Constructive Computer Architecture:

FIFO Lab Comments

Revisiting CF FIFOs

Andy Wright
6.175 TA

Notes

- The FIFO lab has been graded
- Everyone did a great job on the conflicting, pipeline, and bypass FIFOs
- All difficulties came from the Conflict-Free FIFO and the discussion questions

What is a Conflict-Free FIFO?

- Is it a FIFO with a conflict matrix filled with CFs?
  - Not really...
- Is it a FIFO that, in its typical use, doesn’t impose any more conflicts than necessary?
  - Yes!

Typical FIFO use

- Enqueing:
  - check notFull and call enq(x)
- Dequeuing:
  - check notEmpty and call first and deq
- Clearing:
  - call clear

- These are the methods that commonly appear in the same rule
  - A CF FIFO says enqueuing and dequeuing rules will have no scheduling conflict imposed by the FIFO.
  - {notFull, enq} CF {notEmpty, first, deq}

Sequential FIFO Execution

Assume FIFO is empty initially

1. enq(1)
2. first
3. deq
4. enq(2)
5. clear
6. enq(3)
7. enq(4)
8. first
9. deq

How many clock cycles does this take?
Should that change the answer to these questions?

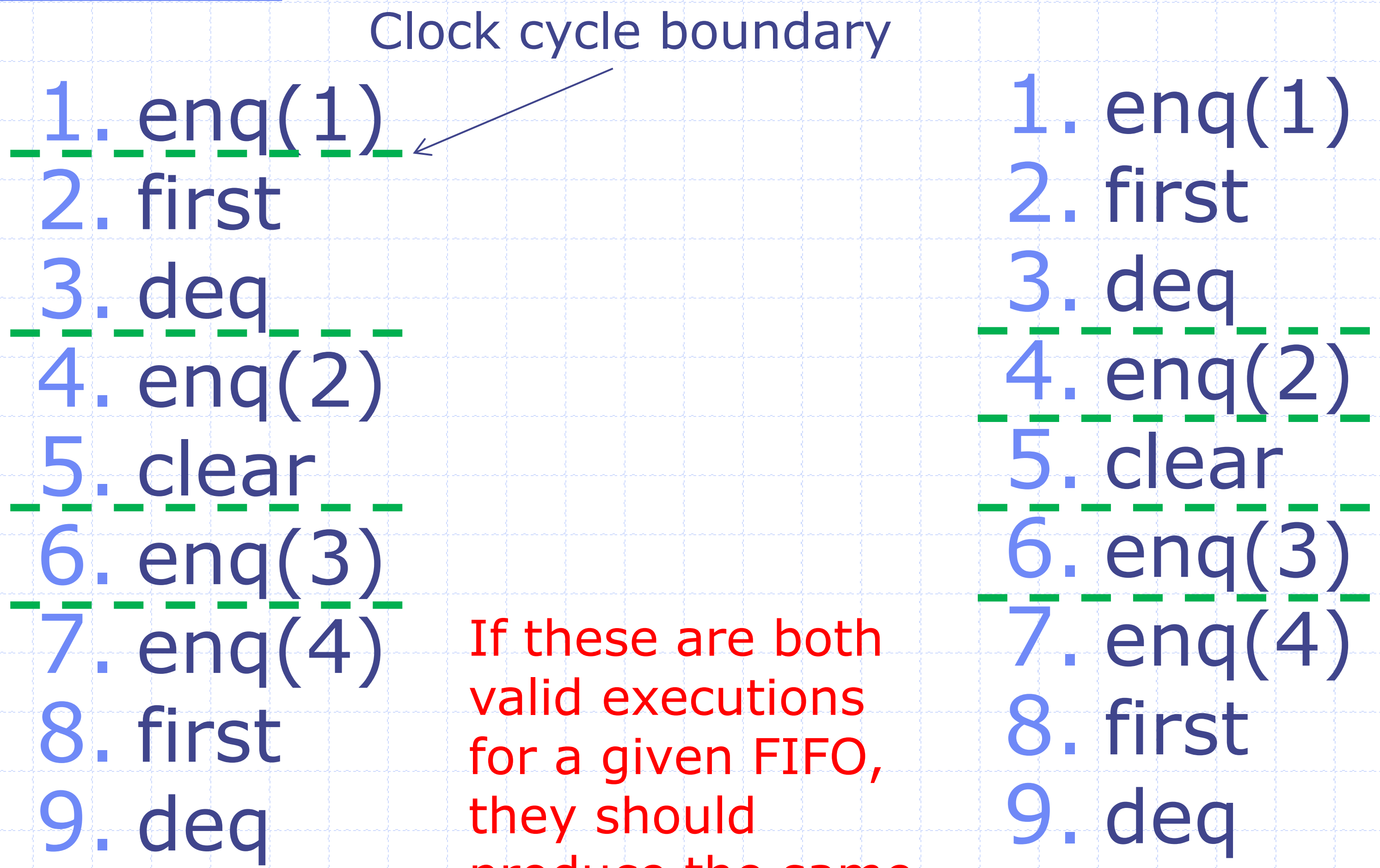
← What is the state of the fifo here?

What does first return?

How many elements are in the FIFO at the end?

Concurrent FIFO Execution

With Clock Cycles



1. enq(1)
2. first
3. deq
4. enq(2)
5. clear
6. enq(3)
7. enq(4)
8. first
9. deq

If these are both valid executions for a given FIFO, they should produce the same results.

1. enq(1)
2. first
3. deq
4. enq(2)
5. clear
6. enq(3)
7. enq(4)
8. first
9. deq

Concurrent FIFO Execution

Clear-Enq Interactions

1. enq(1)
2. first
3. deq
4. enq(2)
5. clear
6. enq(3)
7. enq(4)
8. first
9. deq

What should clear and enq do when they happen in the same cycle?

If both executions are valid, then the action has to be dynamic based on the order.

1. enq(1)
2. first
3. deq
4. enq(2)
5. clear
6. enq(3)
7. enq(4)
8. first
9. deq

Can we detect the order of methods firing?

- In order for a method to know if it fired after another method fired, it has to always be scheduled after that method
- If you want two methods to be able to tell if either of them came after the other, they have to conflict
    - They will never fire in the same cycle making this problem of who fired first trivial...

Concurrent FIFO Execution

CF FIFO solution

1. enq(1)
2. first
3. deq
4. enq(2)
5. clear
6. enq(3)
7. enq(4)
8. first
9. deq

Force enq < clear

Invalid Execution

1. ~~enq(1)~~
2. ~~first~~
3. ~~deq~~
4. ~~enq(2)~~
5. ~~clear~~
6. ~~enq(3)~~
7. ~~enq(4)~~
8. ~~first~~
9. ~~deq~~

What does a CF FIFO do?

- It allows the most flexible scheduling constraints possible while still requiring all valid concurrent executions to match the expected result from sequential execution.
  - That is why {enq, deq} < clear and not {enq, deq} CF clear

What about the canonicalize rule?

- What happens if you have:
  - {enq, deq} < canonicalize < clear
    - If deq and clear occur in the same rule, that rule conflicts with canonicalize. This may or may not be a problem depending on the exact use.
  - {enq, deq} < clear < canonicalize
    - canonicalize can always fire at the end of the cycle independent of how enq, deq, and clear are used in rules