Solution 6.3 (Buffer Loss)

(c) Peter Fischer, SuS@Uni HD, 2019

- FIFO has a depth of M (N is special character in Mathematica)
- Rate of hits, i.e. writing to FIFO is fin
- FIFO is emptied with constant rate fread

- we could use large time steps and calculate how many hits we get per time step, add this to FIFO content, and empty a certain amount of hits. This will need special treatment when only a fraction of hits can be written.

- We will insted use here very small time steps dt:
- The only relevant number is obviously the ratio between fin and fread. We set fin = α fread where α must obviously be smaller than 1.
- For simplicity we set fread = 1, and dt an integer fraction of this, i.e. dt = $\frac{1}{R}$
- in most time steps, nothing happens.
- We assume that we have at most have one hit per time step (which is onyl true if tiem steps are small!), the rate depending on fin. The probability to have a hit in one time step is α / R
- We will empty the FIFO every R time steps

---

1. Program

In[235]:=
```
R = 10;

(* Try compiled version, but it is not faster... *)
```

In[255]:=
```
DoItCompile = Compile[{NEVENT, FIFOSIZE, α},
    Module[
     {FIFO = 0 (* entries in FIFO *),
      GENERATED = 0,
      LOST = 0,
      FOUND = 0},
     For[i = 0, i < NEVENT, i++,
      If[RandomReal[] < α/R, (* We have a hit *)
       GENERATED++;
       If[FIFO < FIFOSIZE, FIFO++, LOST++]
      ];
      If[Mod[i, R] == 0, (* we can read the FIFO *)
       If[FIFO > 0, FOUND++; FIFO--]
      ] ;
     ]; {GENERATED, FOUND, LOST}
    ]];
```

In[262]:=
```
Timing@DoItCompile[100 000, 4, 0.5]
```

Out[262]=
```
{1.86269, {5094, 5082, 12}}
```

In[257]:=
```
(* Normal version *)
```

In[331]:=
```
Clear[DoIt]; DoIt[NEVENT_, FIFOSIZE_, α_] :=
 Module[
  {FIFO = 0 (* entries in FIFO *),
   GENERATED = 0,
   LOST = 0,
   FOUND = 0,
   cut = α/R},
  For[i = 0, i < NEVENT, i++,
   If[RandomReal[] < cut, (* We have a hit *)
    GENERATED++;
    If[FIFO < FIFOSIZE, FIFO++, LOST++]
   ];
   If[Mod[i, R] == 0, (* we can read the FIFO *)
    If[FIFO > 0, FOUND++; FIFO--]
   ] ;
  ]; FOUND += FIFO; (* Empty the rest of the FIFO *)
  {α, GENERATED/NEVENT R // N, α NEVENT/R,
   GENERATED, LOST, LOST/GENERATED // N, GENERATED == FOUND + LOST}
  (* Show α, generated α, expected number of generated events,
  generated events, lost events, percentage, flag  that all is ok.*)
 ]
```

In[332]:=
```
Timing@DoIt[100 000, 3, 0.5]
```

Out[332]=
```
{0.339454, {0.5, 0.5088, 2500., 2544, 51, 0.0200472, True}}
```

Check how the result depends on R

In[339]:=
```
R = 2; DoIt[1 000 000, 3, 0.8]
```

Out[339]=
```
{0.8, 0.800058, 400 000., 400 029, 9832, 0.0245782, True}
```

In[338]:=
```
R = 10; DoIt[1 000 000, 3, 0.8]
```

Out[338]=
```
{0.8, 0.80035, 80 000., 80 035, 5355, 0.0669082, True}
```

In[340]:=
```
R = 100; DoIt[1 000 000, 3, 0.8]
```

Out[340]=
```
{0.8, 0.8013, 8000., 8013, 661, 0.082491, True}
```

(* We see: There is a small dependency of R. Obviously, R->∞ is perfect, but runs slow. For more discussion of this, see section 5*)

In[322]:=
```
R = 20;
```

2. fin > fread

```
(* Make a table for a 'small' FIFO *)
Table[DoIt[100 000, 5, α], {α, 0.1, 2, 0.1}] // TableForm
```

Out[323]//TableForm=

0.1	500.	484	0	0.	True
0.2	1000.	1039	0	0.	True
0.3	1500.	1534	0	0.	True
0.4	2000.	2017	1	0.000495786	True
0.5	2500.	2518	5	0.0019857	True
0.6	3000.	3025	13	0.00429752	True
0.7	3500.	3391	36	0.0106163	True
0.8	4000.	3938	113	0.0286948	True
0.9	4500.	4348	160	0.0367985	True
1.	5000.	4870	378	0.0776181	True
1.1	5500.	5434	708	0.130291	True
1.2	6000.	5992	1134	0.189252	True
1.3	6500.	6474	1537	0.237411	True
1.4	7000.	6970	1991	0.285653	True
1.5	7500.	7534	2552	0.338731	True
1.6	8000.	7995	3006	0.375985	True
1.7	8500.	8312	3316	0.398941	True
1.8	9000.	9053	4053	0.447697	True
1.9	9500.	9522	4518	0.47448	True
2.	10 000.	10 022	5018	0.500698	True

In[324]:=
```
(* And for a large 'large' FIFO *)
Table[DoIt[100 000, 50, α], {α, 0.1, 2, 0.1}] // TableForm
```

Out[324]//TableForm=

0.1	500.	506	0	0.	True
0.2	1000.	1051	0	0.	True
0.3	1500.	1466	0	0.	True
0.4	2000.	1986	0	0.	True
0.5	2500.	2541	0	0.	True
0.6	3000.	2881	0	0.	True
0.7	3500.	3614	0	0.	True
0.8	4000.	4070	0	0.	True
0.9	4500.	4480	0	0.	True
1.	5000.	4988	10	0.00200481	True
1.1	5500.	5461	421	0.0770921	True
1.2	6000.	6004	957	0.159394	True
1.3	6500.	6384	1337	0.20943	True
1.4	7000.	7107	2061	0.289996	True
1.5	7500.	7342	2294	0.312449	True
1.6	8000.	8015	2966	0.370056	True
1.7	8500.	8486	3437	0.40502	True
1.8	9000.	9016	3968	0.440106	True
1.9	9500.	9499	4455	0.468997	True
2.	10 000.	10 084	5035	0.499306	True

Result:

- For α>1, losses become significant, even a large FIFO does not help.
- The losses are not infinite, because we still get hits out of the FIFO, but

that rate is limited.
- For α=2, for example, reading is half as fast as writing and we loose half of the hits

3. α=0.5

In[325]:=
```
(* Try some FIFO sizes *)
Table[DoIt[1 000 000, F, 0.5], {F, 1, 8, 1}] // TableForm
```

Out[325]//TableForm=

0.5	25 000.	24 985	5100	0.204122	True
0.5	25 000.	24 962	1295	0.0518789	True
0.5	25 000.	24 904	352	0.0141343	True
0.5	25 000.	25 064	90	0.00359081	True
0.5	25 000.	24 959	13	0.000520854	True
0.5	25 000.	24 968	5	0.000200256	True
0.5	25 000.	25 045	2	0.0000798563	True
0.5	25 000.	25 109	0	0.	True

Even for a FIFO with only one entry, we lose only 20%. To get down to 1%, we need M=3, and for 0.1% , M=5 is sufficient. FIFOs do not have to be large!

In[326]:=
```
(* Do it again for a higher hit rate *)
Table[DoIt[1 000 000, F, 0.8], {F, 1, 8, 1}] // TableForm
```

Out[326]//TableForm=

0.8	40 000.	39 821	12 022	0.301901	True
0.8	40 000.	40 409	5567	0.137766	True
0.8	40 000.	40 168	3078	0.0766282	True
0.8	40 000.	39 933	1656	0.0414695	True
0.8	40 000.	39 977	1003	0.0250894	True
0.8	40 000.	40 256	672	0.0166932	True
0.8	40 000.	40 119	365	0.00909793	True
0.8	40 000.	40 255	263	0.00653335	True

4. Merge Hits

As stated in the introduction, the result should in principle only depend on α, which is the same for both cases discussed on question 4. In a practical system, there is a time discretization, so there is real the effect of R, discussed below, which will make a small difference for the 2 cases: IF we have many FIFOs, we will typically read them one after the other, so R>1. The case in question 4 comes closer to R=1, so that the losses decrease. We have to compare the case for some R with R/2 to get an answer on question 4.

5. Effect of R

Let us look at 2 extreme R values:
- If R=1, we may have a hit during one readout time step (which we have set to 1),
or not. If we have one, the hit will be read, so there cannot be ANY losses by fluctuations of the FIFO content, independent of (our) α. But note that we do not really simulate the correct thing for α>1, because we cannot generate enough hits.

- If R is large, we MAY have several hits (and thus increase FIFO content) during one readout time step.

To demonstrate that, we repeat (2) for different R values:

```
(* 'Large' R, 'truth' *)
R = 20;
Table[DoIt[100 000, 3, α], {α, 0.5, 1.4, 0.1}] // TableForm
```

Out[344]//TableForm=

0.5	0.4976	2500.	2488	26	0.0104502	True
0.6	0.6068	3000.	3034	118	0.0388926	True
0.7	0.681	3500.	3405	146	0.0428781	True
0.8	0.8178	4000.	4089	354	0.0865737	True
0.9	0.9134	4500.	4567	510	0.111671	True
1.	1.0014	5000.	5007	728	0.145396	True
1.1	1.1224	5500.	5612	1104	0.196721	True
1.2	1.1972	6000.	5986	1367	0.228366	True
1.3	1.2962	6500.	6481	1752	0.270329	True
1.4	1.4428	7000.	7214	2371	0.328666	True

```
(* R=1 *)
R = 1;
Table[DoIt[100 000, 3, α], {α, 0.5, 1.4, 0.1}] // TableForm
```

Out[345]//TableForm=

0.5	0.50117	50 000.	50 117	0	0.	True
0.6	0.59926	60 000.	59 926	0	0.	True
0.7	0.70139	70 000.	70 139	0	0.	True
0.8	0.79869	80 000.	79 869	0	0.	True
0.9	0.90144	90 000.	90 144	0	0.	True
1.	1.	100 000.	100 000	0	0.	True
1.1	1.	110 000.	100 000	0	0.	True
1.2	1.	120 000.	100 000	0	0.	True
1.3	1.	130 000.	100 000	0	0.	True
1.4	1.	140 000.	100 000	0	0.	True

We confirm that we have NO LOSS at all, but the generated rate α gets stuck at 1.

In[346]:=
```
(* R=2 *)
R = 2;
Table[DoIt[100 000, 3, α], {α, 0.5, 1.4, 0.1}] // TableForm
```

Out[346]//TableForm=

0.5	0.4967	25 000.	24 835	30	0.00120797	True
0.6	0.60312	30 000.	30 156	117	0.00387982	True
0.7	0.70406	35 000.	35 203	412	0.0117035	True
0.8	0.79972	40 000.	39 986	983	0.0245836	True
0.9	0.89932	45 000.	44 966	2101	0.0467242	True
1.	0.99698	50 000.	49 849	4092	0.0820879	True
1.1	1.101	55 000.	55 050	7205	0.130881	True
1.2	1.19988	60 000.	59 994	10 942	0.182385	True
1.3	1.3023	65 000.	65 115	15 492	0.237918	True
1.4	1.39674	70 000.	69 837	19 953	0.285708	True

For R=2 it's better, but the losses are still a bit underestimated.