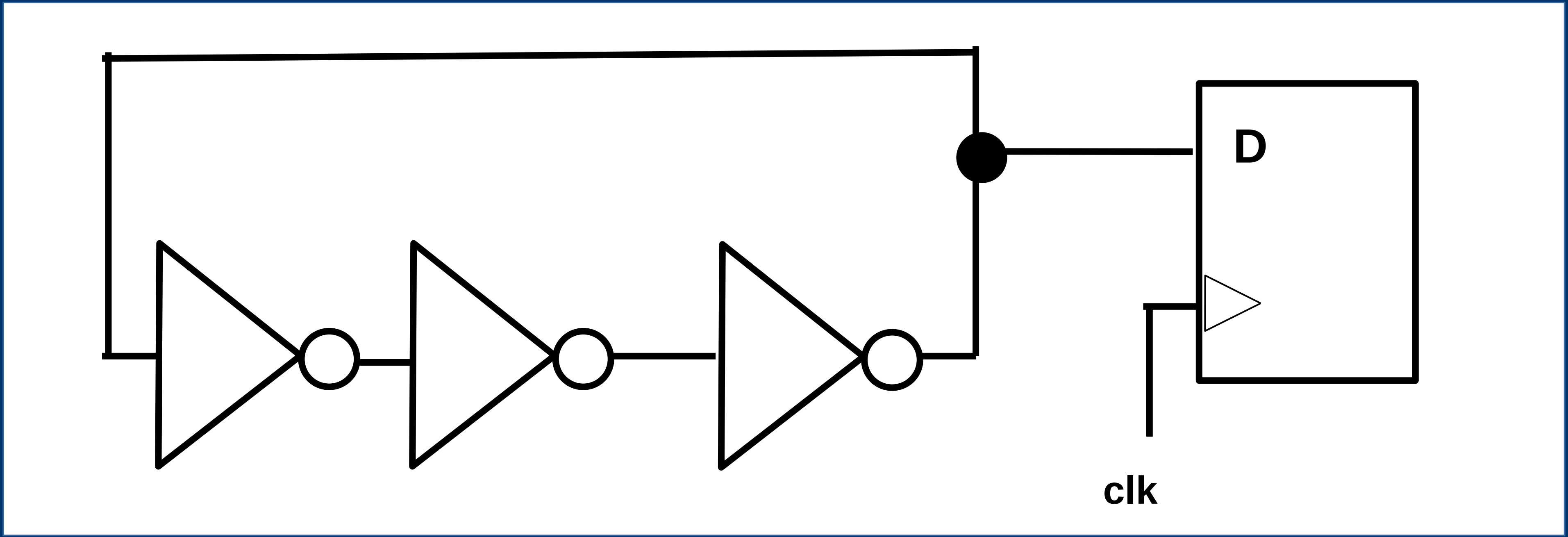
How (Not) To End Up With Dependent Random Bits

Markus Dichtl

Markus.Dichtl@gmail.com



This seems to be a completely trivial and innocent scenario: A ring oscillator is sampled by a D-flip-flop in order to generate random bits.

But wait a minute: we have to respect setup and hold times if we want the flip-flop to work correctly!

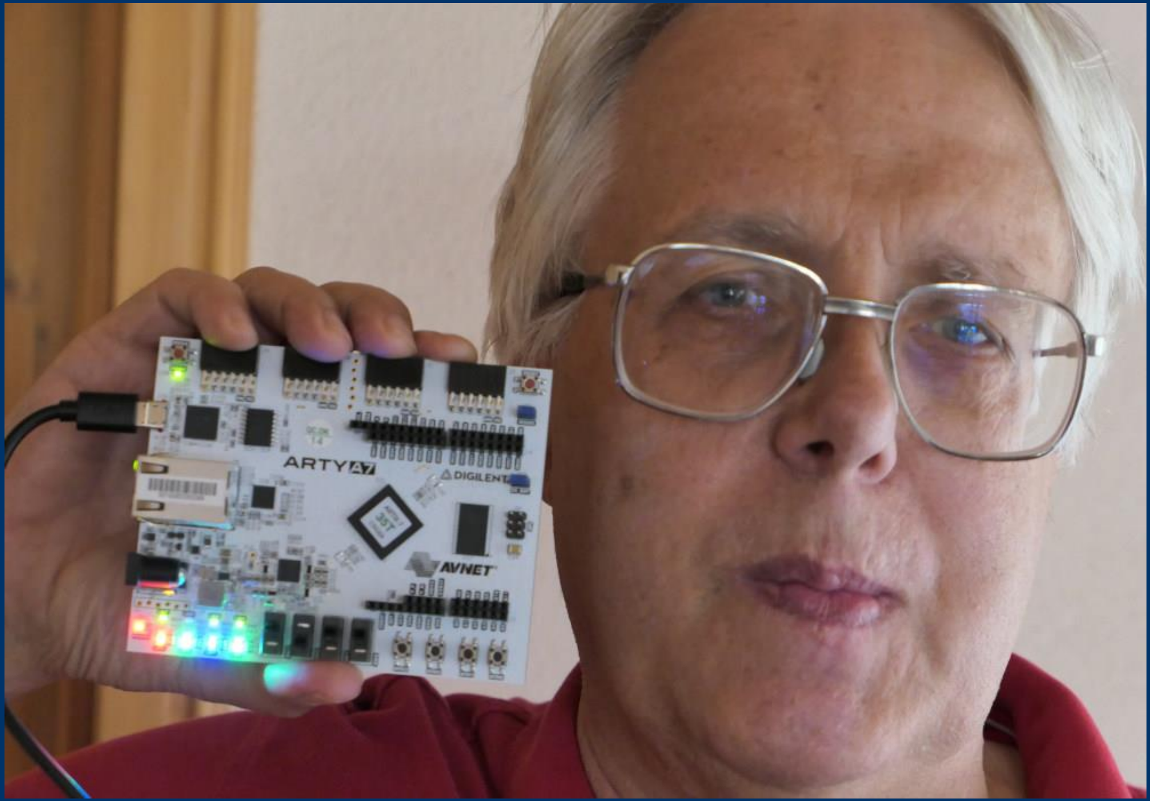
Now one could argue: Forget about it, if we happen to violate this condition, we get into a metastable state, but in what bit this state ends is just an additional source of randomness.

But this argument neglects one important question:

Does the state after metastability depend on the previous bit stored in the flip-flop?

This talk tries to answer this question based on experimental results.

As I am retired now, I had to come up with a possibility to keep playing with FPGAs, so I bought an Xilinx Arty A7-35 board based on an Artix FPGA.
All subsequent experimental results come from this board.



Ring oscillator of length 31,
Restart from a fixed state with one gate implemented as NAND
All 31 bits sampled 10 ns after restart, repeated 100000 times

```
0      100000   0   100000 0       100000   0   100000   0   100000
0      100000   0   100000 0       100000   0   100000   0
       100000
```

Only 10 subsequent numbers of 0 bits shown
Top row: flip-flops preloaded with 0s
Bottom row: flip-flops preloaded with 1s

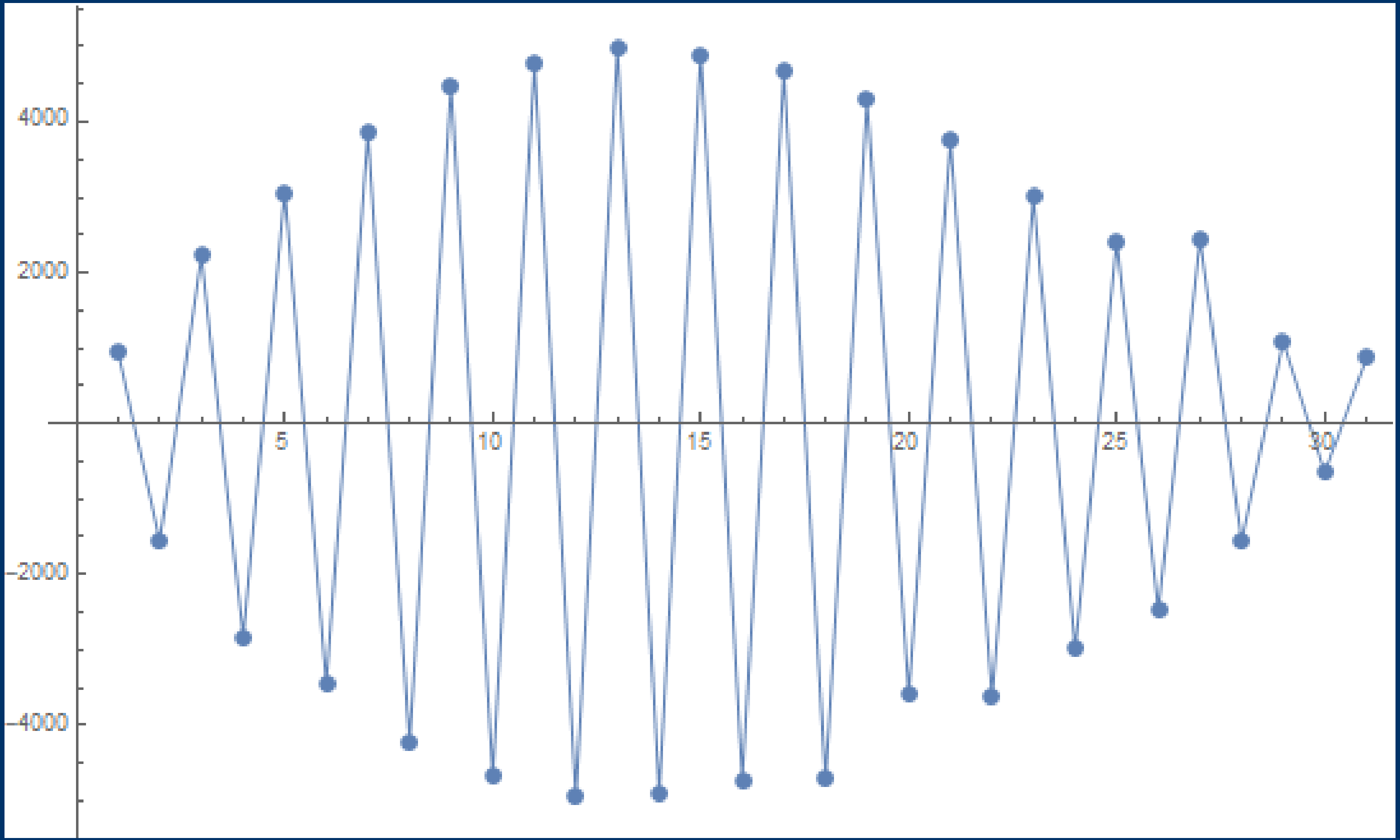
(I have to admit, a not too interesting result, but...)

Ring oscillator of length 31,
Restart from a fixed state with one gate implemented as NAND
All 31 bits sampled 100 ns after restart, repeated 100000 times

0	100000	5	97029	14977	55162	54741	25651	83051	2
0	100000	14	95716	18316	52790	56236	22464	84467	0

Only 10 subsequent numbers of 0 bits shown
Top row: flip-flops preloaded with 0s
Bottom row: flip-flops preloaded with 1s

RO of length 31, all bits sampled 890 ns after restart, 100000 trials for each preload value



x-axis: subsequent bits, y-axis: difference of 0 bits observed when FF preloaded with 0 or 1

With a difference of 4.99% of the probability to sample a 0 bit for 0/1 as prior flip-flop state, these dependencies can be quite strong, but most people do not sample their ROs so fast (though it is good idea in order to maximize entropy harvest)

What about a very conservative approach of sampling only after more than a million ring oscillator periods? I tried this with ring oscillators of length 3. The ring oscillators were not reset, but running freely.

I measured 9 ROs of length 3 running freely, again sampling all inverters.

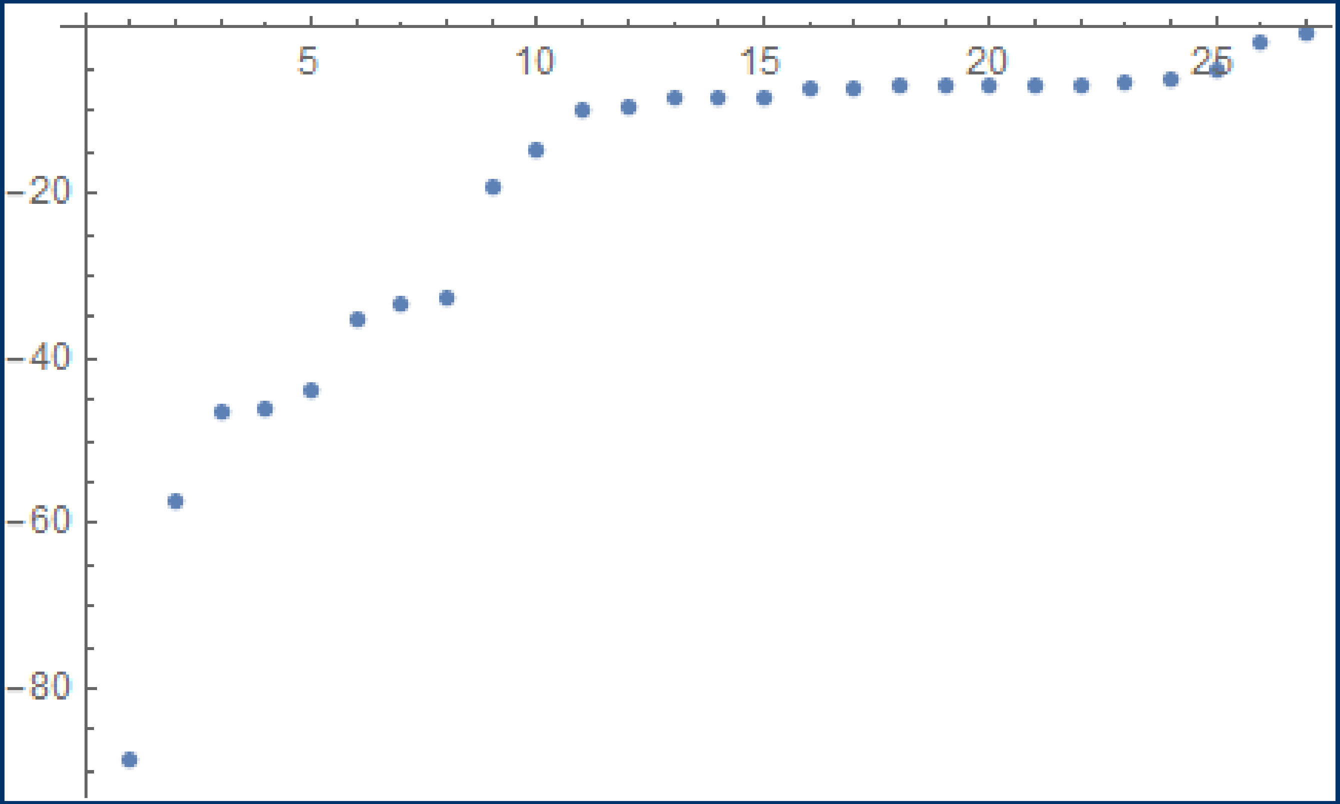
Both for 0 and 1 as preload state 3 million samples were taken.

All flip-flops tend to change the preloaded state.

1502085 1506624	1507301 1519764	1502994 1508370
1513431 1518449	1438208 1450279	1461641 1468447
1507615 1513077	1453096 1465461	1546903 1551930
1493325 1501211	1475761 1480265	1460965 1471782
1520391 1524800	1480520 1484309	1529420 1533832
1455158 1456922	1494746 1499731	1470798 1481174
1515804 1520231	1485485 1489716	1527993 1541858
1510403 1514842	1505562 1505981	1608490 1618957
1511427 1516003	1535601 1539973	1433726 1451027

Numbers of 0 sampled , left columns preloaded with 0

Are the differences statistically significant? Yes!



All percentage levels of acceptance except 2 less than 10^{-5}

The sorted decimal logarithms for the p-values of the number of 0s for prelaoding with 1, according to the binomial distribution with the bias from the bits with preloading 0

Now some people think that XORing hundreds of ring oscillator outputs could lead to provably secure random bits.

So I looked at the XORs of the outputs of 50 freely running ROs of length 3, again for all 3 inverters, again sampling only after more than a million RO periods, again 3 million samples per preload value.

1599258 1626262 1654136 1681490 1551179 1573764

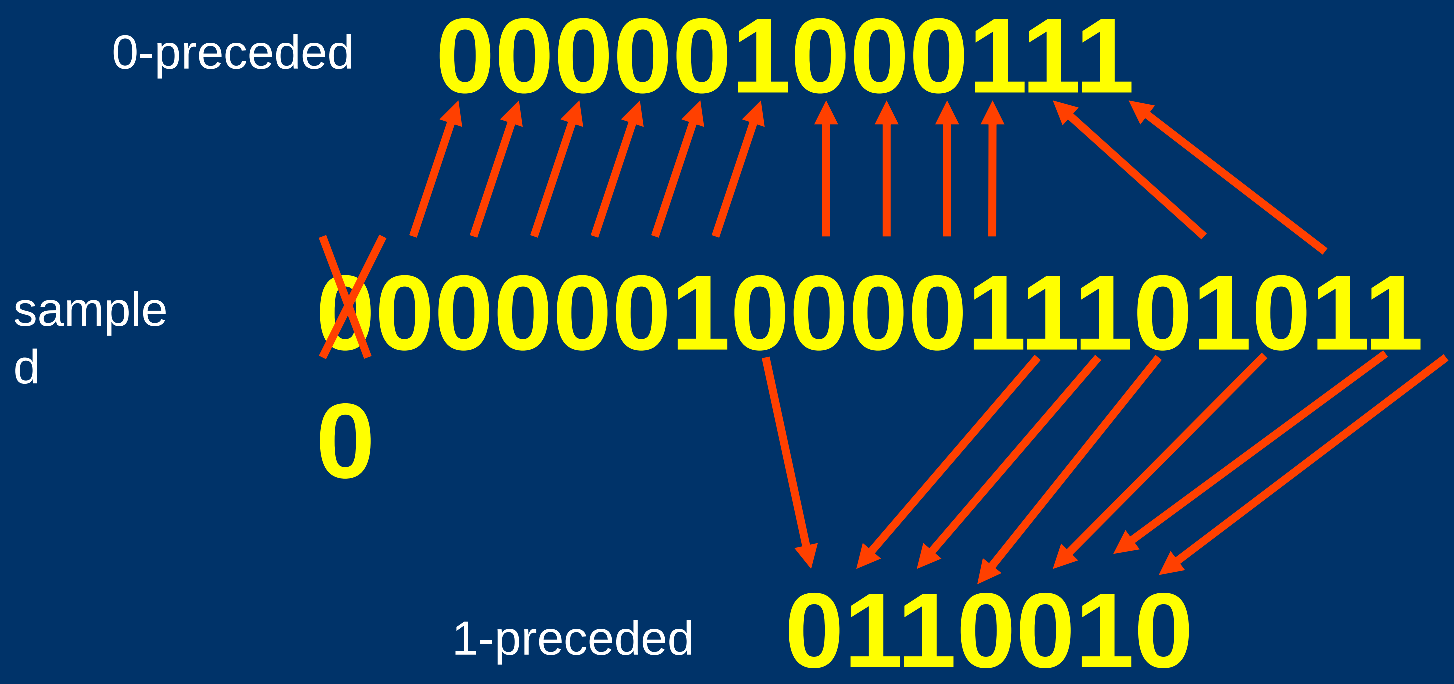
The tendency to change state is considerably stronger than for simple RO outputs (decimal logarithms of the p-values (as before) -214.141,-221.149,-149.735)

How to fix the problem?

The easiest approach: Preload the sampling flip-flop

(performance penalty for very fast sampling)

A bit more involved: Post process the bits preceded by 0 separately from those preceded by 1



For both subsequences, it could make sense to assume independent bits, but of course one has to assume different but fixed biases. Any postprocessing for independent bits, like von Neumann or Juels et al. may be applied to both subsequences.

Conclusion

Once more, generating true random number has turned out to be trickier than assumed.