University of California, Los Angeles
Henry Samueli School of Engineering and Applied Science
Department of Electrical Engineering

ECE216B | D. Marković

ECE216B: SAMPLE FINAL PROBLEMS

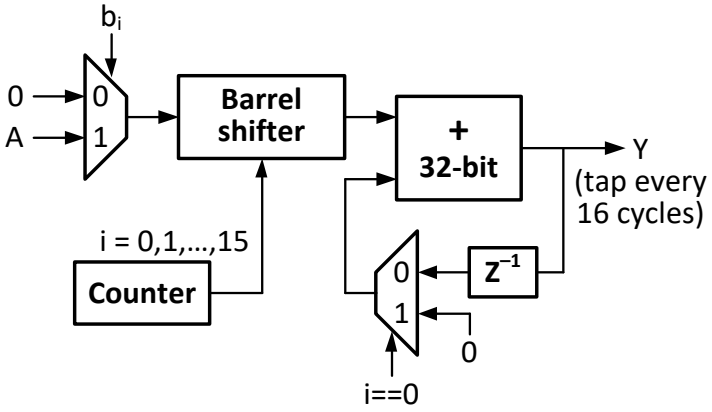
SOLUTION

PROBLEM 1: Multiplier Design (15 pts)

Consider the multiplier shown below. The multiplier accepts two 16-bit unsigned inputs A and B and computes the output Y = A×B.

a) Realize Y using a single 32-bit adder and any other logic blocks you may need (these additional blocks should not be full/half adders or multipliers). You may use more than one clock cycle to complete the computation. How many cycles (latency) do you need to produce the output Y? Draw the architecture, clearly indicating the inputs and the outputs. What is the throughput of the multiplier? (5 pts)

$B = b_{15:0}$

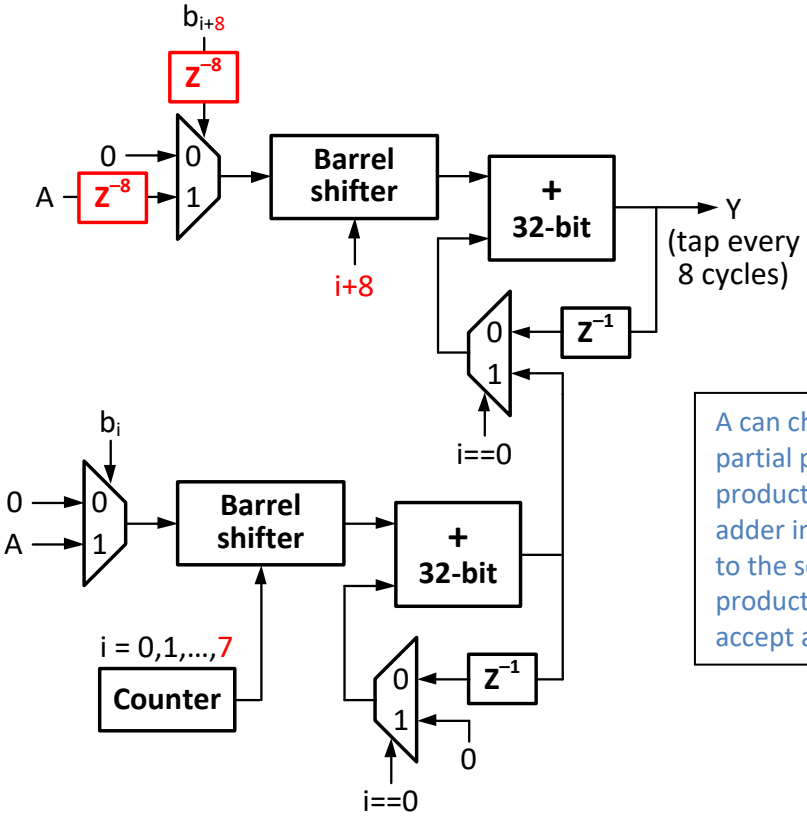


Throughput = $(t_{add} + t_{mux} + t_{shift})/16$

The inputs A and B do not change for 16 cycles. New input can be accepted only after Y has been tapped after 16 cycles. Latency = 16.

Latency: **16**	***Throughput:*** **$(t_{add} + t_{mux} + t_{shift})/16$**

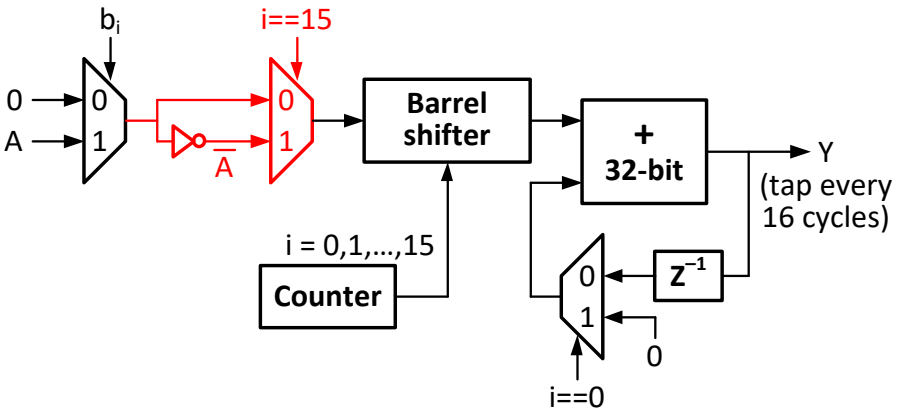
b) Suppose now you have two 32-bit adders to realize the multiplier. How would you improve the throughput obtained in part (a)? Draw the architecture, clearly indicating the inputs and the outputs. What is the throughput of the new design? (5 pts)



A can change every 8 cycles. The addition of partial products is split in two phases. Partial products for b_0 to b_7 are added in the first adder in the first 8 cycles. The result is sent to the second adder that adds the partial products for b_8 to b_{15}. The first adder can accept a new input in the 9th cycle.

Throughput: $(t_{add} + t_{mux} + t_{shift})/8$

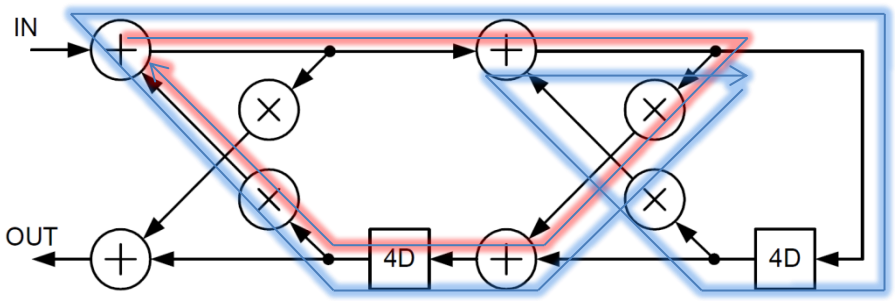
c) Suppose the inputs A and B are signed 2's complement. Modify the architecture in part (a) so as to support signed multiplication. (5 pts)



If MSB b_{15} = 1, then invert A and add. The inputs A and B do not change for 16 cycles.

PROBLEM 2: Filter Design with Retiming and Pipelining (20 pts)

Consider the IIR filter shown below. Assume that addition and multiplication require 10 and 20 ns, respectively.



a) Calculate the iteration bound of this filter. (3 pts)

Red Loop:
Delay = 3 adder + 2 multiplier = 3×10 + 2×20 = 70 ns
Num. of register = 4
Loop bound = 70 / 4 = 17.5 ns
Blue Loop:
Delay = 4 adder + 3 multiplier = 4×10 + 3×20 = 100 ns
Num. of register = 8
Loop bound = 100 / 8 = 12.5 ns
Iteration bound = max(12.5, 17.5) = 17.5 ns

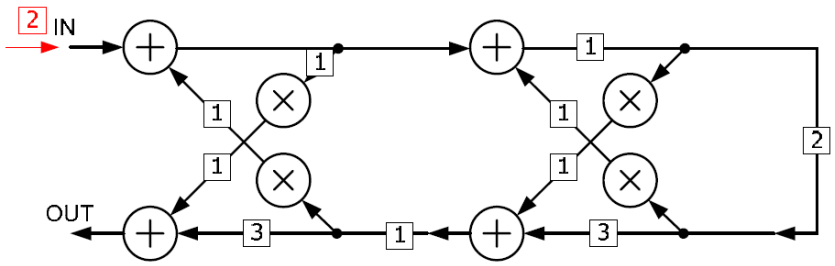
***Iteration bound:* 17.5 ns**

b) Calculate the critical-path delay of the circuit. (3 pts)

Critical path = 3 adder + 2 multiplier = 70 ns

***Critical-path delay:* 70 ns**

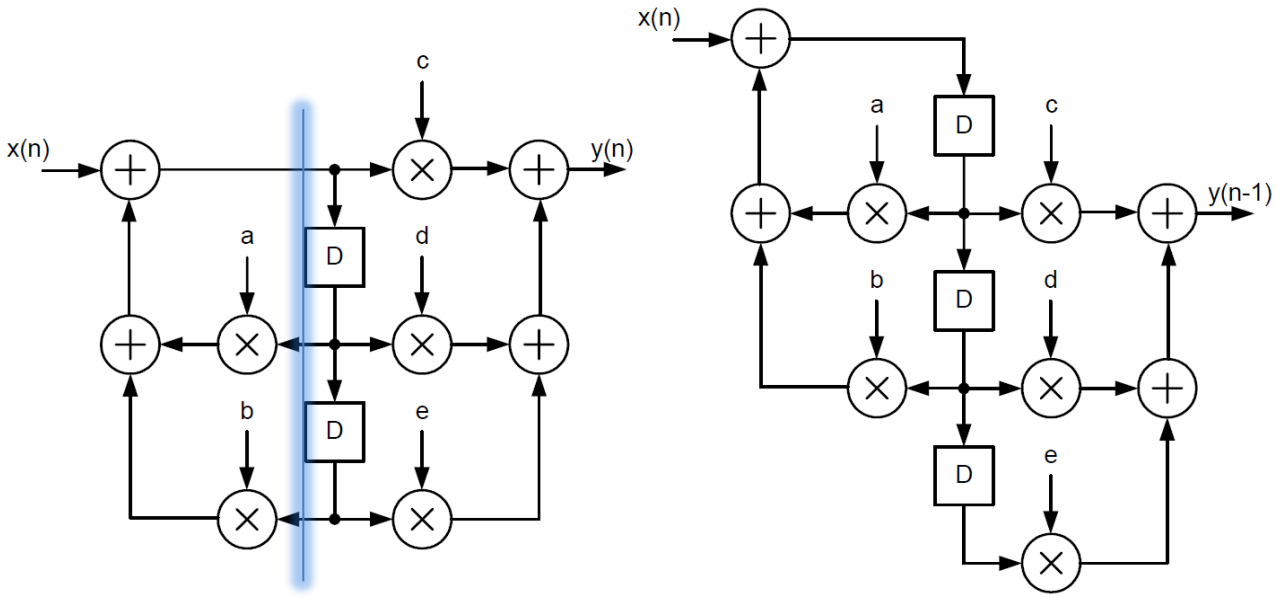
c) Pipeline and/or retime this system to achieve a critical path of 20 ns. (3 pts)



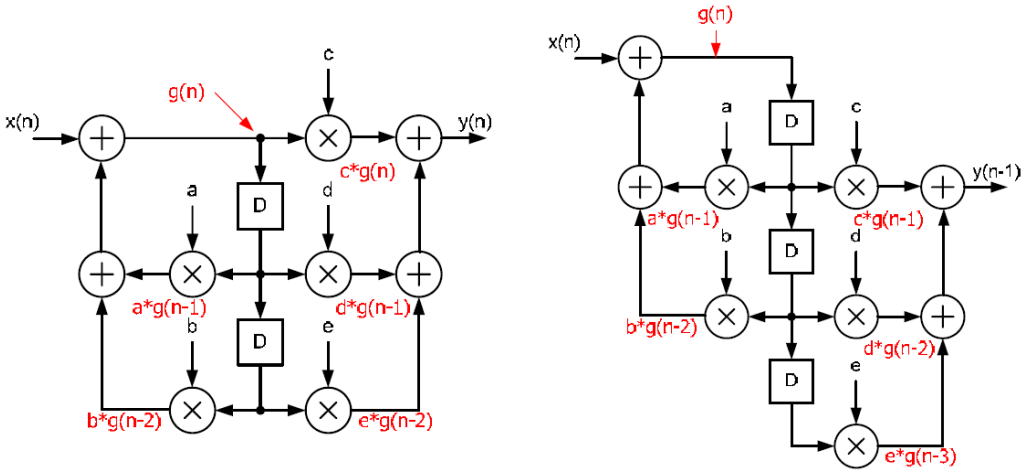
d) Pipeline and/or retime this system to achieve a critical path of 10 ns, where fine-grain retiming is allowed. If you find it impossible, please explain. (3 pts)

Since the iteration bound (= 17.5 ns) is greater than 10 ns, it is not possible to use pipelining or retiming to achieve a critical path of 10 ns.

e) Consider the two IIR filter DFGs shown below.



Prove the two circuits have equivalent functionality (circuit on the right has one-cycle latency). (4 pts)



Left: $g(n) = x(n) + a*g(n-1) + b*g(n-2)$ $y(n-1) = c*g(n-1) + d*g(n-1) + e*g(n-2)$

Right: $g(n) = x(n) + a*g(n-1) + b*g(n-2)$ $y(n-1) = c*g(n-1) + d*g(n-2) + e*g(n-3)$

Therefore they are equivalent, except the one-cycle latency.

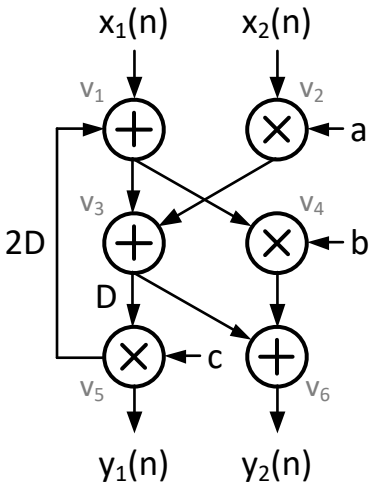
Transform the left circuit to the right circuit by pipelining and/or retiming. (4 pts)

1st step: Pipeline cut set shown above.

2nd step: Merge the registers to get the final result.

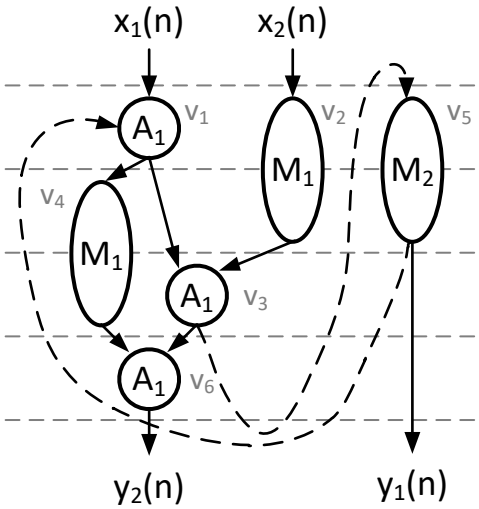
PROBLEM 3: Scheduling and Retiming (20 pts)

Consider the signal flow-graph shown below.



a) Schedule this graph using a maximum of 2 adders and 2 multipliers as available resource units. Assume that the multipliers are 2-stage pipelined and the adders are single stage pipelined in your scheduling. Clearly draw the scheduled graph indicating the time steps and the hardware allocation for the operations $v_1 – v_6$. How many cycles (N) do you need to complete the schedule? Use as many rows in the table below as necessary.

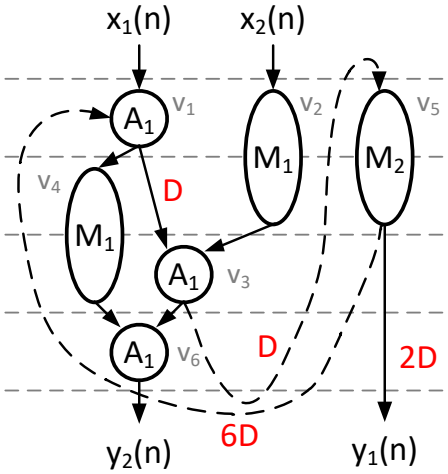
Cycle #	M_1		M_2	A_1	A_2
1	v_2		v_5	v_1	
2		v_4 (stg1)		X	
3	v_4 (stg2)		X	v_3	
4	X		X	v_6	



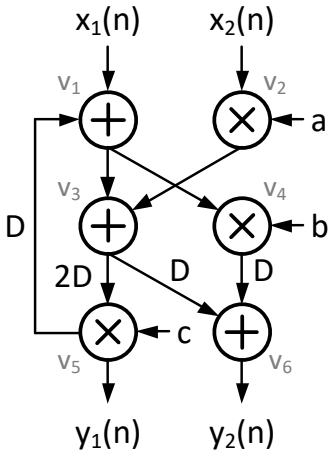
Schedule requires two multipliers and one adder. Two pipeline stages in the multipliers can be used independently in the same cycle (M_1 in cycle 2).

***N* = 4**

b) Compute the number of delays/registers on each graph edge for the schedule in part (a).

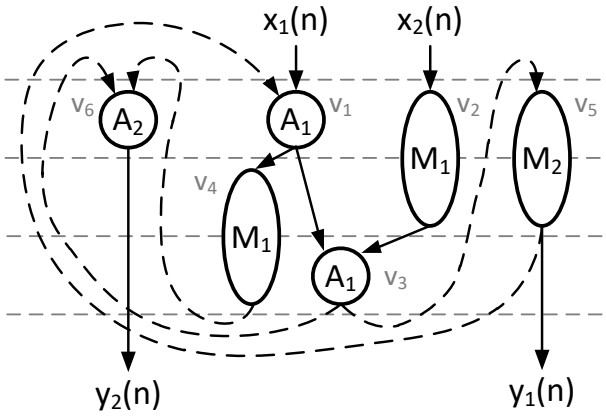


c) Pipeline the original flow graph by inserting one register at both inputs. Now retime the pipelined graph so as to ensure minimum delay.



This is one retimed implementation. Other solutions are also possible.

d) Schedule the retimed flow-graph from part (c) in N = 3 cycles. Assume that multipliers are 2-stage pipelined and the adders are single stage pipelined in your scheduling. Clearly draw the scheduled graph indicating the time steps and the hardware allocation for the operations. How many adders and multipliers do you need to realize this schedule?



Solution will depend on the retimed version in part (c). For the retimed version shown in (c), the scheduled graph is shown here. We need two multipliers and two adders.

PROBLEM 4: General Knowledge (10 pts)

Pipelining a recursive loop is possible by inserting additional registers. True (False) (1)

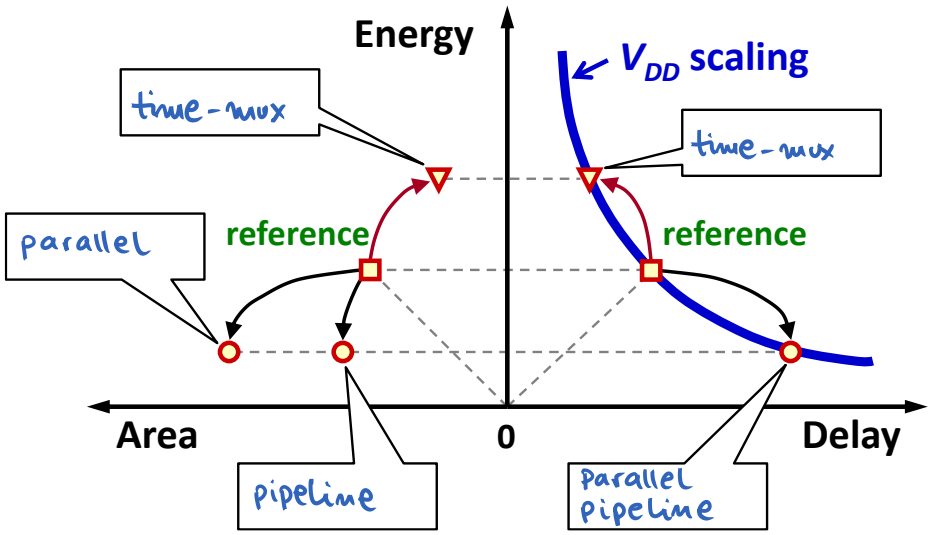
Iteration bound limits the maximum achievable throughput. (True) False (1)

Energy consumption can be perfectly reduced by voltage scaling. True (False) (1)

More parallelism is always better because it improves energy for the same throughput.
True (False) (1)

Pipelining can increase the throughput because of reduced logic depth,
or can reduce the energy consumption by voltage scaling. (1)

Complete the architectural transformation plot below by filling in the related circuit techniques.



(5)