QR Arrays and Schedules

Joseph Cavallaro
Rice University
25 October 2022

Last Lecture

- Vitis HLS Data Types
- Notion of Data Flow and pipeline scheduling – Wolf Chapter 6
- Review of CORDIC and intro to QR
- QR Decomposition array structures

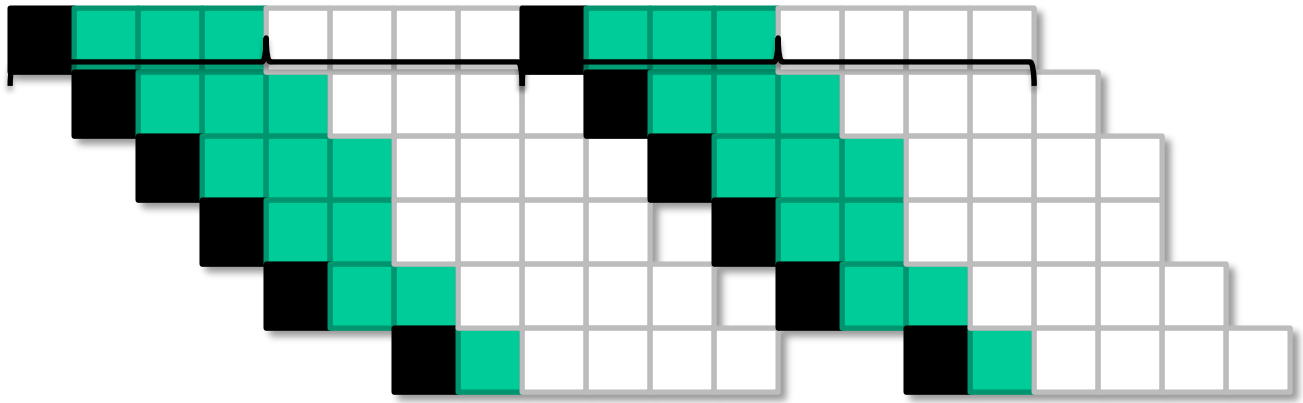
Today

- QR Arrays
  - ⇒ Aspects of scheduling with focus on some of the systolic design issues
- This flow is easier to capture in a block diagram tool such as Model Composer
- As we look at the Vitis HLS tools it become more difficult to create a custom schedule.

Recall: Triangular Systolic-Array for Givens QR

3			
2	5		
1	4	6	

nth iteration (n+1)th iteration



Review of QR Decomposition

- Total 6 multiplications to needed to produce 6 zero entries.

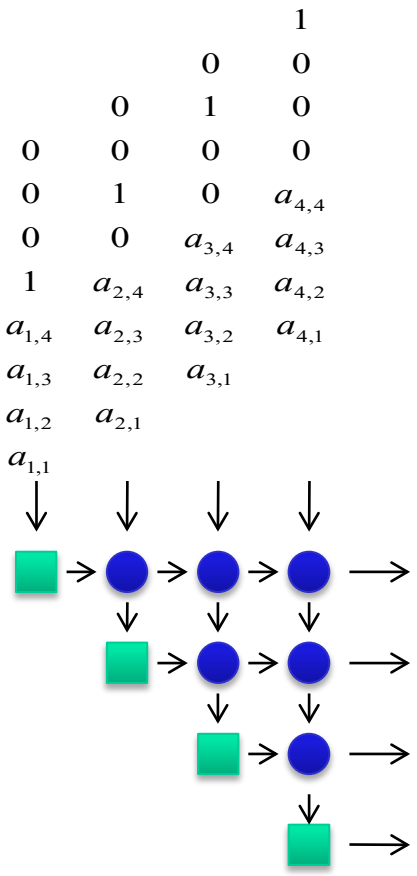
$$A = QR$$

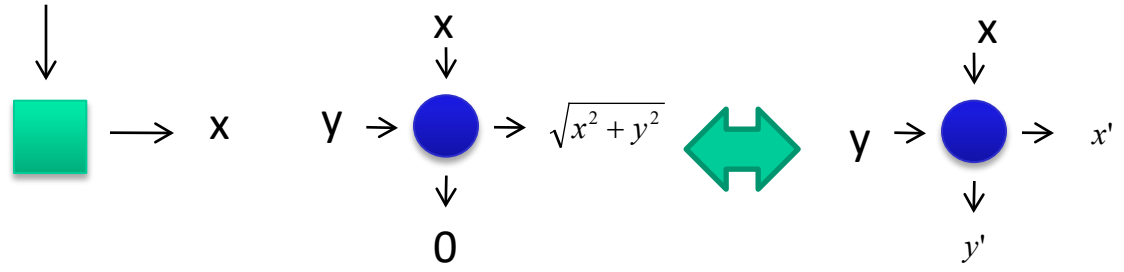
$$A = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ 0 & r_{22} & r_{23} & r_{24} \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & r_{44} \end{bmatrix}$$

3 [2x2] * [2*4] matrix multiplications
2 [2x2] * [2*3] matrix multiplications
1 [2x2] * [2*2] matrix multiplications

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \end{pmatrix} = \begin{pmatrix} \sqrt{{a_{1,1}}^2 + {a_{2,1}}^2} & a_{1,2}' & a_{1,3}' & a_{1,4}' \\ 0 & a_{2,2}' & a_{2,3}' & a_{2,4}' \end{pmatrix}$$

Triangular Systolic-Array for Givens QR

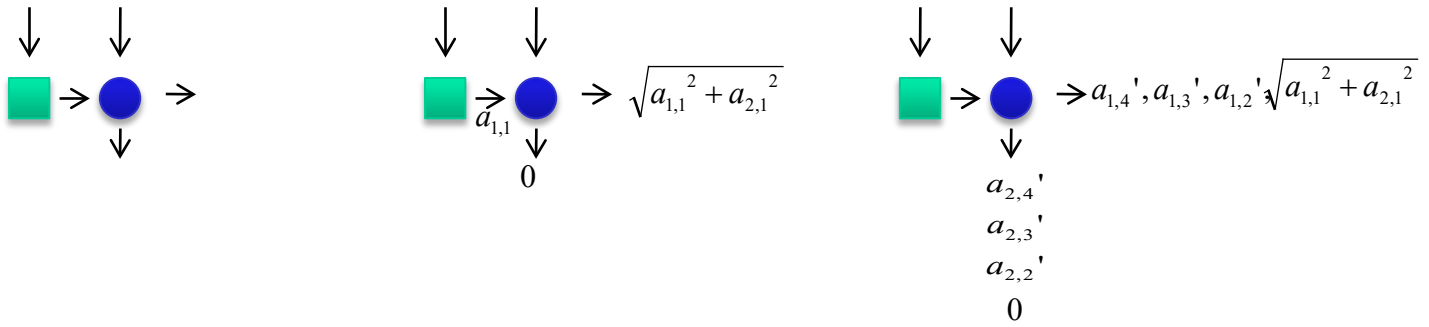




$$\begin{matrix} q_{1,4} & q_{1,3} & q_{1,2} & q_{1,1} & r_{1,4} & r_{1,3} & r_{1,2} & r_{1,1} \\ q_{2,4} & q_{2,3} & q_{2,2} & q_{2,1} & r_{2,4} & r_{2,3} & r_{2,2} & 0 \\ q_{3,4} & q_{3,3} & q_{3,2} & q_{3,1} & r_{3,4} & r_{3,3} & 0 & 0 \\ q_{4,4} & q_{4,3} & q_{4,2} & q_{4,1} & r_{4,4} & 0 & 0 & 0 \end{matrix}$$

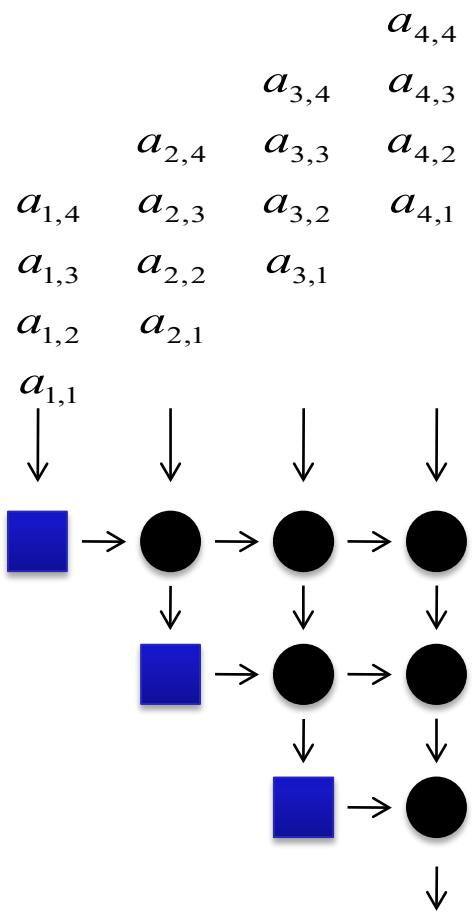
Triangular Systolic-Array for Givens QR

- Each processing element
  - ⇒ Has two modes, ATAN and SINCOS
  - ⇒ Does a matrix multiply



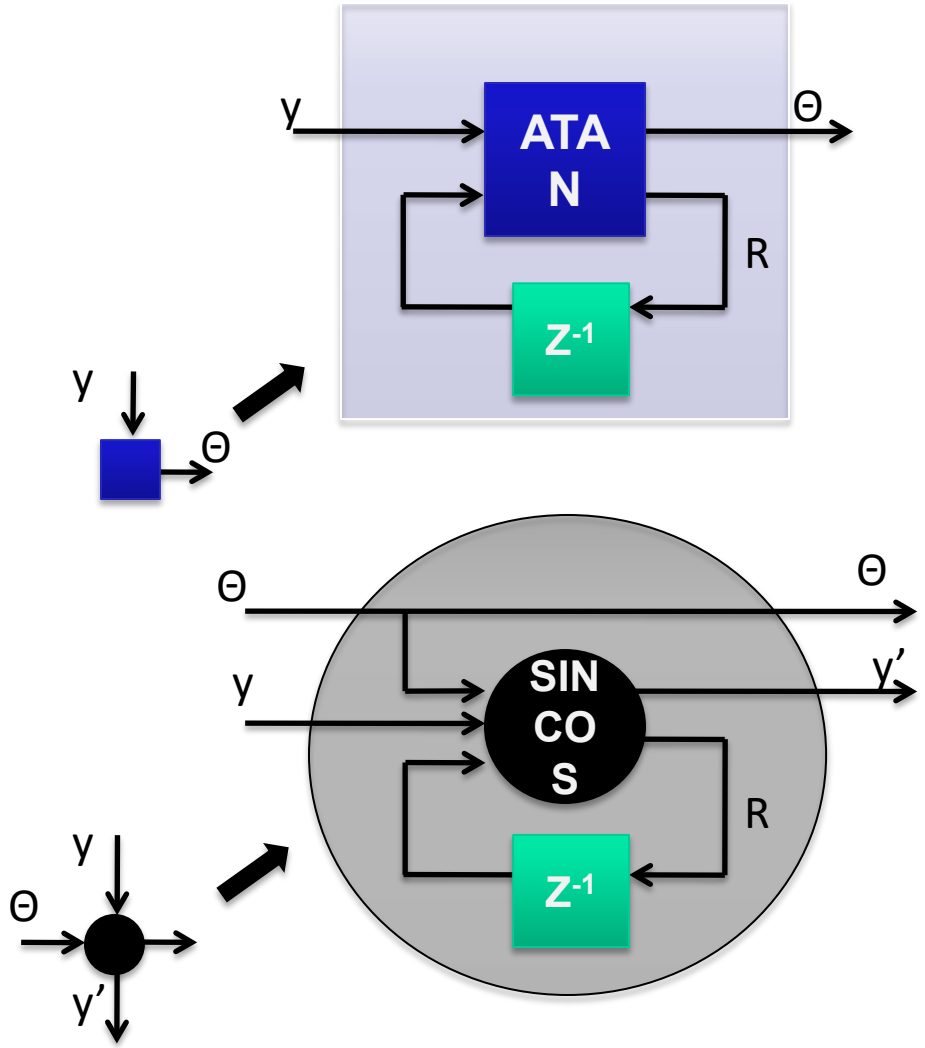
$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \end{pmatrix} = \begin{pmatrix} \sqrt{a_{1,1}^2 + a_{2,1}^2} & a_{1,2}' & a_{1,3}' & a_{1,4}' \\ 0 & a_{2,2}' & a_{2,3}' & a_{2,4}' \end{pmatrix}$$

Alternative Triangular Systolic Array



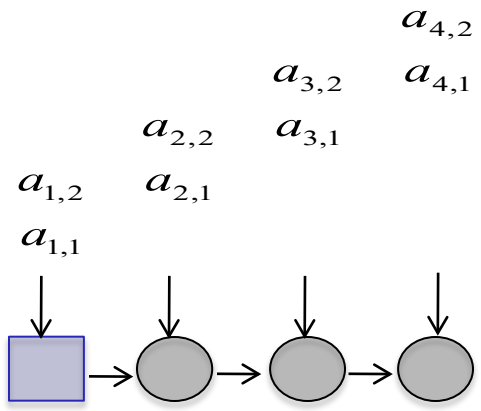
- Each processing element is a functional unit–a vectoring (ATAN) unit ■ or rotation unit (SIN/COS) ●
- The upper triangular R is stored in the PE and an unload stage is needed for this design.

Alternative Triangular Systolic Array

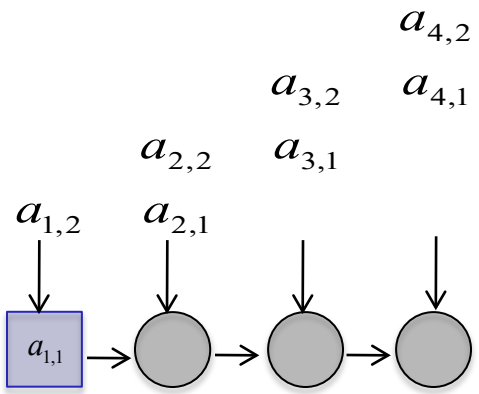


- For both blocks
  - ⇒ Store the first valid input in the internal register
  - ⇒ For each subsequent input, do the operation and store the result R
  - ⇒ Pass the other calculated output to the next block

Alternative Triangular Systolic Array



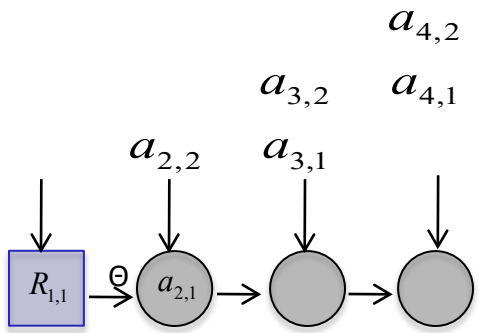
Alternative Triangular Systolic Array



Doing this operation, vectoring

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a_{1,1} \\ a_{1,2} \end{pmatrix} = \begin{pmatrix} \sqrt{{a_{1,1}}^2 + {a_{2,1}}^2} \\ 0 \end{pmatrix} = \begin{pmatrix} R_{1,1} \\ 0 \end{pmatrix}$$

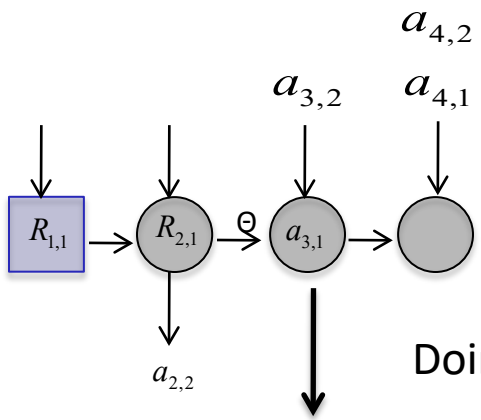
Alternative Triangular Systolic Array



Doing this operation, rotating

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a_{2,1} \\ a_{2,2} \end{pmatrix} = \begin{pmatrix} R_{2,1} \\ a_{2,2}' \end{pmatrix} = \begin{pmatrix} R_{2,1} \\ a_{2,2}' \end{pmatrix}$$

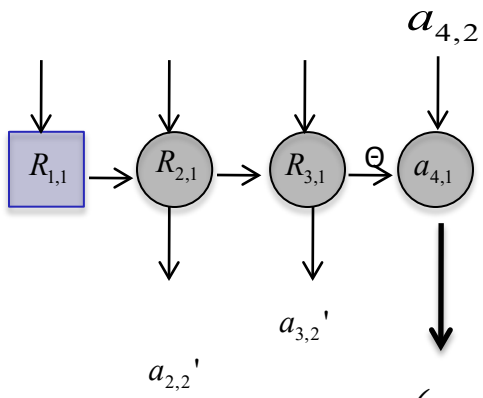
Alternative Triangular Systolic Array



Doing this operation, rotating

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a_{3,1} \\ a_{3,2} \end{pmatrix} = \begin{pmatrix} R_{3,1} \\ a_{3,2}' \end{pmatrix} = \begin{pmatrix} R_{3,1} \\ a_{3,2}' \end{pmatrix}$$

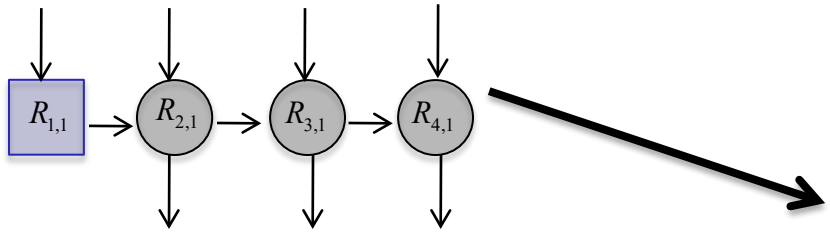
Alternative Triangular Systolic Array



Doing this operation, rotating

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a_{4,1} \\ a_{4,2} \end{pmatrix} = \begin{pmatrix} R_{4,1} \\ a_{4,2}' \end{pmatrix} = \begin{pmatrix} R_{4,1} \\ a_{4,2}' \end{pmatrix}$$

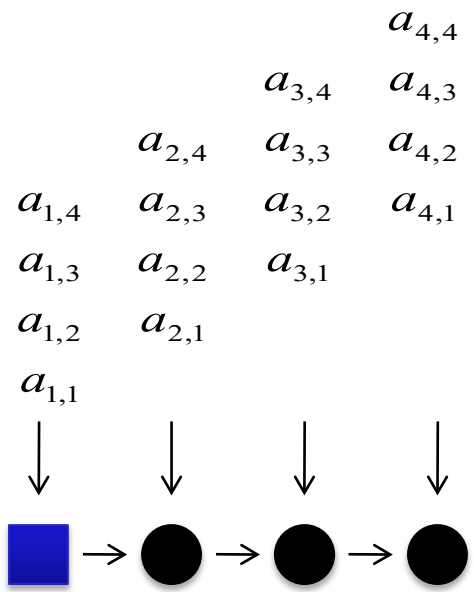
Alternative Triangular Systolic Array



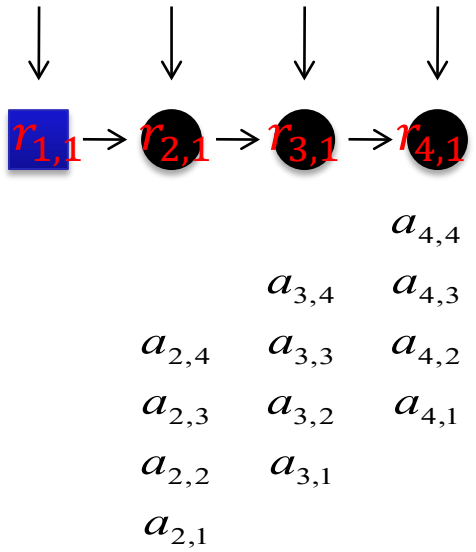
A row of PE does a matrix multiplication

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a_{1,1} & a_{2,1} & a_{3,1} & a_{4,1} \\ a_{1,2} & a_{2,2} & a_{3,2} & a_{4,2} \end{pmatrix} = \begin{pmatrix} R_{1,1} & R_{2,1} & R_{3,1}' & R_{4,1}' \\ 0 & a_{2,2}' & a_{3,2}' & a_{4,2}' \end{pmatrix}$$

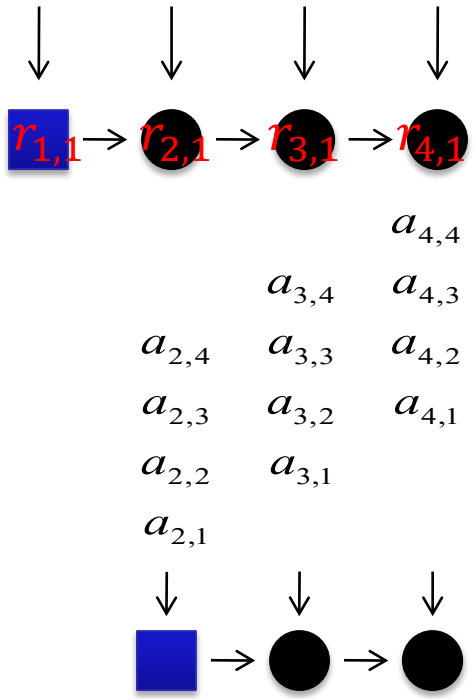
Alternative Triangular Systolic Array



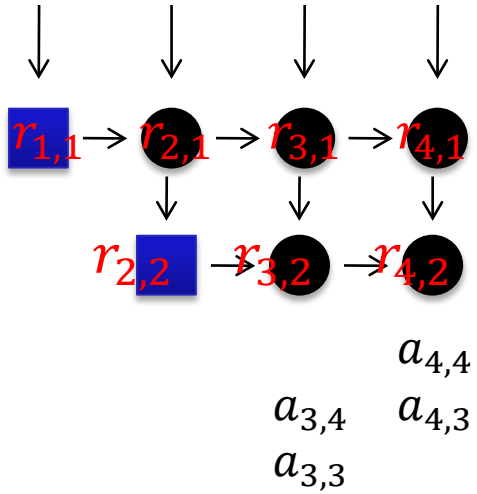
Alternative Triangular Systolic Array



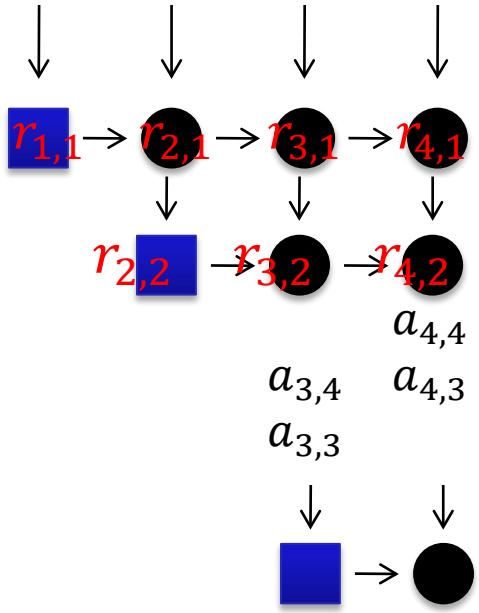
Alternative Triangular Systolic Array



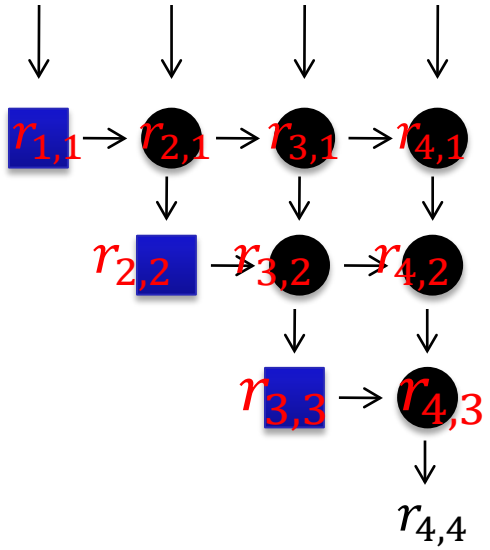
Alternative Triangular Systolic Array



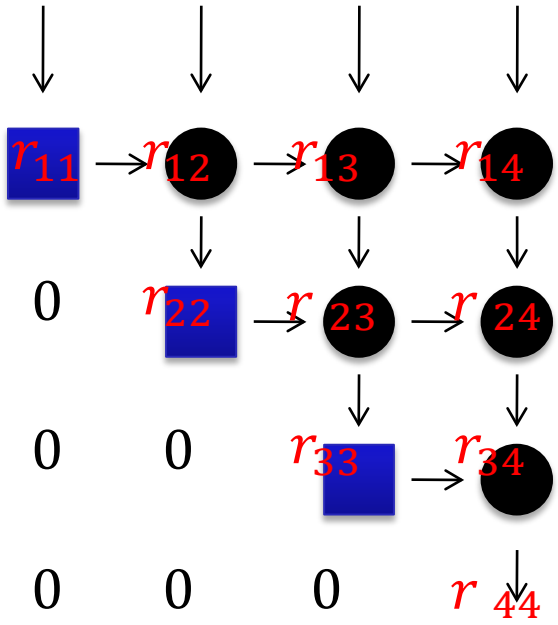
Alternative Triangular Systolic Array



Alternative Triangular SystolicArray



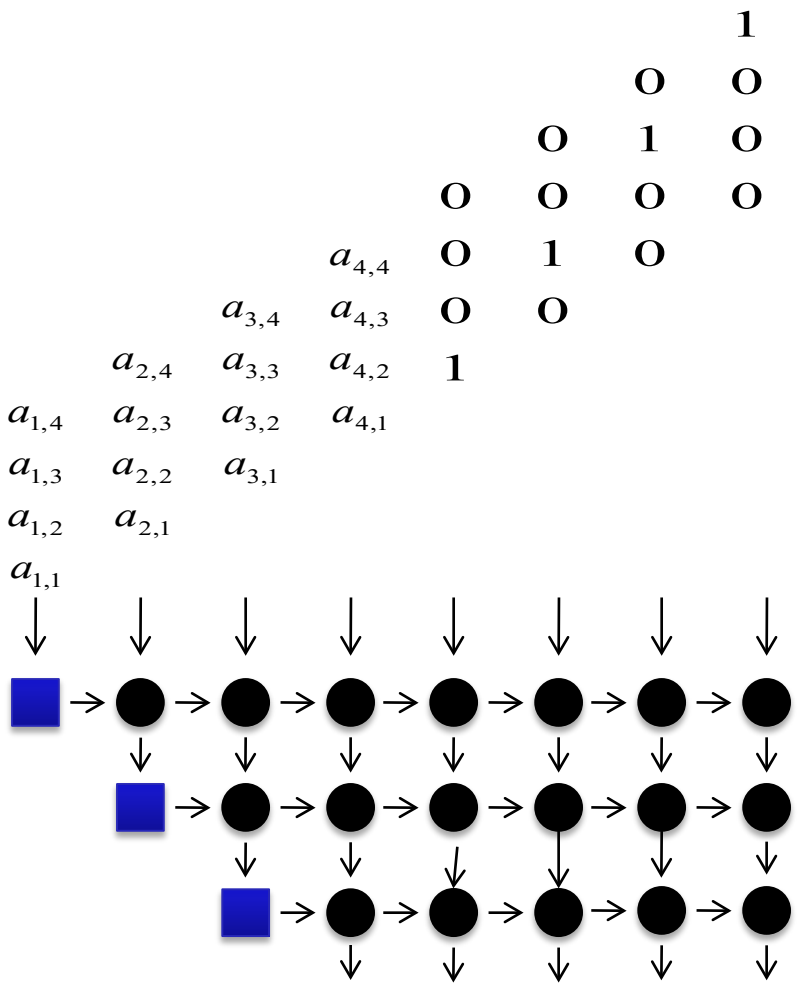
Alternative Triangular Systolic Array



Needs an unload stage!

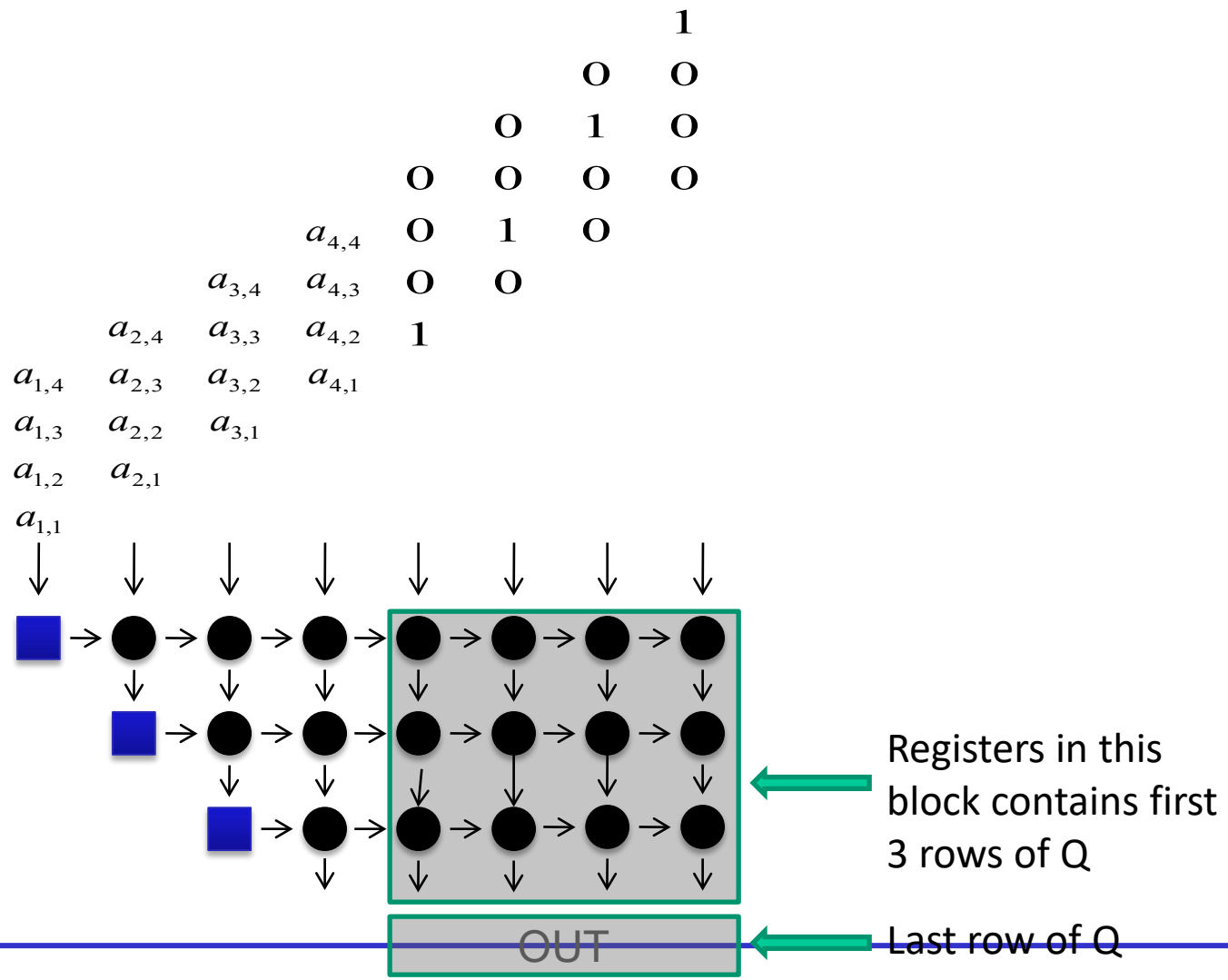
Alternative Triangular Systolic Array

- How do get Q?



Alternative Triangular Systolic Array

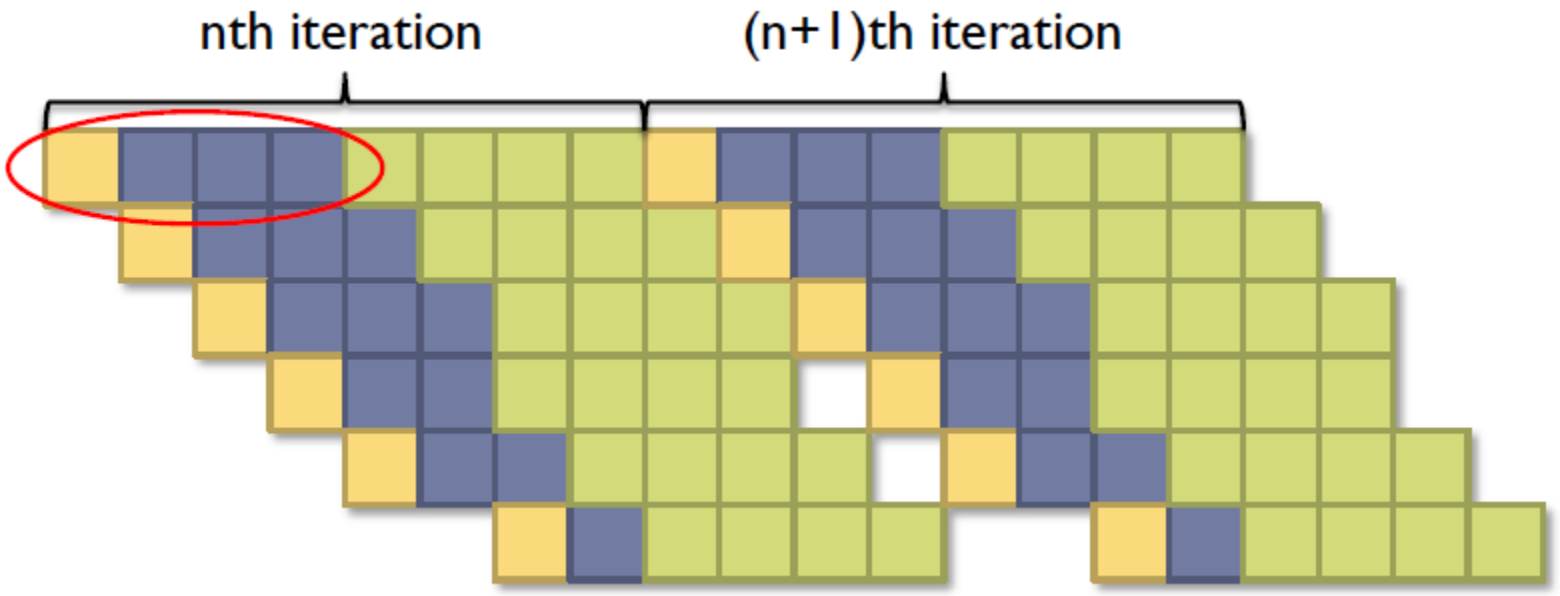
- How do get Q?



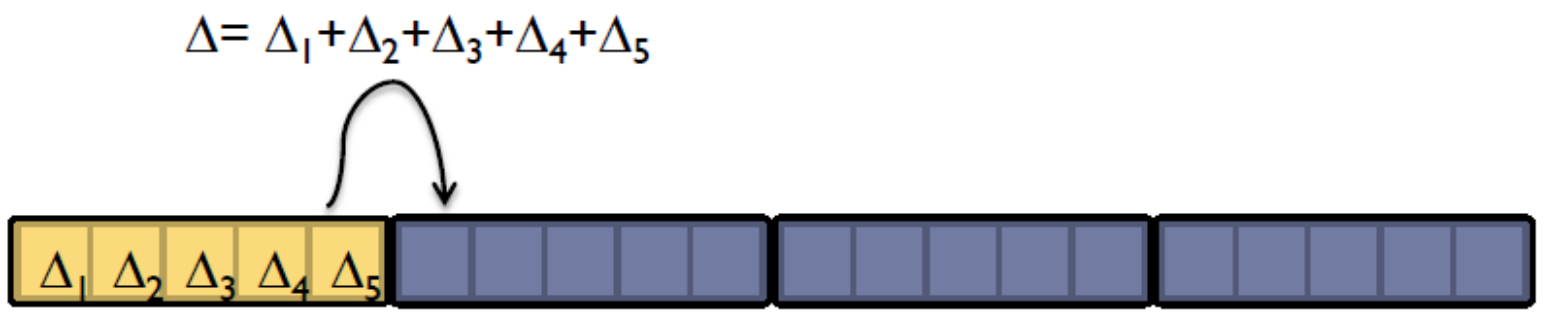
Additional Notes on Micro-Pipelining

- Can we better interface the individual steps in a CORDIC block to pass “micro-rotation” direction vectors instead of full complete “angles”
- Can save the separation of angles into sub-steps or “micro-rotations” at each CORDIC Module

Triangular Systolic-Array for Givens QR

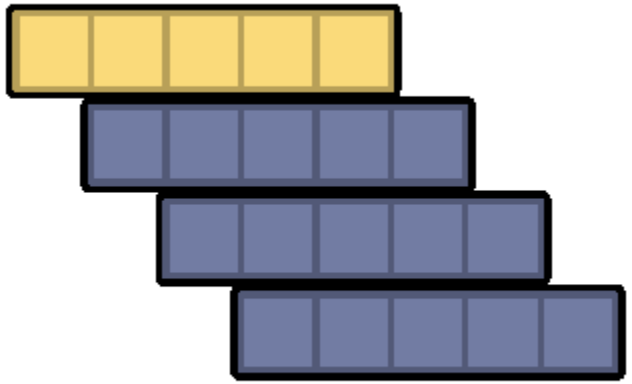


CORDIC Micro-rotations Pipeline

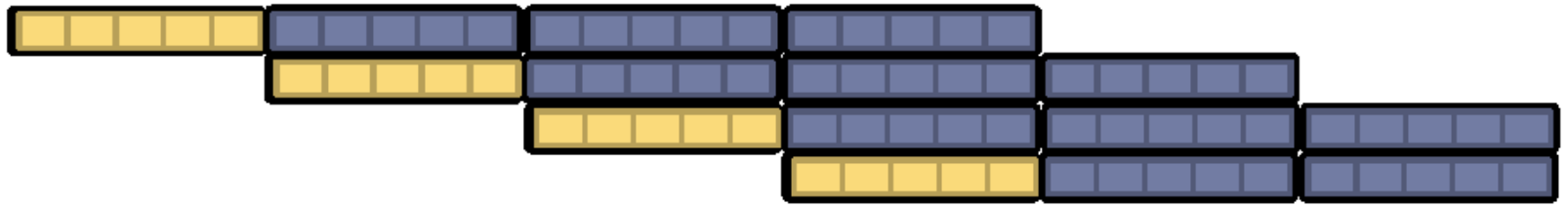


CORDIC Micro-rotations Pipeline

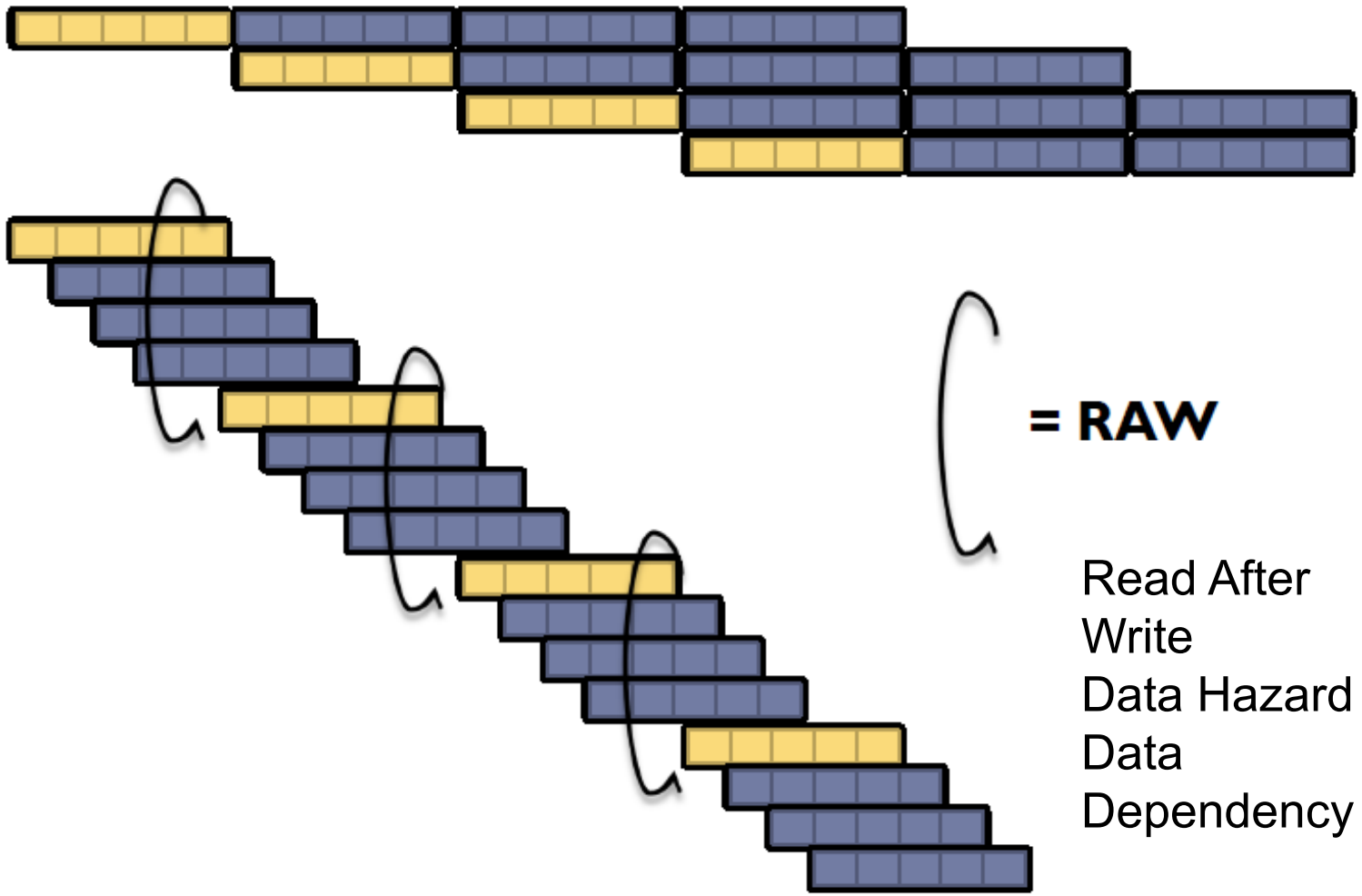
CORDIC Micro-rotations Pipeline



CORDIC Micro-rotations Pipeline



CORDIC Micro-rotations Pipeline

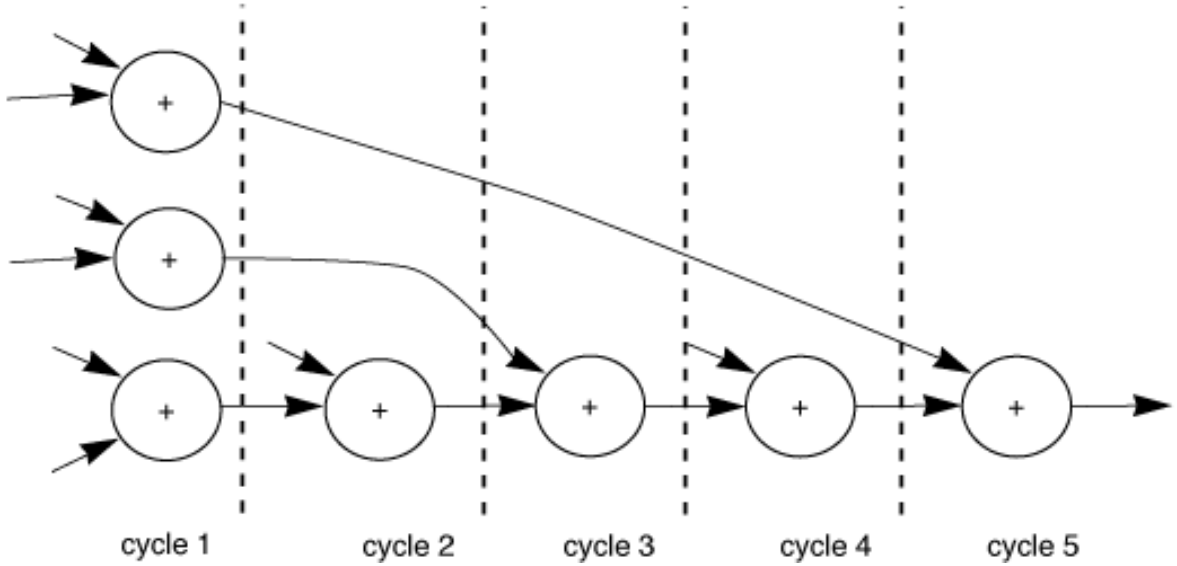


Finding schedules – From Wolf Book Chapter 6

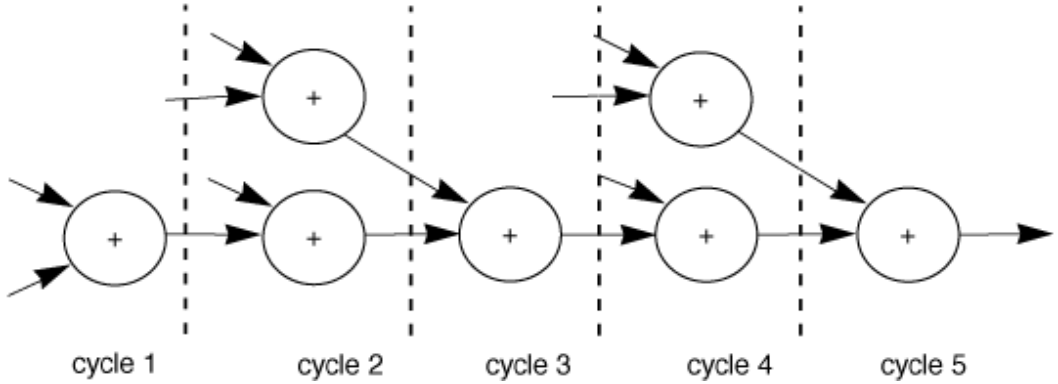
- Two simple schedules:
  - ⇒ As-soon-as-possible (ASAP) schedule puts every operation as early in time as possible.
  - ⇒ As-late-as-possible (ALAP) schedule puts every operation as late in schedule as possible.
- Many schedules exist between ALAP and ASAP extremes.

ASAP and ALAP schedules

ASAP

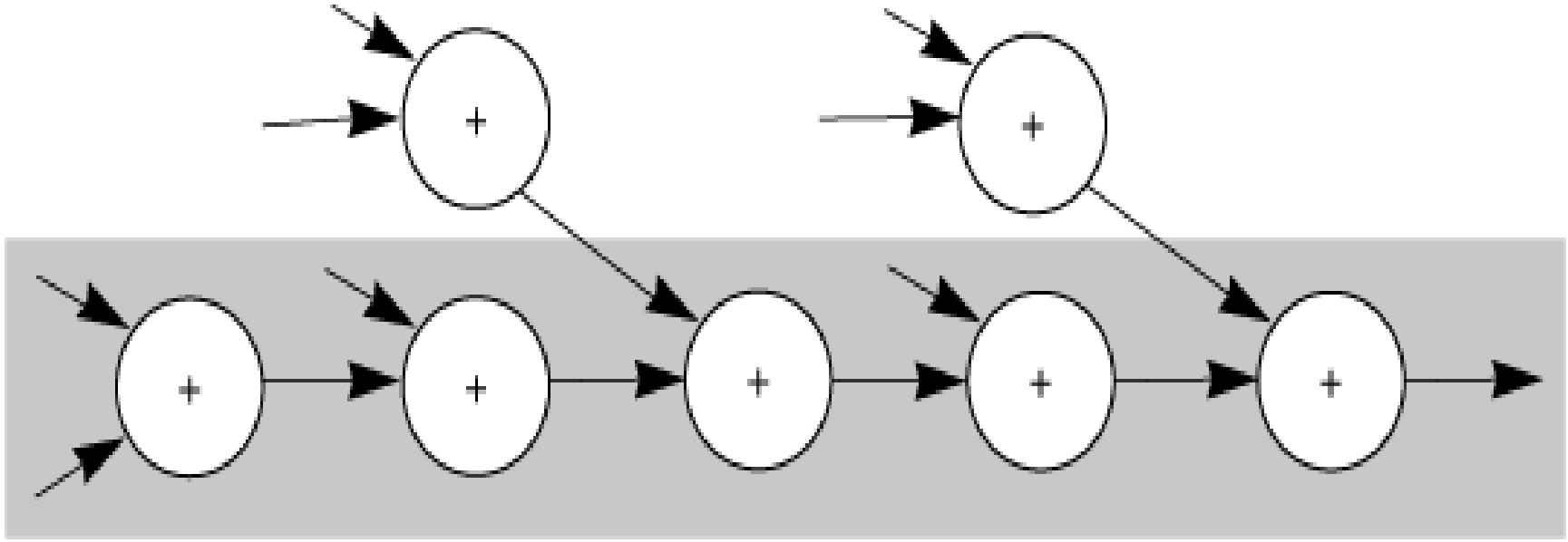


ALAP



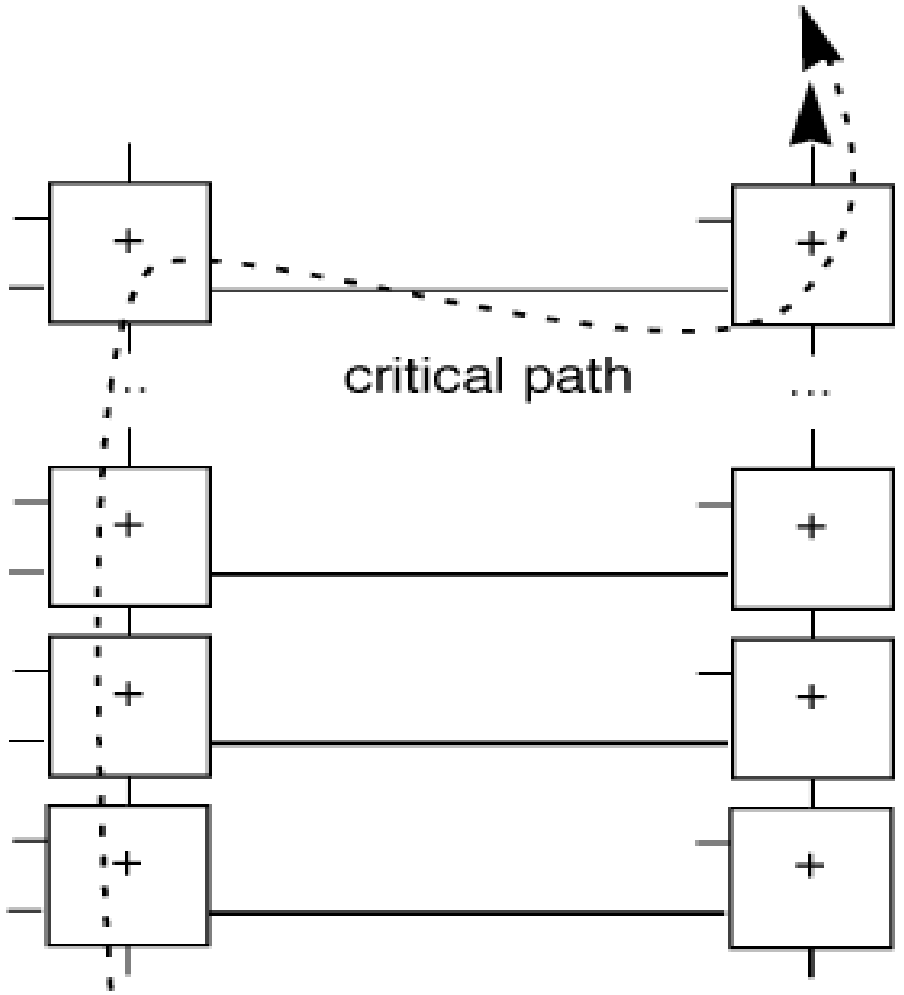
Critical path of schedule

Longest path through data flow determines minimum schedule length:



Operator chaining

- May execute several operations in sequence in one cycle—operator chaining.
- Delay through function units may not be additive, such as through several adders.



Control implementation

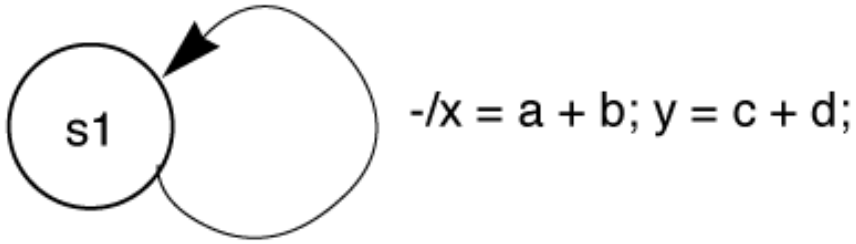
- Clock cycles are also known as control steps.
- Longer schedule means more states in controller.
- Cost of controller may be hard to judge from casual inspection of state transition graph.

Controllers and scheduling

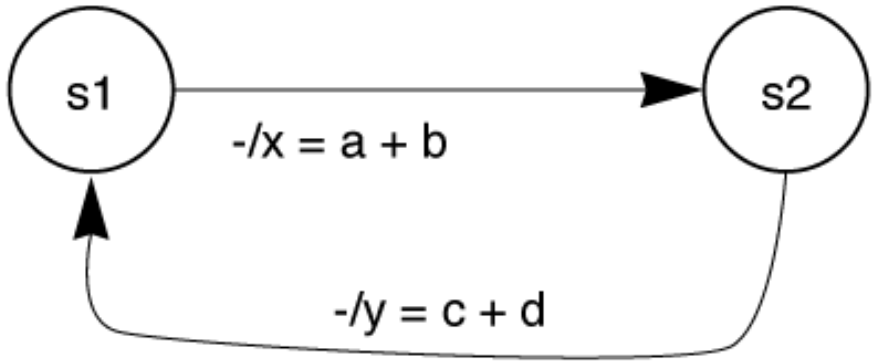
functional model:

x <= a + b;

y <= c + d;

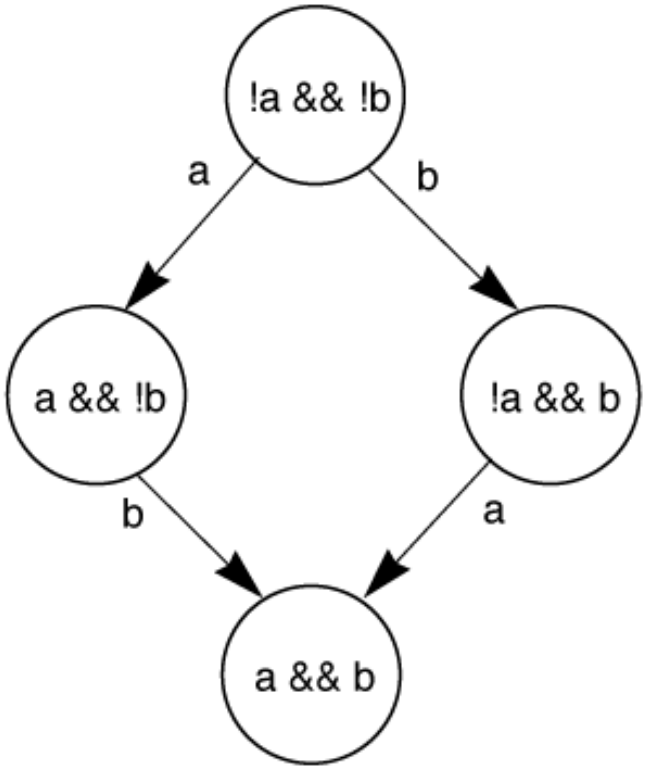


one state

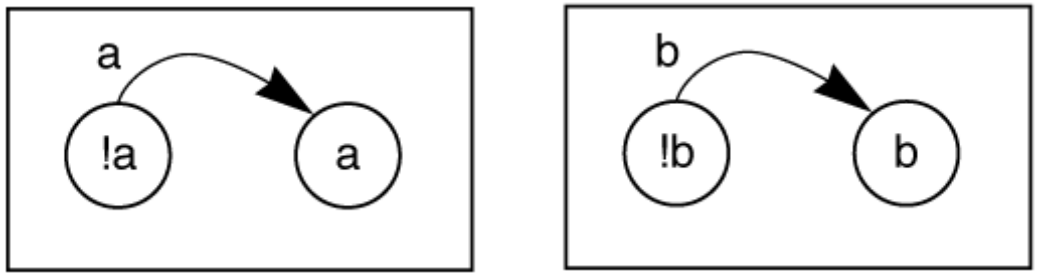


two states

Distributed control



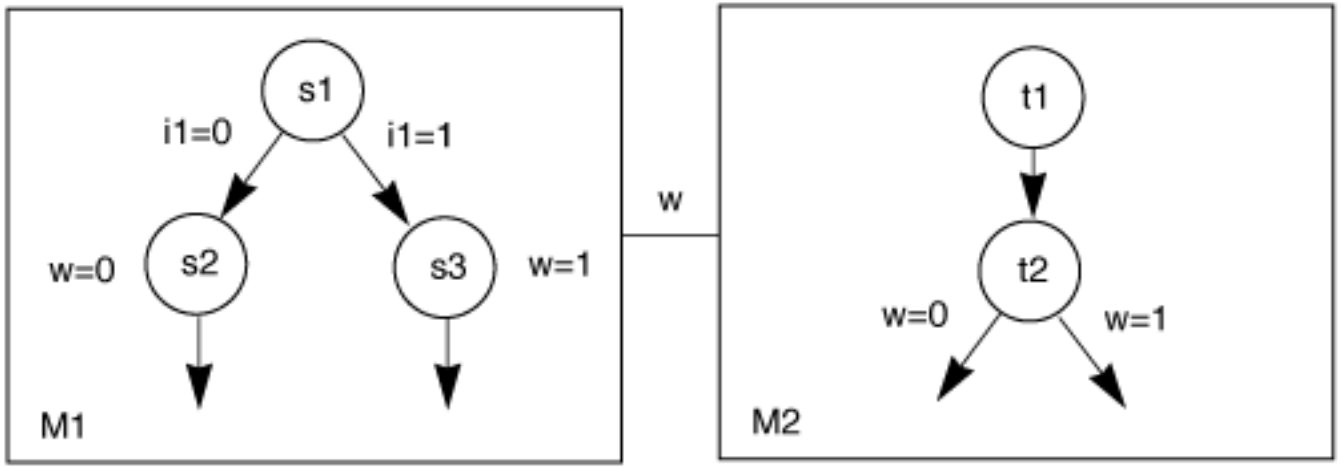
one centralized controller



two distributed controllers

Synchronized communication between FSMs

To pass values between two machines, must schedule output of one machine to coincide with input expected by the other:

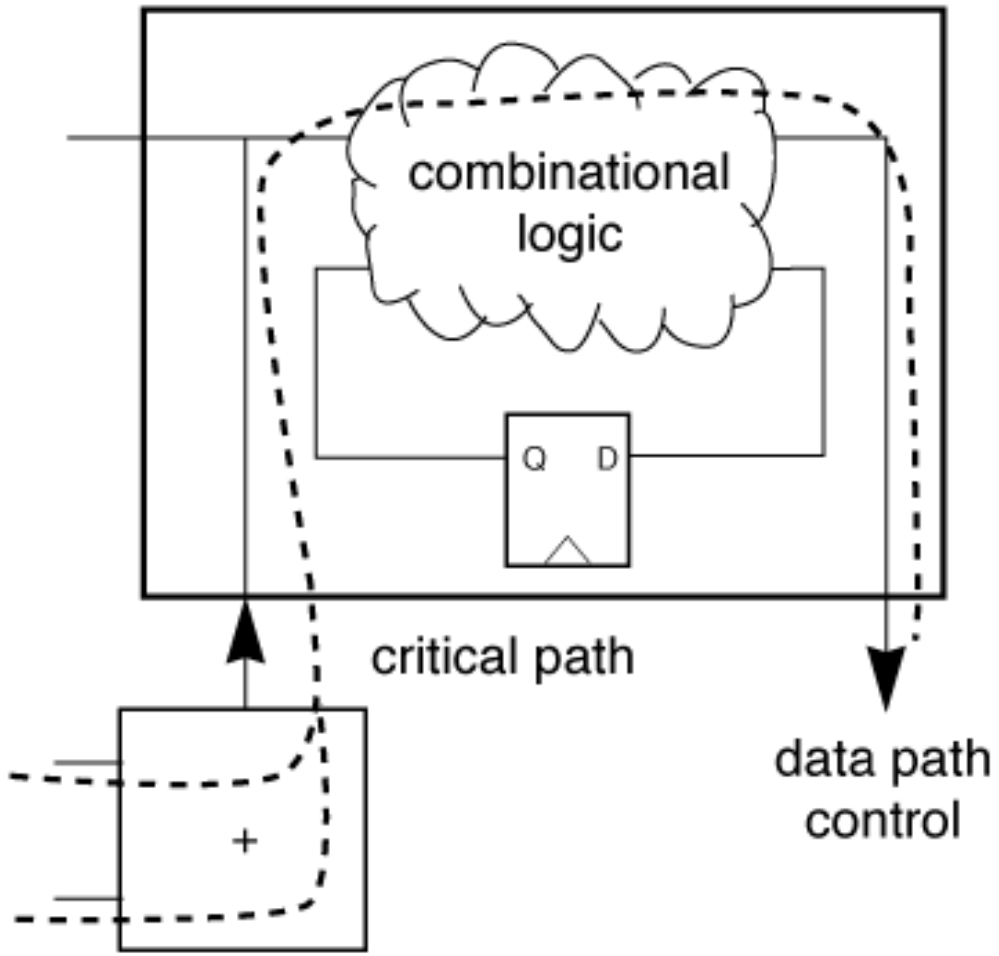


Hardwired vs. microcoded control

- Hardwired control has a state register and “random logic.”
- A microcoded machine has a state register which points into a microcode memory.
- Styles are equivalent; choice depends on implementation considerations.

Data path-controller delay

Watch out for long delay paths created by combination of data path and controller:



Next Lecture

- Project 4 CORDIC discussion on ARM integration and control