Design of Digital Circuits and Systems

Algorithms to Hardware I

Instructor: Vikram Iyer

Teaching Assistants:

Ariel Kao

Selim Saridede

Derek Thorp

Josh Wentzien

Jared Yoder

Adapted from material by Justin Hisa

Relevant Course Information

- Lab 3 reports due Friday (5/2)
- Lab 4 due next Friday (5/9)
- hw4 due on Wednesday (5/7)
- Anonymous mid-quarter survey on Canvas (due 5/5)
- Quiz 3 (ASM, ASMD) next Thursday (5/8)

Arithmetic Mean

- Design a sequential circuit that computes the mean M of k n-bit numbers stored in registers
  - *e.g.*, accessing a RAM or register file with k addresses
  - To save on hardware, you can only use one n-bit adder and have a single read port RAM

- Algorithm Pseudocode:

(1)
```
total = 0
for i = 0 to k-1
    total = total + R[i]
end for
M = total/k
```

(2)
```
for i = k-1 to 0
    total = total + R[i]
end for
M = total/k
```

Aside: Counter Variable

- Many sequential hardware algorithms utilize counters
- If both work, is there a preference?
  - How to implement $C = k - 1$ check?

a	b	xnor
0	0	1
0	1	0
1	0	0
1	1	1

  - How to implement $C = 0$ check?

a	b	nor
0	0	1
0	1	0
1	0	0
1	1	0

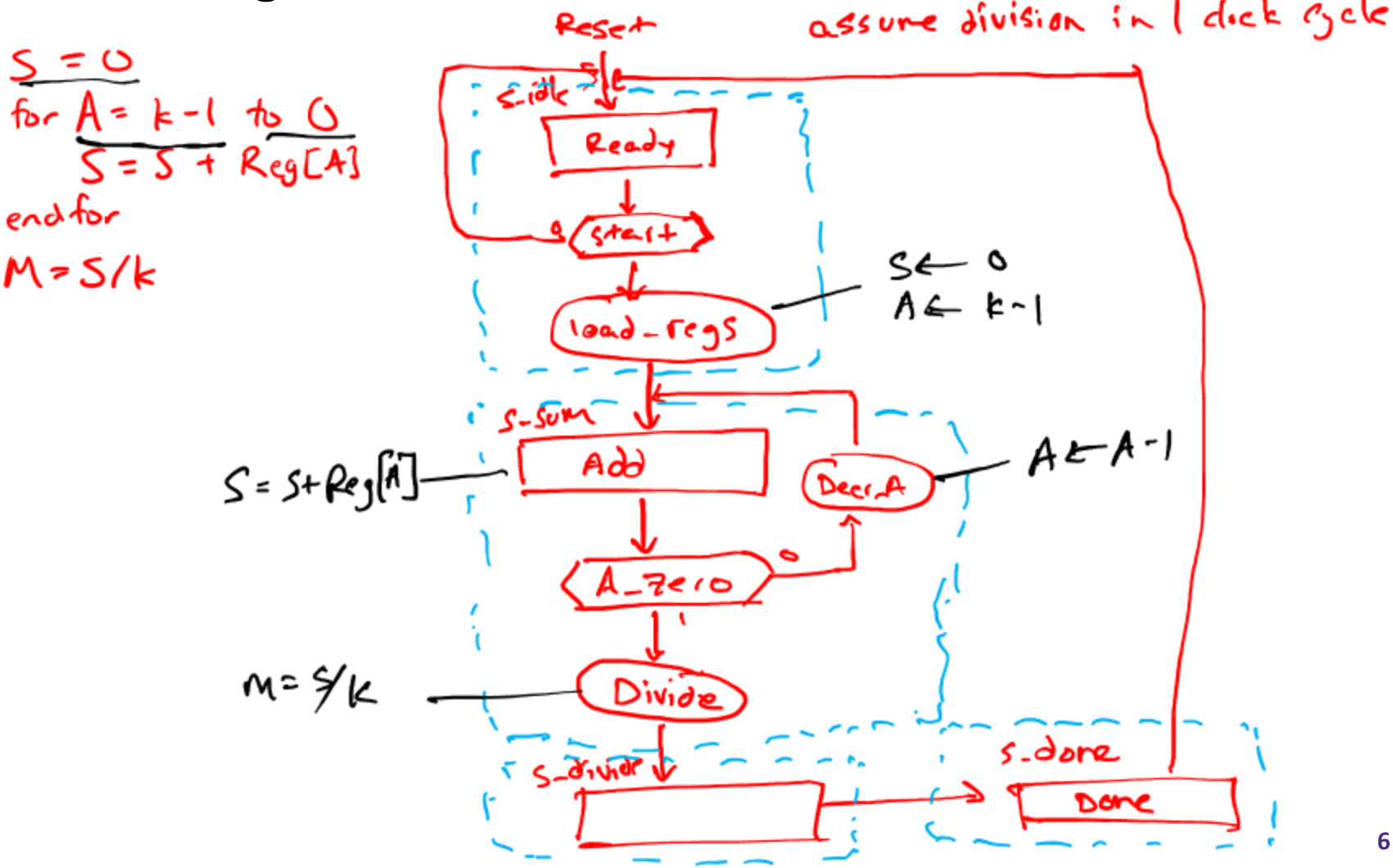
Arithmetic Mean Specification

- Datapath
  - A k-address *register file* (only using `r_addr` and `r_data`)
  - Reg file address stored in $\lceil \log_2(k) \rceil$ *down-counter* A
  - Sum stored in register S
  - An n-bit *divider* circuit, as discussed last lecture

- Control
  - Inputs $Start$ and $Reset$, outputs $Ready$ and $Done$
  - Status signals:
  - Control signals:

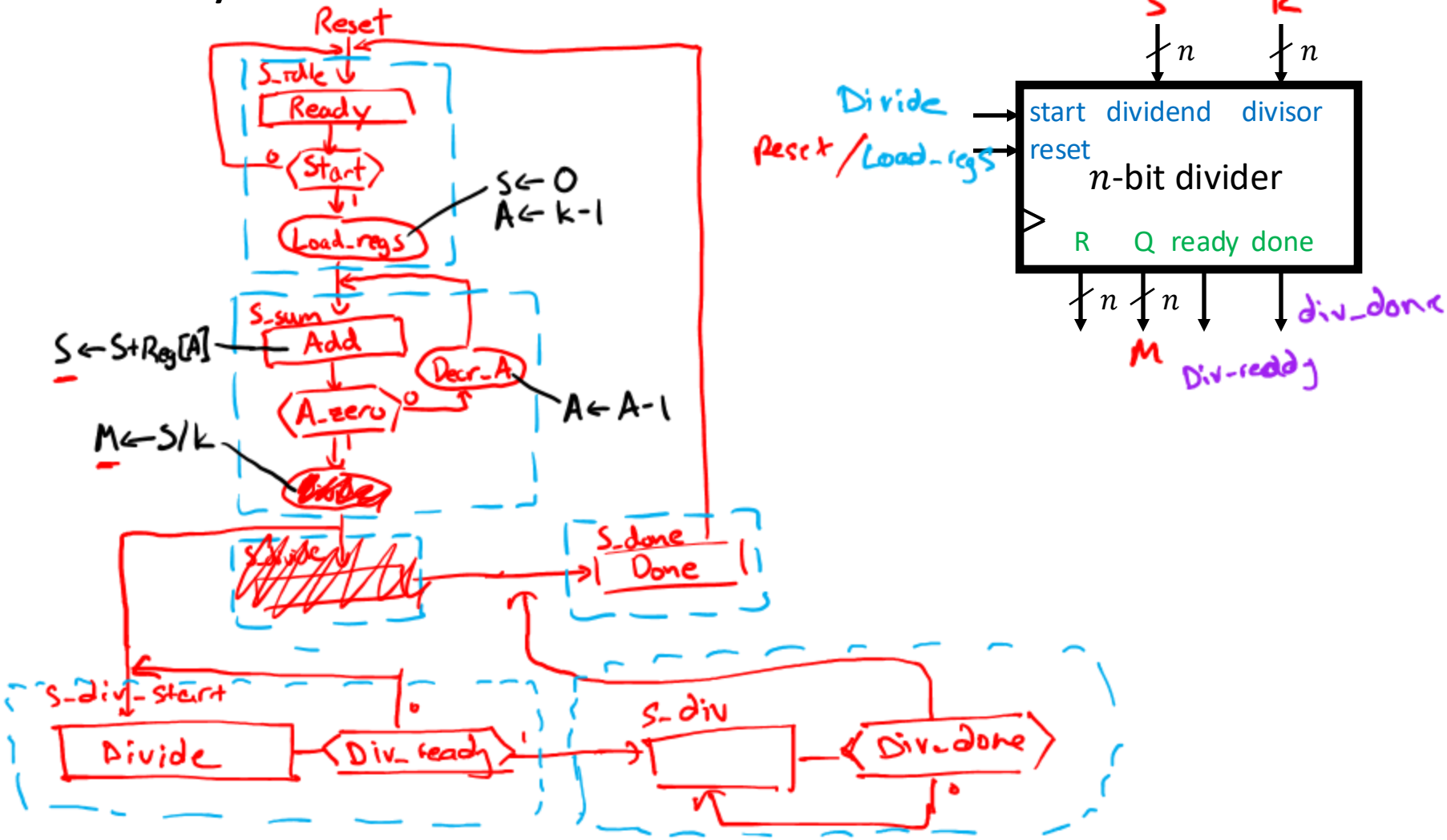
Arithmetic Mean (ASMD Chart, Initial)

- For now, ignore the details of the divider circuit

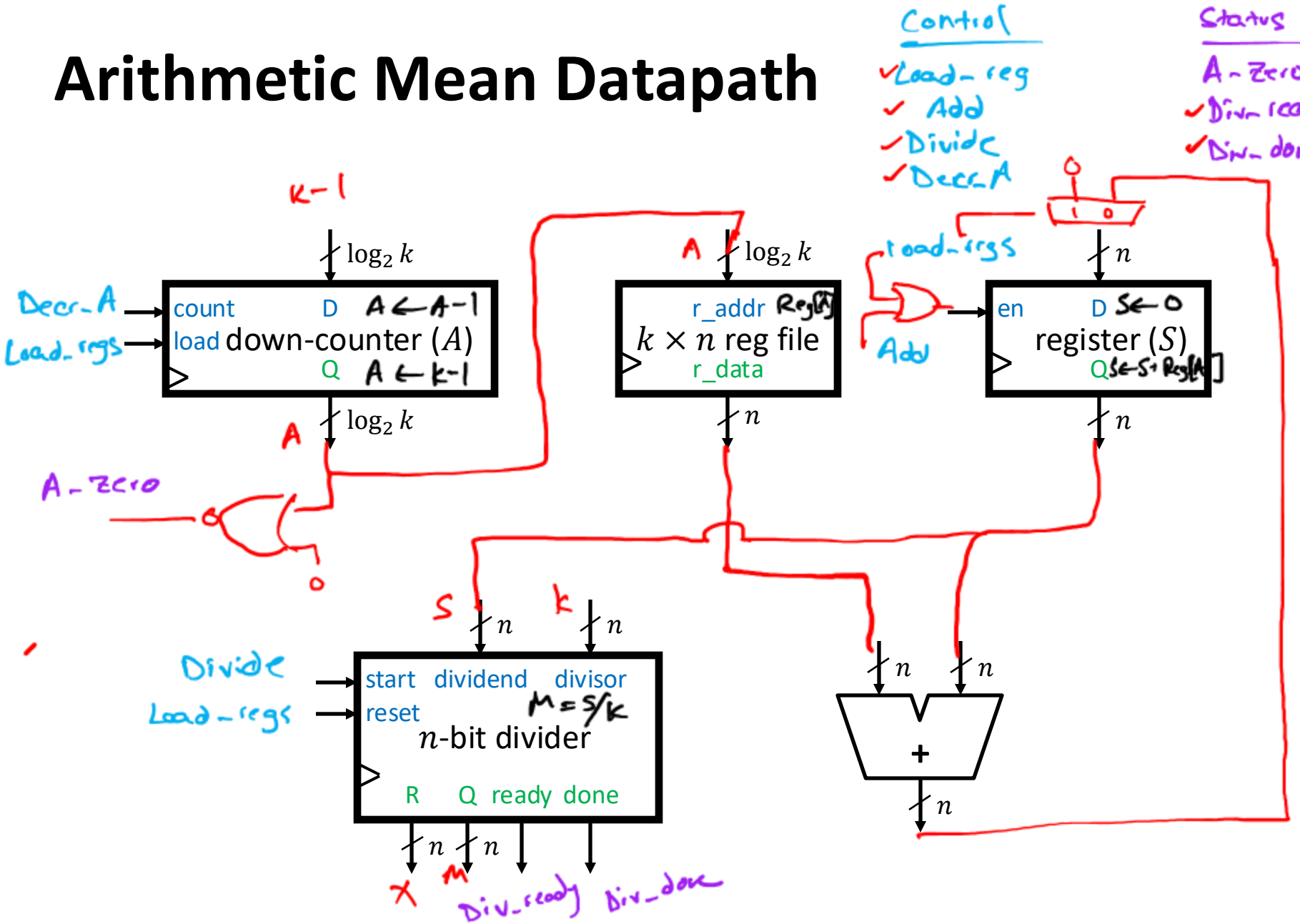


Arithmetic Mean (ASMD Chart)

- Fix your ASMD chart based on the divider circuit:



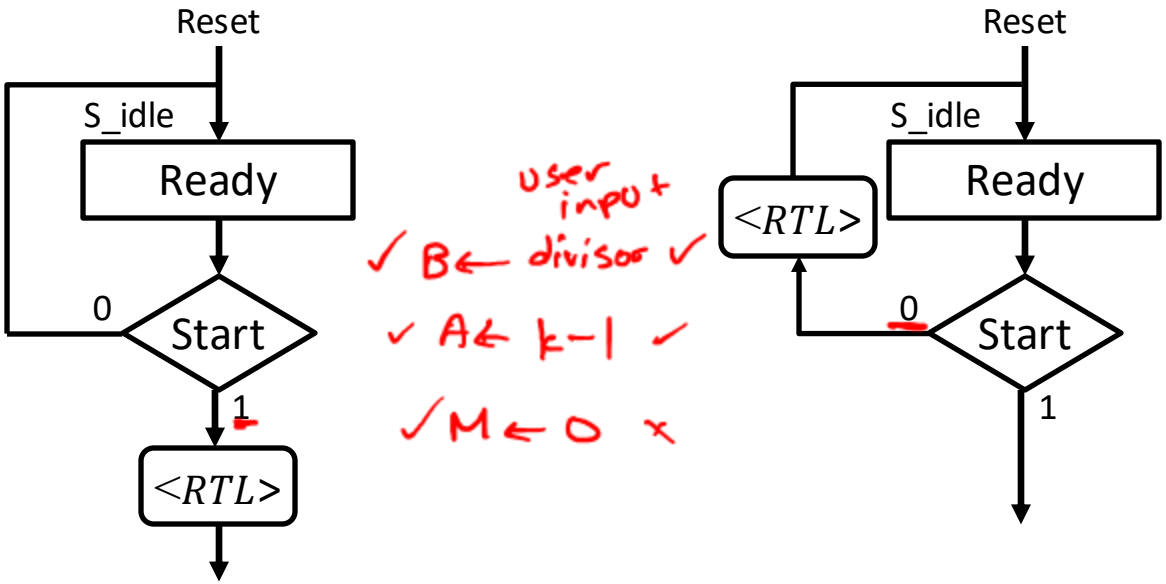
Arithmetic Mean Datapath



Technology Break

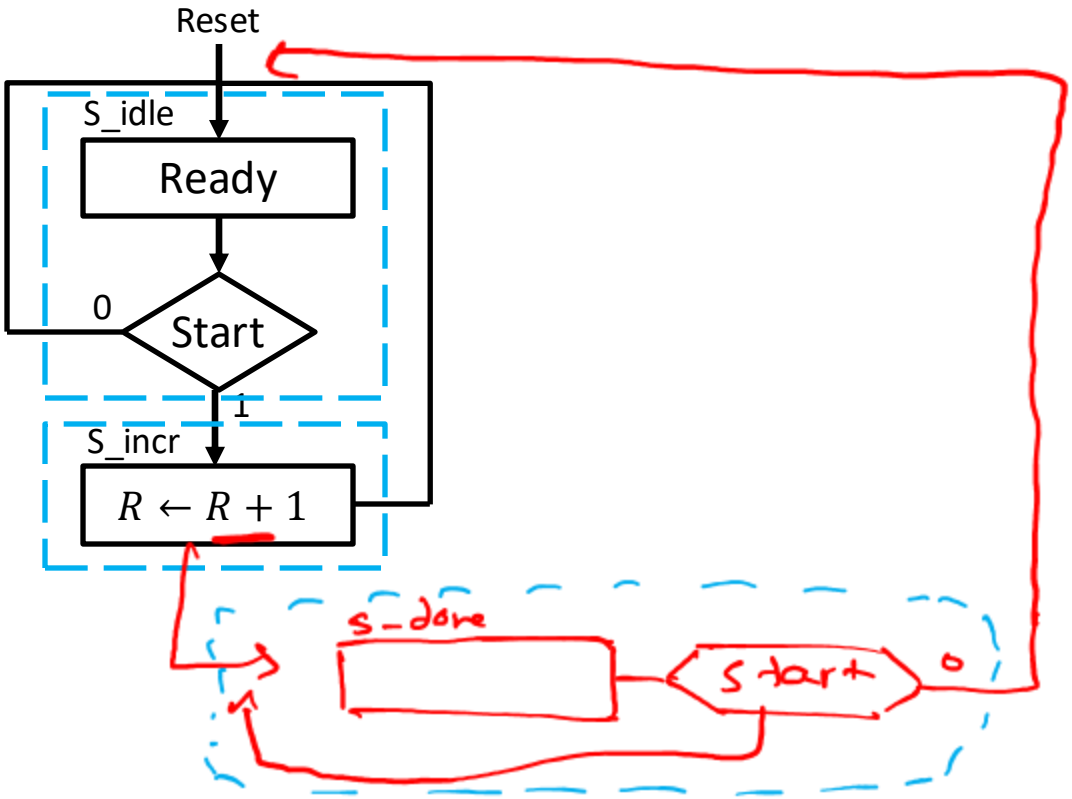
Aside: Load Loops

- For *some* initialization operations, you can get equivalent behavior from either the (1) outgoing edge or the (2) looping edge:



Aside: Start Loops

- What happens if we forget to de-assert $Start$?
- Fix:



Sorting Algorithm

- Design a circuit to sort k n-bit numbers stored in a set of registers in ascending order

Algorithm:

```
for i = 0 to k-2 do
    A = Reg[i]
    for j = i+1 to k-1 do
        B = Reg[j]
        if B < A then
            Reg[i] = B
            Reg[j] = A
            A = Reg[i]
        endif
    endfor
endfor
```

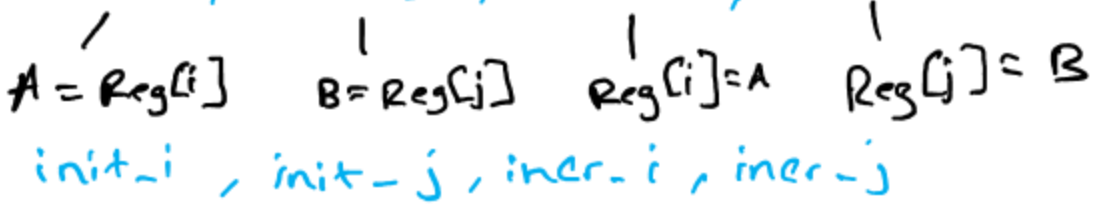
Example ($k = 4$): i to 2, j to 3

i	j	A	B	R[0]	R[1]	R[2]	R[3]
0	1	3	7 x	3	7	1	0
0	2	3	1 ✓	3	7	1	0
0	3	1	0 ✓	1	7	3	0
1	2 (i+1)	7	3 ✓	0	7	3	1
1	3	3	1 ✓	0	3	7	1
2	3 (i+1)	7	3 ✓	0	1	7	3
				0	1	3	7

Sorting Algorithm Specification

- Datapath
  - A k-address *register file* (assume only 1 port)
  - Two $\lceil \log_2(k) \rceil$ *up-counters* i and j
  - Two registers A and B
  - An n-bit *comparator* circuit to check for $B < A$

- Control
  - Inputs $Start$ and $Reset$, outputs $Ready$ and $Done$
  - Status signals: i_done, j_done, B_lt_A
  - Control signals: Load_A, Load_B, Store_A, Store_B



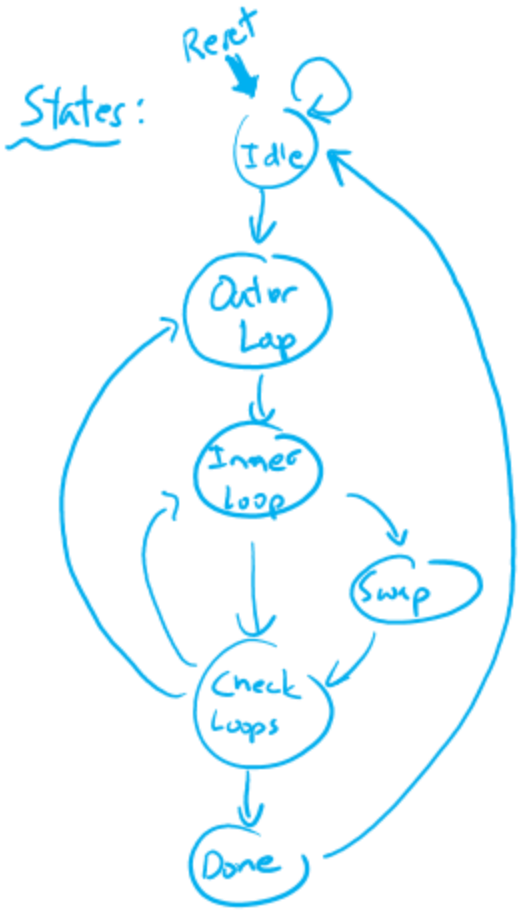
Sorting Algorithm Specification

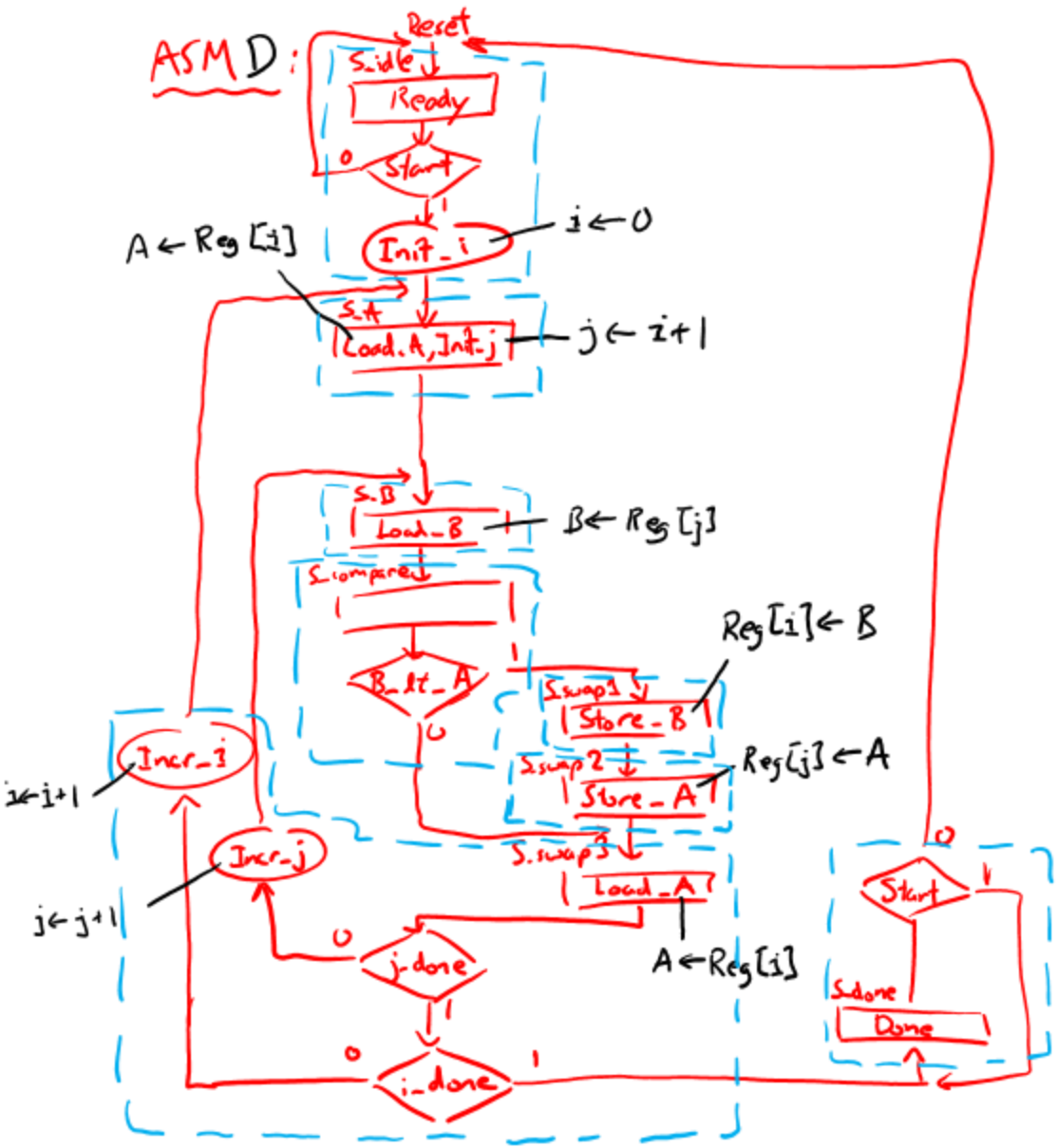
- Datapath
  - A k-address *register file* (assume only 1 port)
  - Two $\lceil \log_2(k) \rceil$ *up-counters* i and j
  - Two registers A and B
  - An n-bit *comparator* circuit to check for $B < A$

- Timing Notes:
  - RTL operations in a state occur on the *next* clock trigger
  - Can `i` ← `x` and `A` ← `Reg[i]` be done simultaneously?
  - Can `Reg[i]` ← `B` and `Reg[j]` ← `A` be done simultaneously?
  - Swap operations *must* be done sequentially

Sorting Algorithm (ASMD Chart)

Sorting Algorithm (ASMD Chart)

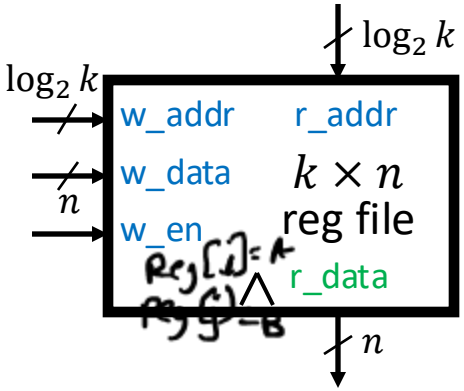


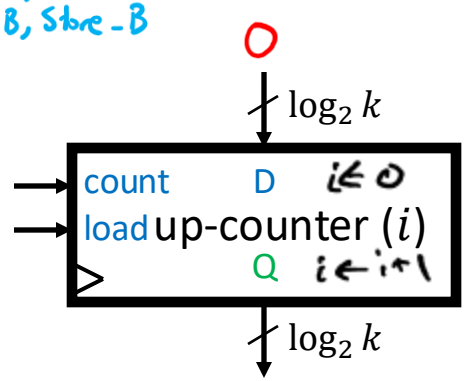


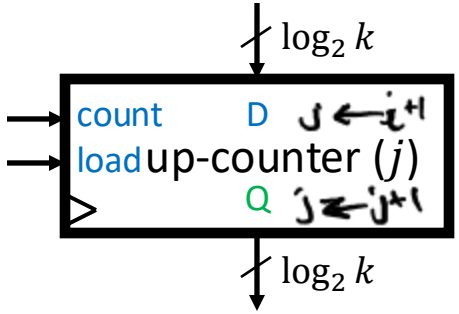
Sorting Algorithm Datapath

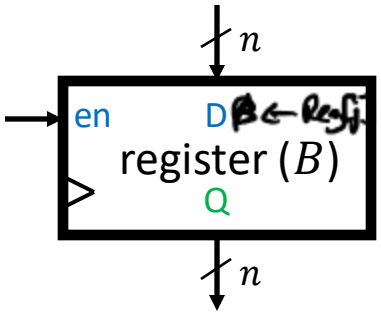
Control Signals:
Init_i, Incr_i
Init_j, Incr_j
Load_A, Store_A
Load_B, Store_B

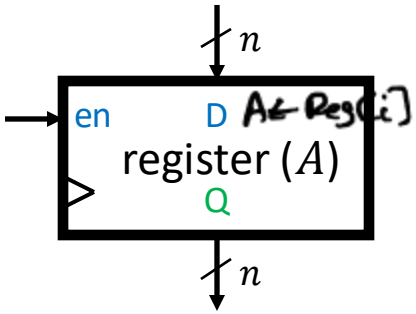
Status Signals:
B_lt_A
i_done
j_done



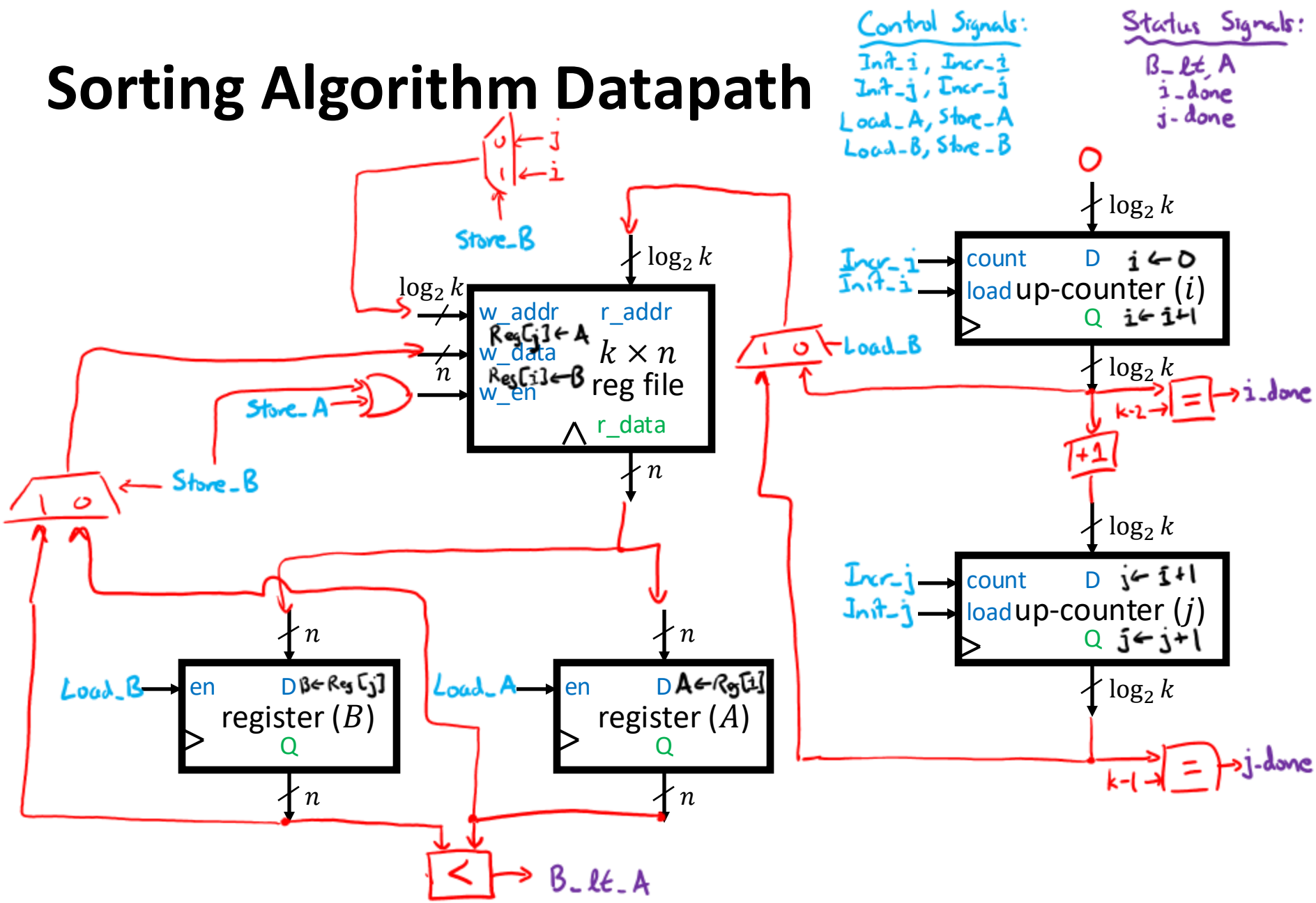




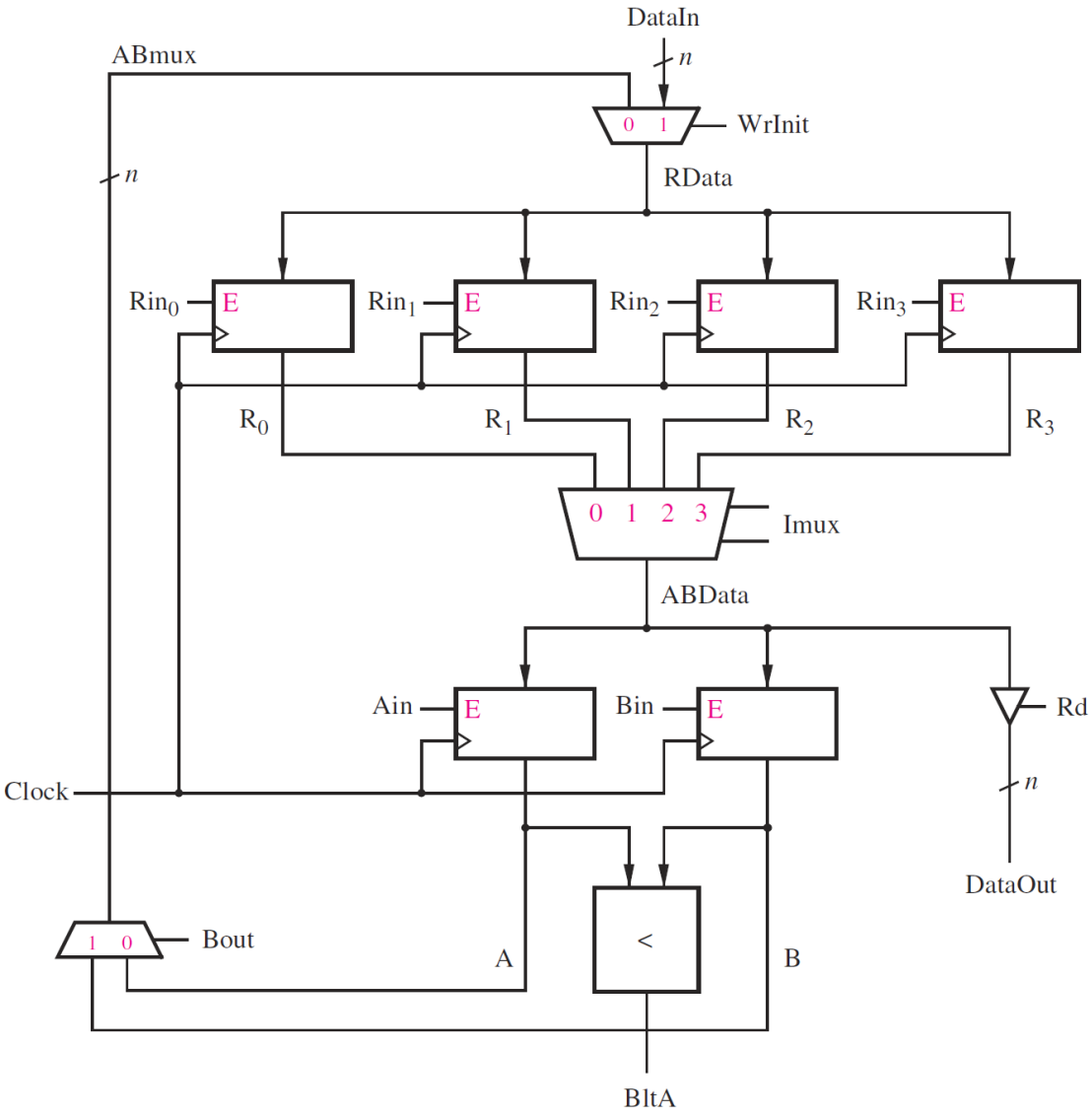




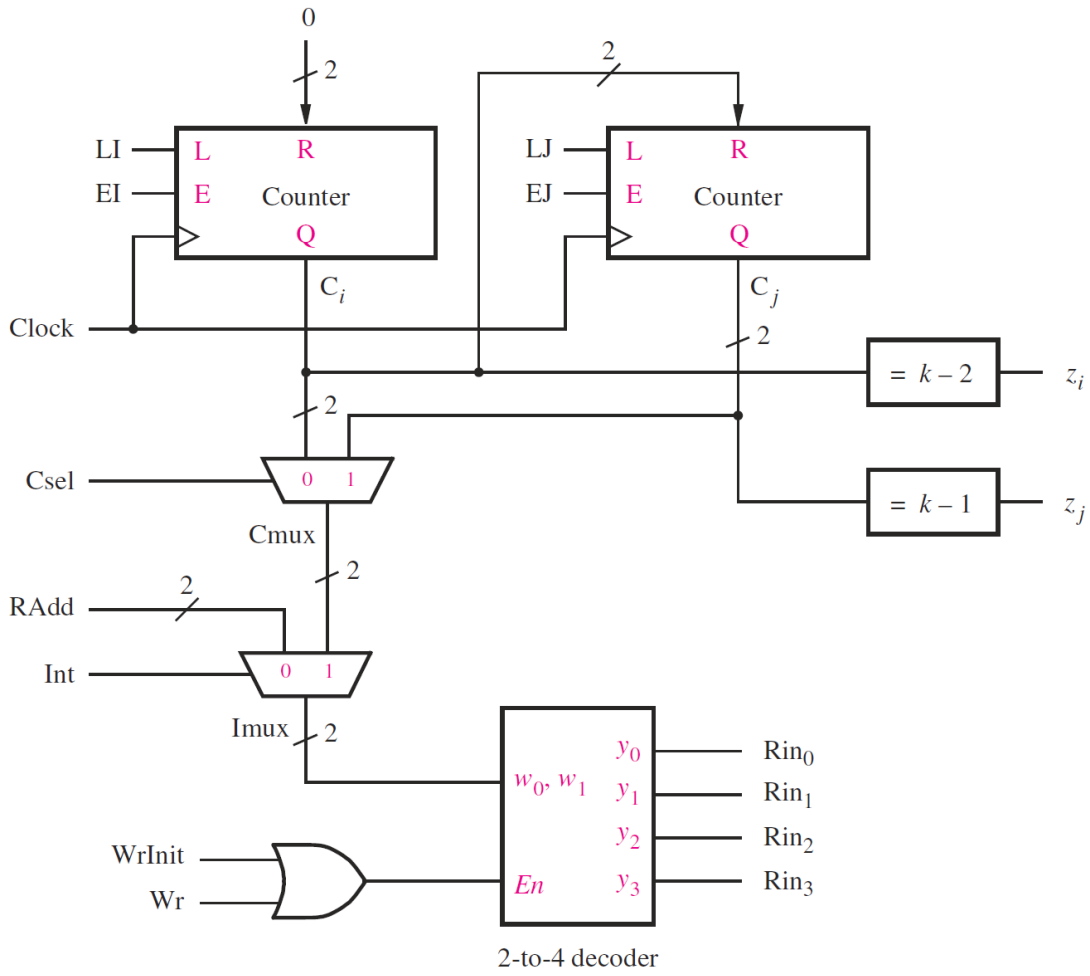
Sorting Algorithm Datapath



Alternate Sort Algorithm Datapath (1/2)



Alternate Sort Algorithm Datapath (2/2)



Lab 4 Preview: Binary Search

- Design a circuit that searches a *sorted* array for a given value by checking the middle element of the remaining portion of the array we would expect to find the given number:

```
L = 0
R = n - 1
while L <= R do
    m = floor((L + R)/2)
    if A[m] < T then
        L = m + 1
    else if A[m] > T then
        R = m - 1
    else
        return m
    endif
endwhile
return unsuccessful
```