1.1 High Throughput

A high-throughput design is one that is concerned with the steady-state data rate but less concerned about the time any specific piece of data requires to propagate through the design (latency).

To increase throughput, a design can be pipelined.

Consider the System Verilog code in *iterative_power3.sv* that calculates the cube of a number and the synthesized design in Figure 1.1a. The same register and computational resources are reused until the computation is finished. With this type of iterative implementation, no new computations can begin until the previous computation has completed.

The performance of this implementation is:

- Throughput = 8/3, or 2.7 bits/clock
- Latency = 3 clocks
- Timing = One multiplier delay in the critical path

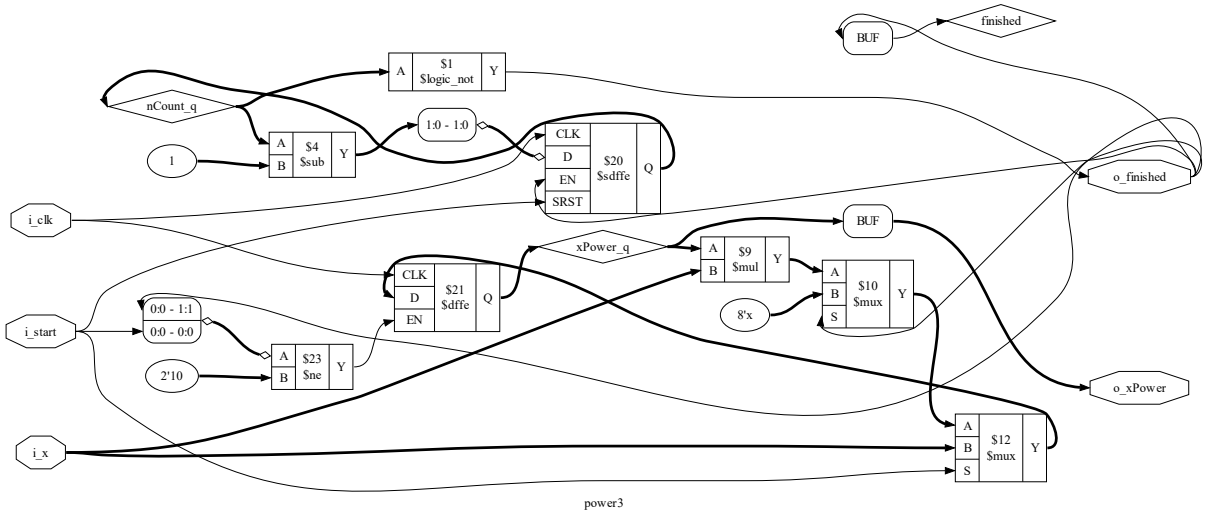

Figure 1.1a: Synthesized design of an iterative implementation to calculate the cube of a number.

Points of Interest of the synthesized design:

- The *xPower* FF is enabled when *start* is asserted and *finished* or finished is deasserted.
- The *nCount* FF is enabled when *finished* is deasserted and is synchronously reset when *start* is asserted.

Contrast this with a pipelined version implemented in *pipelined_Power3.sv* and whose synthesized design is shown in Figure 1.1b.

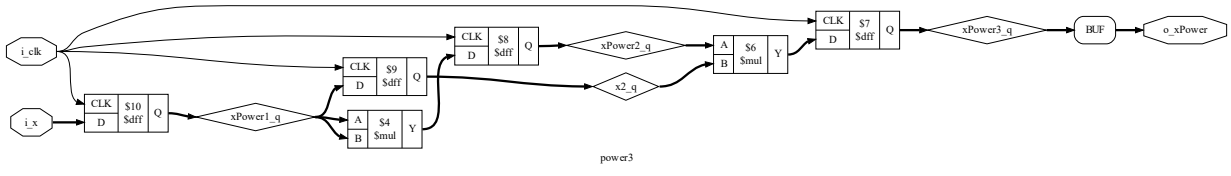

Figure 1.1b: Synthesized design of a pipelined implementation to calculate the cube of a number.

The value of X is passed to both pipeline stages where independent resources compute the corresponding multiply operation. Note, that while X is being used to calculate the final power of 3 in the second pipeline stage, the next value of X can be sent to the first pipeline stage.

Both the final calculation of X3 (XPower3 resources) and the first calculation of the next value of X (XPower2 resources) occur simultaneously.

The performance of this design is:

- Throughput = 8/1, or 8 bits/clock
- Latency = 3 clocks
- Timing = One multiplier delay in the critical path

The throughput performance increased by a factor of 3 over the iterative implementation. There was no penalty in terms of latency as the pipelined implementation still required 3 clocks to propagate the final computation. Likewise, there was no timing penalty as the critical path still contained only one multiplier. The penalty for pipelining was a proportional increase in area.