Design of Digital Circuits and Systems

Algorithms to Hardware I

Instructor: Justin Hsia

Teaching Assistants:

Colton Harris
Deepti Anoop
Gayathri Vadhyan
Jared Yoder
Lancelot Wathieu
Matthew Hung

Relevant Course Information

- Lab 3 reports due Friday (4/26)
- Lab 4 due next Friday (5/3)

- hw4 due on Monday (4/29)

- Anonymous mid-quarter survey on Canvas (due 4/29)

- Quiz 3 (ASM, ASMD) next Thursday (5/2)

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:

Aside: Counter Variable

- Many sequential hardware algorithms utilize counters
- If both work, is there a preference?
  - How to implement $C = k - 1$ check?
  - How to implement $C = 0$ check?

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

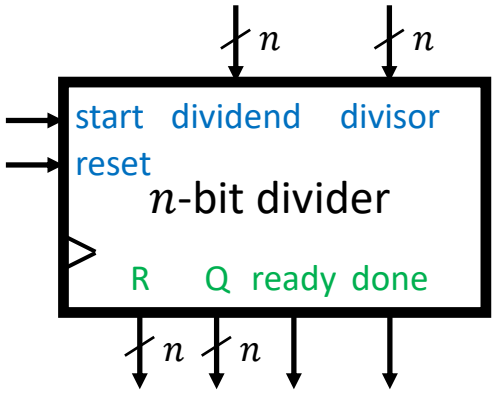
- Control
  - Inputs $Start$ and $Reset$, outputs $Ready$ and $Done$
  - Status signals:
  - Control signals:

Arithmetic Mean (ASMD Chart, Initial)

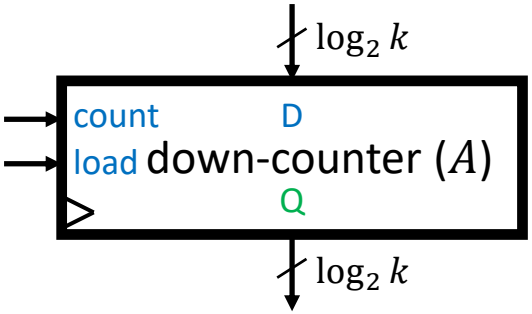
- For now, ignore the details of the divider circuit

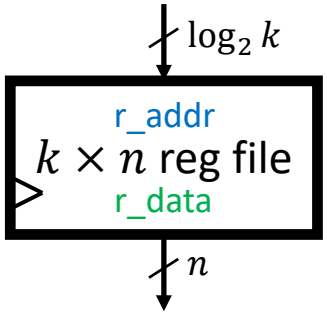
Arithmetic Mean (ASMD Chart)

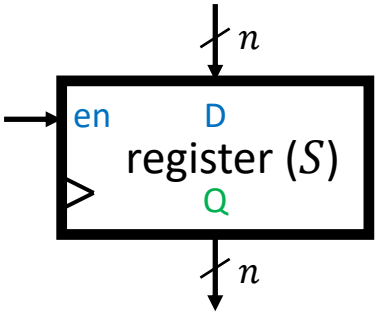
- Fix your ASMD chart based on the divider circuit:

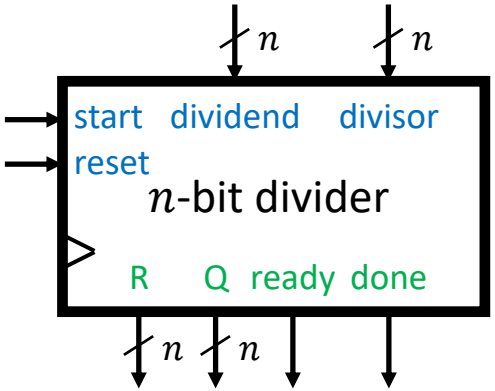


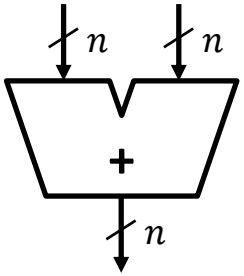
Arithmetic Mean Datapath







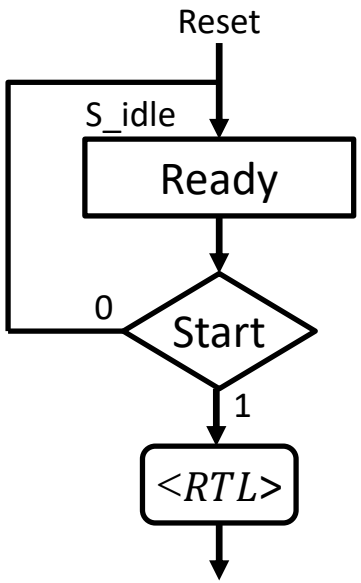


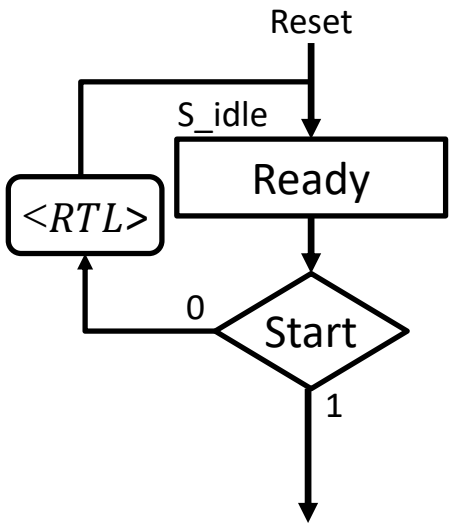


Technology Break

Aside: Load Loops

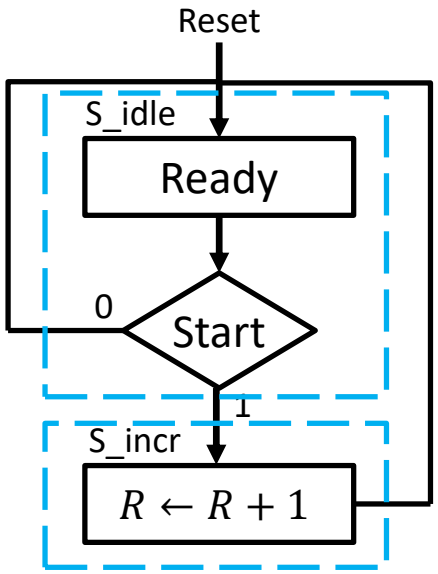
- For *some* initialization operations, you can get equivalent behavior from either the (1) outgoing edge or the (2) looping edge:





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	j	A	B	R[0]	R[1]	R[2]	R[3]
0	1	3	7	3	7	1	0

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:
  - Control signals:

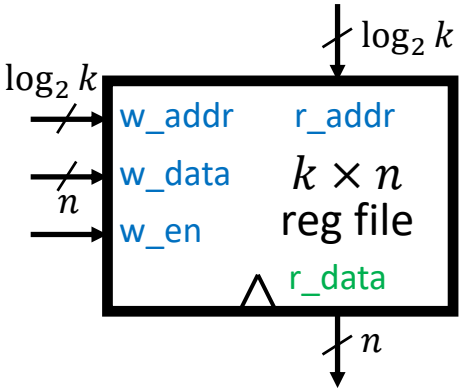
Sorting Algorithm Specification

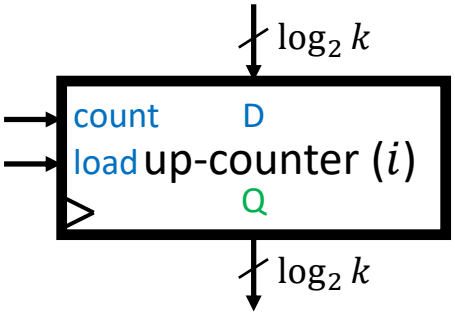
- Datapath
  - A k-address *register file* (assume only 1 port)
  - Two $\lceil \log_2(k) \rceil$ *up-counters* i and j
  - Two registers A and B
  - An n-bit *comparator* circuit to check for $B < A$

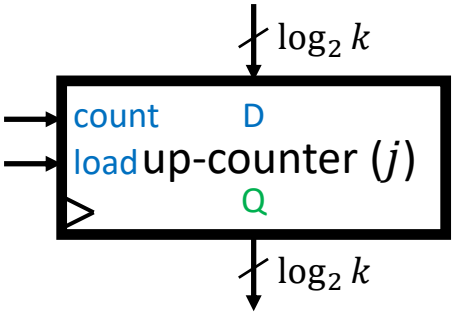
- Timing Notes:
  - RTL operations in a state occur on the *next* clock trigger
  - Can `i` ← `x` and `A` ← `Reg[i]` be done simultaneously?
  - Can `Reg[i]` ← `B` and `Reg[j]` ← `A` be done simultaneously?
  - Swap operations *must* be done sequentially

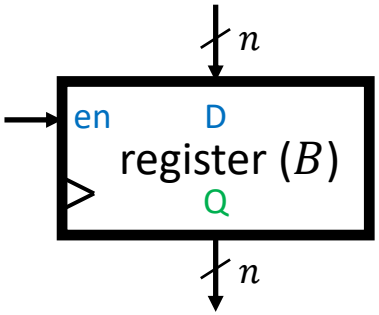
Sorting Algorithm (ASMD Chart)

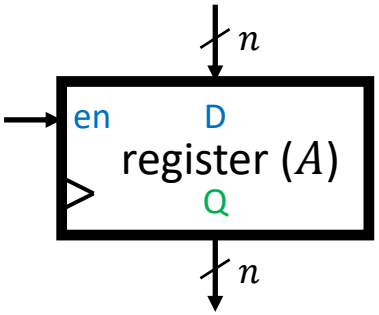
Sorting Algorithm Datapath



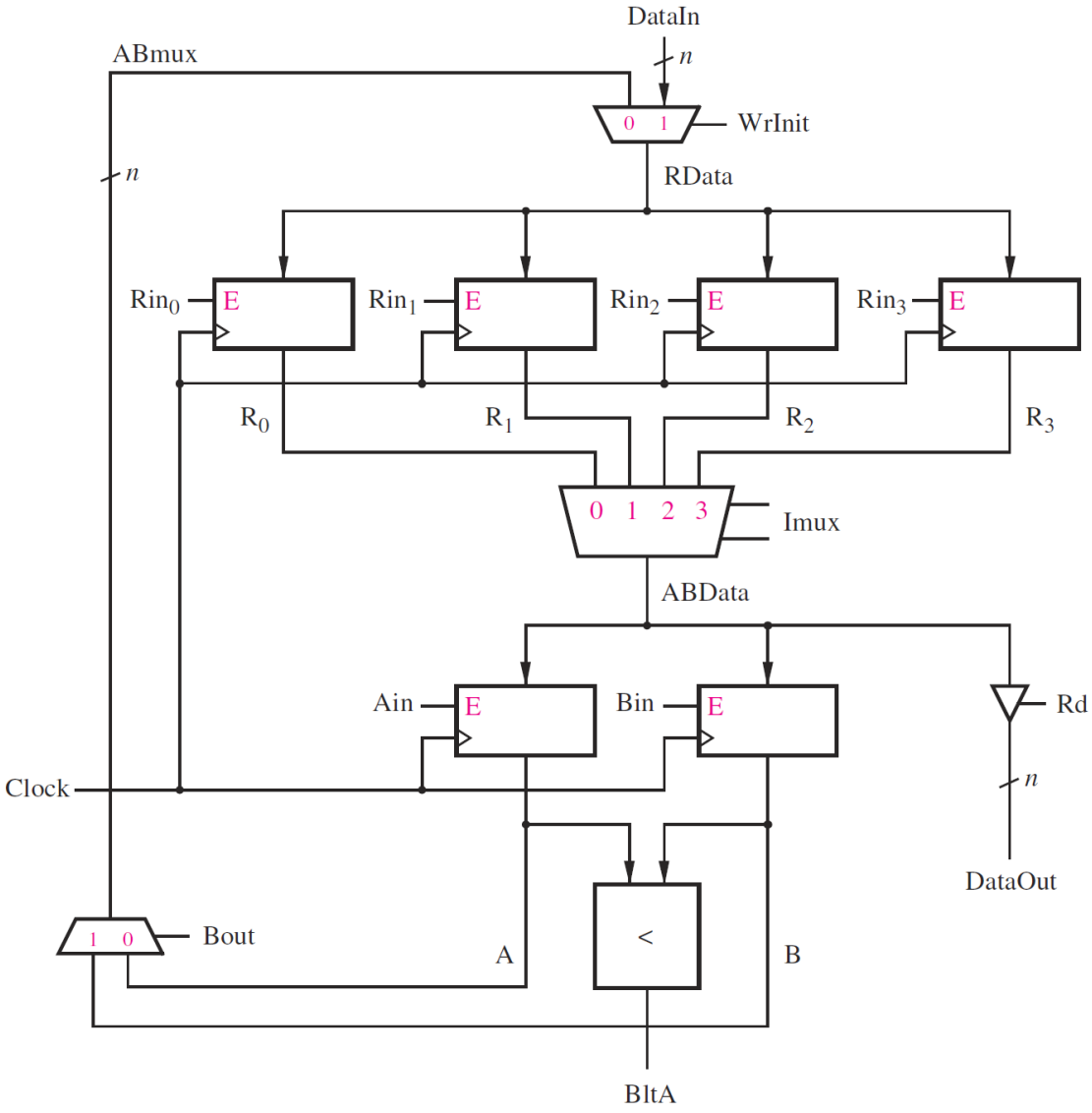




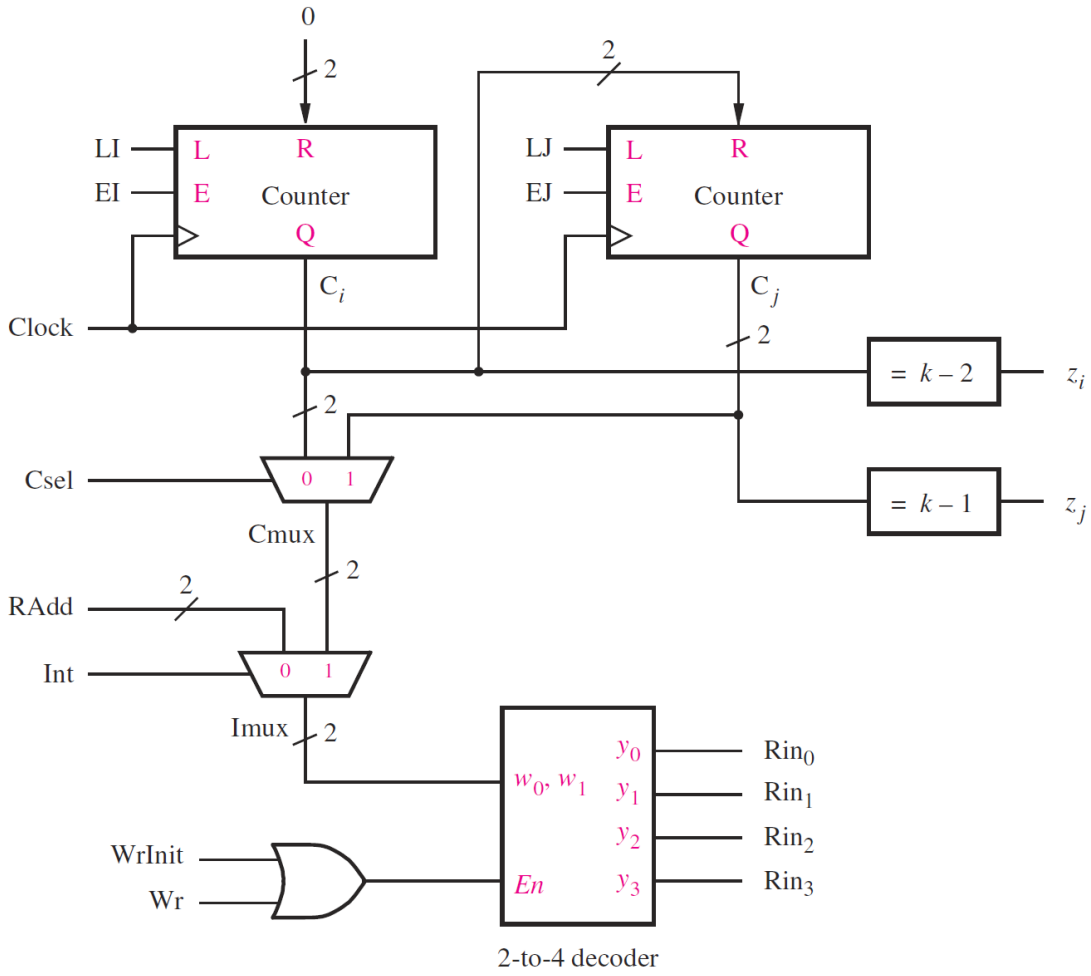




Alternate Sort Algorithm Datapath (1/2)



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
```