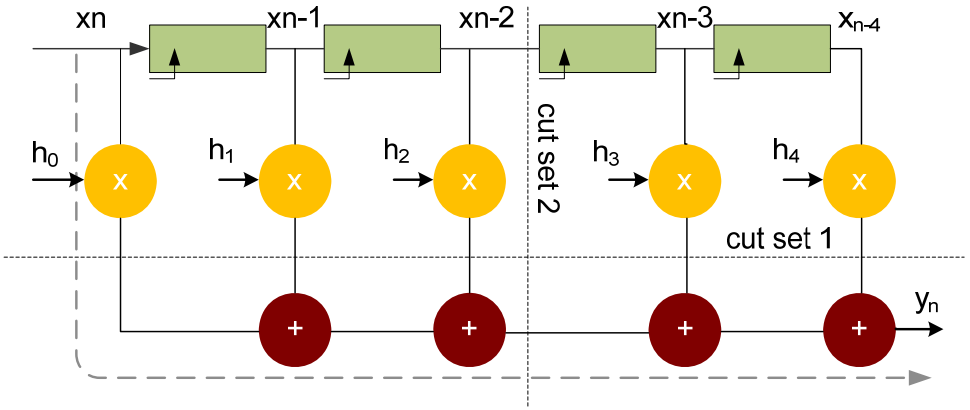
Feed-forward Cut-set & Pipelining

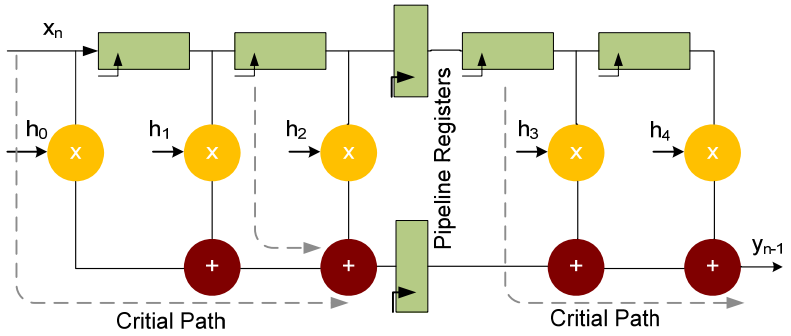
- Feed forward cut-set
- Pipeline registers added on each edge of the cut-set



Pipelining using Feed-forward Cut-set

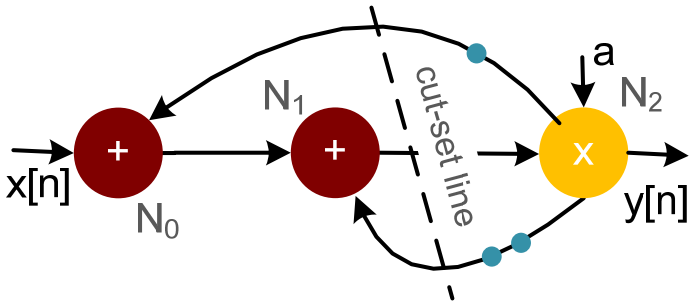
- Two candidate cut-sets in the DFG implementing a 5-coefficient FIR filter.
- One level of pipelining registers added using cut-set 2

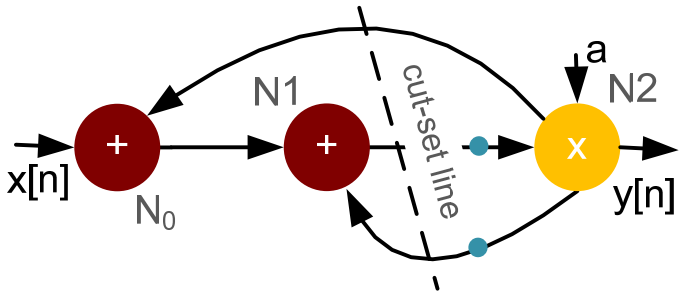




Cut-set Retiming

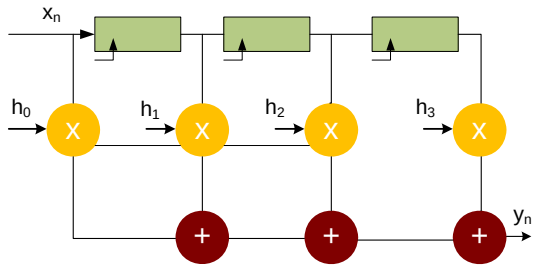
- It involves transferring a number of delays from edges of the same direction across a cut set line of a DFG to all edges of opposing direction across the same line.

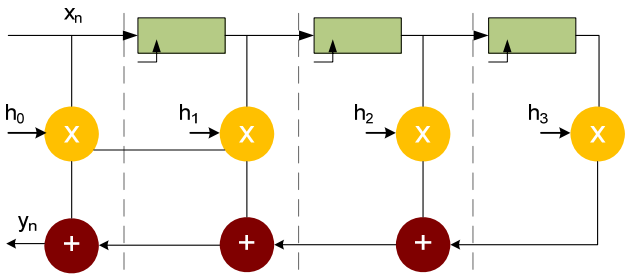


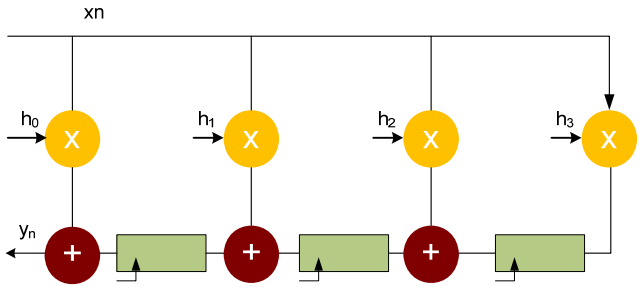


Cut-set Retiming: DF to TDF

- A 4-coefficient FIR filter in DF.
- Reversing the direction of additions in the DF and applying cut-set retiming.
- The retimed filter in TDF





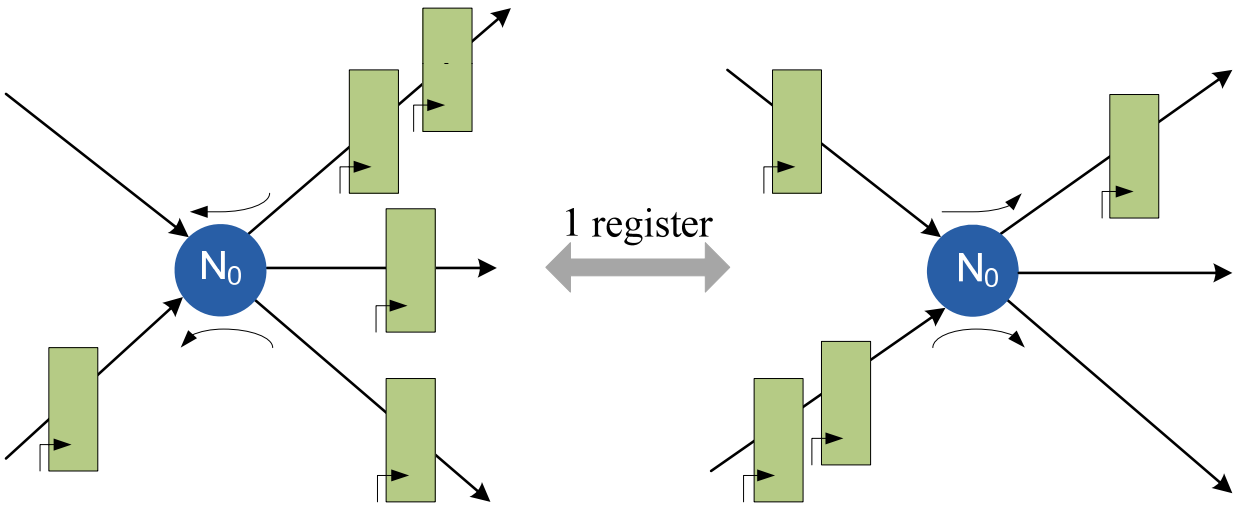


Delay Transfer Theorem

- Without affecting the transfer function of the system, N registers can be transferred from each incoming edge of a node of a DFG to all outgoing edges of the same node,

Node Transfer Theorem

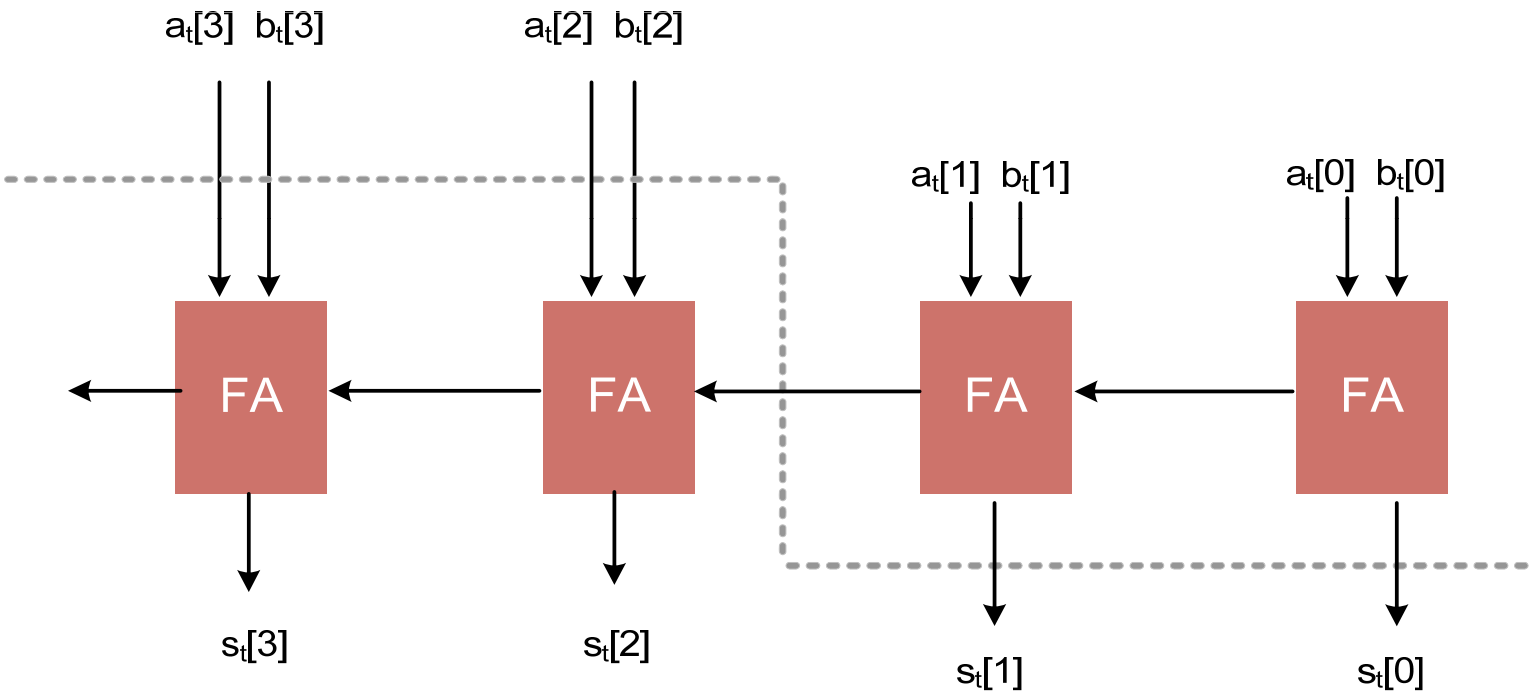
- Delay transfer theorem moves one register each from outgoing edges to incoming edges across N_0



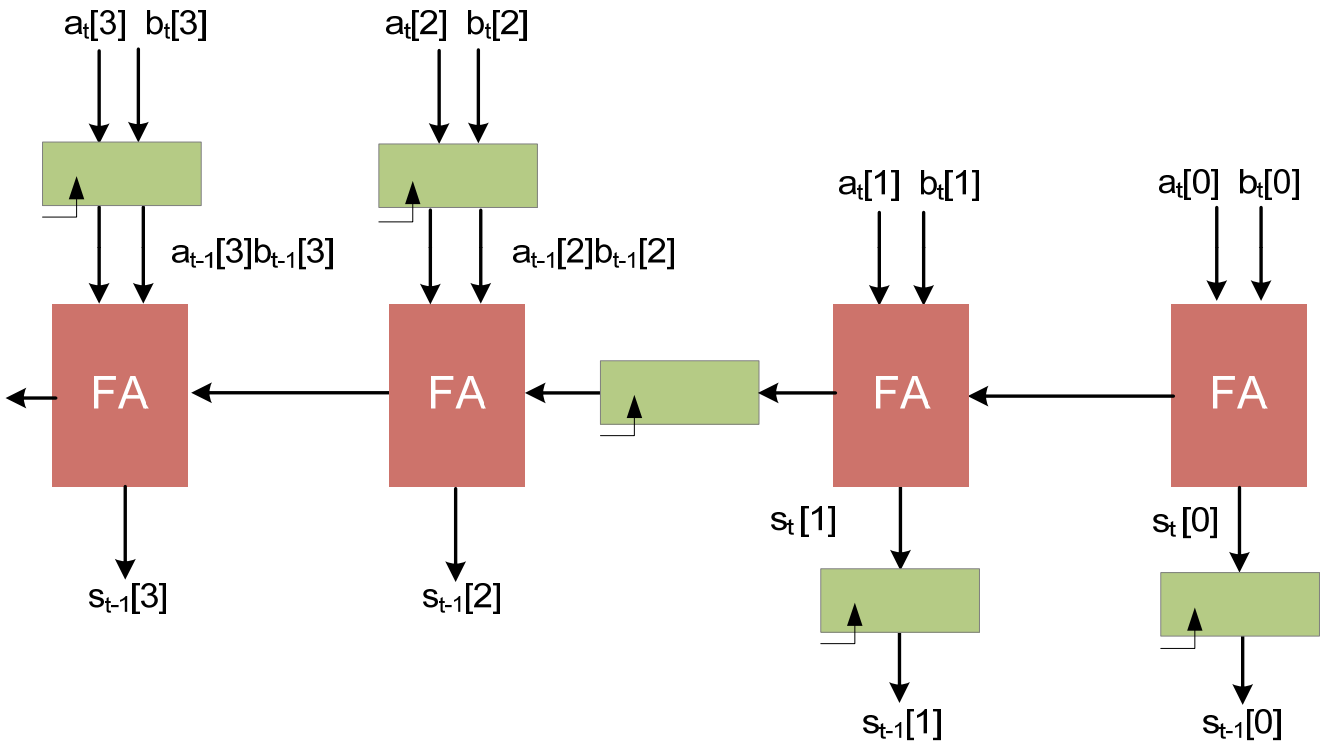
Pipelining using cut-set

- Apply valid cut-set and add pipeline registers and then retime them if required

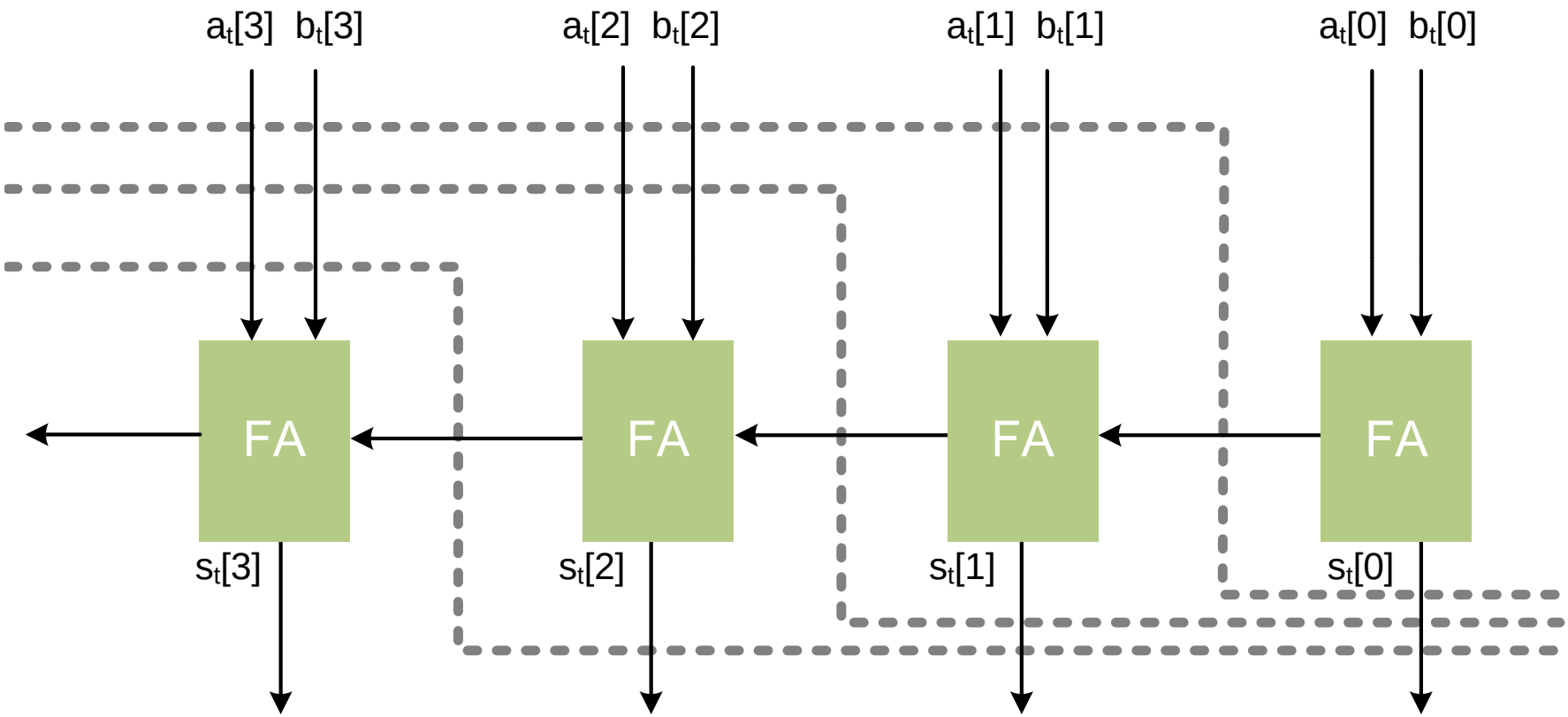
A cut-set to pipeline a 4-bit RCA



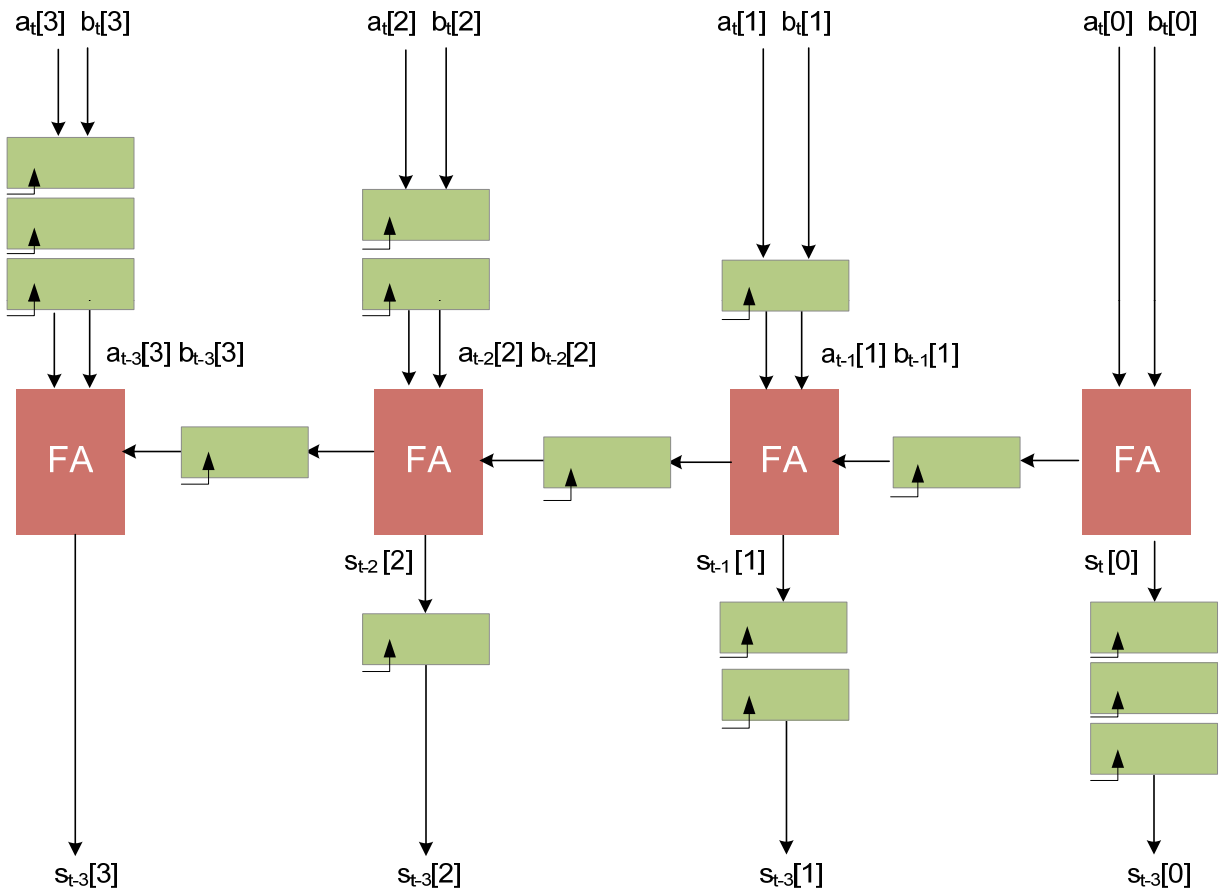
Placing pipeline registers along the cut-set line



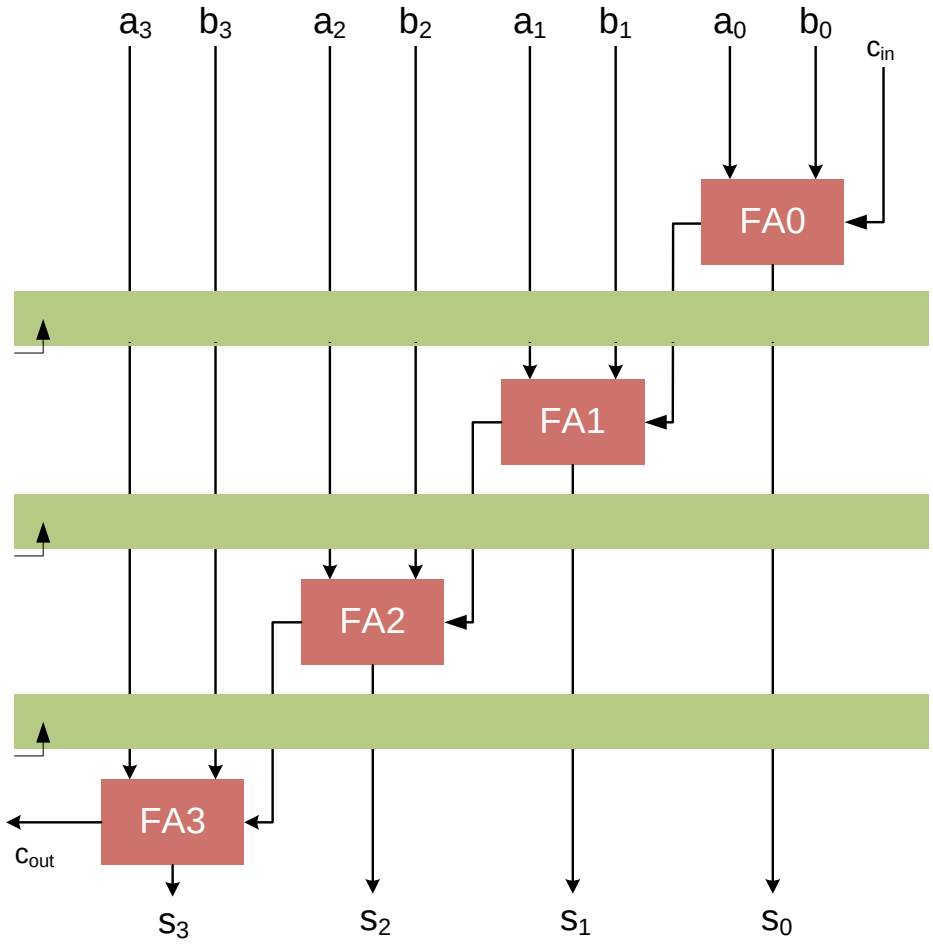
Three cut-sets for adding four pipeline stages in a 4-bit RCA



Four-stage pipelined 4-bit RCA



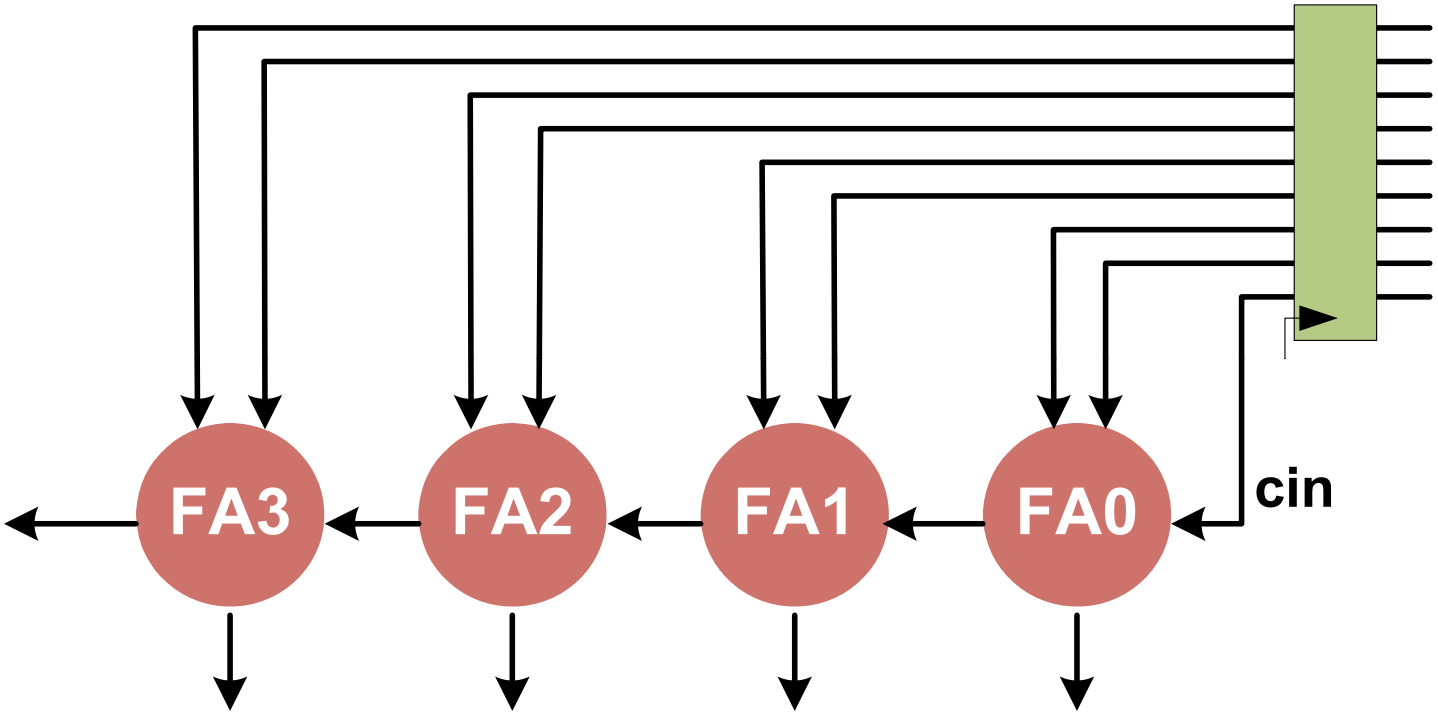
Following good design practice by lining up of different pipeline stages



Pipelining Using Nodal Transfer Theorem

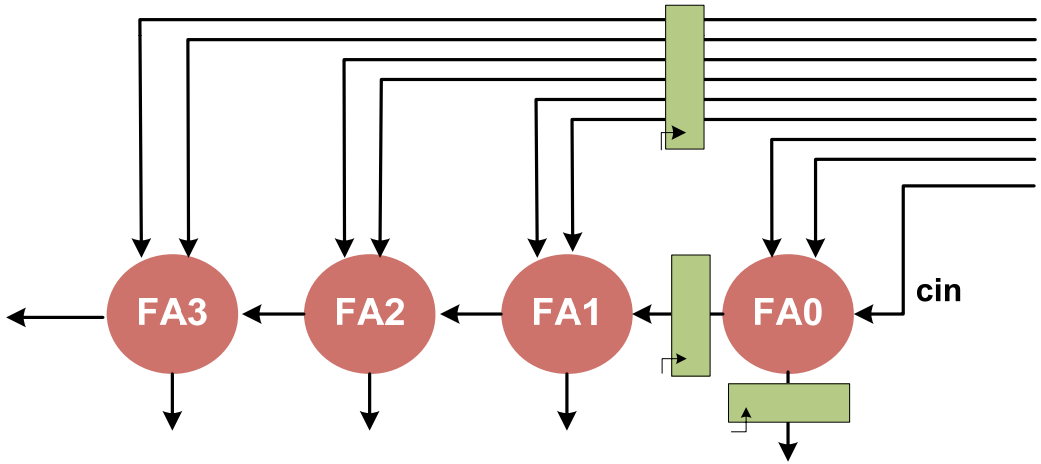
- Add the desired number of registers to all input edges and then, by repeated application of the node transfer theorem, systematically move the registers to break the delay of the critical path

Original DFG

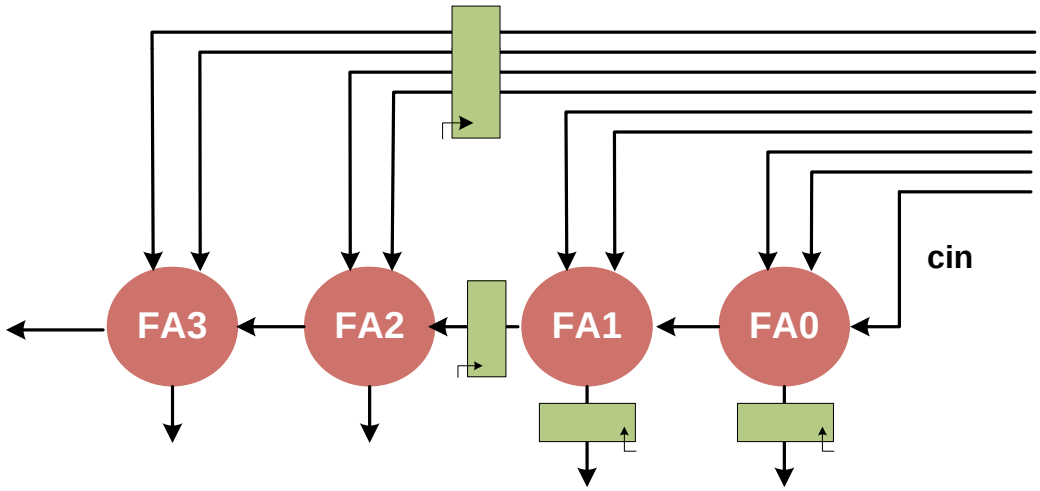


- Node transfer theorem to add one level of pipeline registers in a 4-bit RCA

Node transfer theorem applied around node FA0



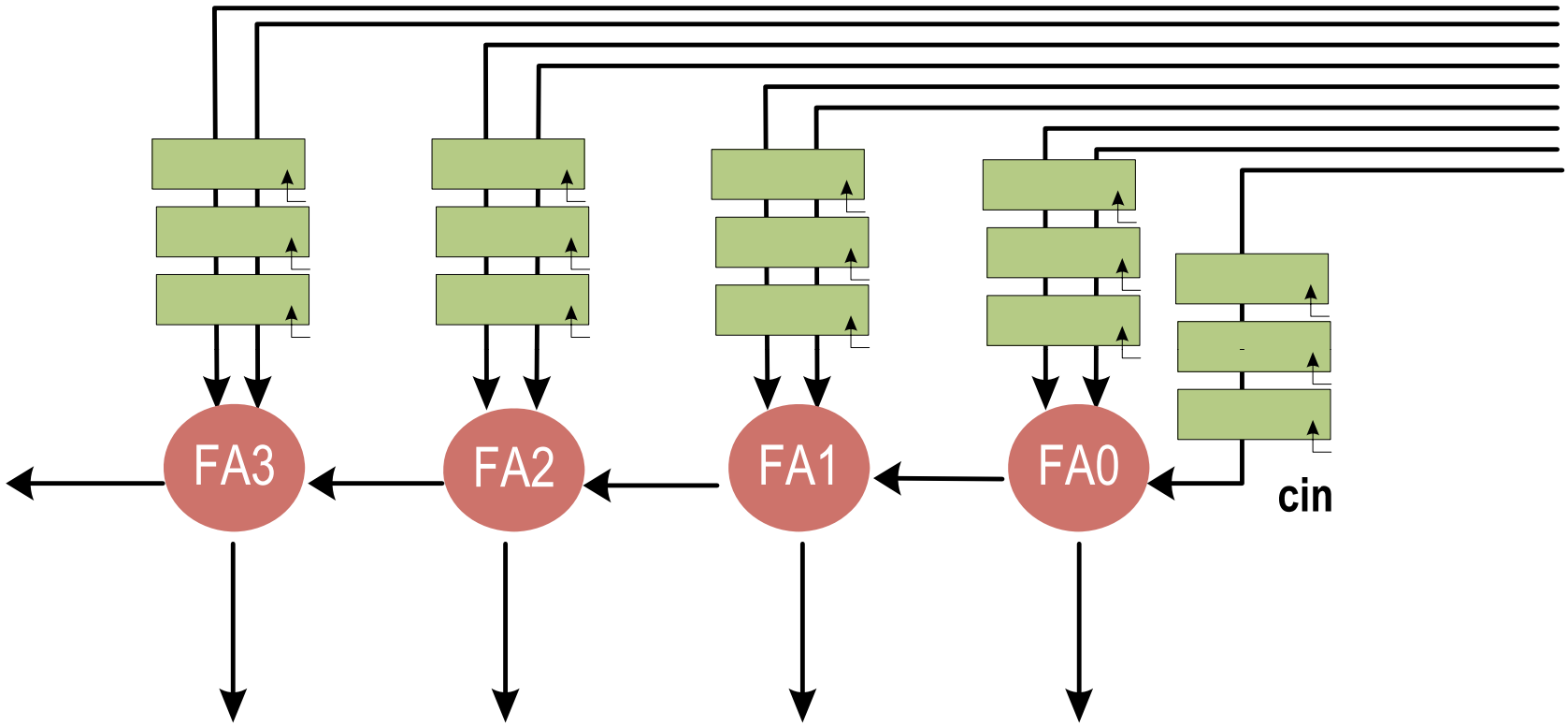
Node transfer theorem applied around FA1



RCA Example

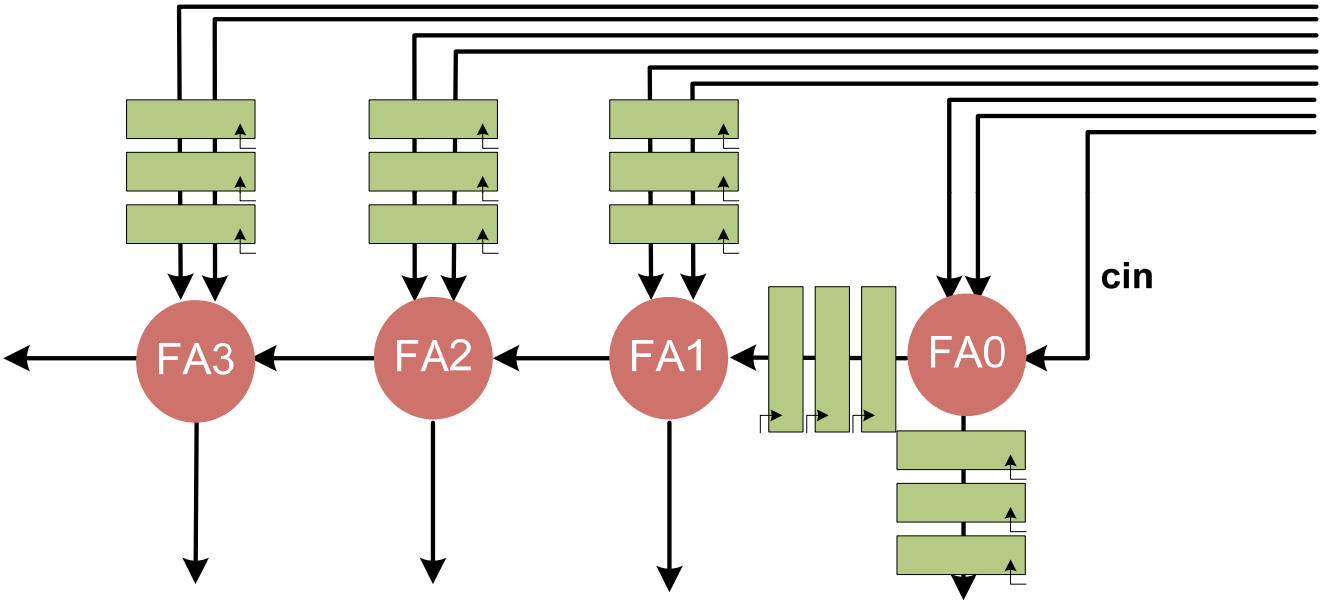
- Adding three stages of pipeline registers by applying the node delay transfer theorem on
  - Original DFG, around
  - FA0
  - FA1
  - FA2

Delay Transfer Theorem

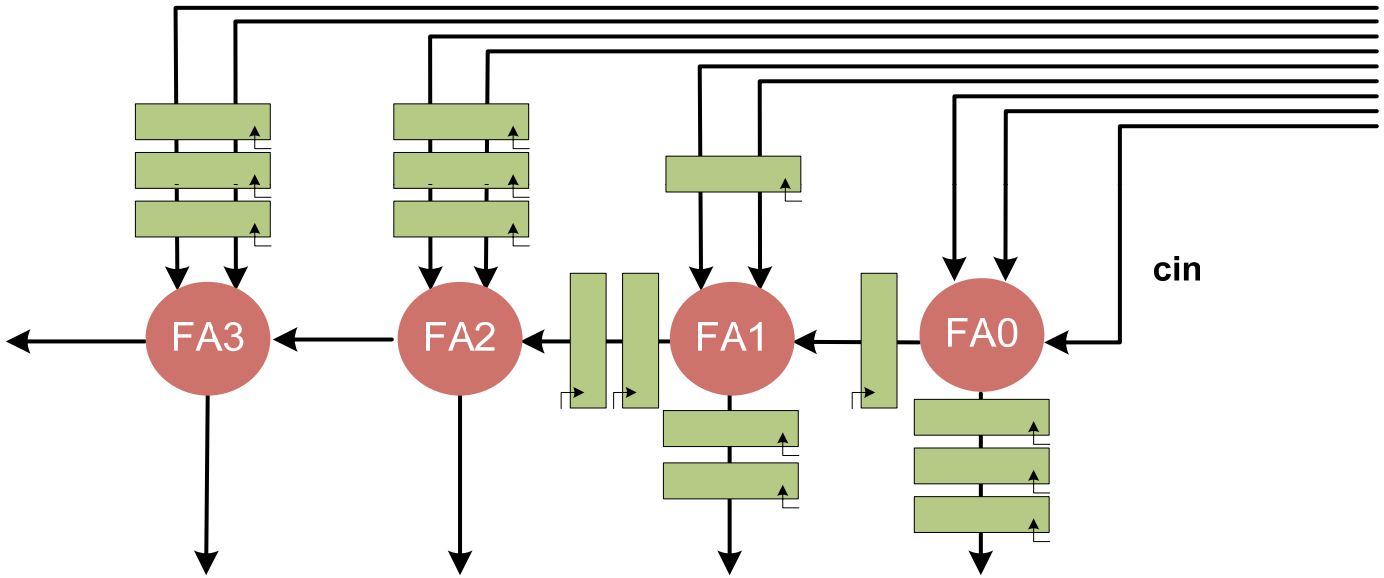


- Adding three stages of pipeline registers by applying the node delay transfer theorem on

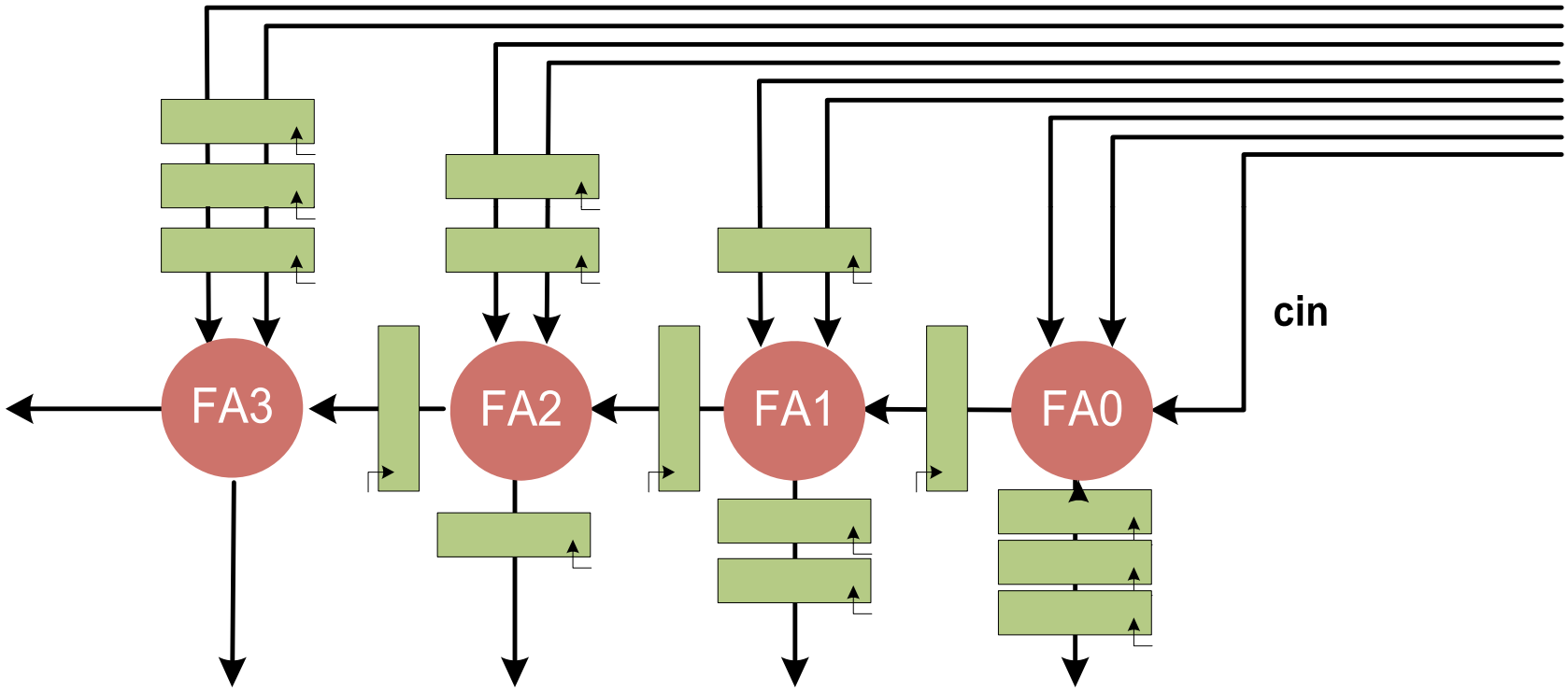
Delay Transfer across FA0



Delay Transfer Across FA1

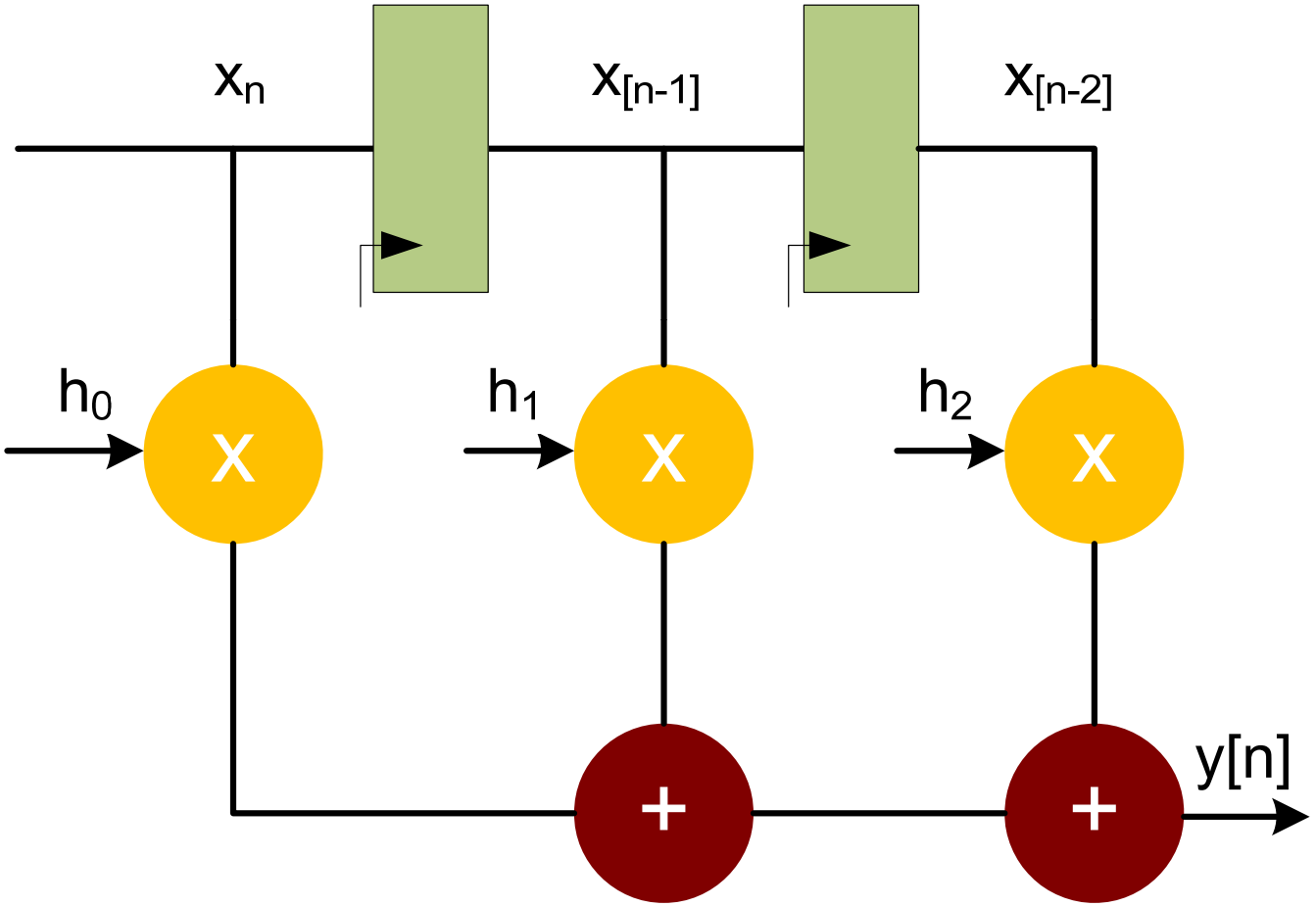


Delay Transfer across FA2

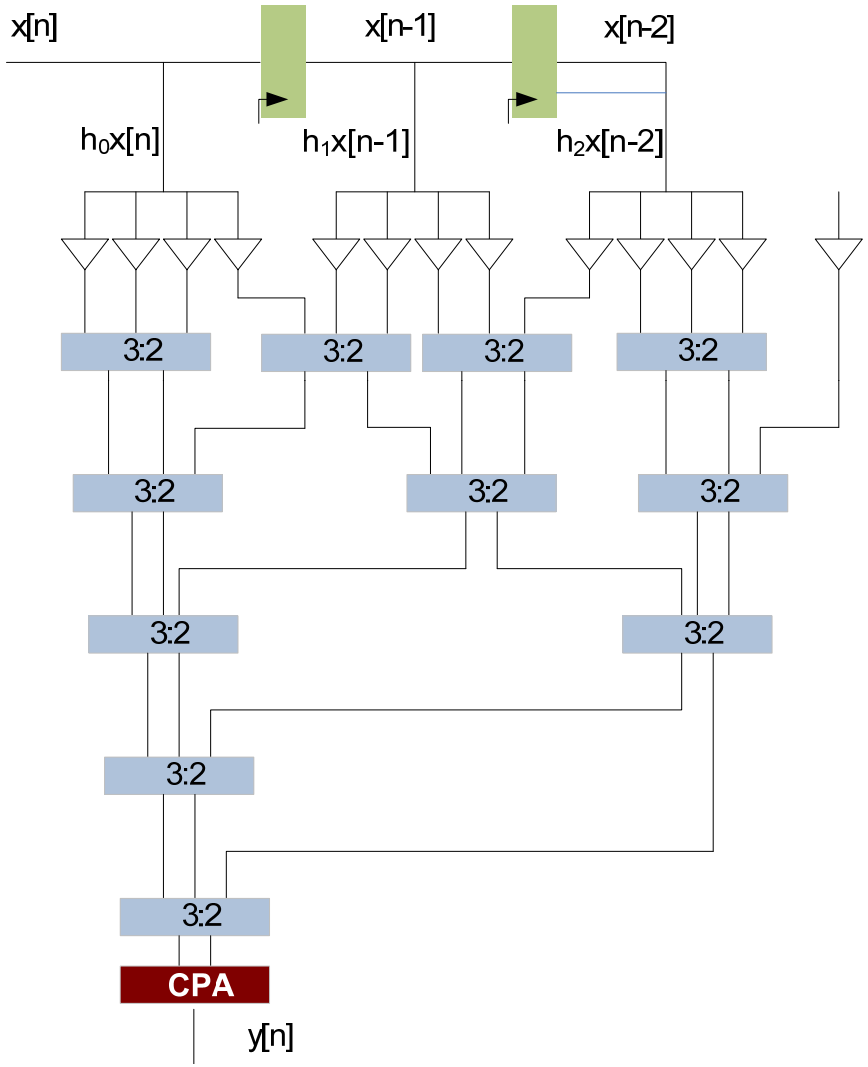


Pipelining Optimized DFG

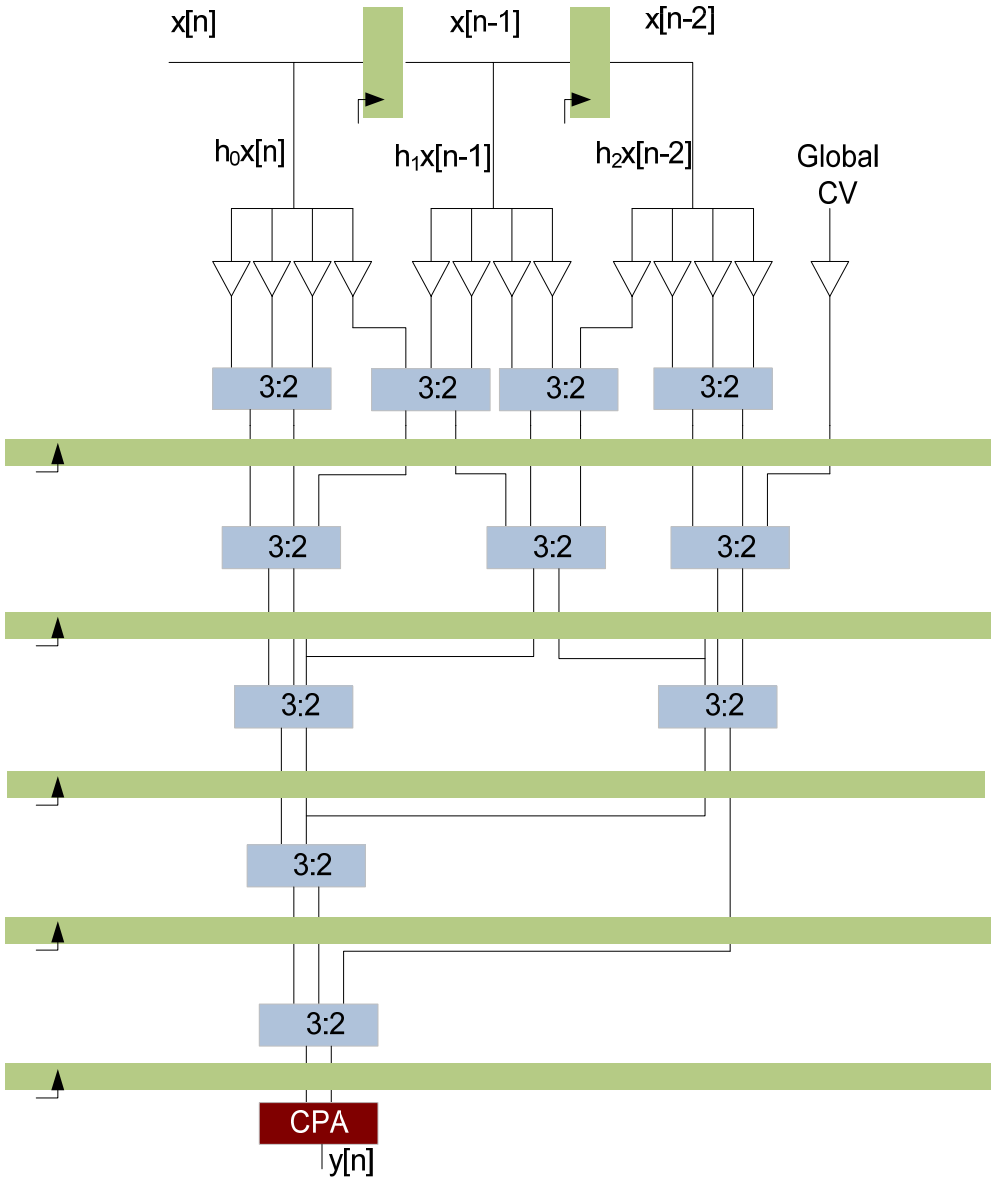
Direct-form FIR filter with three coefficients



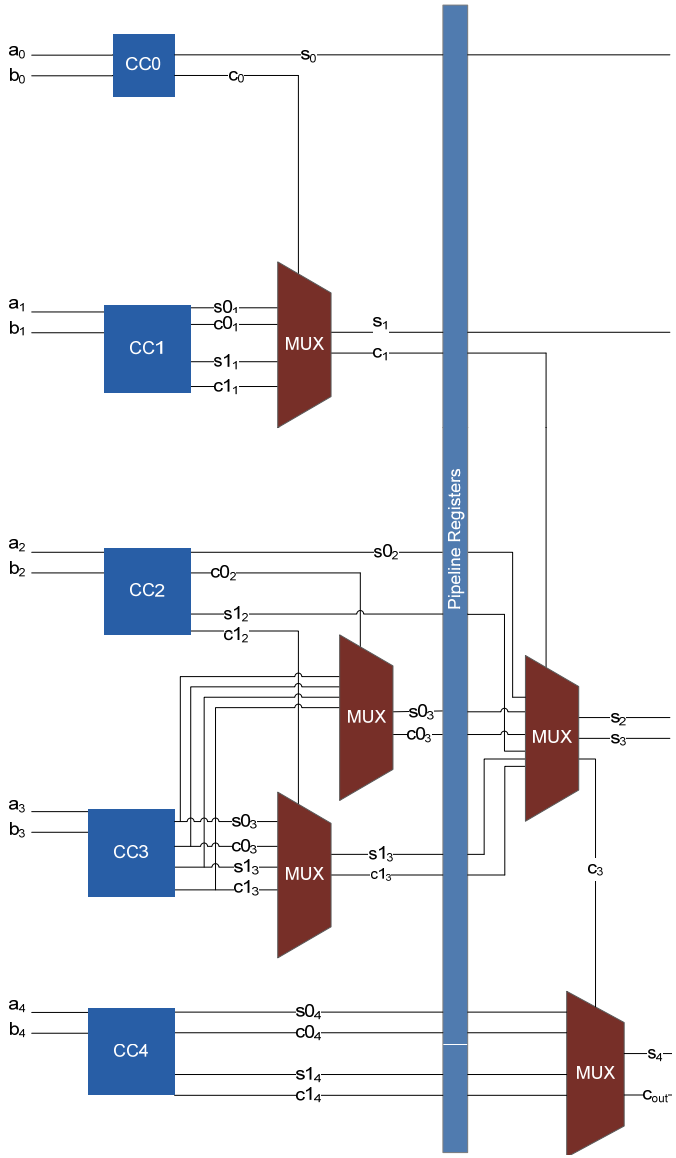
Optimized implementation of 3-coefficient direct-form FIR filter



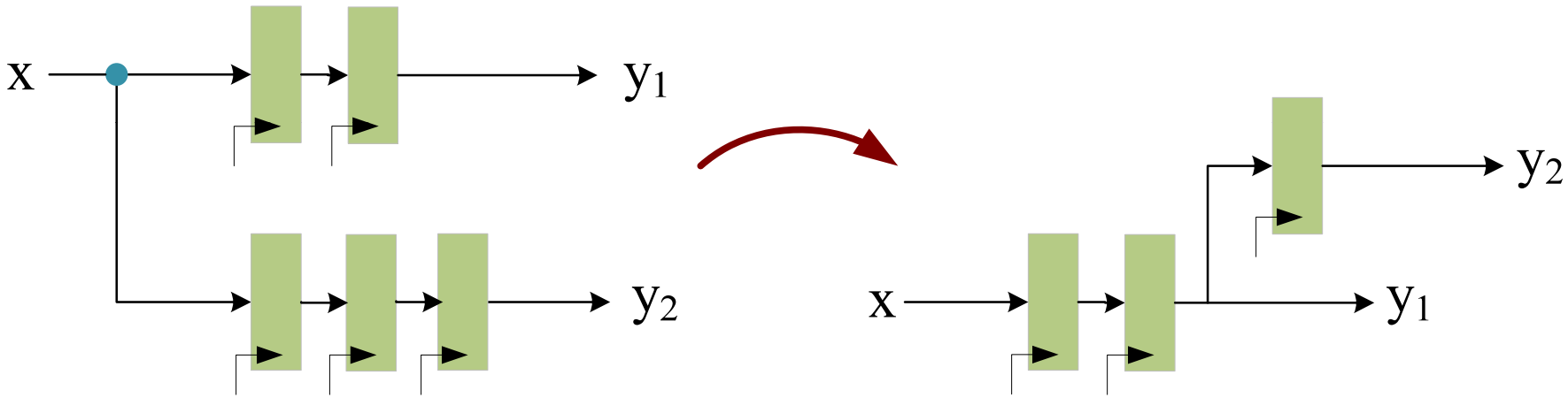
Possible locations to apply cut-set pipelining to a compression tree



Pipelining a 5-bit conditional sum adder (CSA)



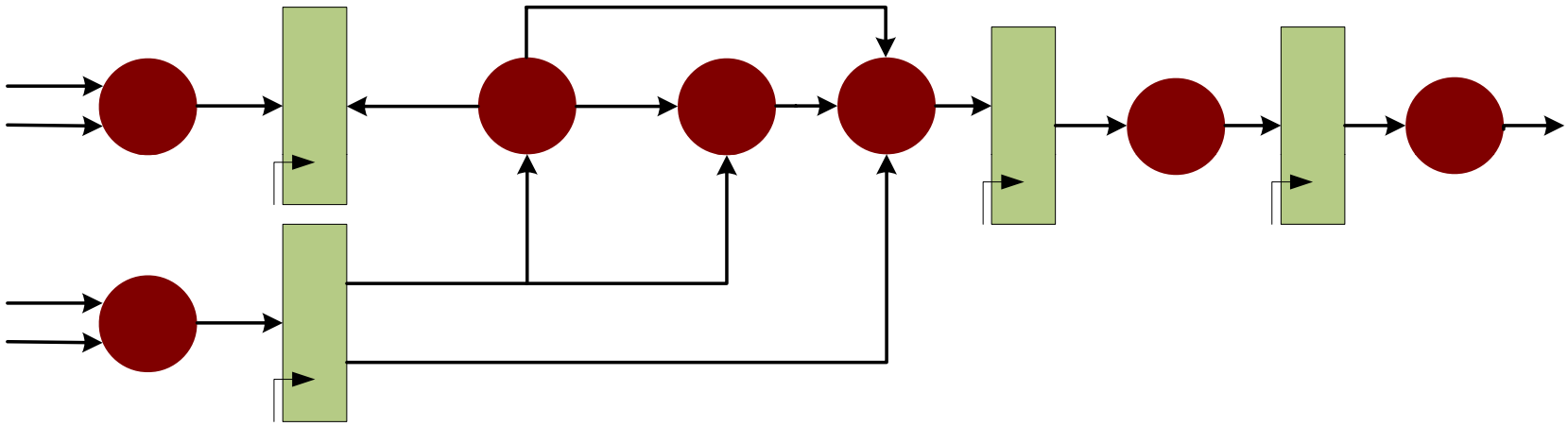
Retiming to minimize the number of registers



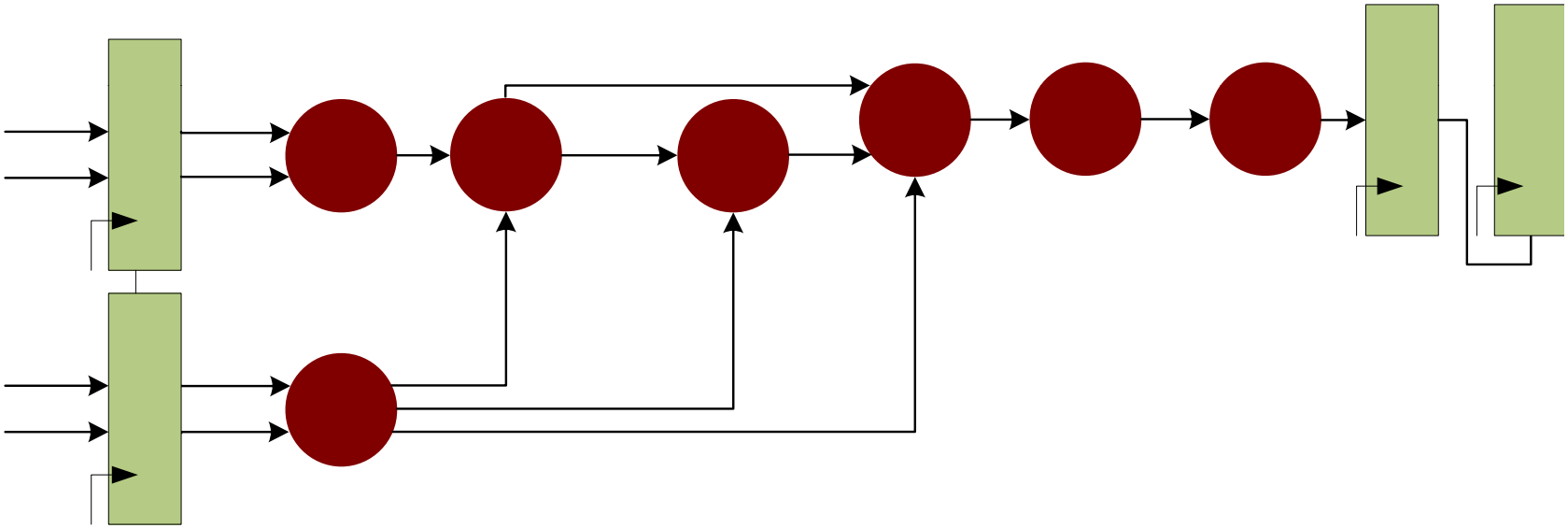
Peripheral retiming

- All the registers are moved to the periphery of the design either at the input or output of the logic
- The combinational logic is then globally optimized and the registers are retimed into the optimized logic for best timing.

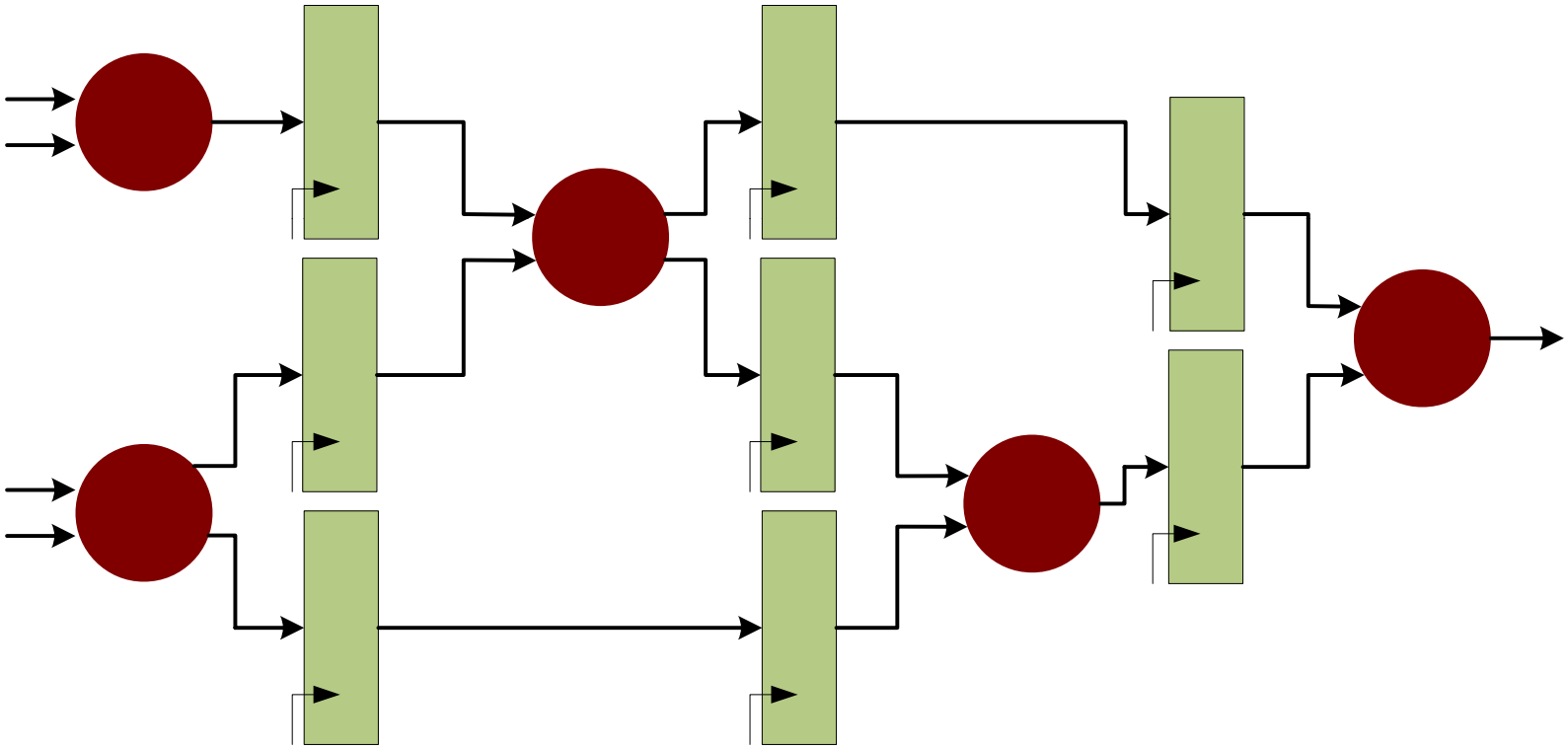
The original design.



Moving registers to the periphery

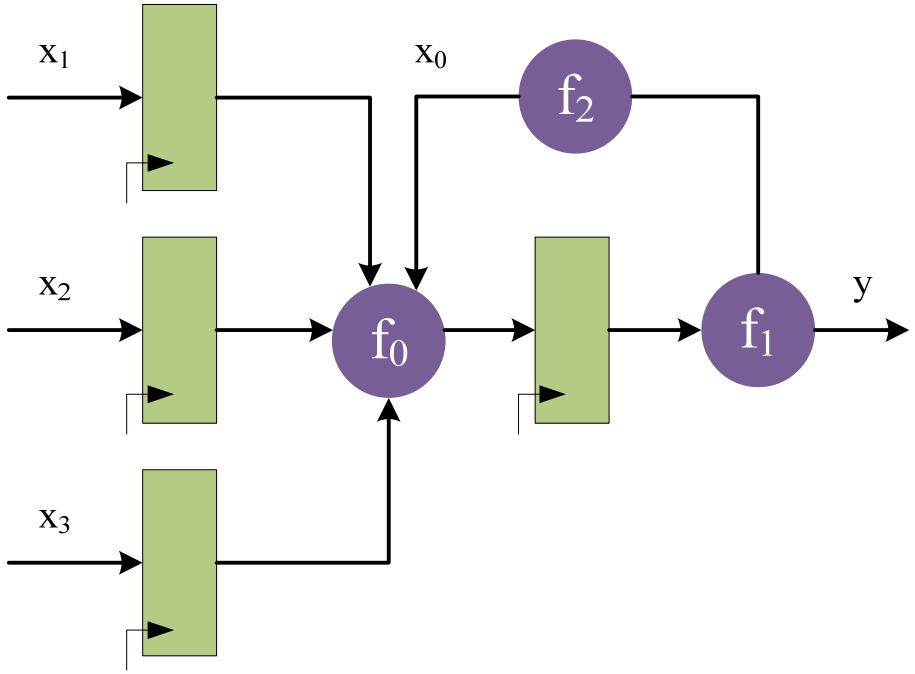


Retiming and optimization

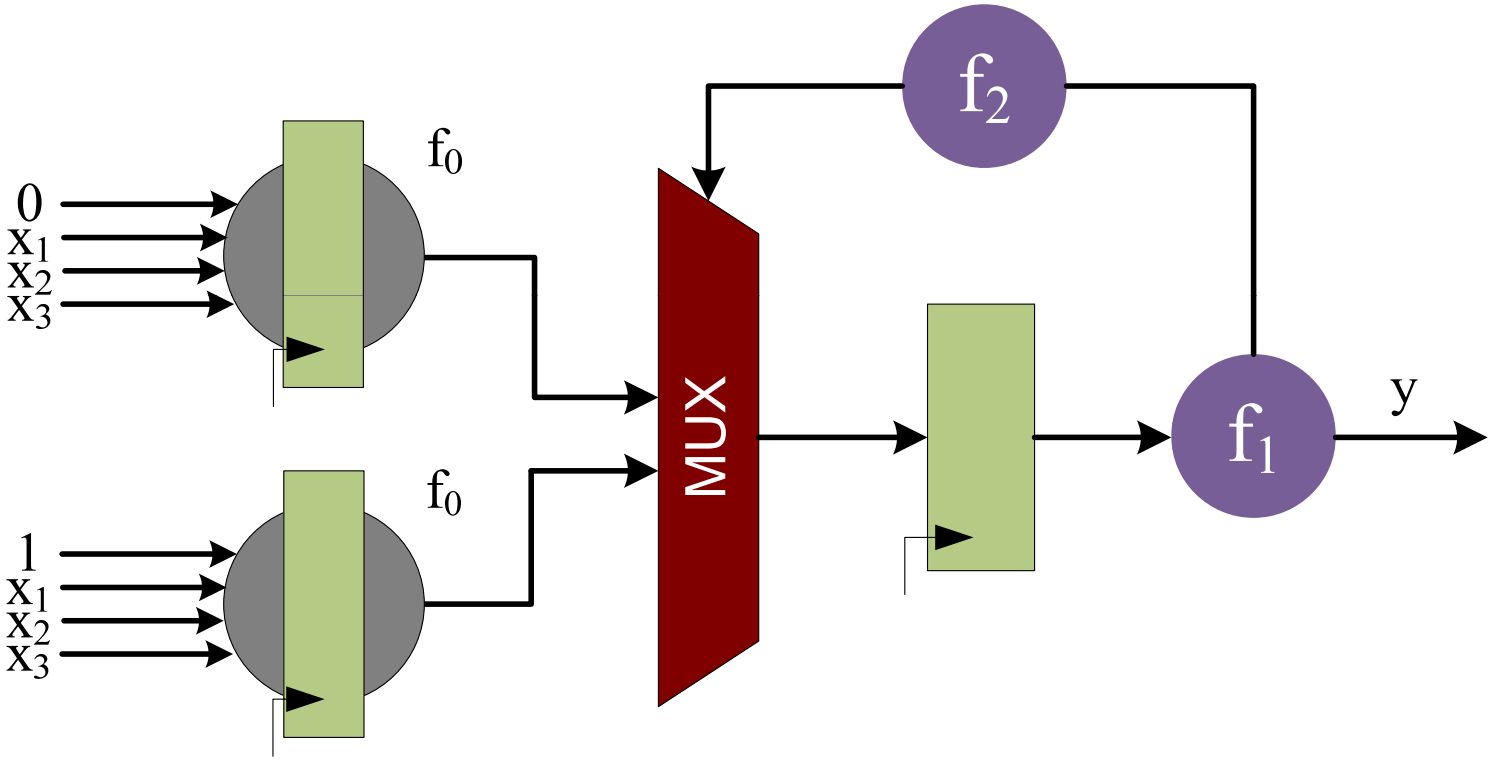


Shannon decomposition

- Shannon decomposition removing the slowest input x0 to f0 and duplicating the logic in f0 with 0 and 1 fixed input values for x0



The design is then retimed for effective timing

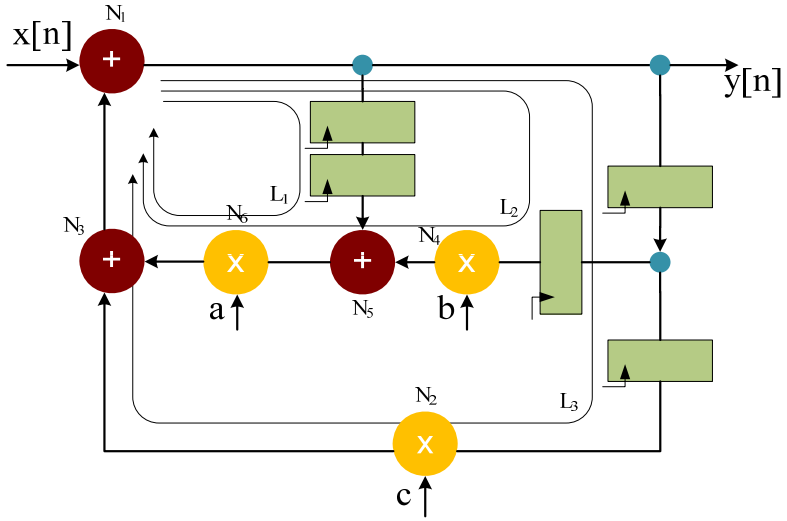


Iteration and Iteration Period

- The iteration period is the time required for execution of one iteration of the algorithm

Critical Path and Critical Path Delay

- The critical path of a DFG is defined as the path with the longest computation time delay among all the paths that contain zero registers
- The computation time delay in the path is called the critical path delay of the DFG

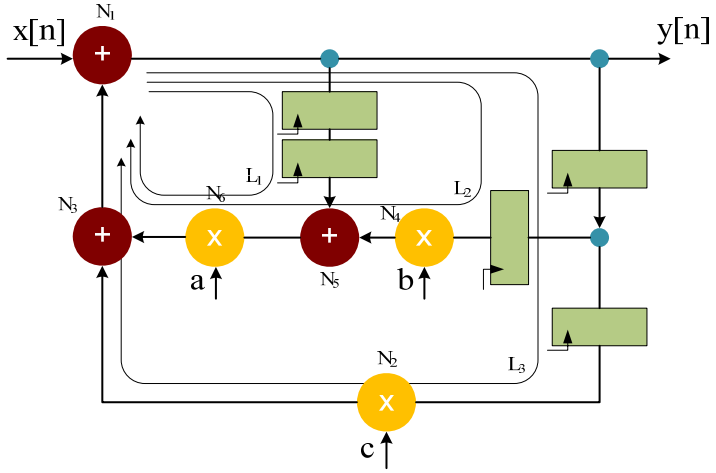


Loop and Loop Bound

- A loop is defined as a directed path that begins and ends at the same node
- The loop bound of the ith loop is defined as

  Ti/Di

  where Ti is the loop computation time and Di is the number of delays in the loop.



Critical Loop and Iteration Bound

- A critical loop of a DFG is defined as the loop with maximum loop bound. The iteration period of a critical loop is called the iteration period bound (IPB)

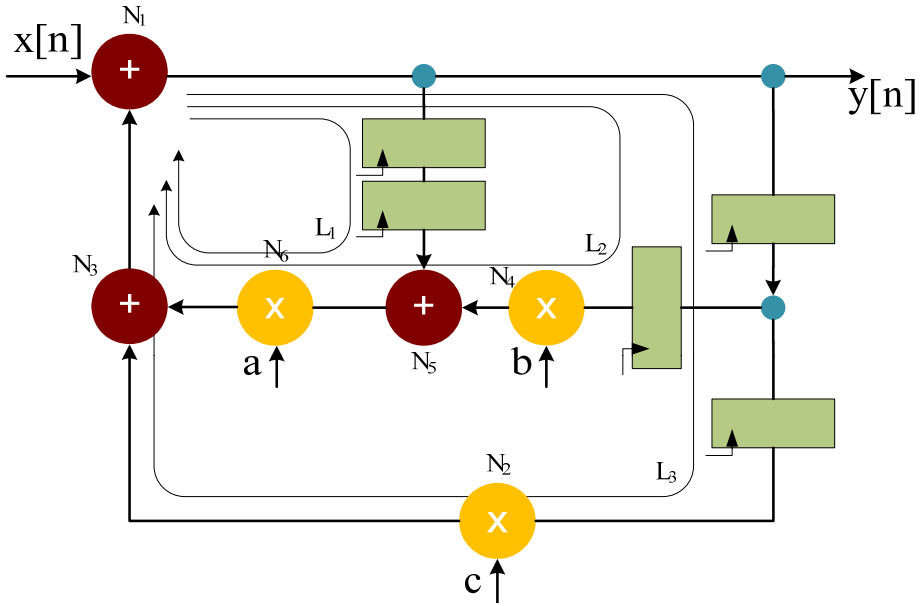
$$IPB = \max_{all\, L_i} \left\{ \frac{T_i}{D_i} \right\}$$

Dataflow graph with three loops

- Mul 2 tu
- Add 1 tu
- The critical loop L2 has maximum loop bound of 3.5 tu
- Critical path is 7 tu

$$N_4 \rightarrow N_5 \rightarrow N_6 \rightarrow N_3 \rightarrow N_1$$

- **IPB is the best achievable critical path for recursive designs**

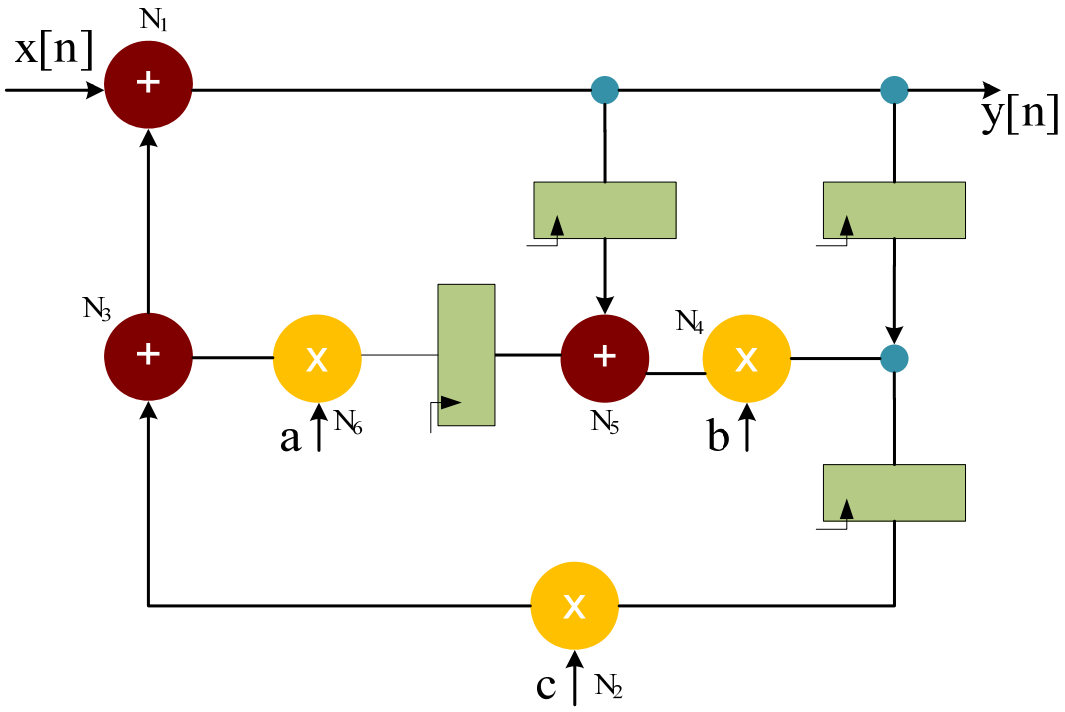


$$LB1 = \frac{T1}{D1} = \frac{(1+1+2+1)}{2} = 2.5$$

$$LB2 = \frac{T2}{D2} = \frac{(1+2+1+2+1)}{2} = 3.5$$

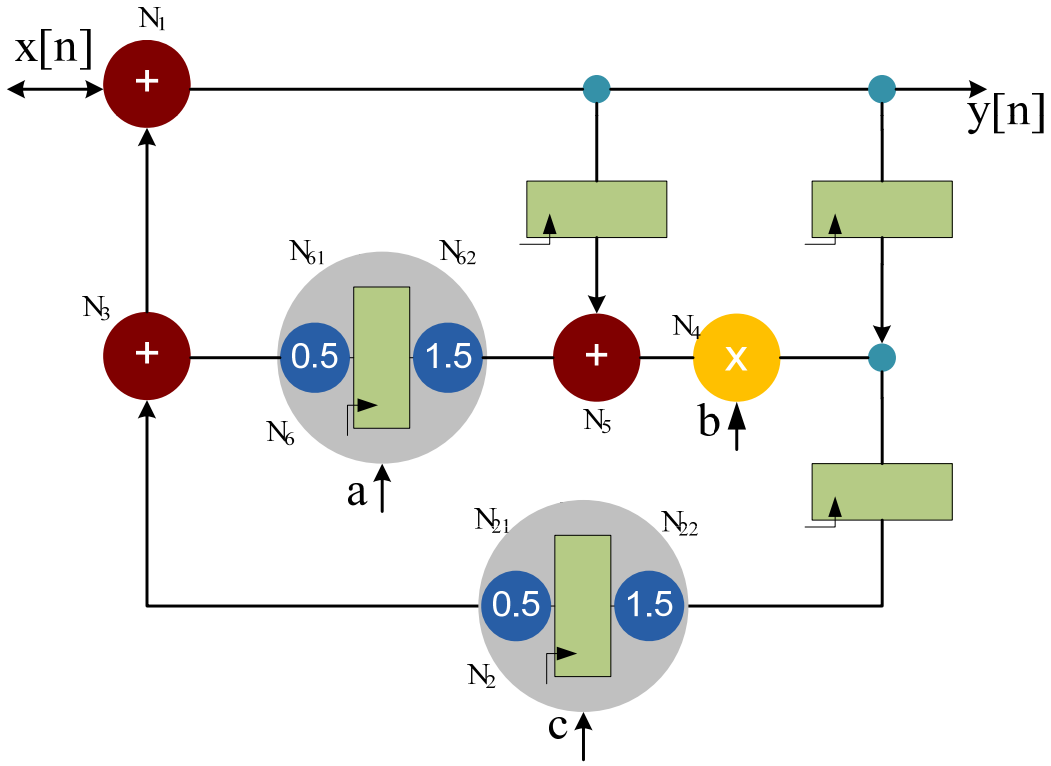
$$LB3 = \frac{T3}{D3} = \frac{(1+2+1)}{2} = 2$$

- Apply nodal transfer theorem across N5



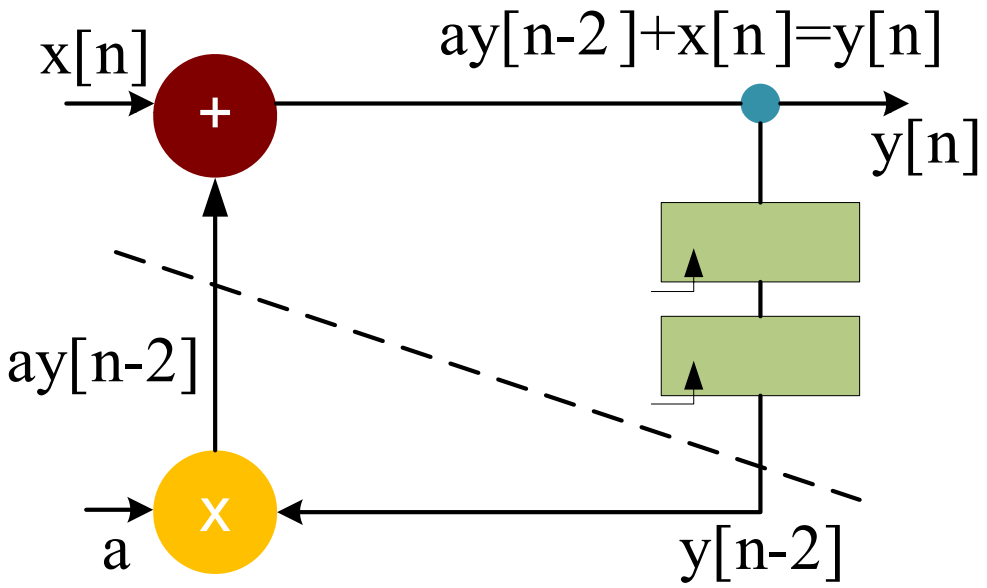
DFG with critical path delay=IPB=3.5

- Can still do better
- Requires pipelining multiplier

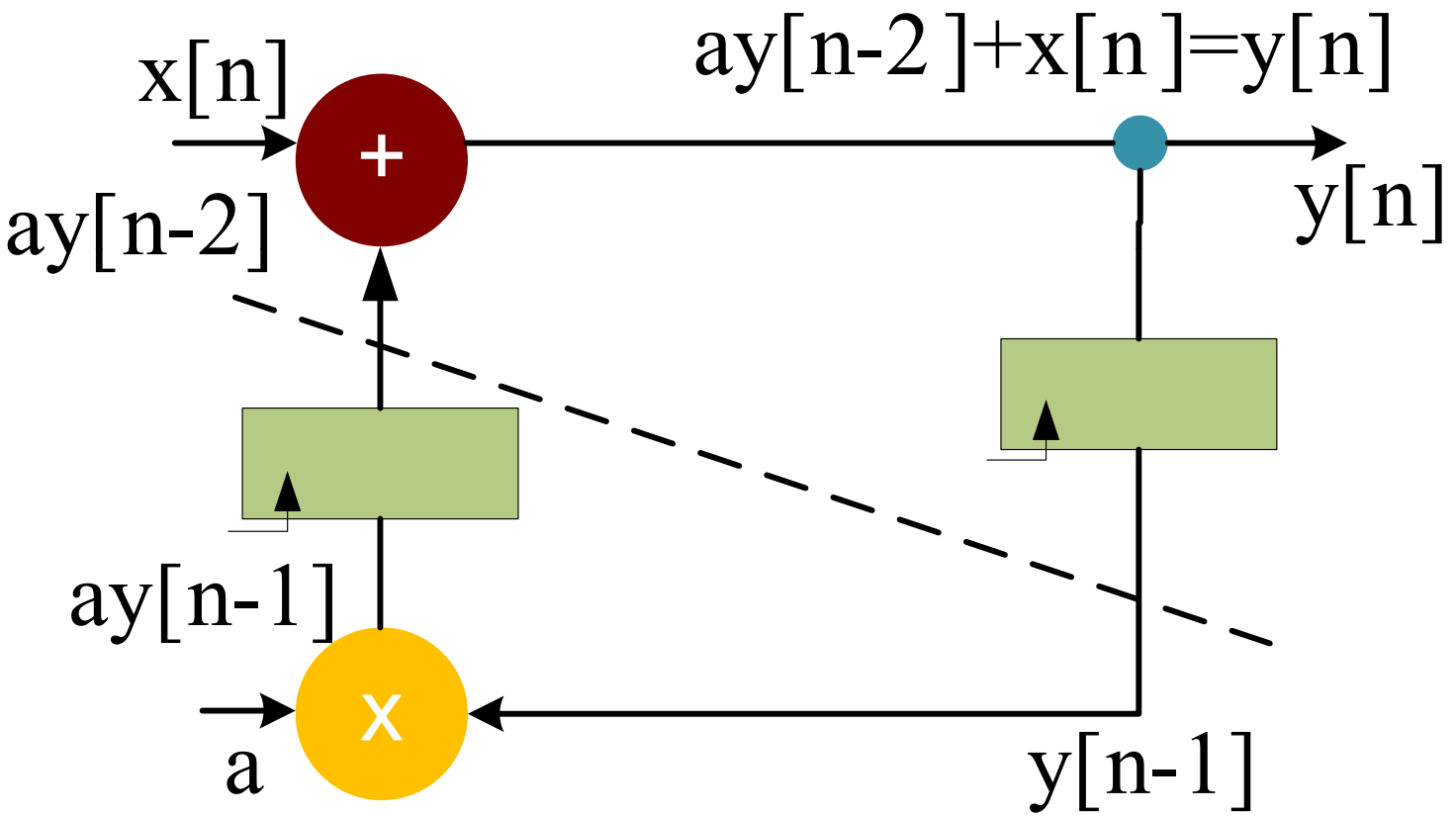


Achieving IPB

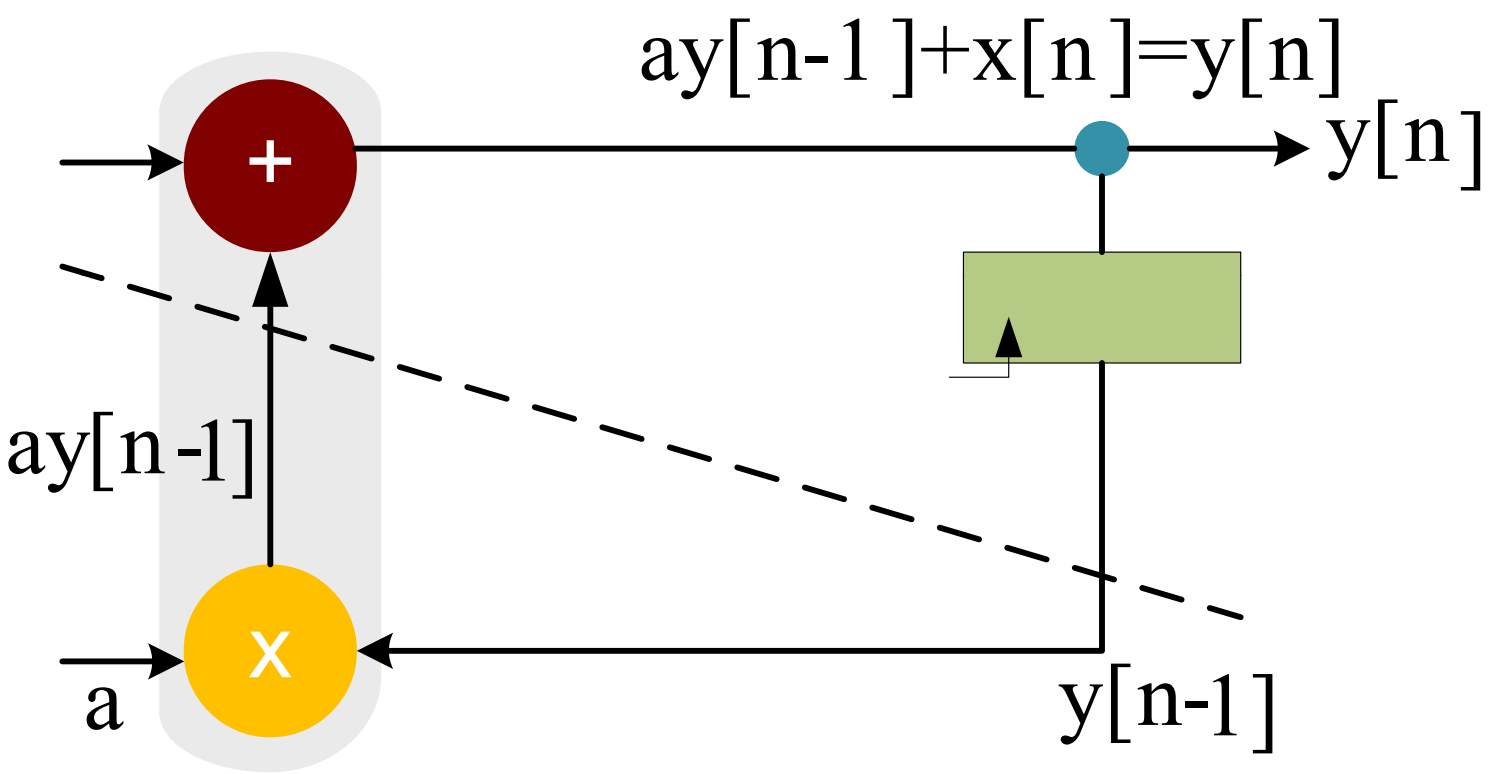
- Let Mul and Add takes 1 tu each
- IPB is 2/2 = 1 tu
- Critical path = 2 tu
- Can be retimed to get IPB
  - Cut-set retiming



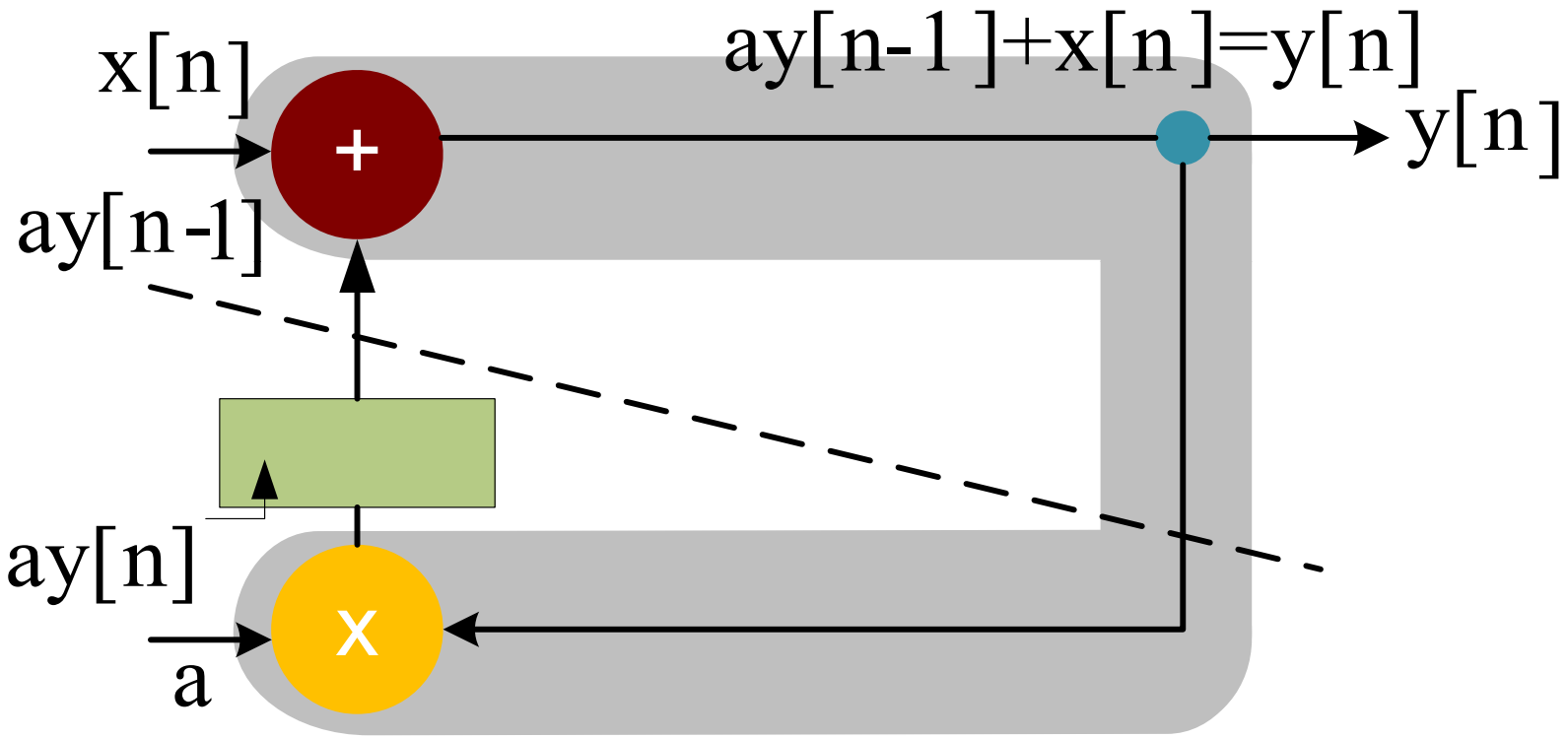
Retimed DFG using cut-set retiming



Critical path delay of a first-order IIR filter

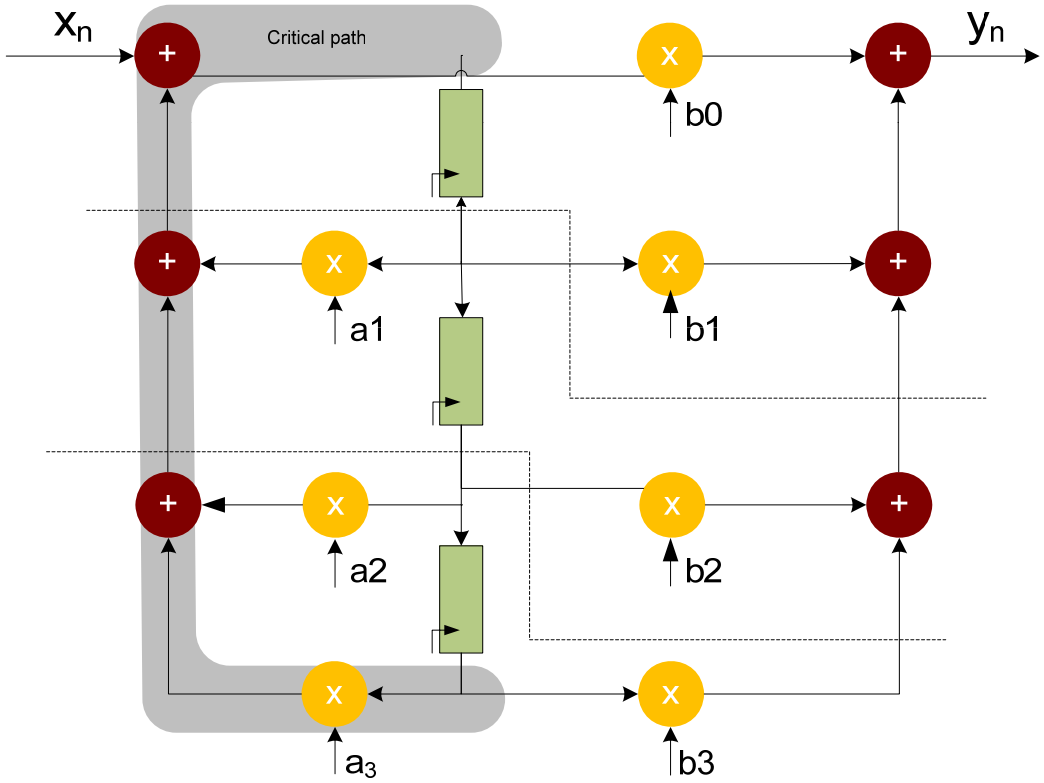


Retiming with one register in a loop will result in the same critical path

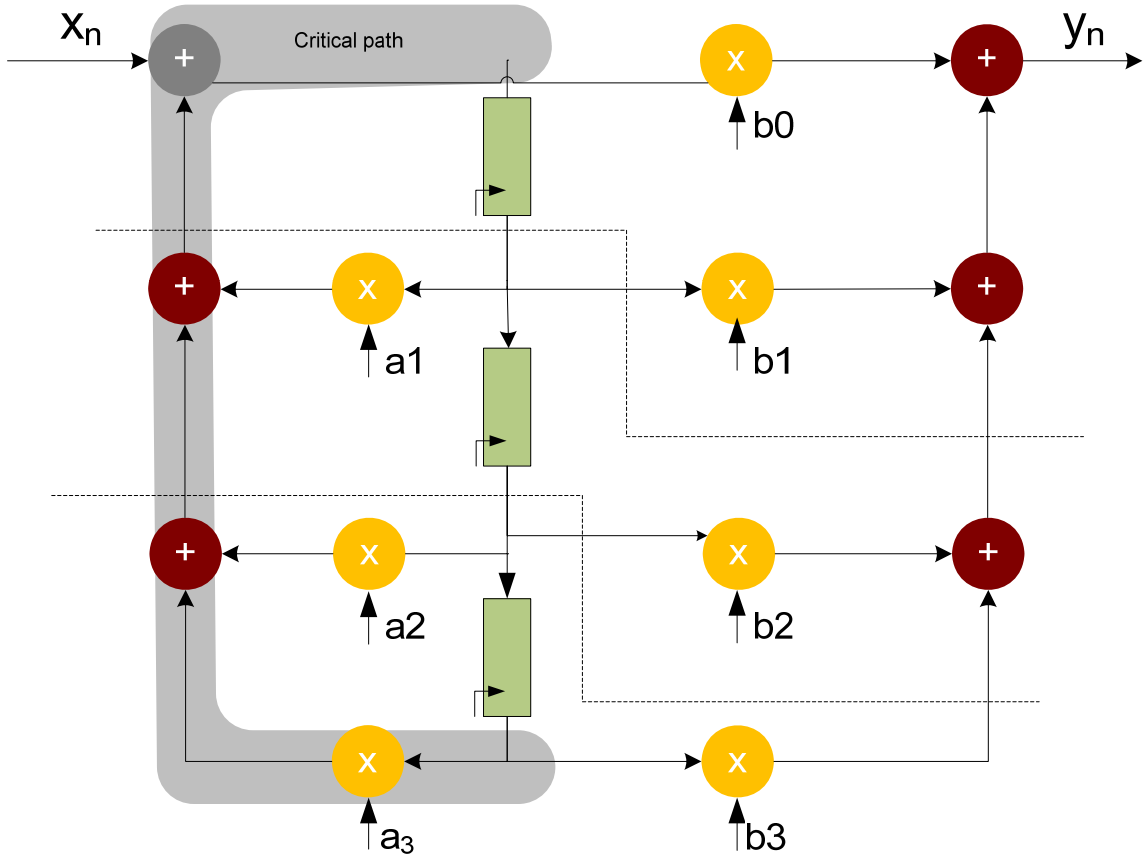


Recursive DFG

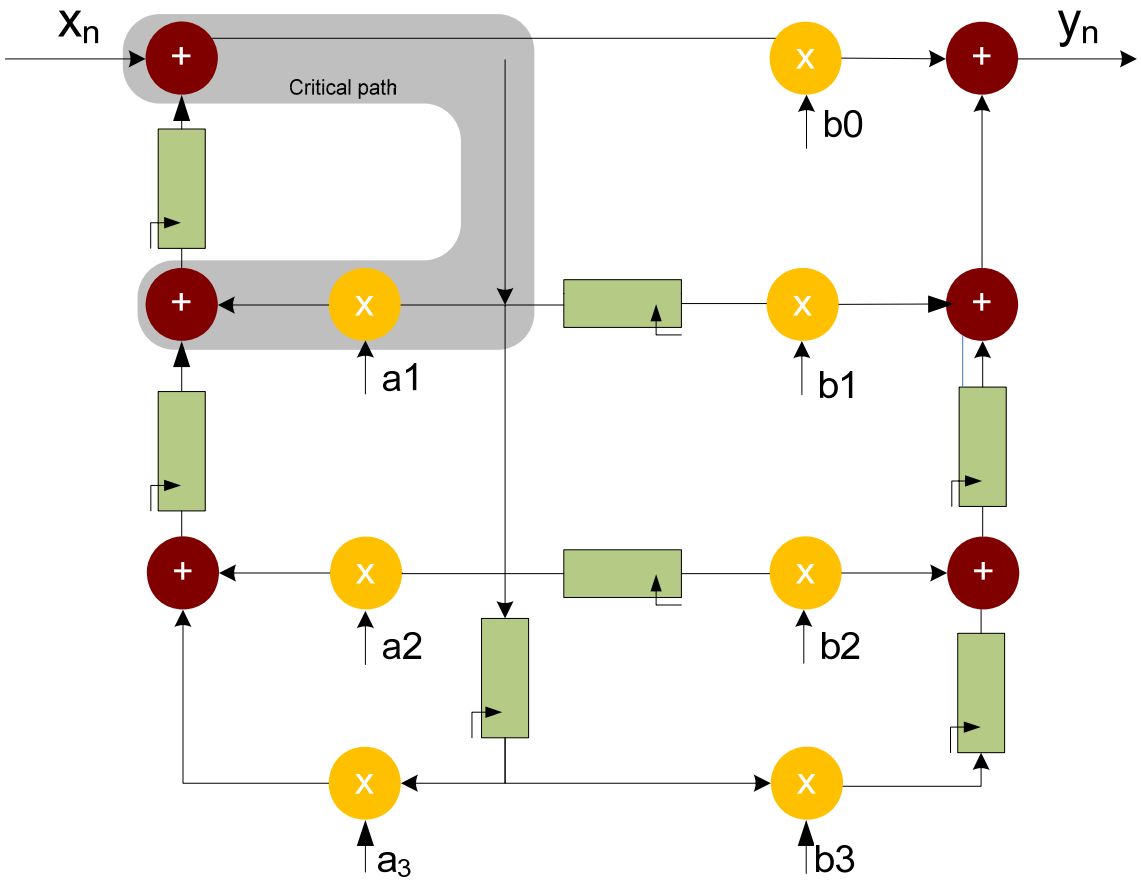
- Let Mul = 2 tu and Add = 1 tu
- IPB = 4 tu
- Critical path = 5 tu



Two cut-sets are applied on the DFG.



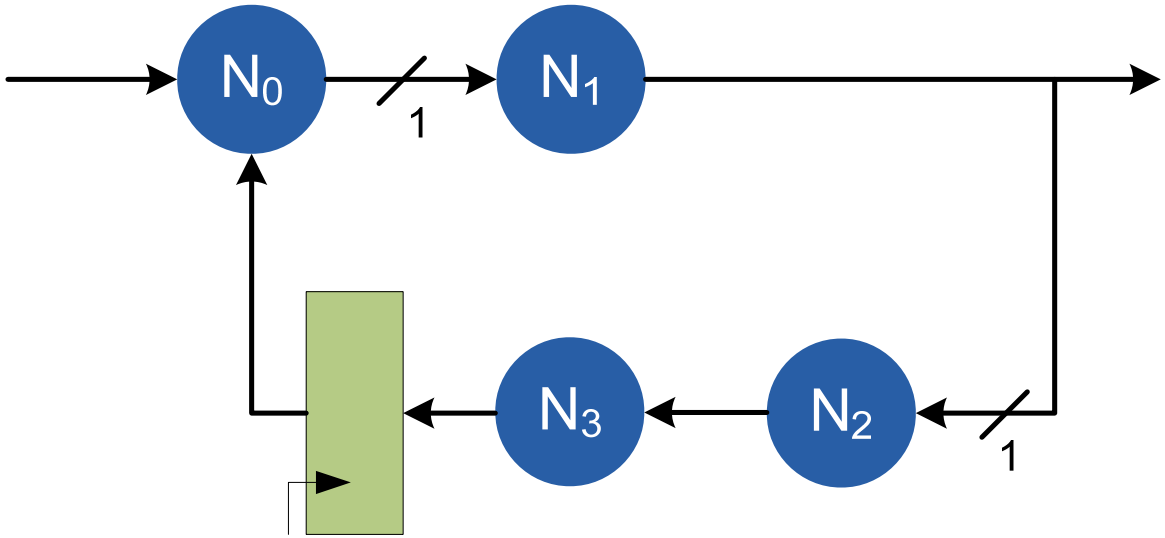
The registers are moved by applying feedback cut-set retiming



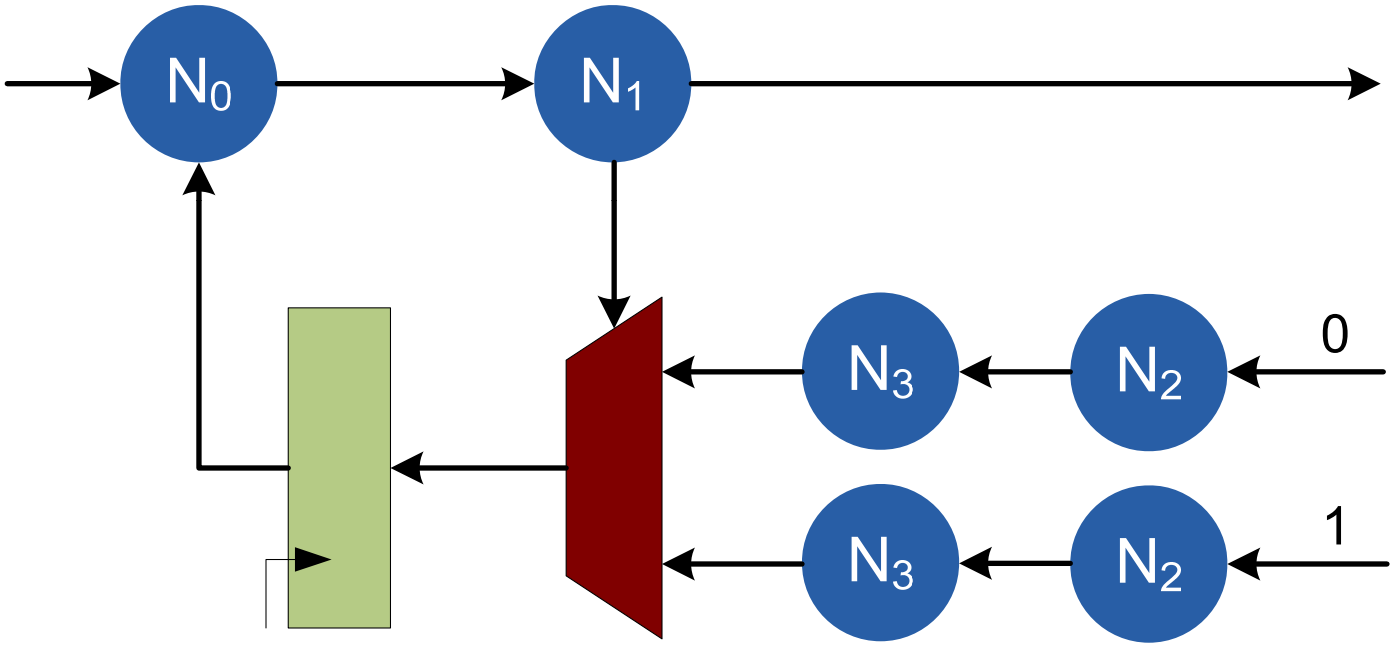
Beyond IPB

Improving beyond IPB

- Each node takes 2 tu
- IPB of 8 timing units



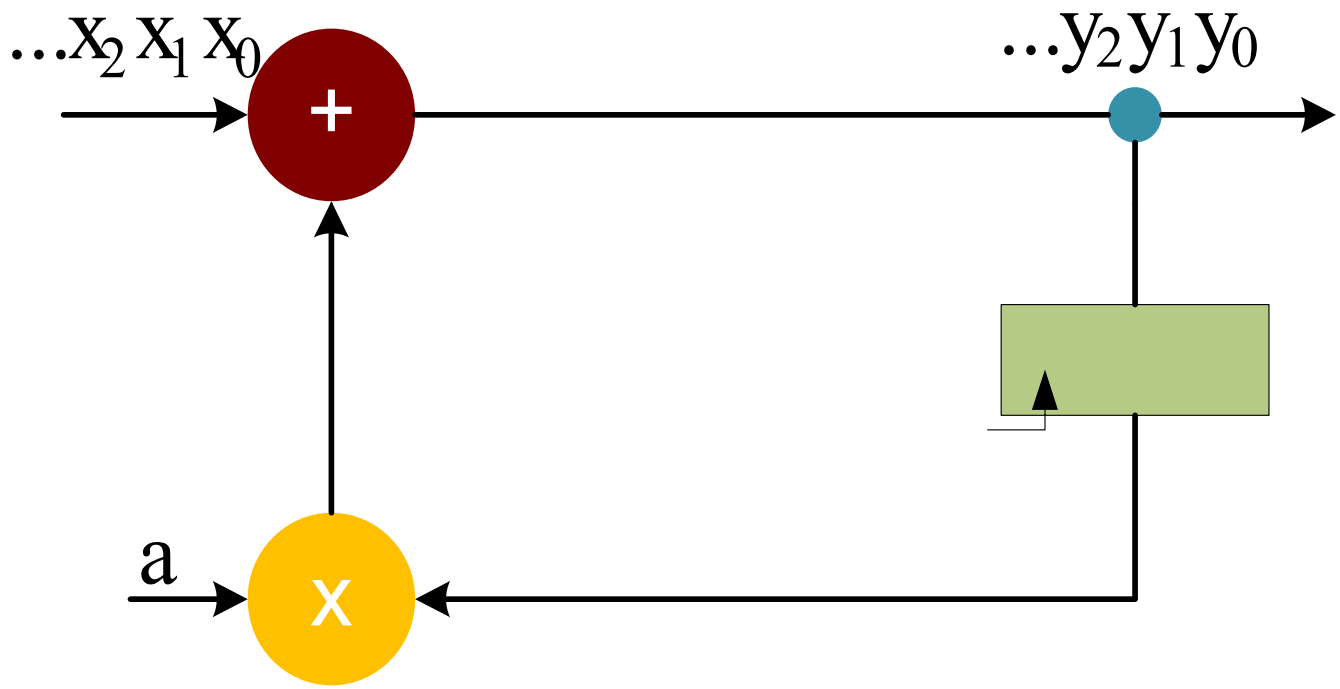
Using Shannon decomposition the IPB is reduced to around 4 timing units



C-Slow Retiming

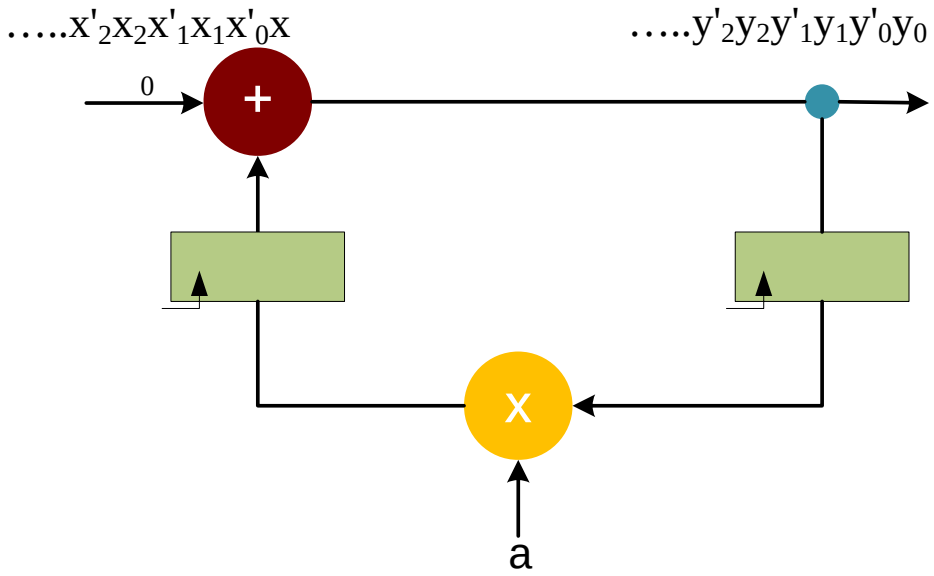
- Replaces every register in a dataflow graph with C registers.
- Retimed these registers to reduce the critical path delay.
- The resultant design can operate on C distinct streams of data

C-slow Retiming

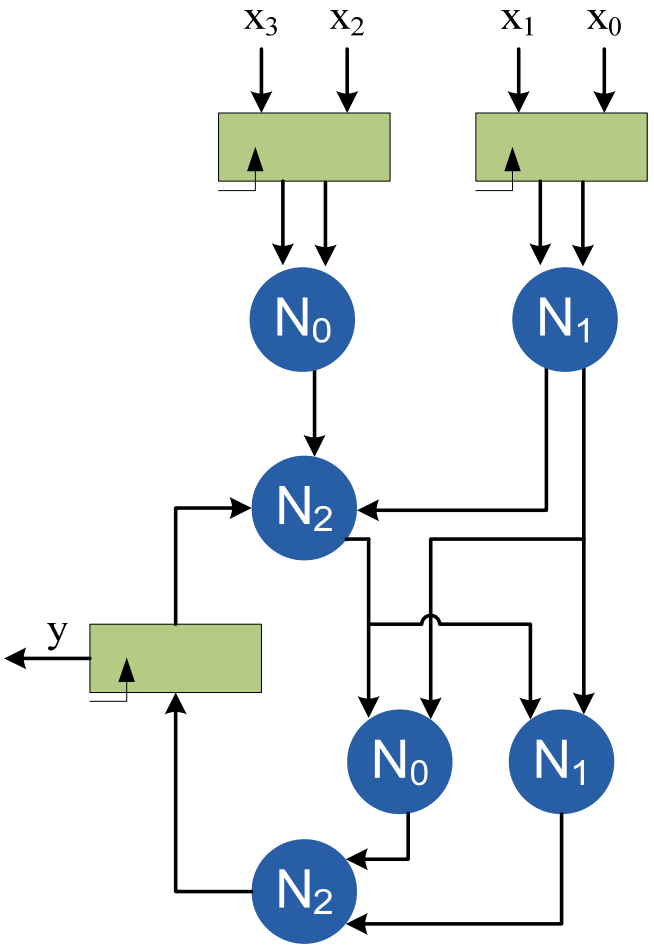


Retiming

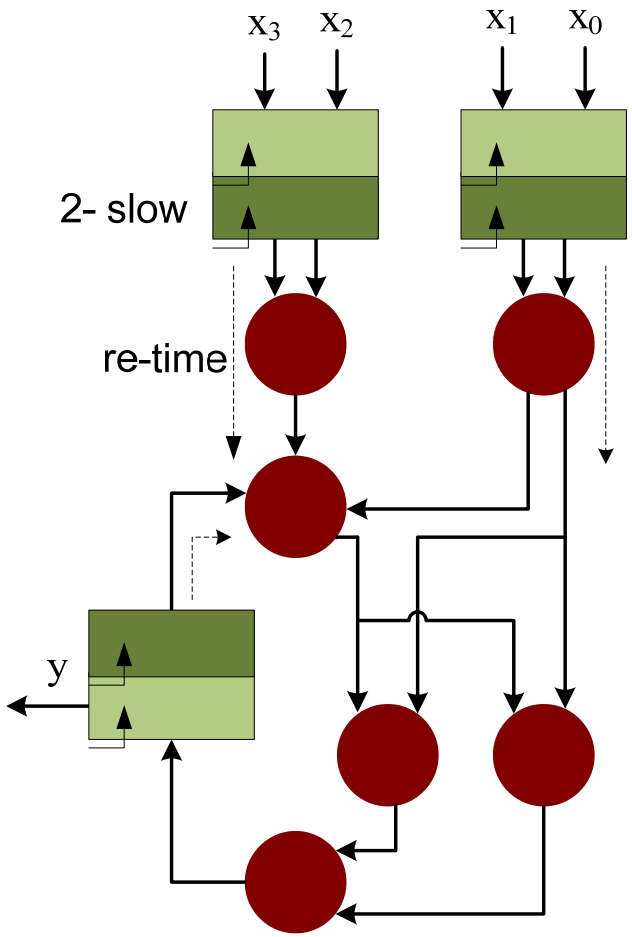
- 2-Slow design working on two streams of data
- 2 Registers are retimed for better timing



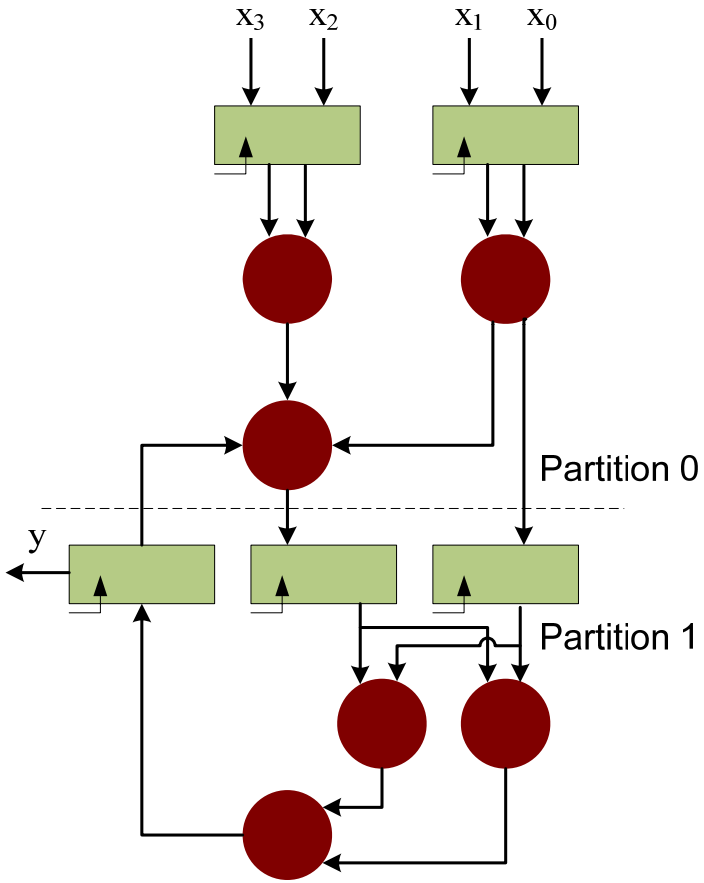
Original DFG with multiple nodes and few registers and large critical path.



2-Slow Design



The DFG partitioned into two equal sets of logic where all the three nodes can be reused in a time-multiplexed design



C-slow for an Instruction Set Processor

- Any processor can be C-slowed and then can run C parallel threads

Lookahead

- Lookahead actually reduces the iteration period bound
  - Pipelining and C-slow do not reduce the iteration period bound
- Design Flow
  - Step 1: Check if sample period is satisfied by IPB
  - Step 2: Lookahead can be used to reduce IPB
  - Step 3: Retiming can then be used to reduce the iteration period to that of the IPB

What is Lookahead?

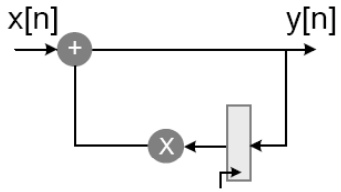
Current iteration : $y[n] = ay[n-1] + x[n]$

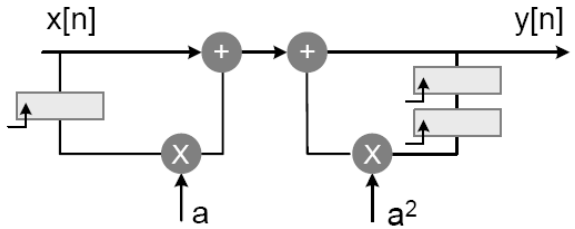
Previous iteration: $y[n-1] = ay[n-2] + x[n-1]$

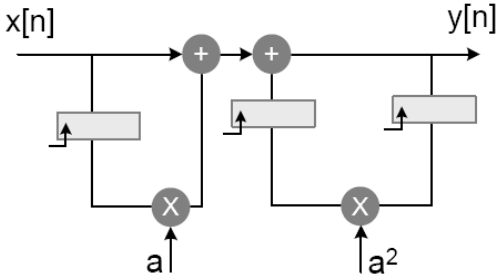
Rewriting the current iteration

$$y[n] = a^2 y[n-2] + ax[n-1] + x[n]$$

Look-ahead Transformation for IIR filters







Lookahead Factor

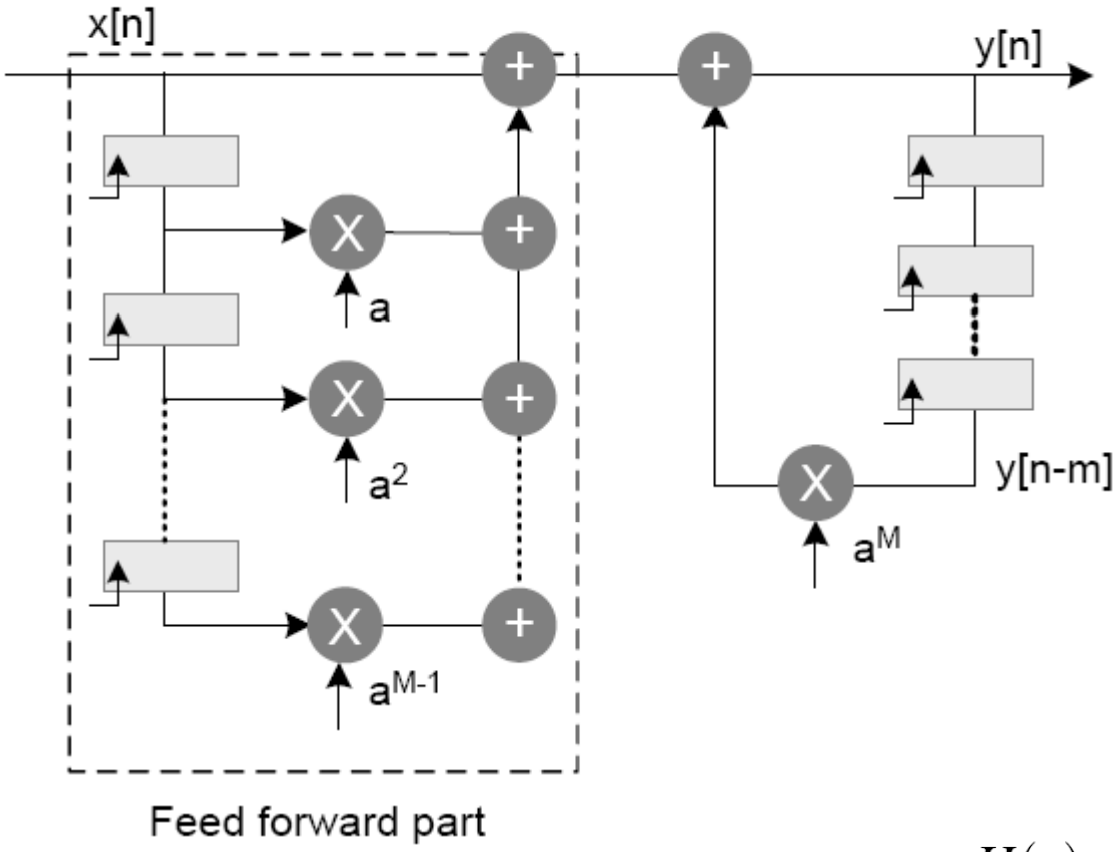
- By an order of 3

$$y[n] = a^3 y[n-3] + a^2 x[n-2] + a x[n-1] + x[n]$$

- By an order of M

$$y[n] = a^M y[n-M] + \sum_{i\ 0}^{M\ 1} a^i x[n-i]$$

Lookahead Transformation



$$H(z) = \frac{1}{1 - az^{-1}} = \frac{\sum_{i=0}^{M-1} a^i z^{-i}}{1 - (az^{-1})^M}$$