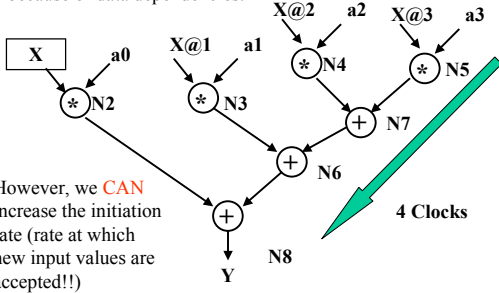


### Increasing the Initiation Rate

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



However, we **CAN** increase the initiation rate (rate at which new input values are accepted!!)

BR 1/99

1

### Initiation Rate = 2

Lets look at the operations needed with initiation rate = 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(+)

BR 1/99

2

### 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 need two adders (A, B).

BR 1/99

3

### Initiation Rate, Latency

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

The **latency** is 4.

When initiation rate  $\neq$  latency, then **pipelining** is being done because the computations for more than one input sample are being done.

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

To schedule, need to generalize the table.

BR 1/99

4

### Generalized Schedule

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

Note: The initiation rate must be evenly divisible into the latency in order to generalize the table.

BR 1/99

5

### Schedule Clk I+0

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

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  $\leftarrow$  RE + RF

Sample J: Input X RE  $\leftarrow$  X

Sample J: N4(x@2\*a2) RG  $\leftarrow$  RB \* a2

Sample J: N5(x@3\*a3) RA  $\leftarrow$  RA \* a1

overwrite N7 value, don't need.  
overwrite N3 value, don't need  
add new register RG to hold N4  
overwrite x@3 value, don't need.

Finished: Added extra Register RG.

BR 1/99

6

### 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 which is:**  
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, don't need  
Very 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!!  
But what about X value that is in RE??? Next clock, X= X@1 for sample J+1,  
so put 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. BR 1/99

7

### 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!**

BR 1/99

8

### 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 InitRate = 4 were not fully utilized!

BR 1/99

9

### Execution Unit Utilization table

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 InitRate = 2, we used this free time in the schedule for InitRate = 4.

BR 1/99

10

### Init Rate = 1

Clk	Operations			Sample D
	Sample A:	Sample B	Sample C	
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(+)	
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).

BR 1/99

11

### 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!

BR 1/99

12

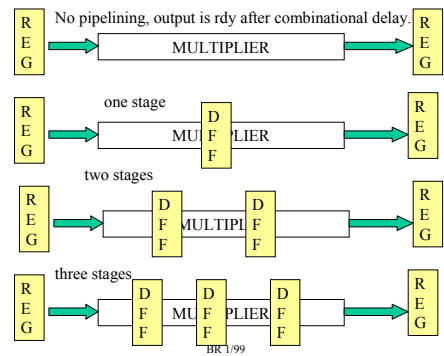
## Less Clock cycles, Lower Clock Period

- Computation Time = #of Clocks \* Clock Period
- Increasing the initiation rate 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

BR 1/99

13

## Recall what Pipelining of Multiplier does



14

## Assume Multiplier is pipelined by 2 stages

Do a solution with Initiation Rate = Latency; 2 mult, 1 adder

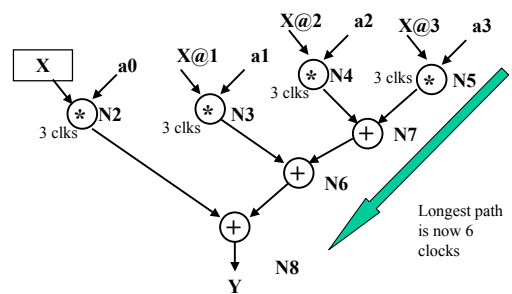
	Adder	MultA	MultB	IO
Clk 1	idle	N5	N4	Input X
Clk 2		N3	N2	
Clk 3		(n5 rdy, saved to register)	(n4 rdy, saved to register)	
Clk 4	N7	(n3 rdy, saved to register)	(n2 rdy, saved to register)	
Clk 5	N6			
Clk 6	N8			

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

BR 1/99

15

## When using Pipelined Execution Units, noting clock cycle cost nodes is helpful



BR 1/99

16

## Increasing Initiation Rate 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(+)

Repeating clocks



BR 1/99

17

## Increasing Initiation Rate 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(+)

Repeating clocks



BR 1/99

18

### Schedule, Init Rate =2,

Do a solution with Initiation Rate = 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 Rate = 2, Latency = 6. Overlapping computations of three samples.