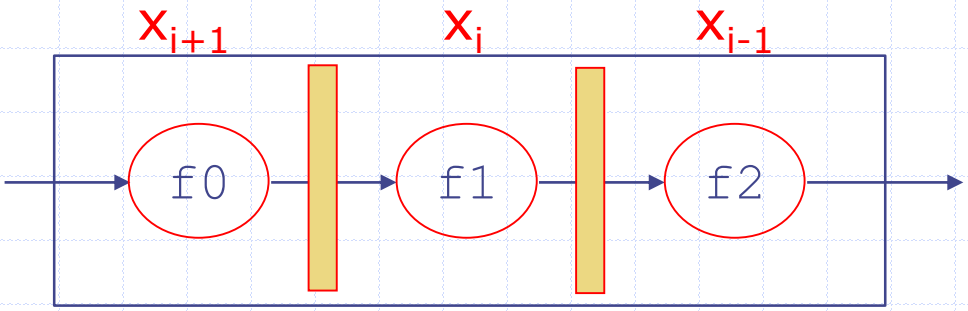
Pipelining combinational circuits

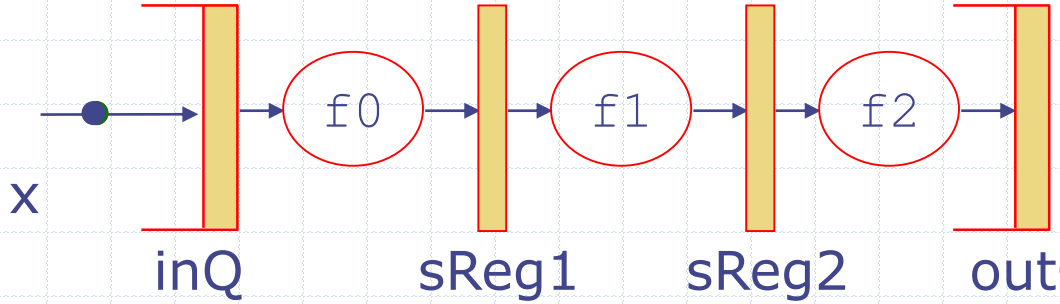
Pipelining Combinational Functions



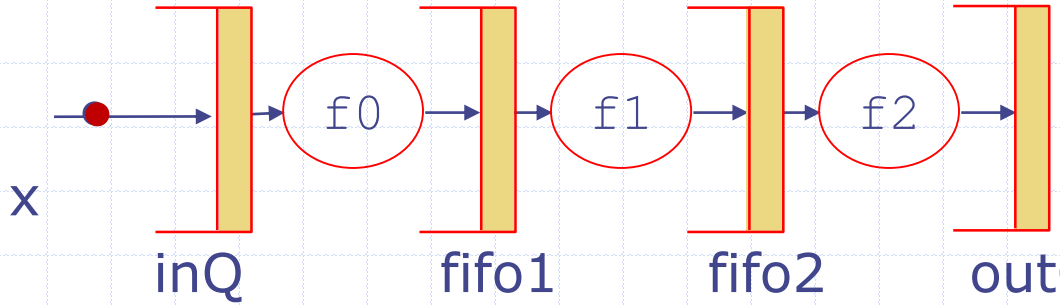
3 different datasets in the pipeline

- Lot of area and long combinational delay
- Folded or multi-cycle version can save area and reduce the combinational delay but throughput per clock cycle gets worse
- Pipelining: a method to increase the circuit throughput by evaluating multiple inputs

Inelastic vs Elastic pipeline



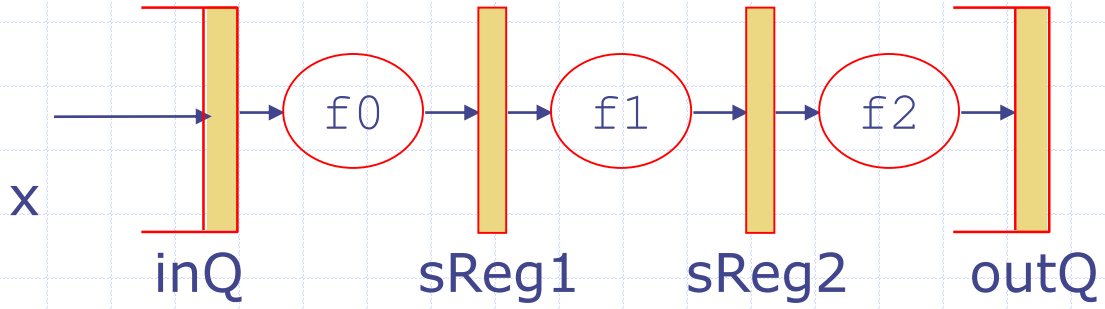
Inelastic: all pipeline stages move synchronously



Elastic: A pipeline stage can process data if its input FIFO is not empty and output FIFO is not Full

Most complex processor pipelines are a combination of the two styles

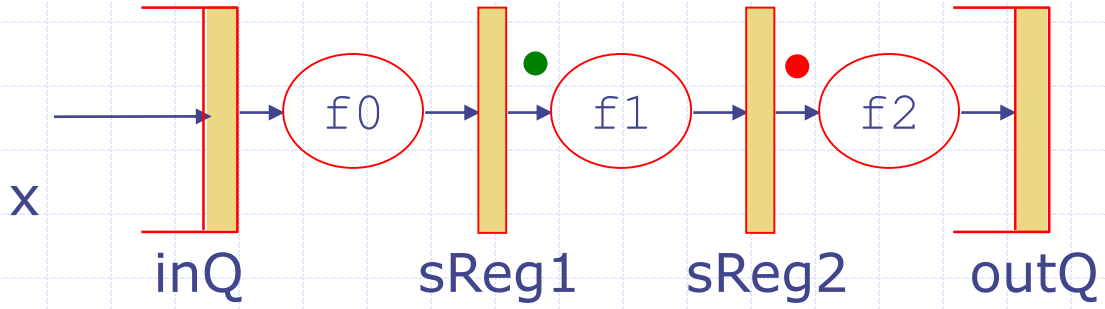
Inelastic pipeline




```
rule sync_pipeline;
    inQ.deq;
    sReg1 <= f0(inQ.first);
    sReg2 <= f1(sReg1);
    outQ.enq(f2(sReg2))
endrule
```

When can this rule execute?

Pipeline bubbles



Some issues

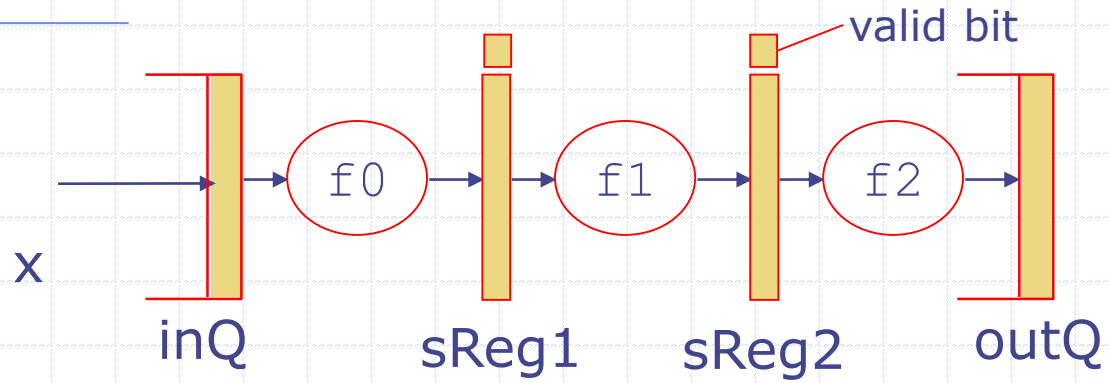
```
rule sync_pipeline;
   inQ.deq;
   sReg1 <= f0(inQ.first);
   sReg2 <= f1(sReg1);
   outQ.enq(f2(sReg2))
endrule
```

- Red and Green tokens must move even if there is nothing in inQ!
- Also if there is no token in sReg2 then nothing should be enqueued in the outQ

Modify the rule to deal with these conditions

Introduce a valid bit

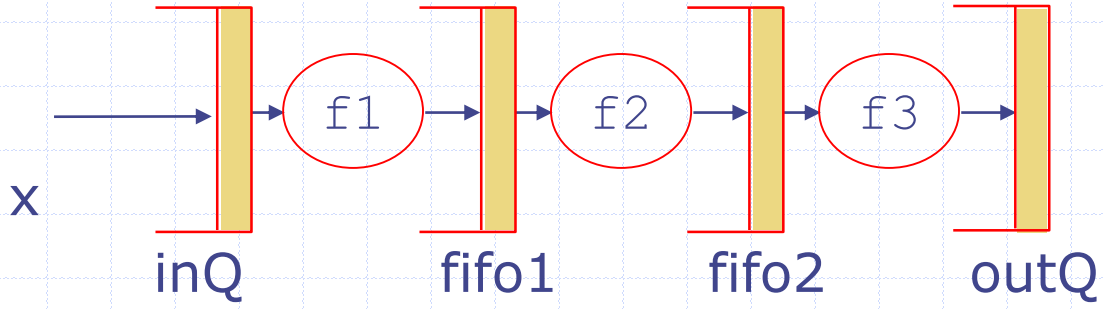
Explicit encoding of data presence




```
rule sync_pipeline;
 if(outQ.notFull || sReg2v != true)
  if (inQ.notEmpty)
  begin sReg1 <= f0(inQ.first); inQ.deq;
        sReg1v <= true end
  else  sReg1v <= false;
 sReg2 <= f1(sReg1); sReg2v <= sReg1v;
  if (sReg2v == true) outQ.enq(f2(sReg2)))
endrule
```

Elastic pipeline

Use FIFOs instead of pipeline registers



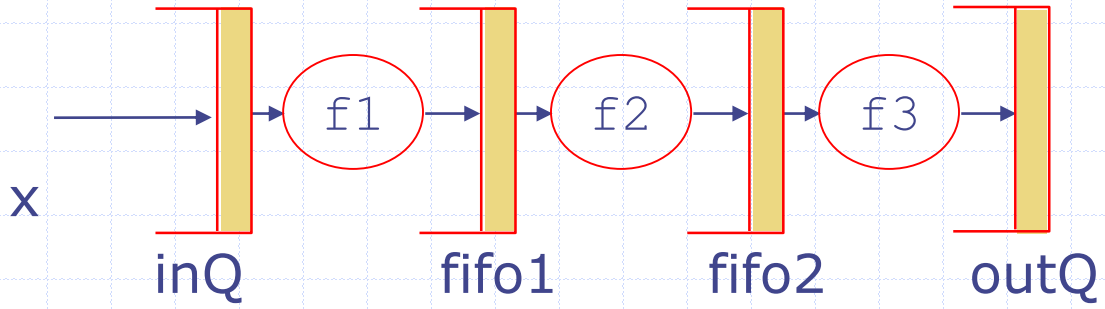
no need for valid bits

```
rule stage1;
  fifo1.enq(f1(inQ.first);
  inQ.deq();          endrule
rule stage2;
  fifo2.enq(f2(fifo1.first);
  fifo1.deq;          endrule
rule stage3;
  outQ.enq(f3(fifo2.first);
  fifo2.deq;          endrule
```

- When can stage1 rule fire?
  - inQ has an element
  - fifo1 has space
- Can tokens be left in the pipeline?

  No
- Can these rules execute concurrently?

Elastic pipeline




```
rule stage1;
  fifo1.enq(f1(inQ.first);
  inQ.deq();          endrule
rule stage2;
  fifo2.enq(f2(fifo1.first);
  fifo1.deq;          endrule
rule stage3;
  outQ.enq(f3(fifo2.first);
  fifo2.deq;          endrule
```

- If these rules cannot execute concurrently, it is hardly a pipelined system
- When can rules execute concurrently?
- What hardware is synthesized to execute rules concurrently?

Multi-rule Systems

Repeatedly:

- Select a rule to execute
- Compute the state updates
- Make the state updates

Non-deterministic choice; User annotations can be used in rule selection

One-rule-at-a-time-semantics: Any legal behavior of a Bluespec program can be explained by observing the state updates obtained by applying only one rule at a time

However, for performance we execute multiple rules concurrently whenever possible

stay tuned …