6.004 Recitation 10
L10 – Sequential Circuits: Methods with Guarded Interfaces

Problem 1

Recall the following interface and the corresponding **mkFifo (Fifo#(1, t))** module implementation of a FIFO queue given in lecture:

```
interface Fifo#(numeric type size, type t);
  method Action enq(t x);
  method Action deq;
  method t first;
endinterface
```

Create an equivalent implementation using the following interface, where `pop` both dequeues and returns the follow of the first element in the Fifo:

```
interface Fifo#(numeric type size, type t);
  method Action enq(t x);
  method ActionValue pop;
endinterface
```

`Explore:` Try changing some of the assignment operators (“< -”, “< =”, “=”) into the other ones. What compiler errors do you receive?

Problem 2

Create a “streaming” rule that uses your Fifo implementation. This rule should be contained in a module that creates 3 Fifos, an “input” Fifo, a “storage” Fifo which takes its enqueues from items dequeued from the input Fifo, and an “output” Fifo, which takes its enqueues from items dequeued from the storage Fifo. *(hint: You will need 2 rules to accomplish this)*

Problem 3

Create a version of the FIFO queue that can hold 2 elements. Note: When this FIFO only has a single element in it, this element should be at the “front” of the queue, or in other words it should be the element about to be dequeued/the “first” element.

Expressing Loops in Bluespec

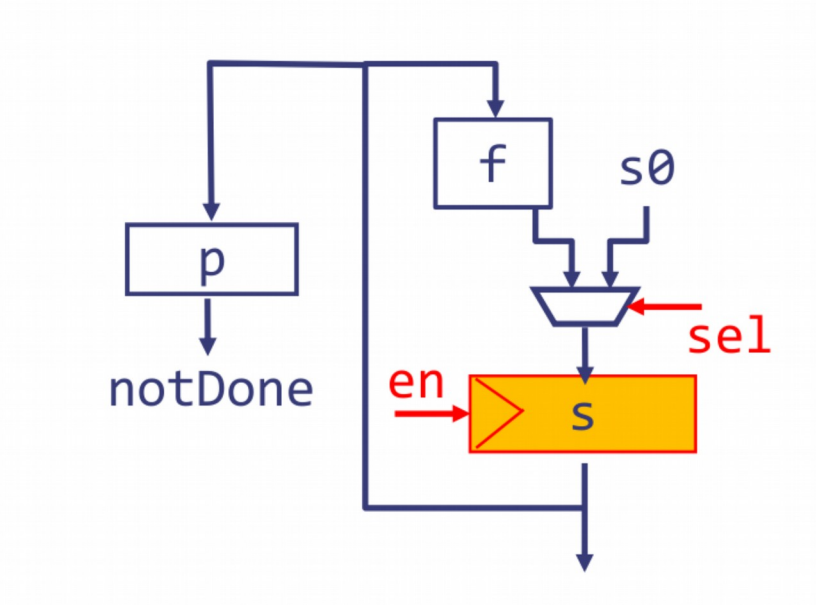
Consider the following while loop in Python:

```
s = s0
while (s > 5):
  s = s - 1
return s
```

The number of iterations that this loop will perform is dependent on the initial value s0. As a result, the loop cannot be described by unfolding. Instead, we can use a **register** to hold the value of *s* from one iteration to the next. Once we've done that, we can use a **rule** to update the value of *s* each cycle until the computation terminates.

The following diagram and Bluespec implementation describe this process:

```
Reg#(Bit#(8)) s <- mkReg(s0);
rule iteration if (s > 5);
  s <= s – 1;
endrule
```



For the described Python code, f(s) = s – 1, and p(s) = (s > 5). 'sel' only picks s0 for the initial value of s, and 'en' is dependent on 'notDone', or the output of p(s)