USC University of Southern California

qSyn 3.0 update

Sumeet Shirgure

USC

Introduction and overview

qSyn is the logic synthesis tool designed as a part of the qPALACE suite

- Consists of two tools – qYosys, and qABC
- qYosys performs RTL logic synthesis given a high level design.
- qABC maps RTL to the given SFQ technology.

New features in qABC

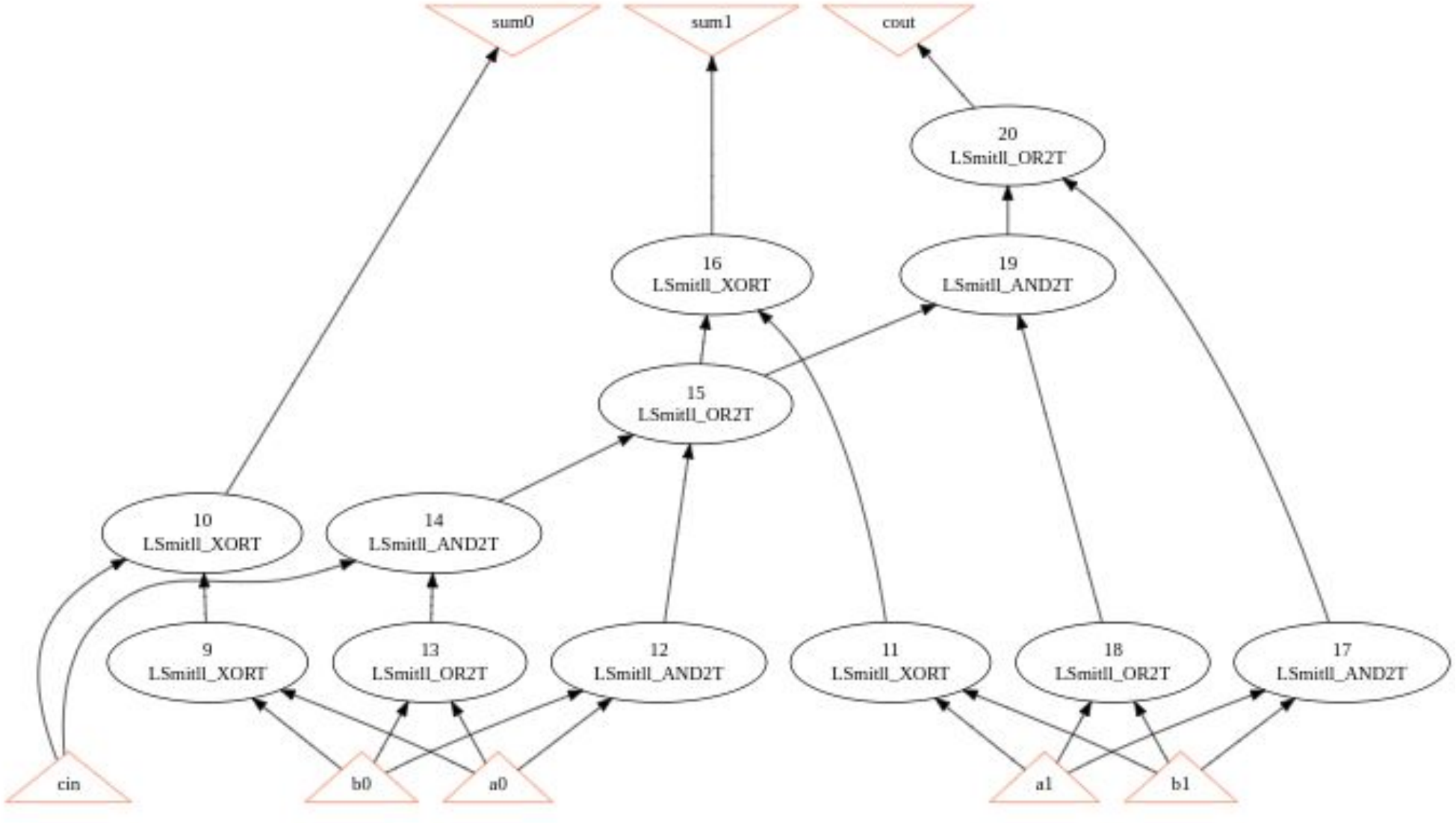
- Improved path balancing algorithm.
- Generalized sequential synthesis method.
- Improved and simplified software implementation of all modules.

Full path balancing

All RSFQ gates are synchronous. To get the correct behaviour, input SFQ pulses must arrive within a specific time interval.

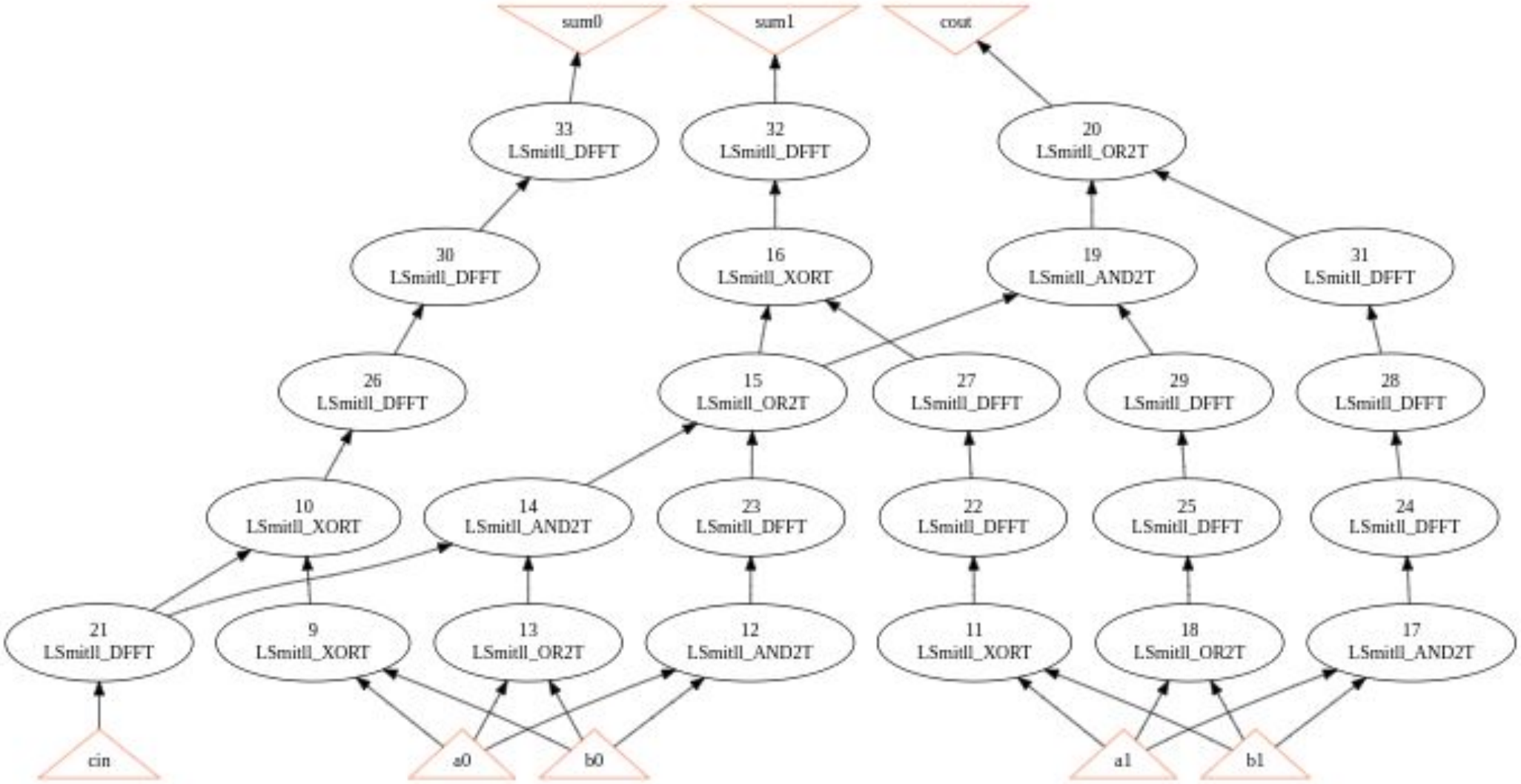
We enforce this by making all paths from all inputs have the same number of gates using SFQ destructive readout DFFs.

Full path balancing - example



USC

Full path balancing - example



USC

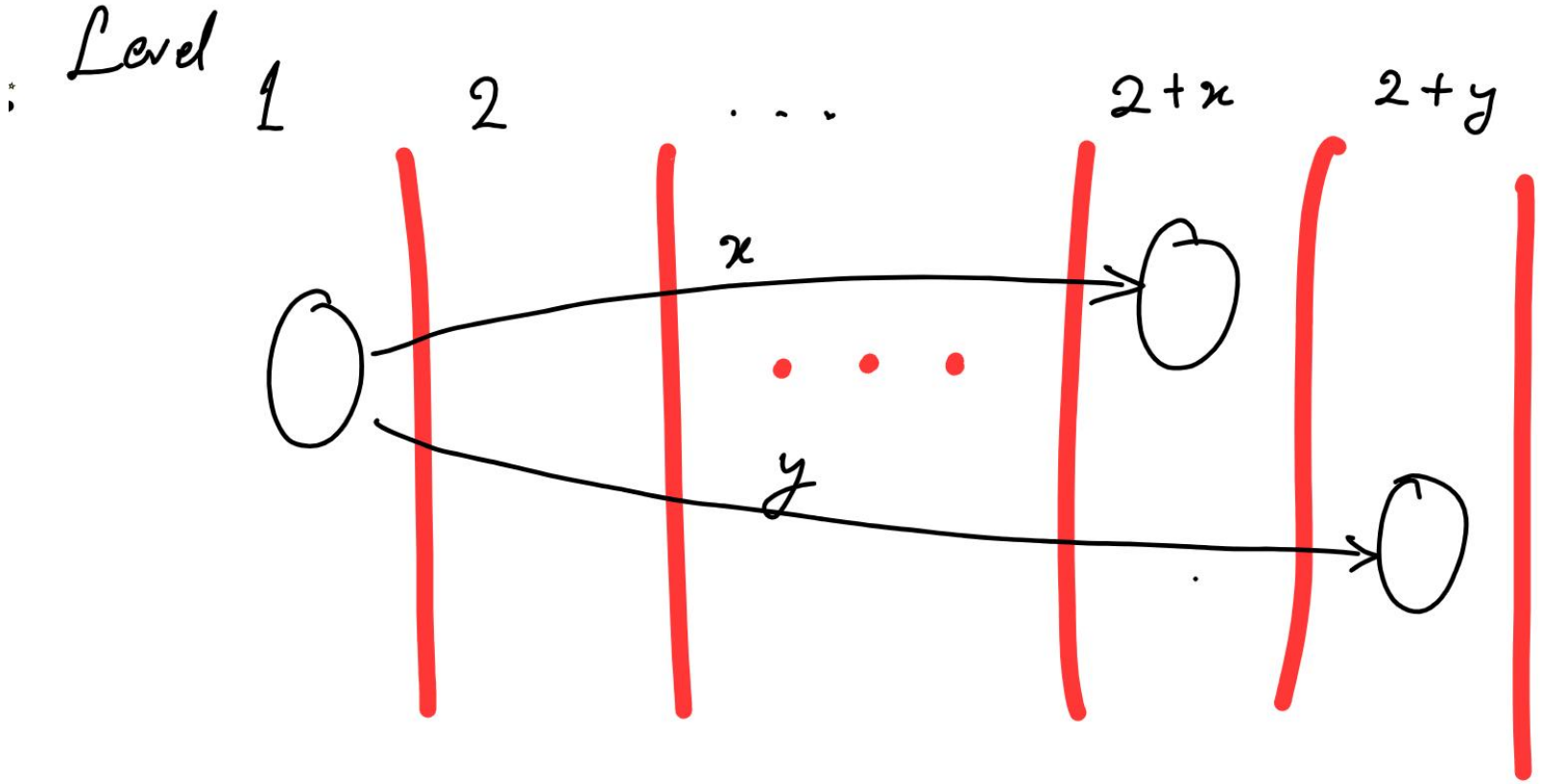
Full path balancing - idea

The earlier version of qABC inserted (level_difference - 1) many DFFs between every node and its fan-in.

This sum of level differences is the same heuristic value that is minimized by dynamic programming.

Full path balancing - idea

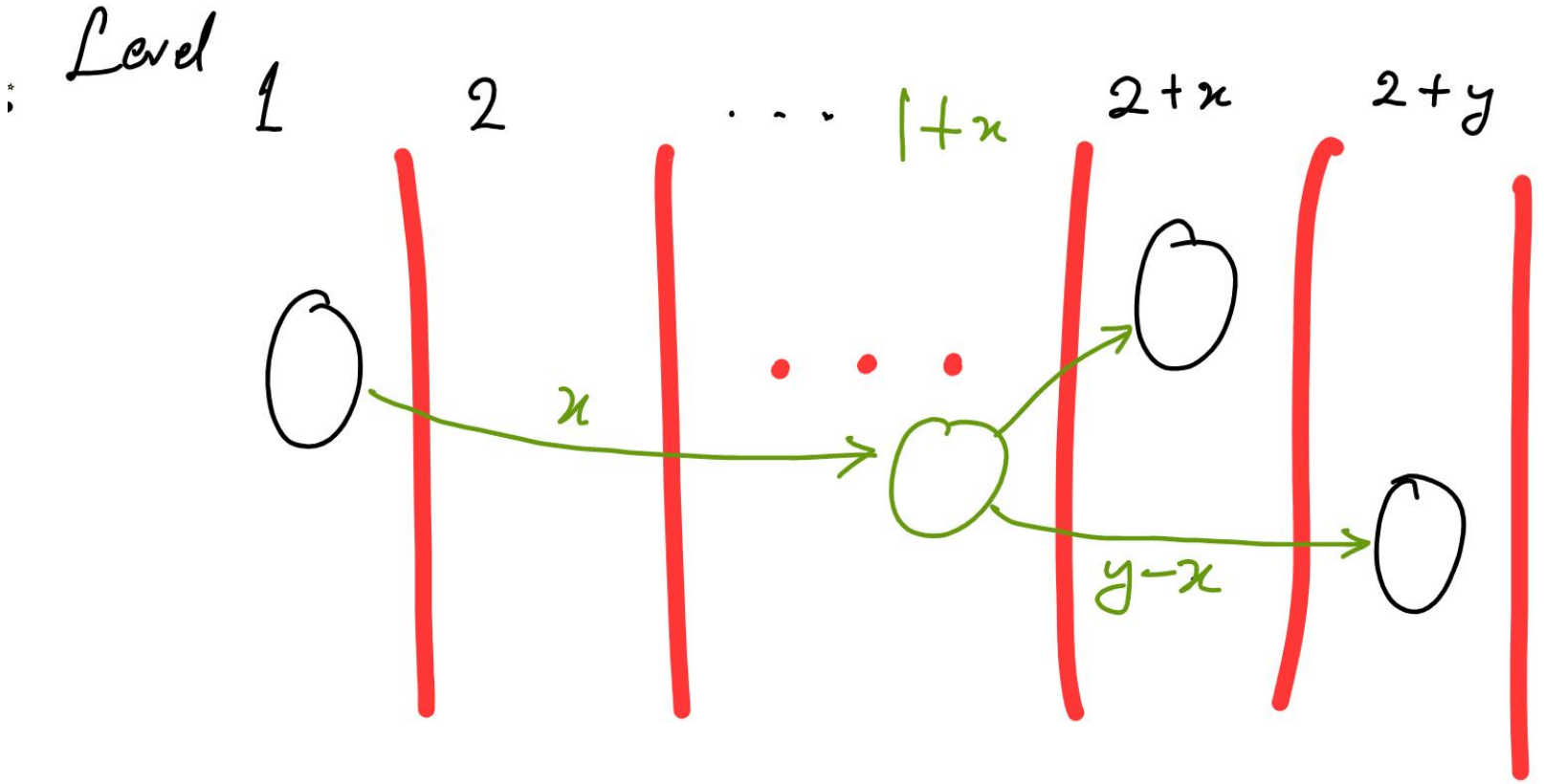
Consider the case where an SFQ signal is duplicated and fed to multiple gates situated at different levels ahead of it.



USC

Full path balancing - idea

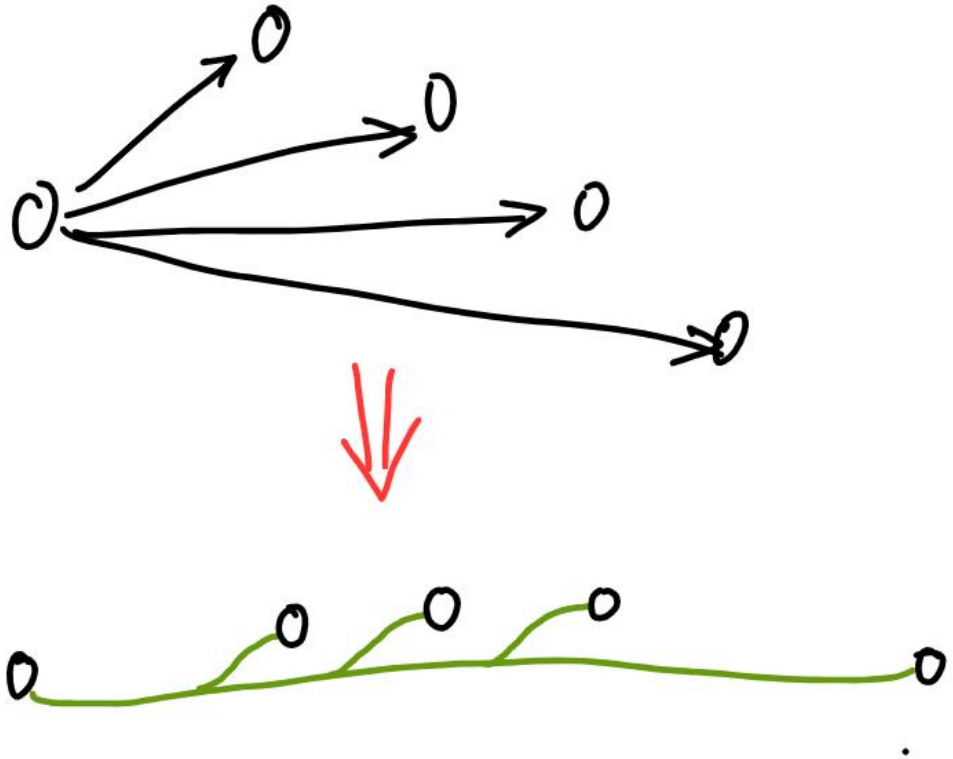
Can be improved like so.

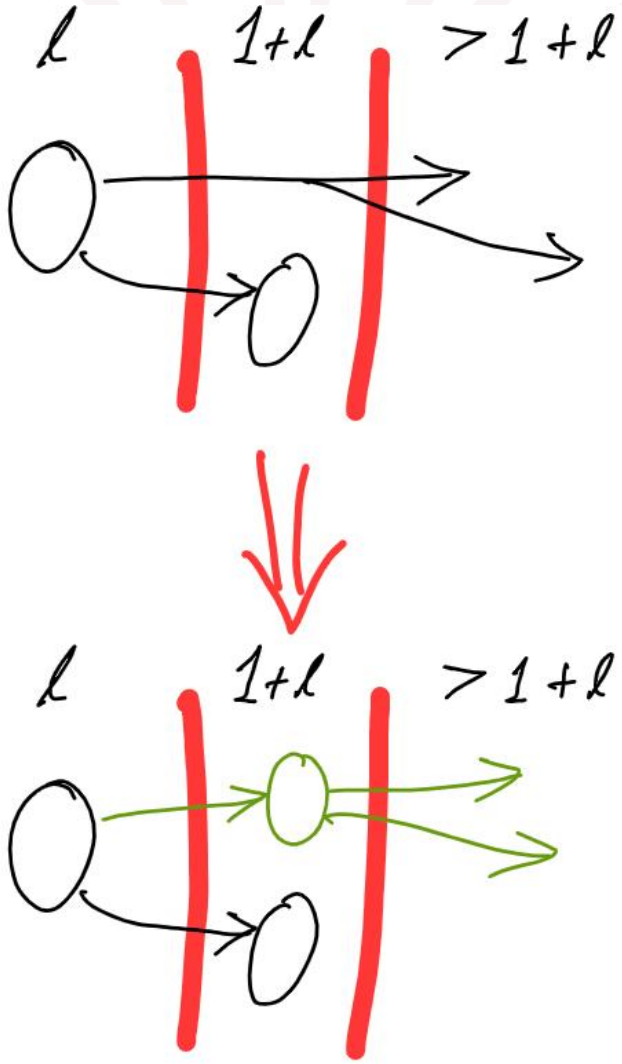


USC

Full path balancing - algorithm

The algorithm “looks like” this :





USC

Full path balancing - metric

If the number of DFFs inserted by this algorithm is ‘N’, and the sum of level differences is ‘SLD’ : clearly N ≤ SLD

Measuring improvement :
(SLD-N)/SLD, – “DFF gain”
(SLD-N)/(total # of gates before this optimization), – “Total gain”

Full path balancing - benchmark results

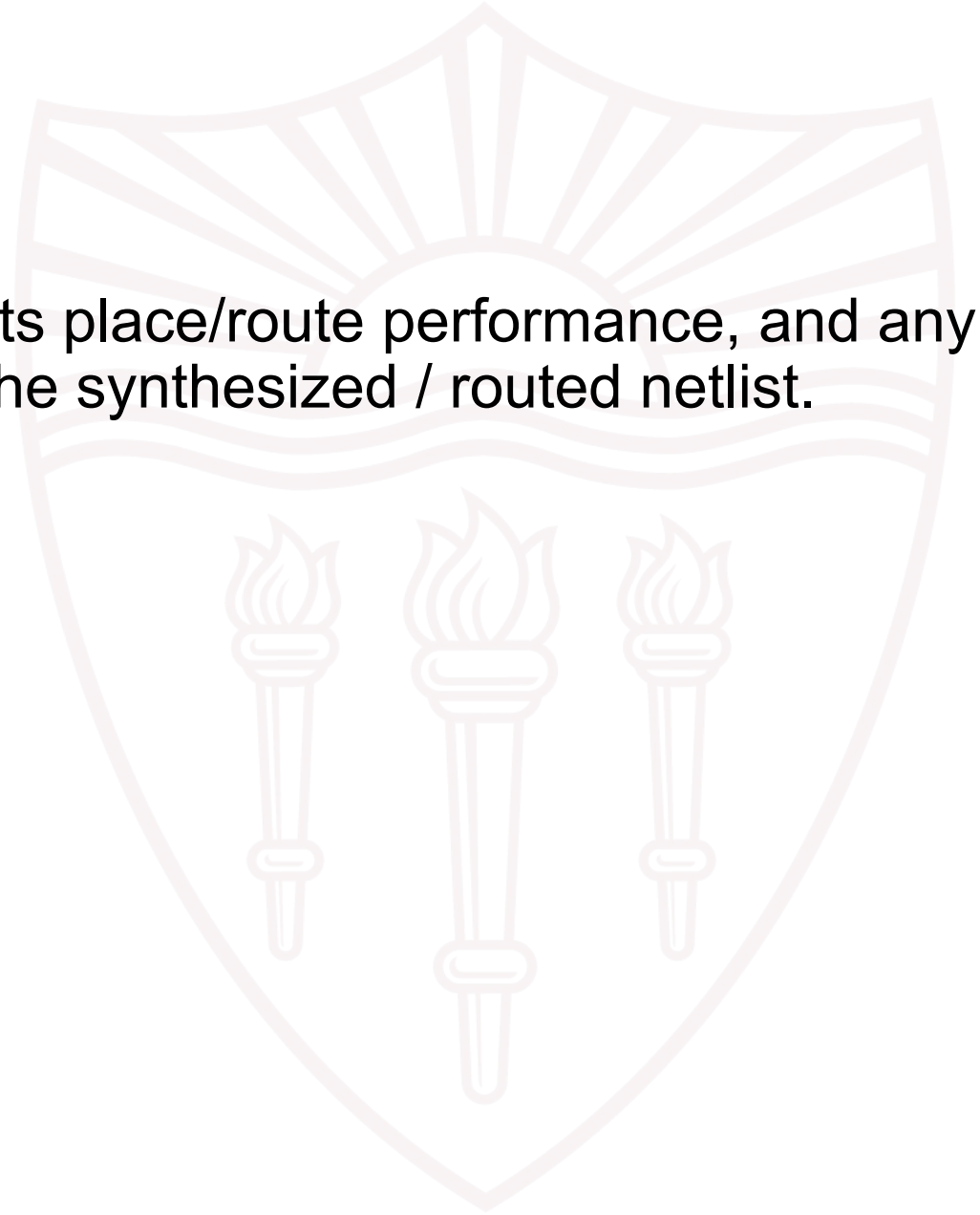
Benchmark	DFF gain (%)	Total gain (%)
EPFL adder	12.52	12.01
EPFL multiplier	22.72	19.57
EPFL sqrt	67.18	66.97
EPFL decoder	40.08	9.35
ISCAS85 c499	13.77	6.65
ISCAS85 c880	21.68	14.14
ISCAS85 c2670	49.91	38.93
ISCAS89 s420	9.00	4.29
ISCAS89 s526	28.7	14.35
ISCAS89 s298	39.8	22.68

Full path balancing - benchmark results

Benchmark	DFF gain (%)		Total gain (%)	
	Average	Max	Average	Max
EPFL arithmetic	43.32	68.21	29.31	68.09
EPFL random_control	31.02	68.27	20.16	64.86
ISCAS85	21.53	49.91	14.39	38.93
ISCAS89	20.57	39.82	11.51	22.68

Full path balancing - impact

Reducing gate count also potentially impacts place/route performance, and any simulation performed by the later tools on the synthesized / routed netlist.



Logic synthesis - possible idea

$N \leq SLD$

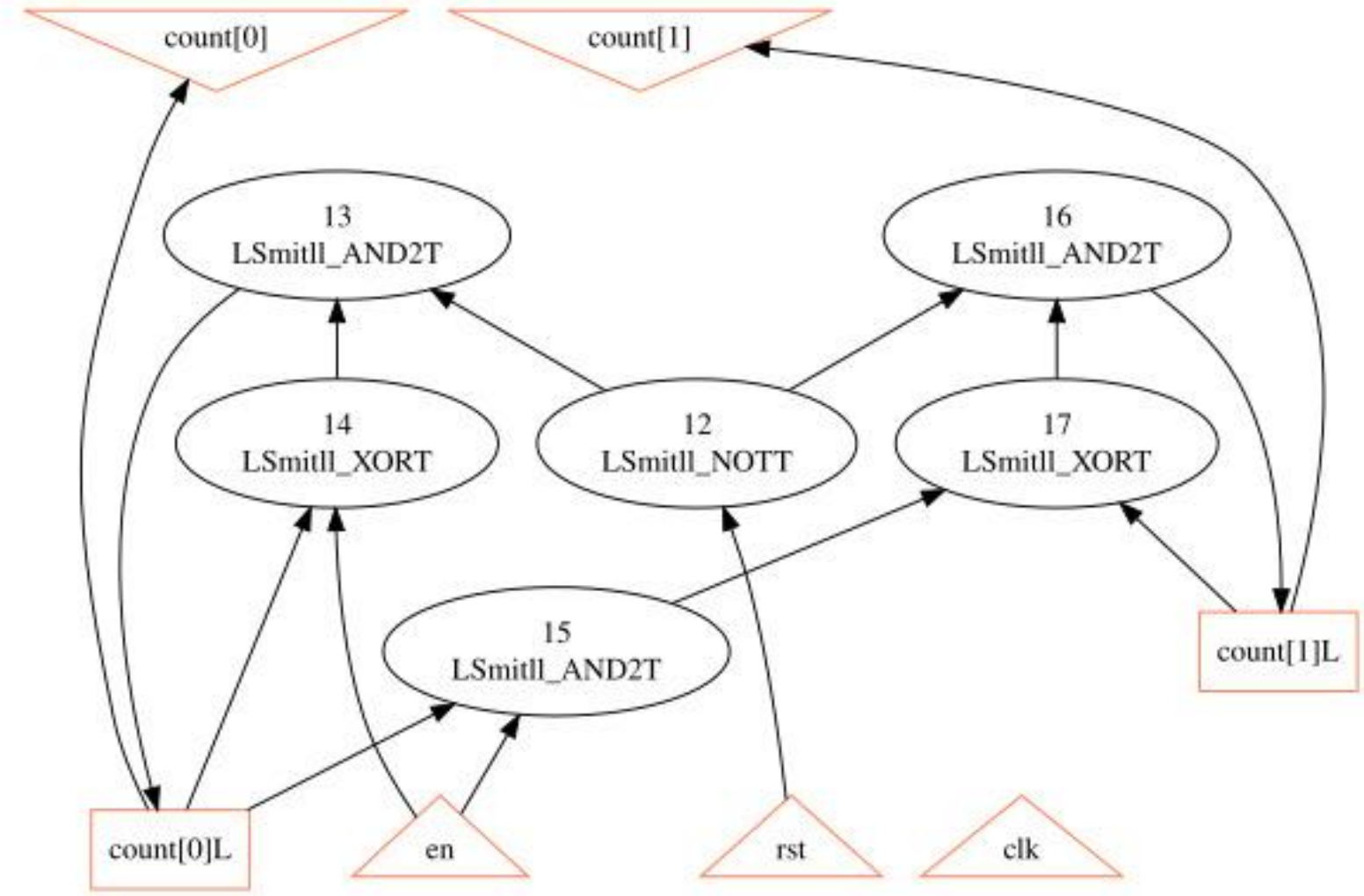
While SLD is an easy to calculate heuristic, attempting to minimize N is better than attempting to minimize the upper bound.

Perhaps try to minimize N directly before path balancing by modifying the dynamic program?

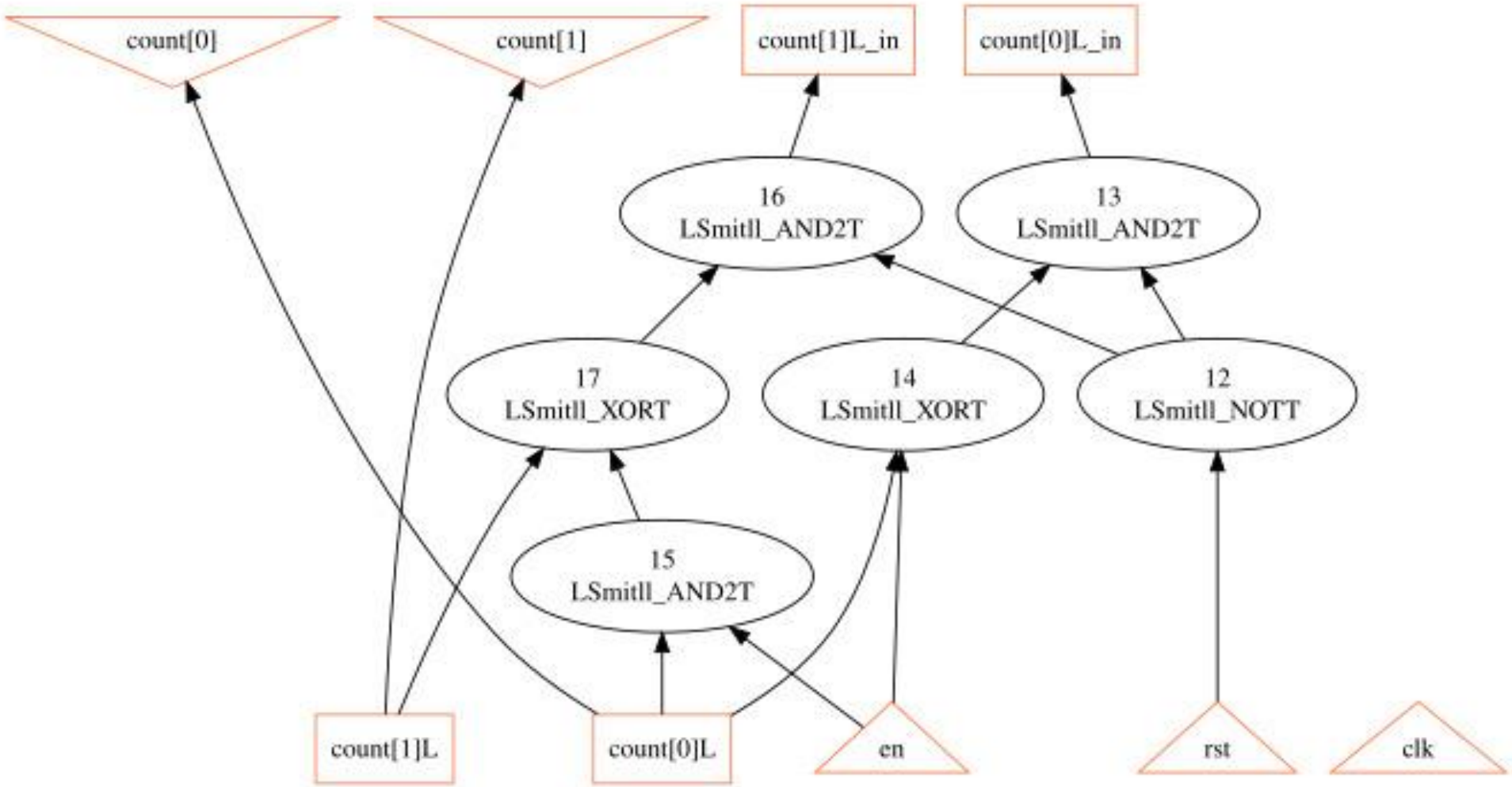
Sequential logic in RSFQ.

- Level assignment is now generalized to handle any sequential circuit.
- Data initiation interval cuts the data processing rate.
- This interval is itself determined by the depth of the combinational part.

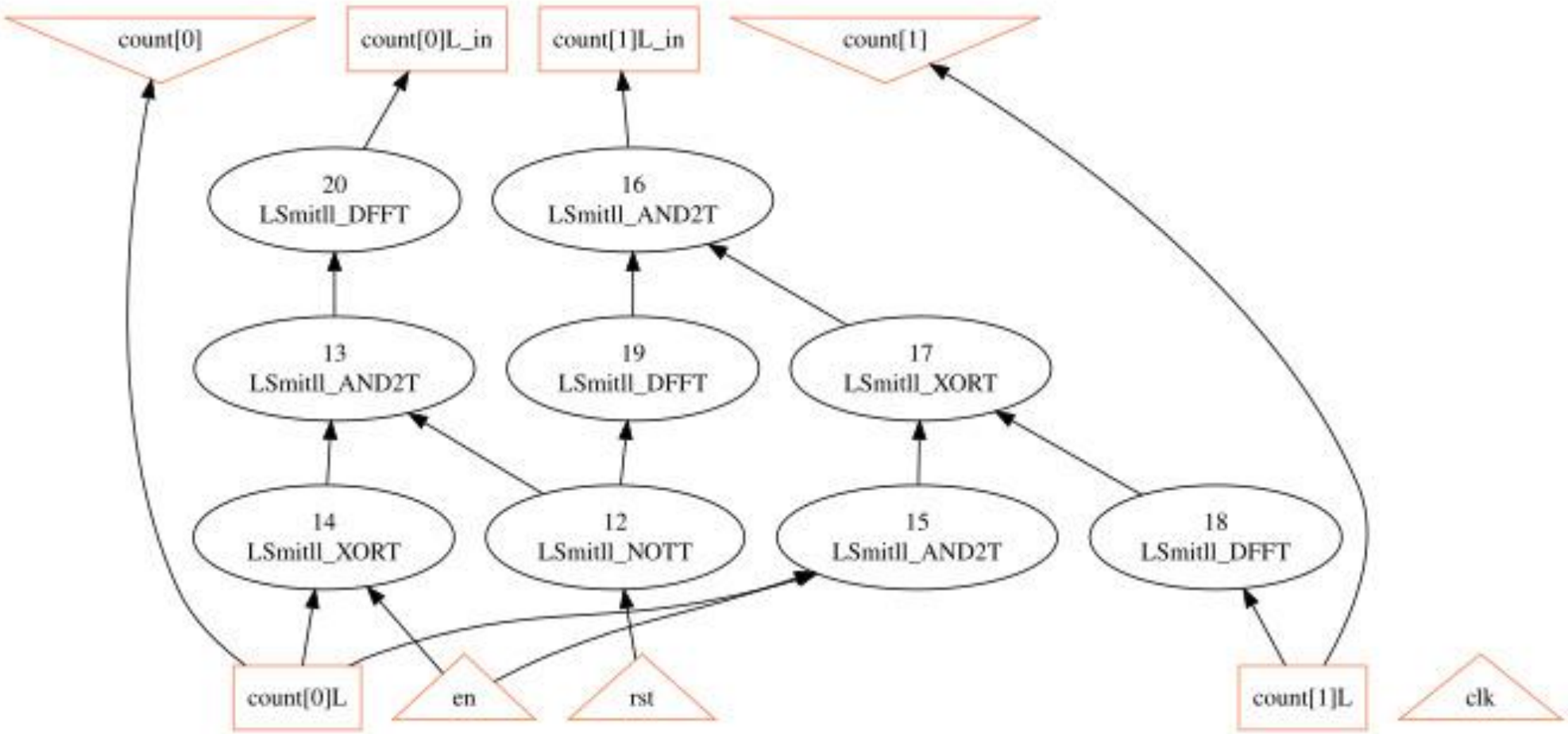
Sequential RSFQ - example



Sequential RSFQ - example

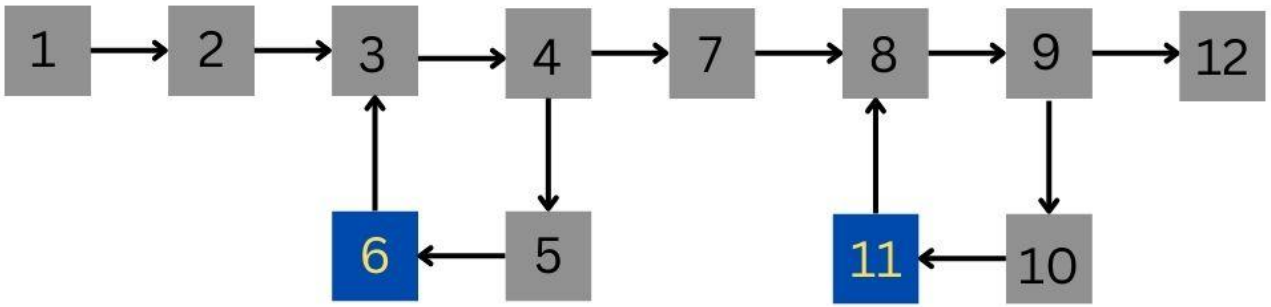


Sequential RSFQ - path balancing



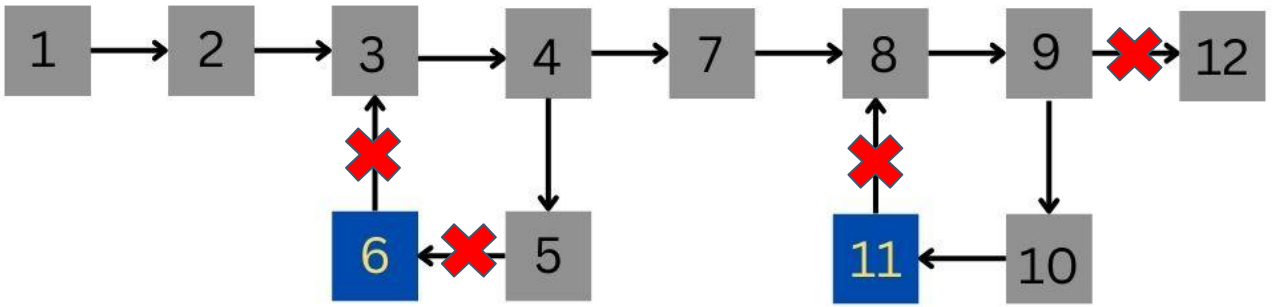
USC

Sequential RSFQ - counterexample



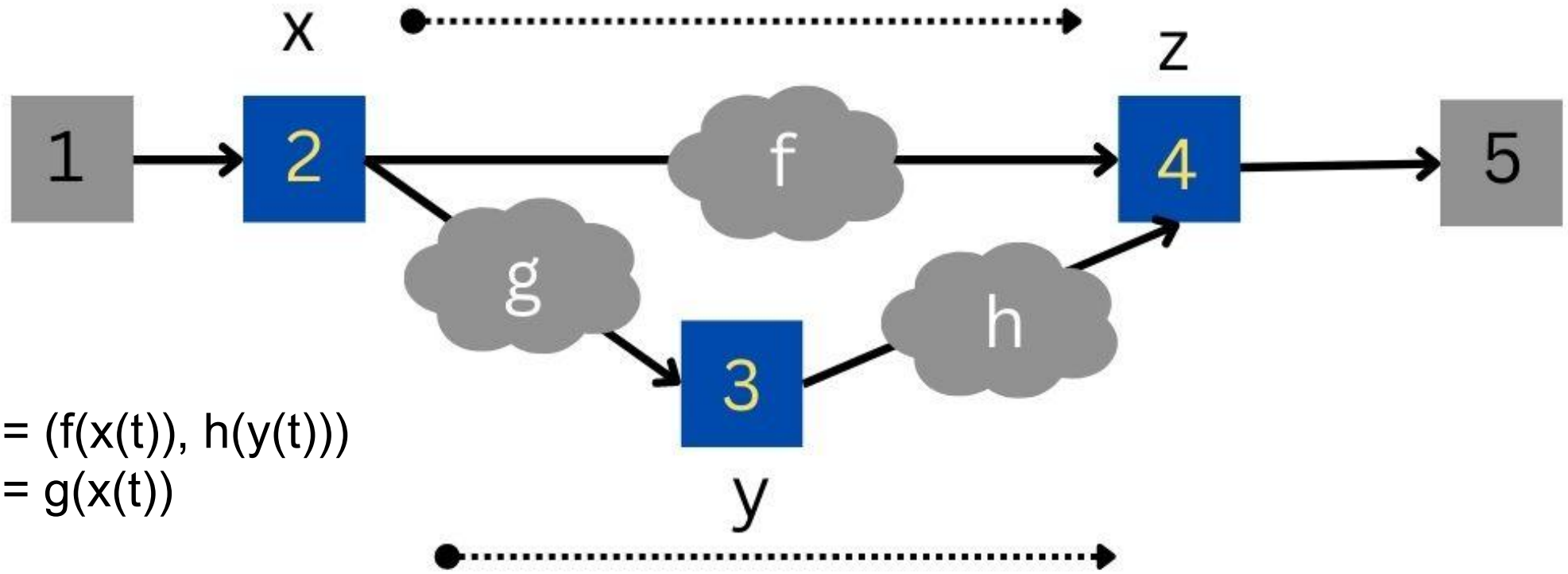
USC

Sequential RSFQ - counterexample



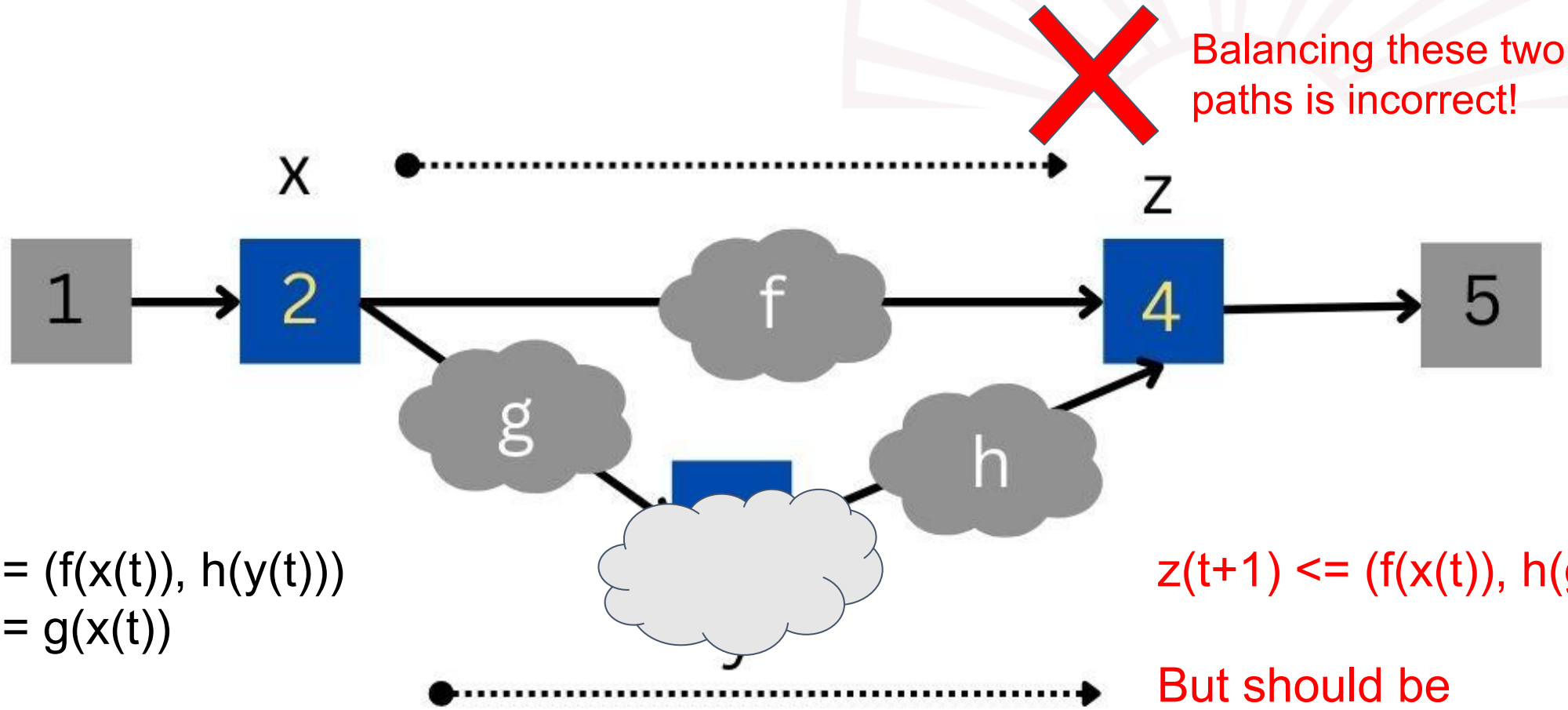
USC

Sequential RSFQ - weak cycles



z(t+1) <= (f(x(t)), h(y(t)))
y(t+1) <= g(x(t))

Sequential RSFQ - weak cycles



z(t+1) <= (f(x(t)), h(y(t)))
y(t+1) <= g(x(t))

z(t+1) <= (f(x(t)), h(g(x(t))))

But should be
(f(x(t)), h(g(x(t-1))))

Sequential RSFQ - drawbacks

Critical path length determines data initiation interval.

E.g 4 cycles for the 2 bit counter.

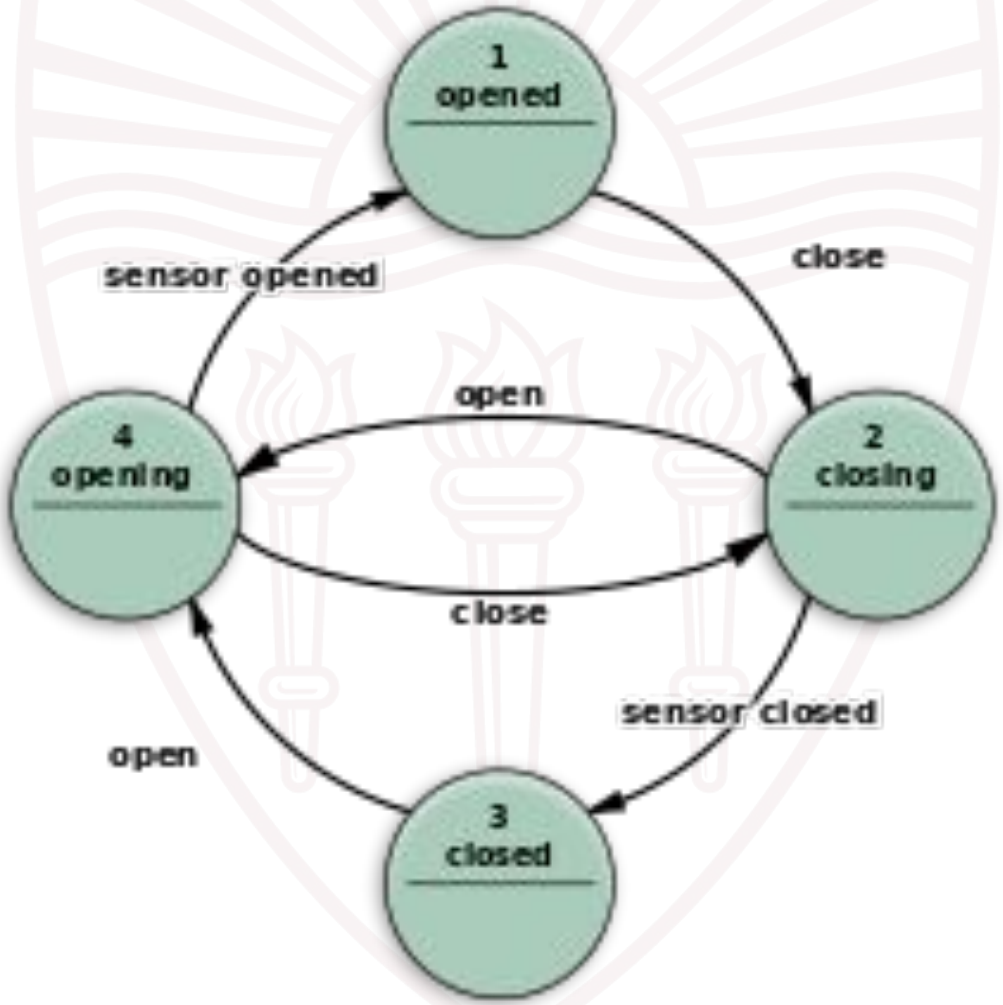
This effectively cuts the data processing rate by 4.

This effect gets more serious when the combinational parts grow deeper, e.g a circuit with just 30 levels of combinational depth can make a 30 GHz chip effectively process data at 1GHz.

The fundamental challenge in making sequential with higher frequency is reducing this critical path length.

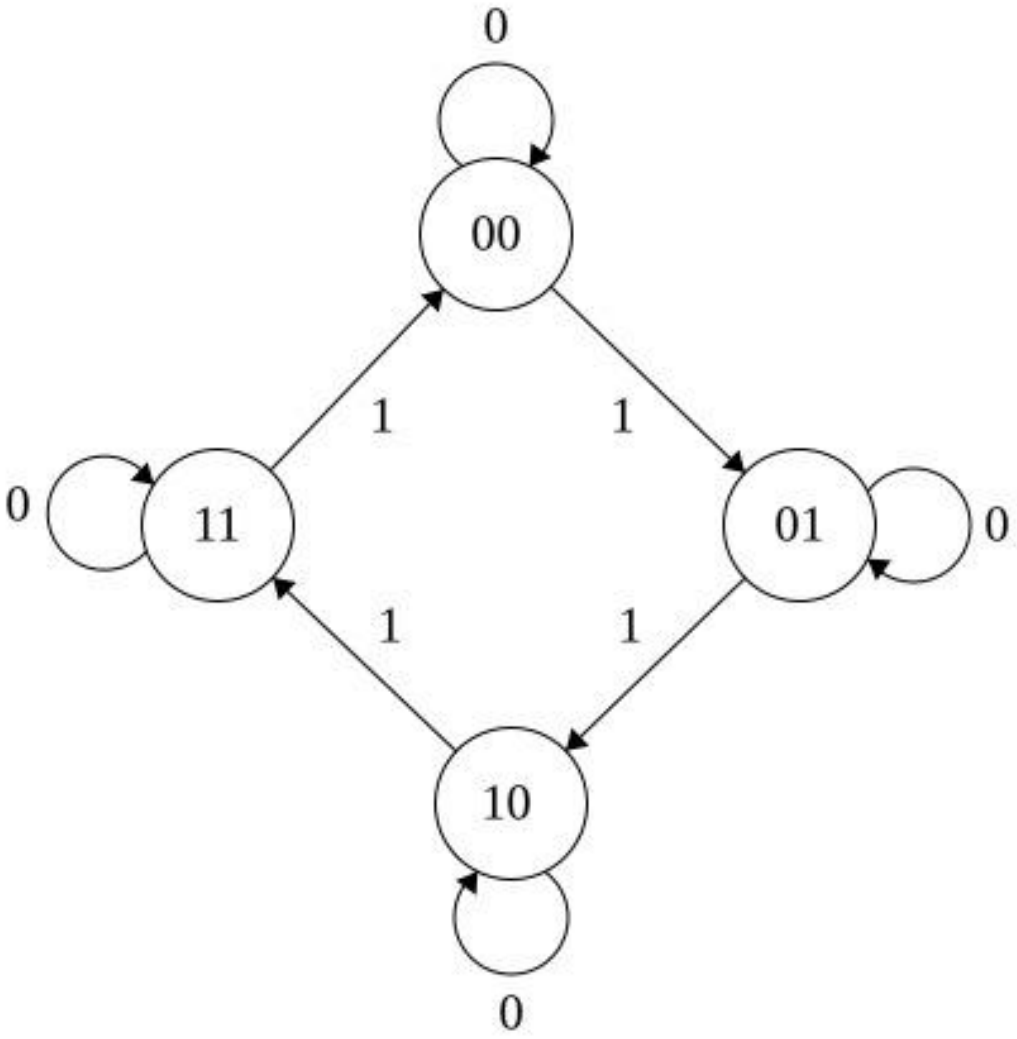
Sequential RSFQ - FSMs

One way to understand sequential circuits is through finite state machines.



Sequential RSFQ - FSMs

It might be possible to bypass RTL mapping (qYosys) and synthesize sequential behaviour by directly using FSM descriptions.

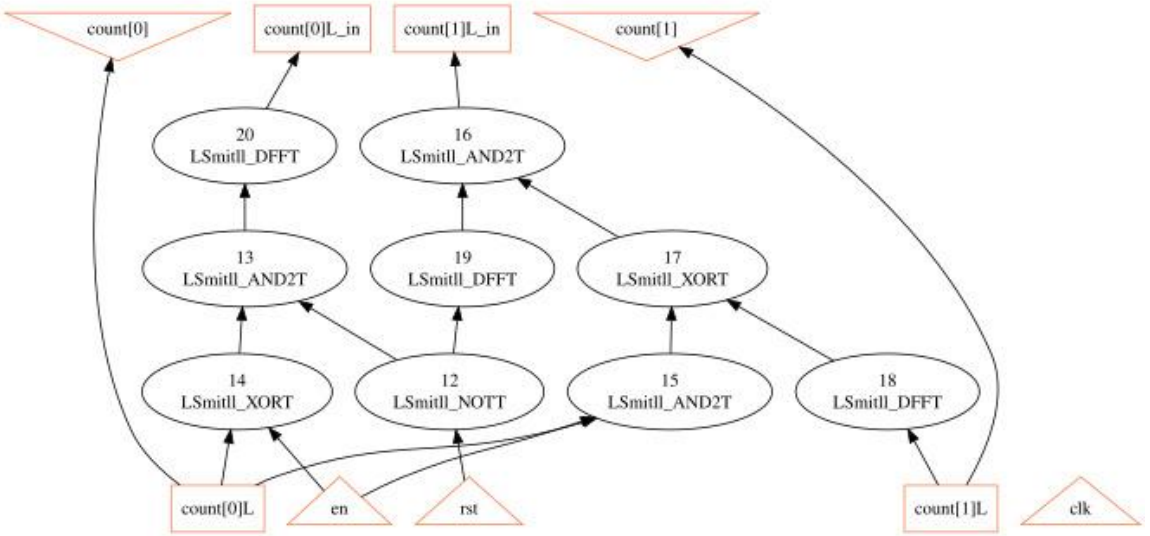


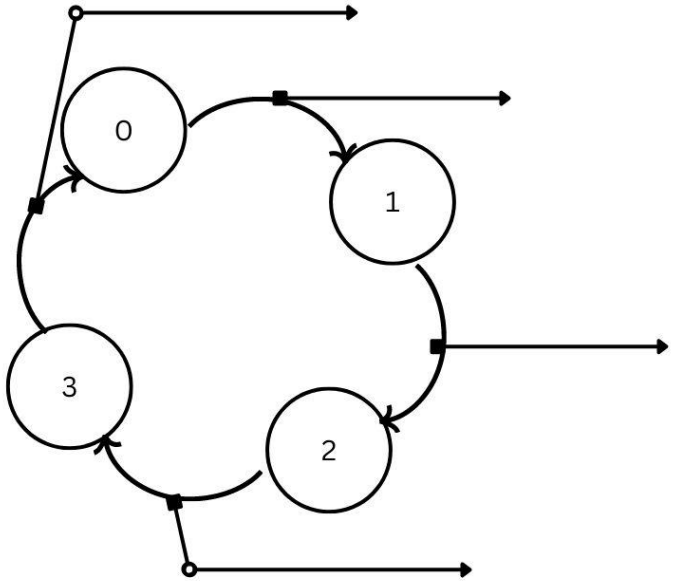
USC

Sequential RSFQ - FSMs

It might be possible to bypass RTL mapping (qYosys) and synthesize sequential behaviour by directly using FSM descriptions.

But it is unlikely that we will obtain better initiation intervals without an exponential blow up in state space.





USC