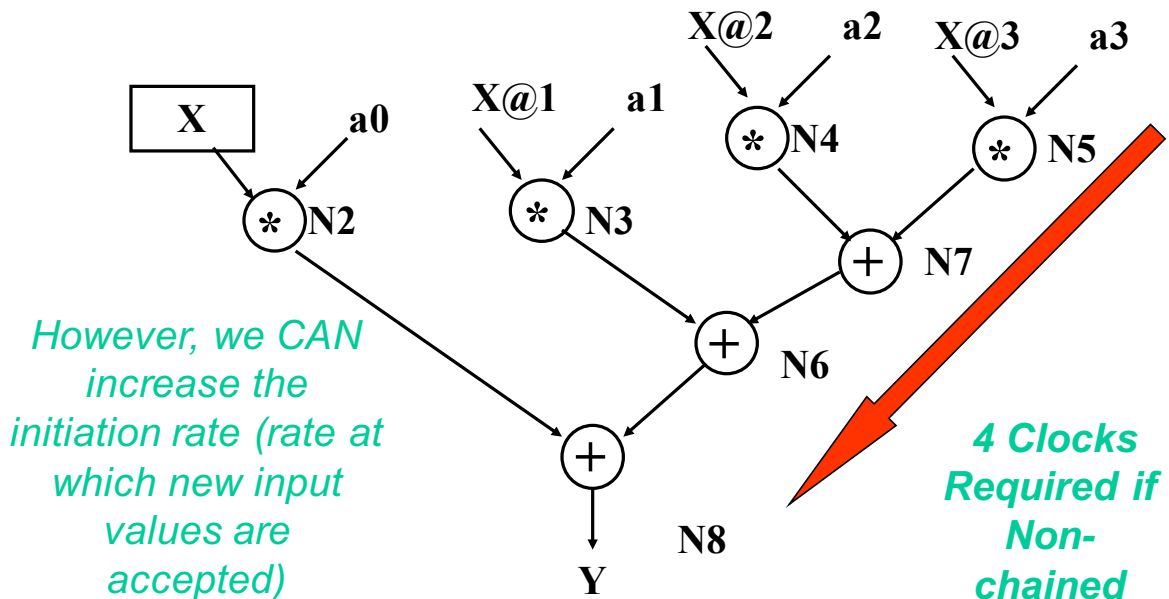


# Datapath Design Approaches

- Previous Topic, estimate minimum number of resources and then try to schedule them
  - May Require Iteration, can Give minimum resource datapath
- Another approach is to build a schedule first (based on the specifications), then determine required resources
  - Can meet latency specification immediately, but may exceed area limitations

## Increasing the Initiation Rate

The flowgraph below has a longest path of 4 clocks. This means the computations cannot be completed in less than 4 clocks (min. latency) because of data dependencies.



# Latency = 4

## Make No Assumptions About Resources Yet

Lets look at the operations needed with latency=4 for several clock cycles.

Successive Sample values are labeled A,B,C etc.

Clk	Operations		
	Sample A	Sample B	Sample C
1	N4(*) N5(*), Input x		
2	N2(*),N3(*),N7(+)		
3	N6 (+)		
4	N8(+)		
5			
6			
7			
8			

# Initiation Period = 2

## Make No Assumptions About Resources Yet

Lets look at the operations needed with initiation period = 2 and latency=4 for several clock cycles.

Successive Sample values are labeled A,B,C etc.

Clk	Operations		
	Sample A	Sample B	Sample C
1	N4(*) N5(*), Input x		
2	N2(*),N3(*),N7(+)		
3	N6 (+)	N4(*) N5(*), Input x	
4	N8(+)	N2(*),N3(*),N7(+)	
5		N6(+)	
6		N8(+)	
7			
8			

# Initiation Period = 2

Lets look at the operations needed with initiation period = 2 for several clock cycles. Successive Sample values are labeled A,B,C etc.

Clk	Operations		
	Sample A	Sample B	Sample C
1	N4(*) N5(*), Input x		
2	N2(*),N3(*),N7(+)		
3	N6(+)	N4(*) N5(*), Input x	
4	N8(+)	N2(*),N3(*),N7(+)	
5		N6(+)	N4(*) N5(*), Input x
6		N8(+)	N2(*),N3(*),N7(+)
7			N6(+)
8			N8(+)

## Resources

Clk	Operations		
	Sample A	Sample B	Sample C
1	N4(*) N5(*), Input x		
2	N2(*),N3(*),N7(+)		
3	N6(+)	N4(*) N5(*), Input x	
4	N8(+)	N2(*),N3(*),N7(+)	
5		N6(+)	N4(*) N5(*), Input x
6		N8(+)	N2(*),N3(*),N7(+)
7			N6(+)
8			N8(+)

Two multiplies per clock, so need two multipliers (A, B).  
 In Clock #4, Clock #6 we have two additions, so need two adders (A, B).

# Initiation Period and Latency

The **initiation period** of this design is 2.

The **latency** is 4.

When initiation period  $\neq$  latency, then system level **pipelining** is occurring because the computations for more than one input sample are being accomplished in a single clock period.

Pipelining implies parallelism (in time) - more than one sample computation is in progress at any given clock cycle.

To schedule, need to generalize the table.

## Generalized Schedule

Clk	Operations		
	Sample J-1:	Sample J	Sample J+1
I-2	N4(*) N5(*), Input X		
I-1	N2(*),N3(*),N7(+)		
I+0	N6(+)	N4(*) N5(*), Input X	
I+1	N8(+)	N2(*),N3(*),N7(+)	
I+2		N6(+)	N4(*) N5(*), Input X
I+3		N8(+)	N2(*),N3(*),N7(+)

# Schedule Clk I+0

What do we need in Registers at Clock I+0?

Schedule Clk I-1:

For Sample J-1: N2, N3, N7

For Sample J: x@3, x@2, x@1

For Sample J+1: No operations

Registers: RA: x@3, RB: x@2, RC: x@1,  
RD: N2, RE: N3, RF:N7

Schedule Clk I+0:

Sample J-1: N6(N3+N7) RF ← RE + RF overwrite N7 value, no longer needed

Sample J: Input x RE ← x overwrite N3 value, no longer needed

Sample J: N4(x@2\*a2) RG ← RB\*a2 add new register RG to hold N4

Sample J: N5(x@3\*a3) RA ← RA\*a1 overwrite x@3 value, no longer needed

*Finished: Added extra Register RG*

# Schedule Clk I+1

Registers: RA: N5, RB: x@2, RC: x@1, RD: N2, RE: x, RF:N6,  
RG: N4

After Clock, Registers need to be setup for next clock:

RA: x@3, RB: x@2, RC: x@1, RD: N2, RE: N3, RF:N7

Schedule Clk I+1:

Sample J-1: N8(N2+N6) Y ← RD + RF output goes to Y bus

Sample J: N2(x\*a0) RD ← RE \*a0 overwrite old N2 value, no longer needed

Important that N2 go into RD because this is needed for next clock cycle.

Sample J: N3(x@1\*a1) RE ← RC\*a1 Need RE=N3 for next clock

x value is currently in RE. In the next clock, x = x@1 for sample J+1,  
so move x into RC register.

Sample J+1: RC=x@1 RC ← RE J+1:x@1 = J:x

Sample J: N7(N4+N5) RF ← RG + RA Need N7 in RF for J+1 sample

Sample J+1: RA=x@3 RA ← RB J+1: x@3 = J:x@2

Sample J+1: RB=x@2 RB ← RC J+1: x@2 = J:x@1

*Finished: no extra registers needed*

## Schedule: Clks I+2, I+3

Schedule for Clk I+2 is repeat of Clk I+0!!!

Schedule for Clk I+3 is repeat of Clk I+1!!

Actually, generalized schedule only needs two clocks

## Resource Comparison

Init Rate	Resources		
	Multipliers	Adders	Registers
4	2	1	10
2	2	2	11

Doubling the initiation rate did NOT double the hardware resources needed. Why?

Because Execution units for Initiation period = 4 were not fully utilized!

# Execution Unit Utilization

Init Rate	Execution Unit Utilization			
	Mult A	Mult B	Add A	Add B
4	50% (2/4)	50 (2/4)	75% (3/4)	N/A
2	100% (2/2)	100% (2/2)	100% (2/2)	50% (1/2)

Note that multipliers were not fully utilized for Initiation period = 2,  
we used this free time in the schedule for Initiation period = 4.

## Init Period = 1

Clk	Operations			
	Sample A	Sample B	Sample C	Sample D
1	N4(*) N5(*), Input x			
2	N2(*),N3(*), N7(+)	N4(*) N5(*), Input x		
3	N6(+)	N2(*),N3(*), N7(+)	N4(*) N5(*), Input x	
4	N8(+)	N6(+)	N2(*),N3(*), N7(+)	N4(*) N5(*), Input x
5		N8(+)	N6(+)	N2(*),N3(*), N7(+)
6			N8(+)	N6(+)
7				N8(+)

Four multiplies in clock 4, so need four multipliers:  
(A, B, C, D).

Three additions in clock 4, so need three adders:  
(A, B, C).

# Resource Comparison Again

Init Rate	Resources		
	Multipliers	Adders	Registers
4	2	1	10
2	2	2	11
1	4	3	??

Latency for all of these designs is 4 clocks.

This table clearly illustrates the time versus area tradeoff in Digital Systems.

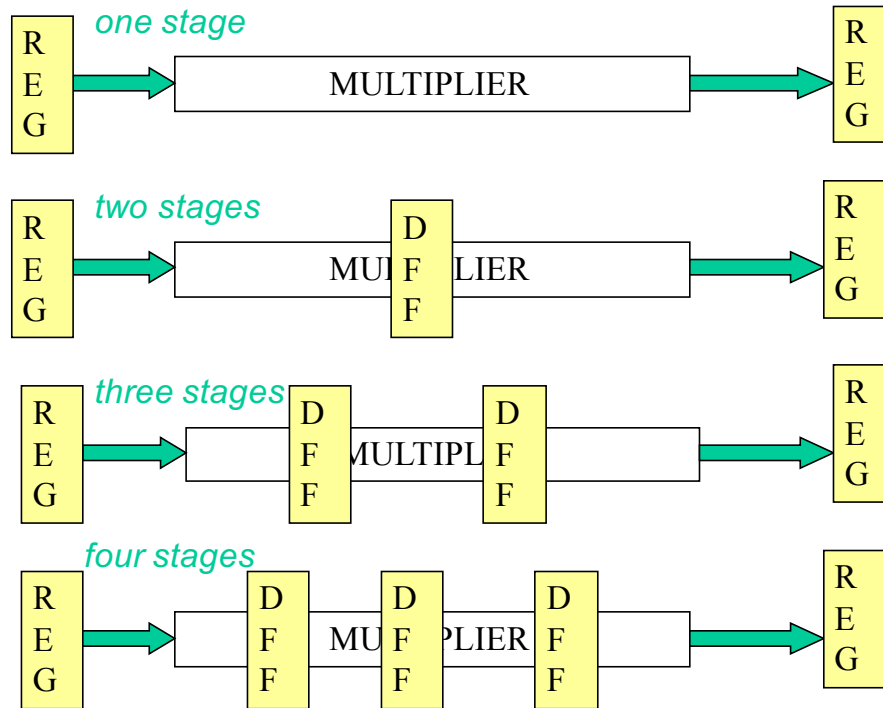
Will cost you MORE resources to do something in LESS Time!

## Less Clock cycles, Lower Clock Period

- Computation Time = #of Clocks×Clock Period
- Throughput = 1/(Computation Time)
- Increasing the initiation period will increase the computation rate in terms of clock cycles
  - Less clock cycles between new outputs
- To decrease the clock period (increase clock frequency), need to have shorter combinational paths in the design
  - PIPELINE the individual execution units
  - Multiplier will have much longer delay than adder, so will want to pipeline this first



# Recall Pipelining



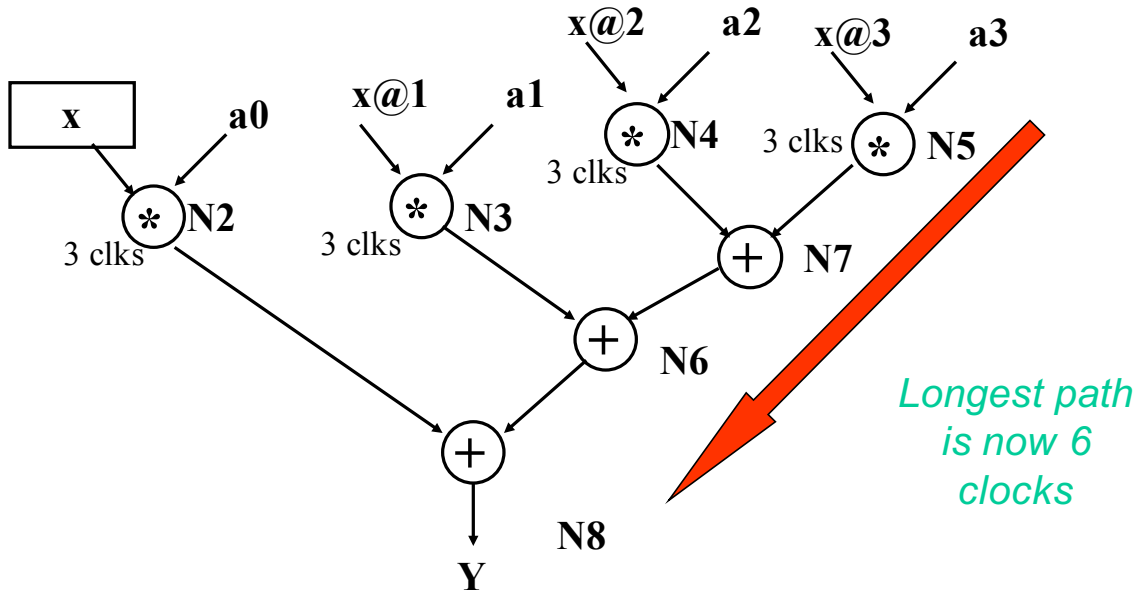
## Assume Multiplier with 2-Stage Pipeline

Schedule with Initiation Period = Latency; 2 mult, 1 adder

	Adder	MultA	MultB	IO
Clk 1	idle	N5	N4	Input x
Clk 2		N3	N2	
Clk 3		(n5 rdy, saved to register)	(n4 rdy, saved to register)	
Clk 4	N7	(n3 rdy, saved to register)	(n2 rdy, saved to register)	
Clk 5	N6			
Clk 6	N8			

Scheduling now takes 6 clocks. N7 depends on N5, N4 - can't do this until Clk 4 because multiplier result not ready.

## Flowgraph Annotated with Delay Values (weighted DAG)



## Decreasing Initiation Period to 2

Clk	Operations			
	Sample A:	Sample B	Sample C	Sample D
1	N4(*), N5(*), Input x			
2	N2(*), N3(*)			
3		N4(*), N5(*), Input x		
4	N7(+)	N2(*), N3(*)		
5	N6(+)		N4(*), N5(*), Input x	
6	N8(+)	N7(+)	N2(*), N3(*)	
7		N6(+)		N4(*), N5(*), Input x
8		N8(+)	N7(+)	N2(*), N3(*)
9			N6(+)	
10			N8(+)	N7(+)

# Decreasing Initiation Period to 2

Clk	Operations			
	Sample J-2	Sample J-1	Sample J	Sample J+1
I-4	N4(*) N5(*), Input x			
I-3	N2(*),N3(*)			
I-2		N4(*) N5(*), Input x		
I-1	N7(+)	N2(*),N3(*)		
I	N6(+)		N4(*) N5(*), Input x	
I+1	N8(+)	N7(+)	N2(*),N3(*)	
I+2		N6(+)		N4(*) N5(*), Input x
I+3		N8(+)	N7(+)	N2(*),N3(*)
I+4			N6(+)	
I+5			N8(+)	N7(+)

## Schedule with Initiation Period =2

Schedule with Initiation Period = Latency; 2 mult, 2 adders

	Adder A	Adder B	MultA	MultB	IO
Clk I	N6 (j-2)	idle	N5 (j)	N4 (j)	Input x
Clk I+1	N8 (j-2)	N7 (j-1)	N3 (j)	N2(j)	
Clk I+2	N6 (j-1)	idle	N5(j+1)	N4(j+1)	Input x
Clk I+3	N8 (j-1)	N7 (j)	N3(j+1)	N3(j+1)	
Clk I+4	N6 (j)	idle	N5(j+2)	N4(j+2)	Input x
Clk I+5	N8 (j)	N7 (j+1)	N3(j+2)	N3(j+2)	

General schedule is I, I+1. 6 clocks shown to complete one computation. Initiation Period = 2, Latency = 6. Overlapping computations of three samples.