Scheduling Examples

Nachiket Kapre
nachiket@uwaterloo.ca
UNIVERSITY OF WATERLOO

Outline

- Scheduling under Area-Time Tradeoff Scenarios
- Scheduling with Xilinx DSP blocks (simplified)

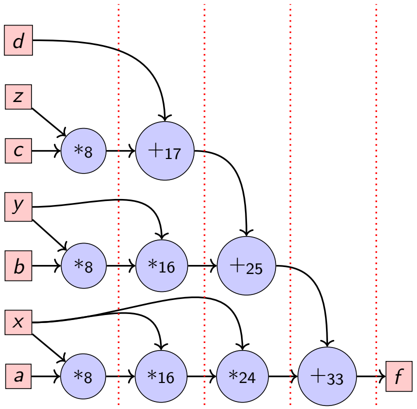
Area-Time Tradeoffs

- Developers must Hold resources accountable for their inclusion
  - Resources cost area on a chip = $$$
  - Is it worth the silicon cost to include a particular hardware resource?
- Two extremes:
  - Fully sequential (one resource of each type) is bare minimum, slow and cheap
  - Fully spatial (parallel) is the best performing, most expensive
- Intermediate cases possible, use extra resources wisely! $\rightarrow$ analysis and optimizations required

Polynomial $f = a \cdot x^3 + b \cdot y^2 + c \cdot z + d$

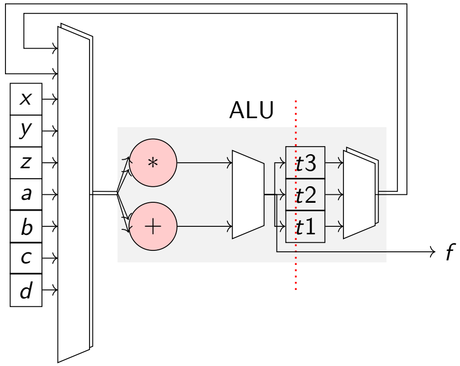
- Each input is an 8-bit unsigned number. Addition does overflow (extra output bit necessary)
- Expression has 3 additions and 6 multiplications!
  - Constraint: We are using 2-input add and 2-input multiply primitives
- Fully sequential design requires one multiplier + one adder
  - Latency = 9 cycles
  - Throughput = $\frac{1}{9}$
- Fully parallel design requires 3 adders and 6 multipliers
  - Latency = 4 cycles
  - Throughput = 1

Fully Spatial (Parallel) Design



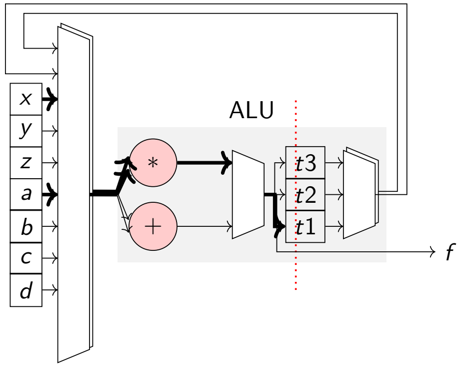
- Throughput=1, Latency=4
- Clock Period=max($T_{*_8}, T_{*_{16}}, T_{*_{24}}, T_{+_{17}}, T_{+_{25}}, T_{+_{33}}$)

Fully Sequential Design



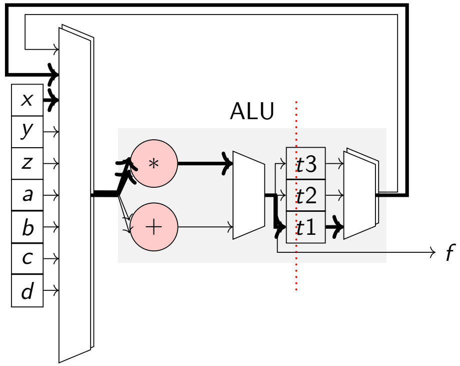
- 1st Cycle $\rightarrow t1 <= a \times x$
- 2nd Cycle $\rightarrow t1 <= t1 \times x$
- 3rd Cycle $\rightarrow t1 <= t1 \times x$
- 4th Cycle $\rightarrow t2 <= b \times y$
- 5th Cycle $\rightarrow t2 <= t2 \times y$
- 6th Cycle $\rightarrow t3 <= c \times z$
- 7th Cycle $\rightarrow t3 <= t3 + d$
- 8th Cycle $\rightarrow t3 <= t3 + t2$
- 9th Cycle $\rightarrow f <= t3 + t1$

Fully Sequential Design



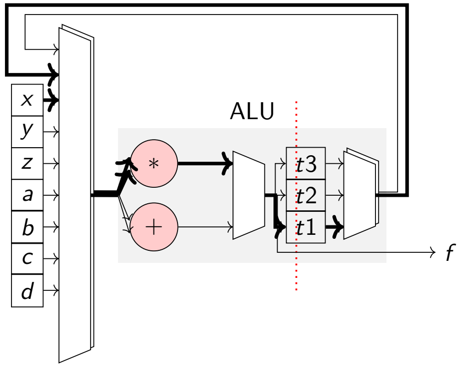
- 1st Cycle $\rightarrow t1 <= a \times x$
- 2nd Cycle $\rightarrow t1 <= t1 \times x$
- 3rd Cycle $\rightarrow t1 <= t1 \times x$
- 4th Cycle $\rightarrow t2 <= b \times y$
- 5th Cycle $\rightarrow t2 <= t2 \times y$
- 6th Cycle $\rightarrow t3 <= c \times z$
- 7th Cycle $\rightarrow t3 <= t3 + d$
- 8th Cycle $\rightarrow t3 <= t3 + t2$
- 9th Cycle $\rightarrow f <= t3 + t1$

Fully Sequential Design



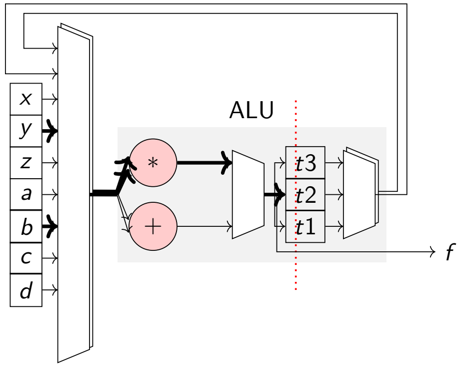
- 1st Cycle $\rightarrow t1 <= a \times x$
- 2nd Cycle $\rightarrow t1 <= t1 \times x$
- 3rd Cycle $\rightarrow t1 <= t1 \times x$
- 4th Cycle $\rightarrow t2 <= b \times y$
- 5th Cycle $\rightarrow t2 <= t2 \times y$
- 6th Cycle $\rightarrow t3 <= c \times z$
- 7th Cycle $\rightarrow t3 <= t3 + d$
- 8th Cycle $\rightarrow t3 <= t3 + t2$
- 9th Cycle $\rightarrow f <= t3 + t1$

Fully Sequential Design



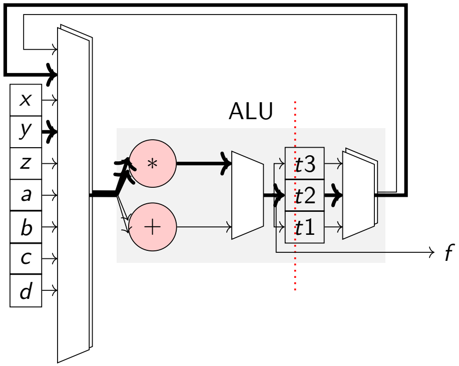
- 1st Cycle $\rightarrow t1 <= a \times x$
- 2nd Cycle $\rightarrow t1 <= t1 \times x$
- 3rd Cycle $\rightarrow t1 <= t1 \times x$
- 4th Cycle $\rightarrow t2 <= b \times y$
- 5th Cycle $\rightarrow t2 <= t2 \times y$
- 6th Cycle $\rightarrow t3 <= c \times z$
- 7th Cycle $\rightarrow t3 <= t3 + d$
- 8th Cycle $\rightarrow t3 <= t3 + t2$
- 9th Cycle $\rightarrow f <= t3 + t1$

Fully Sequential Design



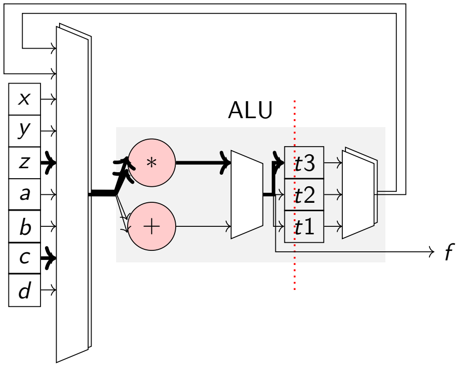
- 1st Cycle $\rightarrow t1 <= a \times x$
- 2nd Cycle $\rightarrow t1 <= t1 \times x$
- 3rd Cycle $\rightarrow t1 <= t1 \times x$
- 4th Cycle $\rightarrow t2 <= b \times y$
- 5th Cycle $\rightarrow t2 <= t2 \times y$
- 6th Cycle $\rightarrow t3 <= c \times z$
- 7th Cycle $\rightarrow t3 <= t3 + d$
- 8th Cycle $\rightarrow t3 <= t3 + t2$
- 9th Cycle $\rightarrow f <= t3 + t1$

Fully Sequential Design



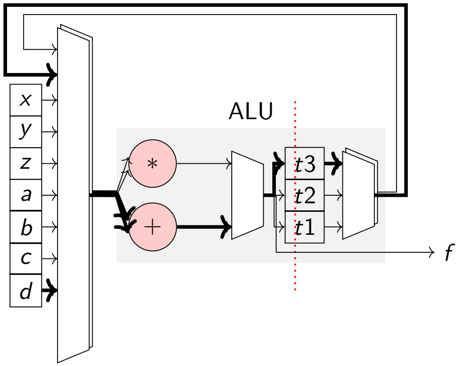
- 1st Cycle $\rightarrow$ $t1 <= a \times x$
- 2nd Cycle $\rightarrow$ $t1 <= t1 \times x$
- 3rd Cycle $\rightarrow$ $t1 <= t1 \times x$
- 4th Cycle $\rightarrow$ $t2 <= b \times y$
- 5th Cycle $\rightarrow$ $t2 <= t2 \times y$
- 6th Cycle $\rightarrow$ $t3 <= c \times z$
- 7th Cycle $\rightarrow$ $t3 <= t3 + d$
- 8th Cycle $\rightarrow$ $t3 <= t3 + t2$
- 9th Cycle $\rightarrow$ $f <= t3 + t1$

Fully Sequential Design



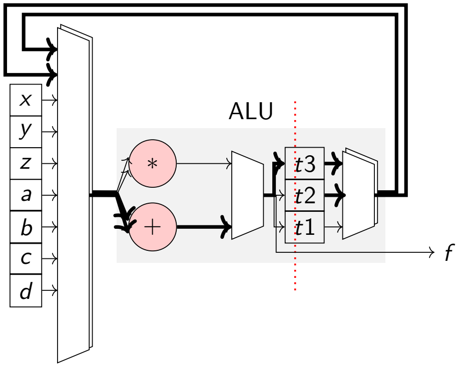
- 1st Cycle $\rightarrow t1 <= a \times x$
- 2nd Cycle $\rightarrow t1 <= t1 \times x$
- 3rd Cycle $\rightarrow t1 <= t1 \times x$
- 4th Cycle $\rightarrow t2 <= b \times y$
- 5th Cycle $\rightarrow t2 <= t2 \times y$
- 6th Cycle $\rightarrow t3 <= c \times z$
- 7th Cycle $\rightarrow t3 <= t3 + d$
- 8th Cycle $\rightarrow t3 <= t3 + t2$
- 9th Cycle $\rightarrow f <= t3 + t1$

Fully Sequential Design



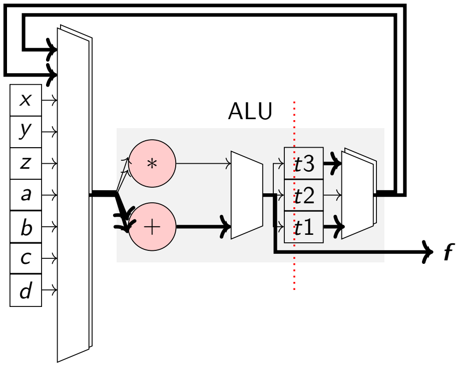
- 1st Cycle $\rightarrow t1 <= a \times x$
- 2nd Cycle $\rightarrow t1 <= t1 \times x$
- 3rd Cycle $\rightarrow t1 <= t1 \times x$
- 4th Cycle $\rightarrow t2 <= b \times y$
- 5th Cycle $\rightarrow t2 <= t2 \times y$
- 6th Cycle $\rightarrow t3 <= c \times z$
- 7th Cycle $\rightarrow t3 <= t3 + d$
- 8th Cycle $\rightarrow t3 <= t3 + t2$
- 9th Cycle $\rightarrow f <= t3 + t1$

Fully Sequential Design



- 1st Cycle $\rightarrow t1 <= a \times x$
- 2nd Cycle $\rightarrow t1 <= t1 \times x$
- 3rd Cycle $\rightarrow t1 <= t1 \times x$
- 4th Cycle $\rightarrow t2 <= b \times y$
- 5th Cycle $\rightarrow t2 <= t2 \times y$
- 6th Cycle $\rightarrow t3 <= c \times z$
- 7th Cycle $\rightarrow t3 <= t3 + d$
- 8th Cycle $\rightarrow t3 <= t3 + t2$
- 9th Cycle $\rightarrow f <= t3 + t1$

Fully Sequential Design

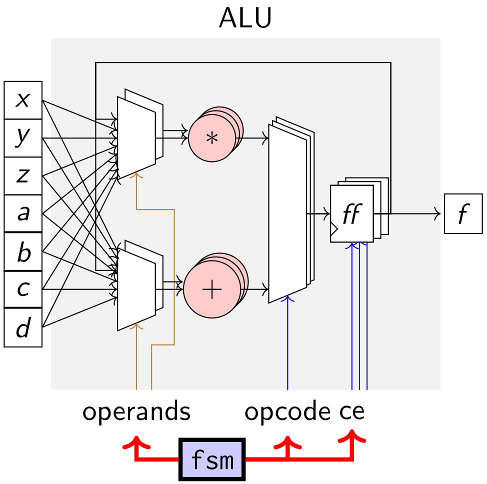


- 1st Cycle $\rightarrow t1 <= a \times x$
- 2nd Cycle $\rightarrow t1 <= t1 \times x$
- 3rd Cycle $\rightarrow t1 <= t1 \times x$
- 4th Cycle $\rightarrow t2 <= b \times y$
- 5th Cycle $\rightarrow t2 <= t2 \times y$
- 6th Cycle $\rightarrow t3 <= c \times z$
- 7th Cycle $\rightarrow t3 <= t3 + d$
- 8th Cycle $\rightarrow t3 <= t3 + t2$
- 9th Cycle $\rightarrow f <= t3 + t1$

Intermediate Design Points

- Often, we can afford to allocate more resources $\rightarrow$ where to spend this resource?
- Consider design alternatives $\rightarrow$ add more multipliers, and/or adders, and/or registers
- Among alternatives, pick the one that most improves throughput (or latency, if that is the designer constraint)

Generic design template



Schedule Table for one multiplier, one adder

Cycle	**Operators**	
	add_0	$mult_0$
0	-	$a \cdot x$
1	-	$a \cdot x^2$
2	-	$a \cdot x^3$
3	$a \cdot x^3 + d$	$b \cdot y$
4	-	$b \cdot y^2$
5	$a \cdot x^3 + b \cdot y^2 + d$	$c \cdot z$
6	$a \cdot x^3 + b \cdot y^2 + c \cdot z + d$	-

- Latency = 7 cycles, Throughput = $\frac{1}{7}$
- Problem: Lot of wasted cycles on the adder!
- Efficiency = Useful Cycles/Total Cycles = $\frac{9}{14}$

Schedule Table for two multipliers, one adder

Cycle	**Operators**		
	add_0	$mult_0$	$mult_1$
0	-	$a \cdot x$	$b \cdot y$
1	-	$a \cdot x^2$	$b \cdot y^2$
2	$b \cdot y^2 + d$	$a \cdot x^3$	$c \cdot z$
3	$a \cdot x^3 + b \cdot y^2 + d$	-	-
4	$a \cdot x^3 + b \cdot y^2 + c \cdot z + d$	-	-

- Latency = 5 cycles, Throughput = $\frac{1}{5}$
- Efficiency = $\frac{9}{15}$

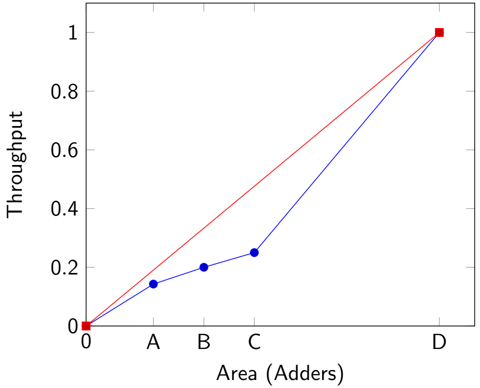
Schedule Table for three multipliers, one adder

Cycle	**Operators**			
	add_0	$mult_0$	$mult_1$	$mult_2$
0	-	$a \cdot x$	$b \cdot y$	$c \cdot z$
1	$c \cdot z + d$	$a \cdot x^2$	$b \cdot y^2$	-
2	$b \cdot y^2 + c \cdot z + d$	$a \cdot x^3$	-	-
3	$a \cdot x^3 + b \cdot y^2 + c \cdot z + d$	-	-	-

- Latency $= 4$ cycles, Throughput $= \frac{1}{4}$
- Efficiency $= \frac{9}{16}$

Area-Time Tradeoff Visualization

- **Assumption**: Area of multiplier = 3*Area of adder
  - A = one multiplier + one adder = 4 units
  - B = two multipliers + one adder = 7 units
  - C = three multipliers + one adder = 10 units
  - D = six multipliers + three adders = 21 units



Scheduling with Variable Resource Constraints

- Area-Time tradeoffs are at the heart of hardware design $\rightarrow$ every piece of hardware must justify its existence $\rightarrow$ millions of dollars of manufacturing!
- Aggressively schedule to minimize latency, boost throughput
- Always check if replicating cheap block is a possibility?

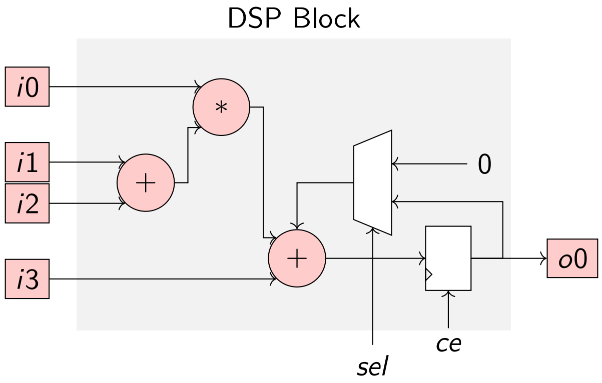
Replication Scenarios – Good

- Replicate A to fit the area occupied by C
  - Area(C) $= 10$ units, Area(A) $= 4$ units.
  - Area(two copies of A) $= 4$ units $\times\ 2 = 8$ units
  - Thus, we can easily fit $2\times$ A in area of one C
- Check throughputs:
  - Throughput of A $= \frac{1}{7}$
  - Throughput of C $= \frac{1}{5}$
  - Throughput of two copies of A $= 2 \times \frac{1}{7} = \frac{1}{3.5}$
  - Thus, Throughput(two copies of A) $>$ Throughout(C)
    - $\frac{1}{3.5} > \frac{1}{5}$
- **What this means**? If you can afford to pay for C, just implement instances of A instead!

Replication Scenarios – Bad

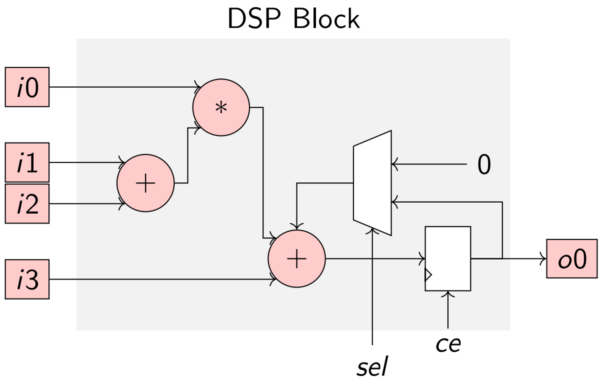
- Replicate A to fit the area occupied by D
  - Area(D) $=$ 21 units, Area(A) $=$ 4 units
  - Area(five copies of A) $=$ 4 units $\times$ 5 $=$ 20 units
  - Thus, we can easily fit 5$\times$ A in area of one C
- Check throughputs:
  - Throughput of A $= \frac{1}{7}$
  - Throughput of D $= 1$
  - Throughput of five copies of A $= 5 \times \frac{1}{7} = \frac{5}{7}$
  - Thus, Throughput(five copies of A) $<$ Throughout(D)
    - $\frac{5}{7}<1$
- **What this means**? If you can afford to pay for D, just implement D instead of being a cheapskate and replicating A!

Scheduling with Fixed-Function Units



- Modern FPGAs contain add-multiply blocks that can be programmed to suite customer requirements
- Embedded FPGA blocks are fast structures $\rightarrow$ more on FPGA architecture after midterm

Scheduling with Fixed-Function Units



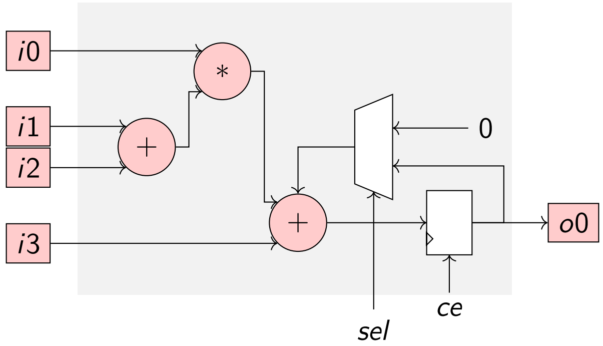
- Three arithmetic units $\rightarrow$
  - pre-adder ($i1 + i2$)
  - multiplier ($i0 \times (i1 + i2)$)
  - accumulator ($o0 = o0 + i0 \times (i1 + i2) + i3$)
- Programmable *ce* and *sel* inputs to steer and control data inside the DSP

Computational Folding over FPGA DSPs

- Time multiplexing of a DSP block is an example of scheduling
- Another word for it $\rightarrow$ computational folding
  - Why? Because computational origami was taken
- Split your input expression into multiple DSP operations
- Precision of adders and multipliers determine how to slice up operations $\rightarrow$ for this lecture, assume sufficient

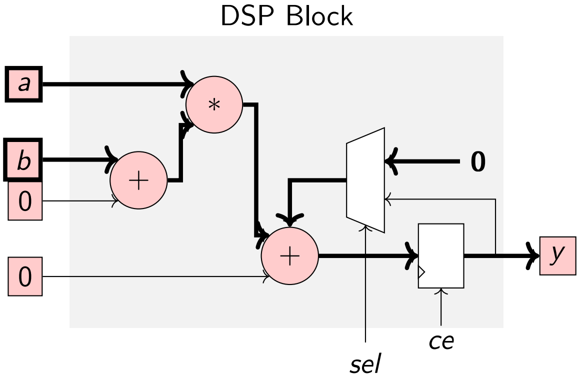
Types of Operations possible

DSP Block



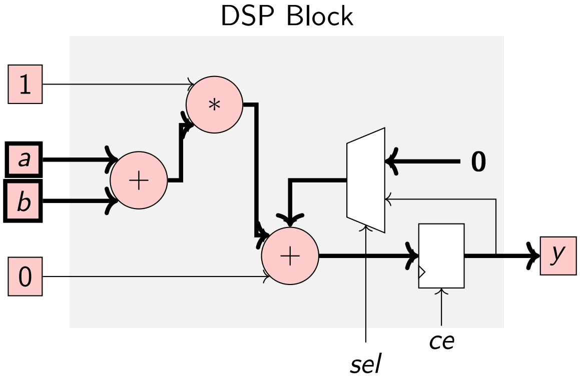
Highly flexible datapath

Types of Operations possible



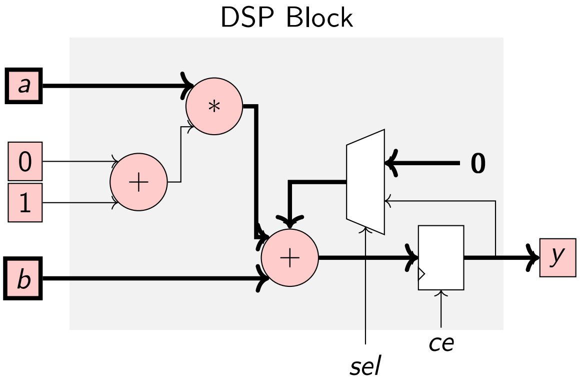
$$y = a \cdot b$$

Types of Operations possible



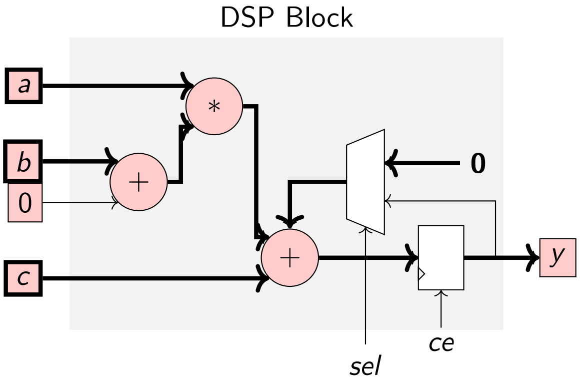
$$y = a + b$$

Types of Operations possible



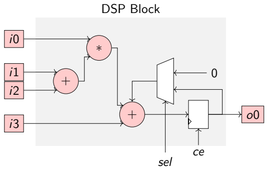
$$y = a + b$$

Types of Operations possible



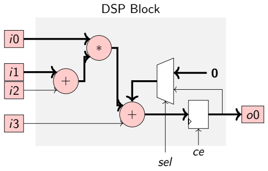
$$y = a \cdot b + c$$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP



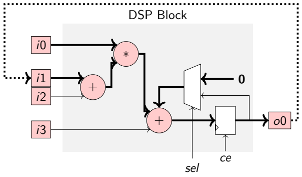
Cycle	**Intra-DSP Operations**			**Output**
	$preadd(p)$	$mult(m)$	$accum$	$o0 = accum$
0	$a + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x$
1	$a \cdot x + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x^2$
2	$b + 0$	$p \cdot x$	$m + o0 + 0$	$a \cdot x^2 + b \cdot x$
3	$0 + 0$	$p \cdot x$	$m + o0 + c$	$a \cdot x^2 + b \cdot x + c$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP



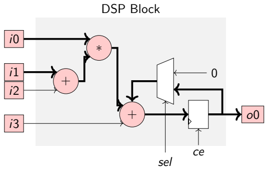
Cycle	Intra-DSP Operations			Output
	preadd(p)	*mult*(m)	*accum*	$o0 = accum$
0	$a + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x$
1	$a \cdot x + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x^2$
2	$b + 0$	$p \cdot x$	$m + o0 + 0$	$a \cdot x^2 + b \cdot x$
3	$0 + 0$	$p \cdot x$	$m + o0 + c$	$a \cdot x^2 + b \cdot x + c$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP



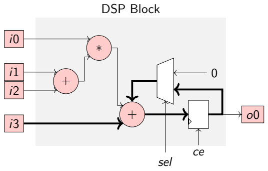
Cycle	**Intra-DSP Operations**			**Output**
	preadd(p)	*mult*(m)	*accum*	$o0 = accum$
0	$a + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x$
1	$a \cdot x + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x^2$
2	$b + 0$	$p \cdot x$	$m + o0 + 0$	$a \cdot x^2 + b \cdot x$
3	$0 + 0$	$p \cdot x$	$m + o0 + c$	$a \cdot x^2 + b \cdot x + c$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP



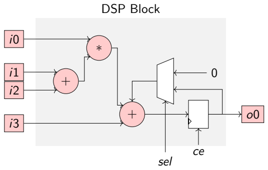
Cycle	Intra-DSP Operations			Output
	preadd(p)	*mult(m)*	*accum*	$o0 = accum$
0	$a + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x$
1	$a \cdot x + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x^2$
2	$b + 0$	$p \cdot x$	$m + o0 + 0$	$a \cdot x^2 + b \cdot x$
3	$0 + 0$	$p \cdot x$	$m + o0 + c$	$a \cdot x^2 + b \cdot x + c$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP



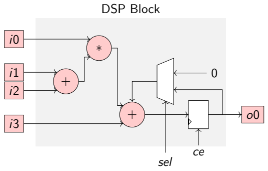
Cycle	**Intra-DSP Operations**			**Output**
	preadd(p)	*mult*(m)	*accum*	$o0 = accum$
0	$a + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x$
1	$a \cdot x + 0$	$p \cdot x$	$m + 0 + 0$	$a \cdot x^2$
2	$b + 0$	$p \cdot x$	$m + o0 + 0$	$a \cdot x^2 + b \cdot x$
3	$0 + 0$	$p \cdot x$	$m + o0 + c$	$a \cdot x^2 + b \cdot x + c$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP



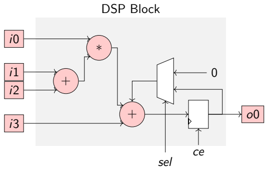
Cycle	**IO Table**			
	$i0$	$i1$	$i2$	$i3$
0	x	a	0	0
1	x	$a \cdot x$	0	0
2	x	b	0	0
3	0	-	0	c

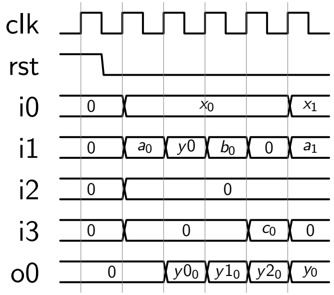
Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP



Cycle	Selection	
	sel	*ce*
0	0	1
1	0	1
2	1	1
3	1	1

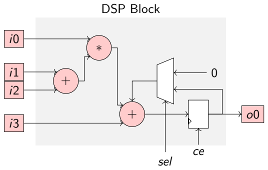
Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP





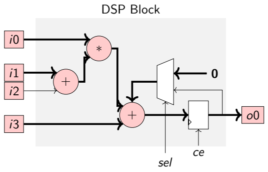
- $y0 = a \cdot x$
- $y1 = a \cdot x^2$
- $y2 = a \cdot x^2 + b \cdot x$
- $y = a \cdot x^2 + b \cdot x + c$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP (Optimized)



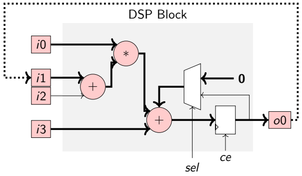
Cycle	Intra-DSP Operations			Output
	preadd(p)	*mult*(m)	*accum*	$o0 = accum$
0	$a + 0$	$p \cdot x$	$m + 0 + b$	$a \cdot x + b$
1	$(a \cdot x + b) + 0$	$p \cdot x$	$m + 0 + c$	$(a \cdot x + b) \cdot x + c$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP (Optimized)



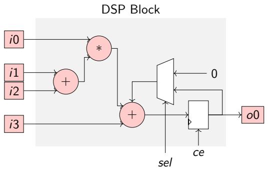
Cycle	Intra-DSP Operations			Output
	preadd(p)	*mult(m)*	*accum*	$o0 = accum$
0	$a + 0$	$p \cdot x$	$m + 0 + b$	$a \cdot x + b$
1	$(a \cdot x + b) + 0$	$p \cdot x$	$m + 0 + c$	$(a \cdot x + b) \cdot x + c$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP (Optimized)



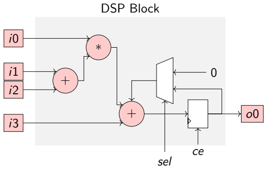
Cycle	Intra-DSP Operations			Output
	preadd(p)	*mult*(m)	*accum*	$o0 = accum$
0	$a + 0$	$p \cdot x$	$m + 0 + b$	$a \cdot x + b$
1	$(a \cdot x + b) + 0$	$p \cdot x$	$m + 0 + c$	$(a \cdot x + b) \cdot x + c$

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP (Optimized)



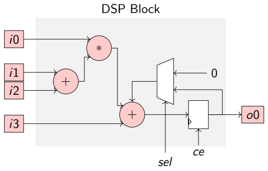
Cycle	IO Table			
	$i0$	$i1$	$i2$	$i3$
0	x	a	0	b
1	x	$a \cdot x + b$	0	c

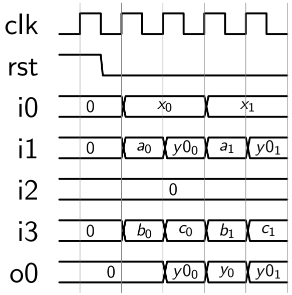
Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP (Optimized)



Cycle	Selection	
	sel	*ce*
0	0	1
1	0	1

Mapping poly $a \cdot x^2 + b \cdot x + c$ to a DSP (Optimized)





- $y0 = a \cdot x + b$
- $y = (a \cdot x + b) \cdot x + c$

Mapping Technique

- Different ways to perform algebraic factorization
- Core DSP computes $o0 = (i1 + i2) \cdot i0 + i3 + o0$
- Attempt to extract this expression from your application
- Use as many multiplier and adder blocks in same cycle
- External feedback sometimes necessary!
- Minimize the number of cycles

Class Wrapup

- Area-Time tradeoffs fundamental aspect of hardware design
- Resources must be introduced carefully with high utilization
- Reduce the number of cycles, and enhance utilization of hardware resources
- FPGAs today ship with DSP blocks that contain fast multiply and addition blocks $\rightarrow$ scheduling is needed to use them properly.