

# ESE5320: System-on-a-Chip Architecture

Day 23: November 18, 2024  
VLIW  
(Very Long Instruction Word Processors)

Penn ESE5320 Fall 2024 -- DeHon



1

## Today

VLIW (Very Large Instruction Word)  
Exploiting Instruction-Level Parallelism (ILP)

- Part 1:
  - Demand, Basic Model
- Part 2:
  - Tuning, Unrolling
- Part 3:
  - Data Movement
- Part 4 (bonus, probably not cover):
  - Zero-Overhead Loops, Pipelined Operators

Penn ESE5320 Fall 2024 -- DeHon

2

## Message

- VLIW as a Model for
  - Instruction-Level Parallelism (ILP)
  - Customizing Datapaths
  - Area-Time Tradeoffs
  - Real-Time, Scheduled Sequentialization

Penn ESE5320 Fall 2024 -- DeHon

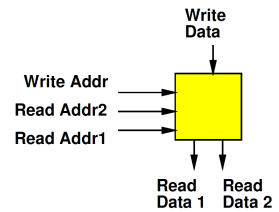
3

3

Day 6

## Register File

- Small Memory
- Usually with multiple ports
  - Ability to perform multiple reads and writes simultaneously
- Small
  - To make it fast (small memories fast)
  - Multiple ports are expensive



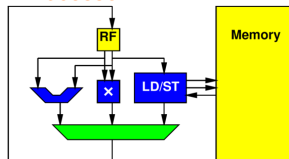
Penn ESE5320 Fall 2024 -- DeHon

4

4

## Preclass 1

- Cycles per multiply-accumulate (throughput question, not latency)
  - Spatial Pipeline
  - Processor



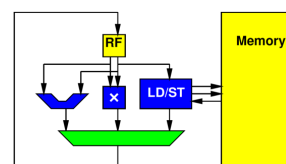
Penn ESE5320 Fall 2024 -- DeHon

5

5

## Preclass 1

- How different?
  - Resources
    - Number of units can perform adds
    - Number of simultaneous memory reads
  - Ability to use resources
    - What resources can use in same cycle



Penn ESE5320 Fall 2024 -- DeHon

6

6

## Computing Forms

- Processor – does one thing at a time
- Spatial Pipeline – can do many things, but always the same
- Vector – can do the same things on many pieces of data

Penn ESE5320 Fall 2024 -- DeHon

7

7

## In Between

Independent Instructions

VLIW

What if...

- Want to
  - Do many things at a time (ILP)
  - But not the same (DLP)

SIMD

Shared Instruction

Penn ESE5320 Fall 2024 -- DeHon

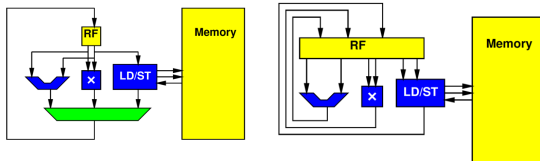
8

8

## In Between

What if...

- Want to
  - Do many things at a time (ILP)
  - But not the same (DLP)
- Want to use resources concurrently



Penn ESE5320 Fall 2024 -- DeHon

9

9

## In Between

What if...

- Want to
  - Do many things at a time (ILP)
  - But not the same (DLP)
- Want to use resources concurrently
- Want to
  - Accelerate specific task
  - But not go to spatial pipeline extreme

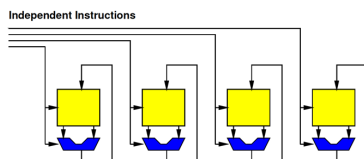
Penn ESE5320 Fall 2024 -- DeHon

10

10

## VLIW Feature: Supply Independent Instructions

- Provide instruction per ALU (resource)
- Instructions more expensive than Vector
  - But more flexible



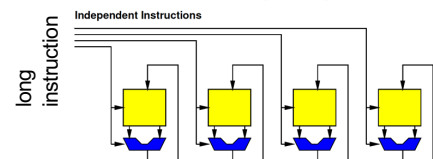
Penn ESE5320 Fall 2024 -- DeHon

11

11

## VLIW

- The "instruction"
  - The bits controlling the datapath
- ...becomes long
- Hence:
  - Very Long Instruction Word (VLIW)



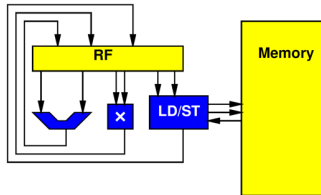
Penn ESE5320 Fall 2024 -- DeHon

12

12

## Control Heterogeneous Units

- Control each unit simultaneously and independently
  - More expensive than processor
    - Memory ports and/or interconnect
  - But more parallelism



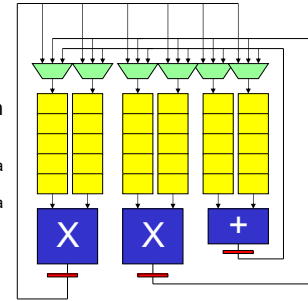
Penn ESE5320 Fall 2024 -- DeHon

13

13

## VLIW

- Very Long Instruction Word
- Set of operators
  - Parameterize number, distribution (X, +, sqrt...)
    - More operators → less time, more area
    - Fewer operators → more time, less area
- Memories for intermediate state



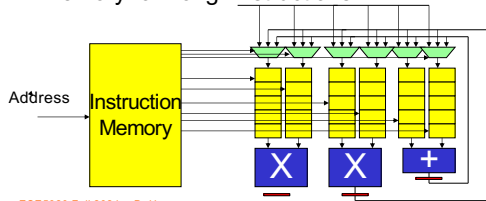
Penn ESE5320 Fall 2024 -- DeHon

14

14

## VLIW

- Very Long Instruction Word
- Set of operators
  - Parameterize number, distribution (X, +, sqrt...)
    - More operators → less time, more area
    - Fewer operators → more time, less area
- Memories for intermediate state
- Memory for "long" instructions

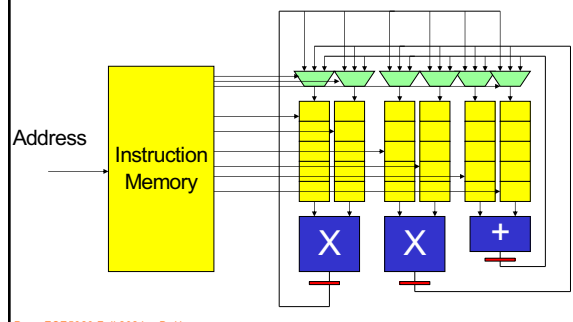


Penn ESE5320 Fall 2024 -- DeHon

15

15

## VLIW



Penn ESE5320 Fall 2024 -- DeHon

16

16

## VLIW

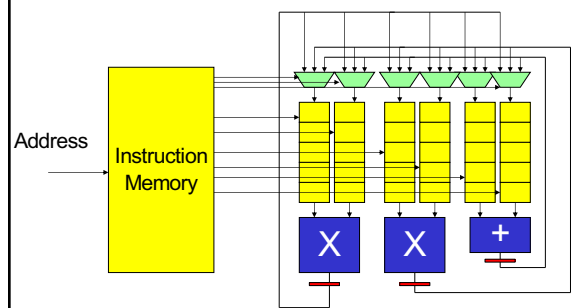
- Very Long Instruction Word
- Set of operators
  - Parameterize number, distribution (X, +, sqrt...)
    - More operators → less time, more area
    - Fewer operators → more time, less area
    - Continuum for operator allocation
- Memories for intermediate state
- Memory for "long" instructions
- General framework for specializing to problem
  - Wiring, memories get expensive
  - Opportunity for further optimizations
- General way to tradeoff area and time

Penn ESE5320 Fall 2024 -- DeHon

17

17

## VLIW



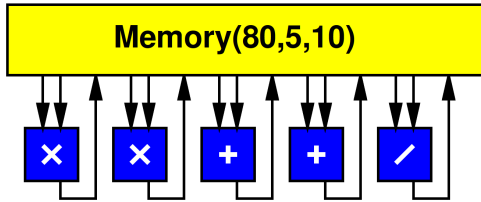
Penn ESE5320 Fall 2024 -- DeHon

18

18

## VLIW w/ Multiport RF

- Simple, full-featured model use common Register File
  - Memory(Words, WritePorts, ReadPorts)



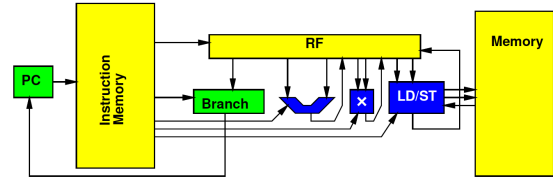
Penn ESE5320 Fall 2024 -- DeHon

19

19

## Processor Unbound

- Can (design to) use all operators at once



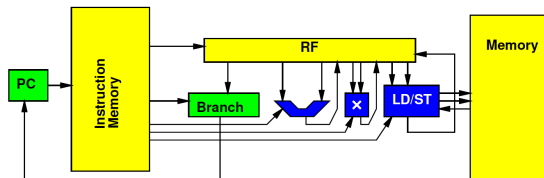
Penn ESE5320 Fall 2024 -- DeHon

20

20

## Processor Unbound

- Schedule Preclass 1 on this VLIW (next slide)



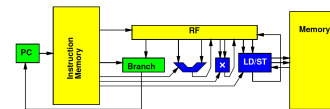
Penn ESE5320 Fall 2024 -- DeHon

21

21

## Schedule Preclass 1

Cycle	Branch	ALU	Multiply	LD/ST
0				
1				
2				
3				
4				
5				



Penn ESE5320 Fall 2024 -- DeHon

22

22

## Schedule

Cycle	Branch	ALU	Multiply	LD/ST
0		Sub r2,r1,r3		
1	Bzneq r3,end	Add r4		
2		Add r5		Ld r4,r6
3				Ld r5,r7
4		Add r1,#1,r1	Mpy r7,r8,r8	
5	B top	Add r7,r8,r8		

Penn ESE5320 Fall 2024 -- DeHon

23

23

## Resource Bound

- Can tune VLIW to directly target resource bound
  - Add operators to target a particular resource bound

Penn ESE5320 Fall 2024 -- DeHon

24

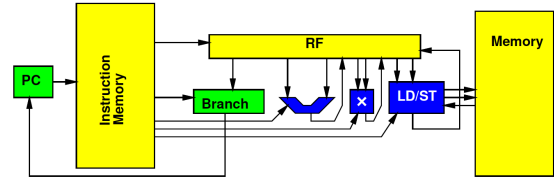
24

# Tuning, Unrolling

## Part 2

# VLIW Operator Knobs (Design Space Axes)

- Choose collection of operators and the numbers of each
  - Match task
  - Tune resources



# Schedule

- Choose collection of operators and the numbers of each
    - Match task
    - Tune resources
- What operator might we add to accelerate this loop?

Cycle	Branch	ALU	Multiply	LD/ST
0		Sub r2,r1,r3		
1	Bzneq r3,end	Add r4		
2		Add r5		Ld r4,r6
3				Ld r5,r7
4		Add r1,#1,r1	Mpy r7,r8,r8	
5	B top	Add r7,r8,r8		

# Preclass 2a

- $res[i]=sqrt(x[i]*x[i]+y[i]*y[i]+z[i]*z[i]);$
- II with one operator of each?

# Schedule

Cycle	LD	ST	Multiply	Add	incr	sqrt
0				i<MAX	&X[i]	
1	X[i]				&Y[i]	
2	Y[i]		X[i]*X[i]		&Z[i]	
3	Z[i]		Y[i]*Y[i]			
4			Z[i]*Z[i]	X <sup>2</sup> +Y <sup>2</sup>		
5				(X <sup>2</sup> +Y <sup>2</sup> )+Z <sup>2</sup>		
6						Sqrt()
7		Res[i]			i	

# Preclass 2b

- $res[i]=sqrt(x[i]*x[i]+y[i]*y[i]+z[i]*z[i]);$
- Minimum II achievable?
  - Latency lower bound

## Critical Path

1. (Increment pointers / branch)
2. Load
3. Multiplies
4. Add
5. Add
6. Squareroot
7. Writeback

Penn ESE5320 Fall 2024 -- DeHon

31

31

## Preclass 2c

- $res[i]=\sqrt{x[i]*x[i]+y[i]*y[i]+z[i]*z[i]}$ ;
- How many operators of each type to achieve minimum II (latency lower bound)?

Penn ESE5320 Fall 2024 -- DeHon

32

32

## Schedule w/ 2c Resources

	LD	LD	ST	*	*	+	i	i	sqrt
0						<	&x	&y	
1	X[i]	Y[i]						&z	
2	Z[i]			x	y				
3								x+y	
4								+z	
5									sqrt
6			Res[i]					i	

Penn ESE5320 Fall 2024 -- DeHon

33

33

## Schedule w/ 2d Resources

	LD	LD	LD	ST	*	*	*	+	i	i	i	sqrt
0								<	&x	&y	&z	
1	X[i]	Y[i]	Z[i]									
2					x	y	z					
3										X+y		
4										+z		
5												sqrt
6				Res[i]						i		

- What is disappointing about this schedule?

Penn ESE5320 Fall 2024 -- DeHon

34

34

## Preclass 2d

- $res[i]=\sqrt{x[i]*x[i]+y[i]*y[i]+z[i]*z[i]}$ ;
- $res[i+1]=\sqrt{x[i+1]*x[i+1]+y[i+1]*y[i+1]+z[i+1]*z[i+1]}$ ;
- $res[i+2]=\sqrt{x[i+2]*x[i+2]+y[i+2]*y[i+2]+z[i+2]*z[i+2]}$ ;
- $res[i+3]=\sqrt{x[i+3]*x[i+3]+y[i+3]*y[i+3]+z[i+3]*z[i+3]}$ ;

- Schedule (next slide)

Penn ESE5320 Fall 2024 -- DeHon

35

35

## Unroll 4 Schedule

	LD	LD	LD	ST	*	*	*	+	+	i	i	i	sqrt
0													
1													
2													
3													
4													
5													
6													
7													
8													
9													

Penn ESE5320 Fall 2024 -- DeHon

36

36

## Time Points

- 4 iterations in 10 cycles = 2.5 cycles/iter
- Compared to 1 iteration in 7
- Compared to 1 iteration in 8

Penn ESE5320 Fall 2024 -- DeHon

37

37

## Unroll 4

	LD	LD	LD	ST	*	*	*	+	+	i	i	i	sqrt
0								<		x0	y0	z0	
1	x0	y0	z0							x1	y1	z1	
2	x1	y1	z1		x0	y0	z0			x2	y2	z2	
3	x2	y2	z2		x1	y1	z1	xy0		x3	y2	z3	
4	x3	y2	z3		x2	y2	z2	xy1	+z0				
5					x3	y2	z3	xy2	+z1				0
6				0				xy3	+z2				1
7				1					+z3				2
8				2									3
9				3						i			

What happens to II (cycle/iter) as unroll more?

Penn ESE5320 Fall 2024 -- DeHon

38

38

## Observation

- With no data dependencies
  - Unroll gets closer to resource bound

Penn ESE5320 Fall 2024 -- DeHon

39

39

## Preclass 2e

- $res[i]=sqrt(x[i]*x[i]+y[i]*y[i]+z[i]*z[i]);$
- Area comparison?

Op	read	write	*	+	incr	sqrt	Area	II
A	20	30	10	2	1	25		
2a	1	1	1	1	1	1		8
2c								7
2d	3	1	3	2	3	1		2.5

Penn ESE5320 Fall 2024 -- DeHon

40

40

## Take Aways

- VLIW: Tune Operators to select Area-Time point
- Running single iteration of loop to completion before start next
  - Inefficient
  - Latency bound for loop
  - Benefit schedule multiple loop bodies
    - Can get closer to resource bound

Penn ESE5320 Fall 2024 -- DeHon

41

41

## Data Storage and Movement

### Part 3

Penn ESE5320 Fall 2024 -- DeHon

42

42

## Multiport RF

- Multiported memories are expensive
  - Need input/output lines for each port
  - Makes large, slow
- Simplified preclass model:
  - $\text{Area}(\text{Memory}(n,w,r)) = n \cdot (w+r+1)/2$

Penn ESE5320 Fall 2024 -- DeHon

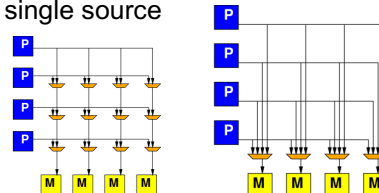
43

43

Day 12

## Alternate: Crossbar (Xbar)

- Provide programmable connection between all sources and destinations
- Any destination can be connected to any single source

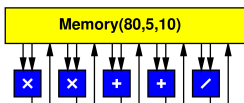


Penn ESE5320 Fall 2024 -- DeHon

44

44

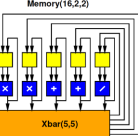
## Preclass 3



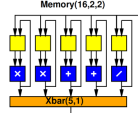
- Areas? (table)
- How does area of memories, xbar compare to datapath operators in each case?

	Multiport RF	Split RF Xbar(5,5)	Split RF Xbar(5,1)
Memory			
Operators			
Xbar			
Total			

Each of 5 memories is: Memory(16,2,2)



Each of 5 memories is: Memory(16,2,2)



Penn ESE5320 Fall 2024 -- DeHon

45

45

## Split RF Cheaper

- At same capacity, split register file cheaper
  - $2W+2R \rightarrow 2.5$  per word
  - $5W+10R \rightarrow 8$  per word

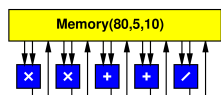
Penn ESE5320 Fall 2024 -- DeHon

46

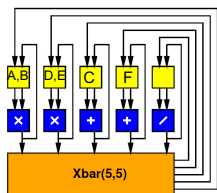
46

## Split RF Full Crossbar

- Cycles each for computing:  $(A*B+C)/(D*E+F)$ 
  - Assume A..F start as shown
  - Hint: left first
    - How work on right?



Each of 5 memories is: Memory(16,2,2)



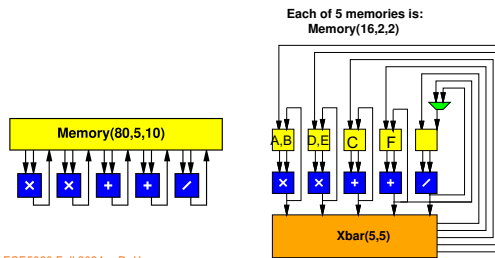
Penn ESE5320 Fall 2024 -- DeHon

47

47

## Split RF Full Crossbar

- Specialize connections to avoid bottleneck



Penn ESE5320 Fall 2024 -- DeHon

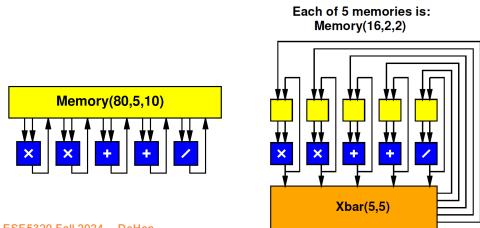
48

48



## VLIW Memory Tuning

- Can select how much sharing or independence in local memories



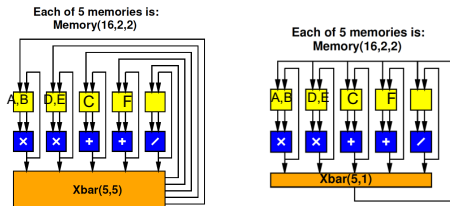
Penn ESE5320 Fall 2024 -- DeHon

49

49

## Split RF, Limited Crossbar

- What limitation does the one crossbar output pose?
  - Cycles for same task:  $(A*B+C)/(D*E+F)$



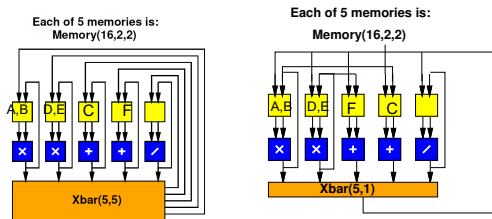
Penn ESE5320 Fall 2024 -- DeHon

50

50

## Split RF, Limited Crossbar

- Could specialize connections for computation



Penn ESE5320 Fall 2024 -- DeHon

51

51

## VLIW Schedule

Need to schedule Xbar output(s) as well as operators.

cycle	*	*	+	+	/	Xbar
0						
1						
2						
3						
4						

Penn ESE5320 Fall 2024 -- DeHon

52

52

## VLIW Interconnect Tuning

- Can decide how rich to make the interconnect
  - Number of outputs to support
  - How to depopulate (restrict) crossbar
  - Use more restricted network

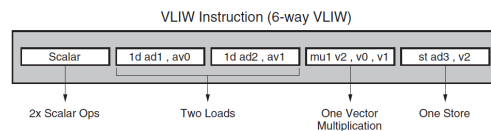
Penn ESE5320 Fall 2024 -- DeHon

53

53

## Commercial: Xilinx AI Engine

- 6-way VLIW Vector



[https://www.xilinx.com/support/documentation/white\\_papers/wp506-ai-engine.pdf](https://www.xilinx.com/support/documentation/white_papers/wp506-ai-engine.pdf)

Xilinx WP506

Penn ESE5320 Fall 2024 -- DeHon

54

54

## VLIW vs. SuperScalar

- Modern, high-end proc. (incl. ARM on Zynq)
  - Do support ILP
  - Issue multiple instructions per cycle
  - ...but, from a single, sequential instruction stream
- SuperScalar – dynamic issue and interlock on data hazards – hide # operators
  - Must have shared, multiport RF
- VLIW – offline scheduled (like done today)
  - No interlocks, allow split RF
  - Lower area/operator – need to recompile code

Penn ESE5320 Fall 2024 -- DeHon

55

55

## Big Ideas:

- VLIW as a Model for
  - Instruction-Level Parallelism (ILP)
  - Customizing Datapaths
  - Area-Time Tradeoffs
  - Real-Time, Scheduled Sequentialization
- Customize VLIW
  - Operator selection
  - Memory/register file setup
  - Inter-functional unit communication network

Penn ESE5320 Fall 2024 -- DeHon

56

56

## Admin

- Feedback (including P3)
- Reading for Wednesday online
- P4 due Friday

Penn ESE5320 Fall 2024 -- DeHon

57

57

## Loop Overhead

Part 4: Bonus slides:  
not expect to cover in lecture

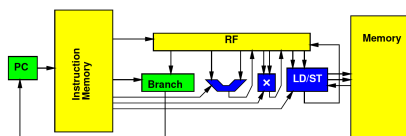
Penn ESE5320 Fall 2024 -- DeHon

58

58

## Loop Overhead

- Can handle loop overhead in ILP on VLIW
  - Increment counters, branches as independent functional units



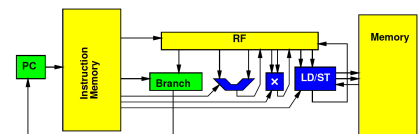
Penn ESE5320 Fall 2024 -- DeHon

59

59

## VLIW Loop Overhead

- Can handle loop overhead in ILP on VLIW
- ...but paying a full issue unit and instruction costs overhead



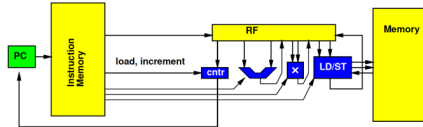
Penn ESE5320 Fall 2024 -- DeHon

60

60

## Zero-Overhead Loops

- Specialize the instructions, state, branching for loops
  - Counter rather than RF
  - One bit to indicate if counter decrement
  - Exit loop when decrement to 0

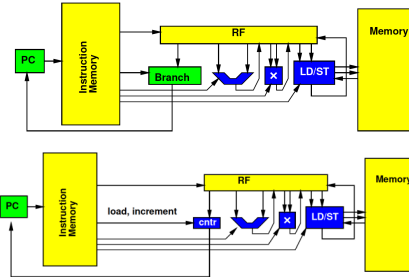


Penn ESE5320 Fall 2024 -- DeHon

61

61

## Simplification



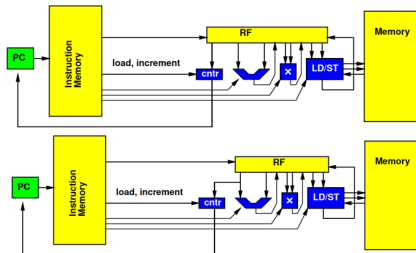
Penn ESE5320 Fall 2024 -- DeHon

62

62

## Zero-Overhead Loop Simplify

- Share port – simplify further



Penn ESE5320 Fall 2024 -- DeHon

63

63

## Zero-Overhead Loop Example (preclass 1)

repeat r3:

```

addi r4,#4,r4;
addi r5,#4,r5; ld r4,r6
ld r5,r7
mul r6,r7,r7
add r7,r8,r8
    
```

Penn ESE5320 Fall 2024 -- DeHon

64

64

## Zero-Overhead Loop

- Potentially generalize to multiple loop nests and counters
- Common in highly optimized DSPs, Vector units
- Can think about design space options within this VLIW model framework

Penn ESE5320 Fall 2024 -- DeHon

65

65

## Pipelined VLIW Operators

Penn ESE5320 Fall 2024 -- DeHon

66

66

## Pipelined Operators

- Often will have pipelined operators
  - E.g. 3 cycles multiply
- How complicate?

Penn ESE5320 Fall 2024 -- DeHon

67

67

## Accommodating Pipeline

- Schedule for when data becomes available
  - Dependencies
  - Use of resources

cycle	*	+	+	+	/	Xbar
0	X*X					
1	Y*Y					
2						X*X
3						Y*Y
4			X <sup>2</sup> +Y <sup>2</sup>			X <sup>2</sup> +Y <sup>2</sup>
5					X <sup>2</sup> +Y <sup>2</sup> / Z	
6						

Penn ESE5320 Fall 2024

68

68

## Accommodating Pipeline

- Schedule for when data becomes available
    - Dependencies
    - Use of resources
- Impossible schedule; Conflict on single Xbar output

cycle	*	+	+	+	/	Xbar
0	X*X					
1	Y*Y					
2						X*X
3			Q+R			Y*Y, Q+R
4			X <sup>2</sup> +Y <sup>2</sup>			X <sup>2</sup> +Y <sup>2</sup>
5					X <sup>2</sup> +Y <sup>2</sup> / Z	
6						

Penn ESE5320 Fall 2024

69

69

## Pipelined Operators

- Instance of what we saw last time
- Need to schedule use on all resources
  - Memory port
  - Communications (busses, crossbars ports)
  - I/O

Penn ESE5320 Fall 2024 -- DeHon

70

70