Register-Based FIFO

XAPP 005.002

Application Note By BERNIE NEW AND WOLFGANG HÖFLICH

Summary

While XC3000-series LCA devices do not provide RAM, it is possible to construct small register-based FIFOs. A basic synchronous FIFO requires one CLB for each two bits of FIFO capacity, plus one CLB for each word in the FIFO. Optional asynchronous input and output circuits are provided. Design files are available for two implementations of this design. The fastest of the two implementations uses a constraints file to achieve better placement.

Specifications		***Xilinx Family***
Size	8 x 8 Bits	XC3000A/XC3100A
Maximum Clock Frequency XC3100A-2	49 MHz	
Number of CLBs	40	

Introduction

In the absence of RAM, XC3000 FIFOs must be constructed with registers. Using both flip-flops, one CLB is required for each two bits of FIFO capacity. For a synchronous FIFO, an additional one CLB per word is required for control. Thus an 8-word by 8-bit FIFO can be implemented in 40 CLBs. Speed is a function of depth, with an 8-word FIFO able to achieve speeds of up to 42 MHz.

Asynchronous inputs and outputs may be added if desired. Each of these adds n/2 CLBs for an n-bit wide FIFO, plus a few additional CLBs for control logic. Typically, asynchronous inputs and outputs operate more slowly because of the handshake required for synchronization. Where burst input or output speed is required for data transfer, the FIFO should be operated in synchronism with the high-speed port.

The basic designs shown use simple flags that permit the input and output of single words. For block transfers, flags could be generated for signaling the availability of a block of data or space for a block of data.

Synchronous FIFOs

The basic FIFO design, shown in Figure 1, comprises a broadside shift register; each word has a separate shift enable. A control flip-flop, associated with each word, contains a valid flag that is shifted with the data. The shift-control logic uses these valid flags to generate shift enables and control the flow of data through the FIFO.

Whenever a register does not contain valid data, shift is enabled for that register, and for all the registers upstream from it. This causes data to continuously shift through the FIFO, with valid words backing-up at the output. They remain there until a POP command enables the shift in all the registers in the FIFO. Invalid data is not retained.

Figure 2 shows the detail of the FIFO. For simplicity, only two data bits are shown (the top two rows of flip-flops); all other data bits are identical. The bottom row of flip-flops contains the valid bits. The shift control logic is the chain of OR gates; a column of flip-flops is enabled if its valid bit, or any valid bit to the right, is not asserted.

The POP command acts like an additional active-Low valid bit, which is to the right of all the columns in the FIFO. When it is High, all the registers shift. If the second to last register contains valid data, this is shifted into the last register, and the VALID flag remains High. Otherwise, invalid data is shifted into the last register, and the VALID flag goes Low. The last register continues shifting until it receives valid data, when the VALID flag goes High.

Data can only be written into the FIFO if the first register contains invalid data or valid data that is about to be shifted out. This condition is signaled by the RDY flag, that is also the shift enable for the first register. Conse-

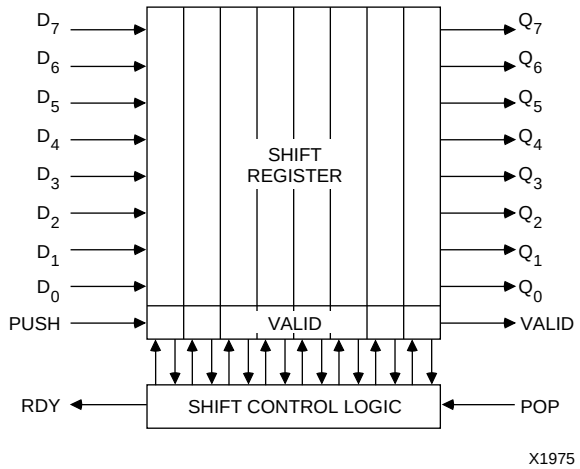

Figure 1. 8-Word x 8-Bit Synchronous FIFO (40 CLBs)

Supporting design files are available on the XACT CD-ROM and on the Xilinx Technical Bulletin Board under the names XAPP005V (VIEWlogic) and XAPP005O (OrCAD)

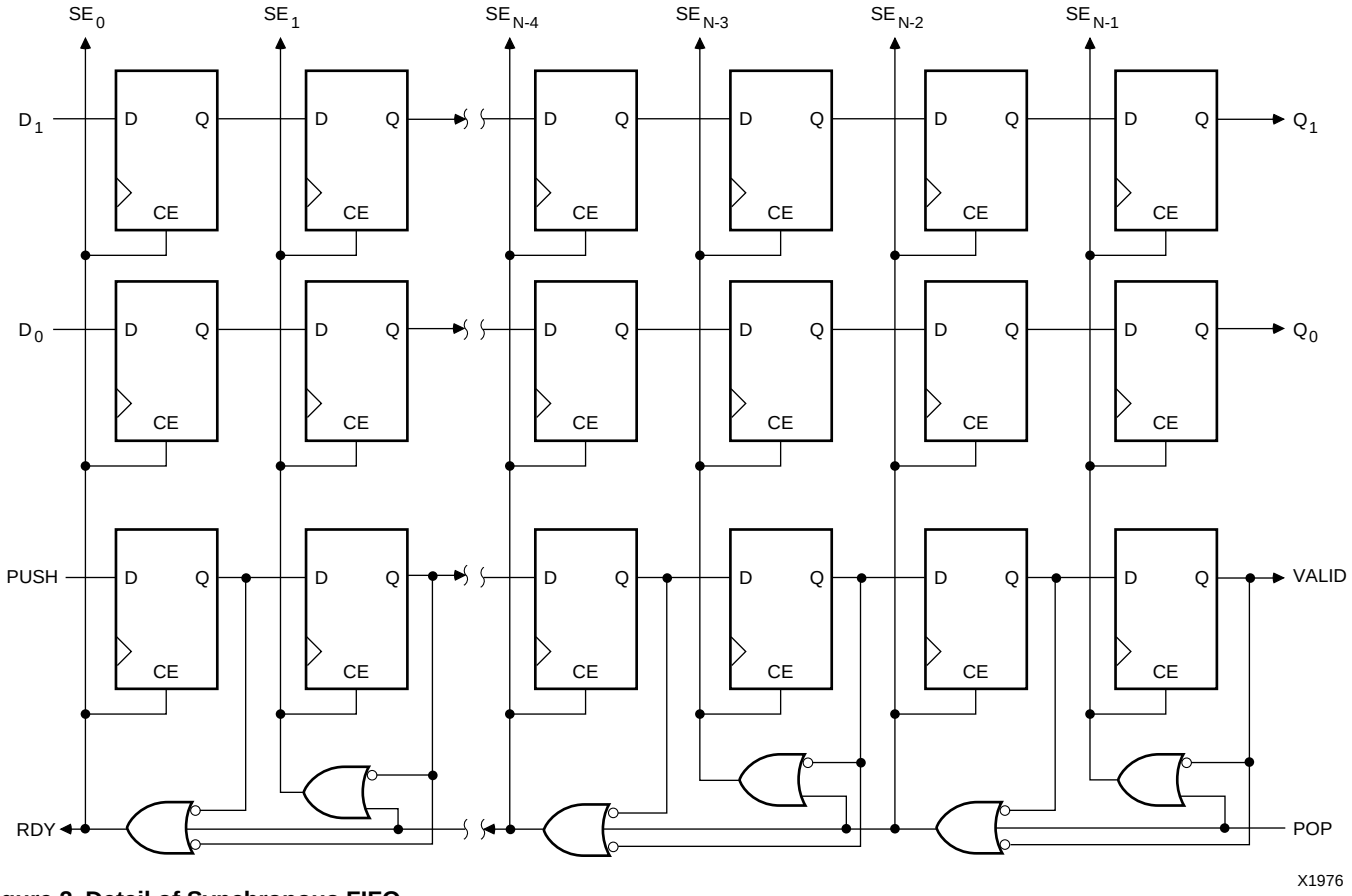

Figure 2. Detail of Synchronous FIFO

quently, data is always being shifted in when the FIFO is ready. The function of PUSH is simply to identify the data being shifted in as valid, so that it is retained in the FIFO.

In the diagram, the CLB clock enable (CE) is used as shift enable. When combining pairs of flip-flops into CLBs, CE can only be used if adjacent bits of the same register are combined. If it is more convenient, bits of equal weight from adjacent registers may be combined. In this case, function generators must be used to implement shift enable. This entails a simple 2-input multiplexer that selects input data when shift is enabled, and selects existing data from the flip-flop when it is not enabled.

The speed of the FIFO is determined by the ripple-OR time of the shift-control logic, and the distribution and set-up times of the shift-enable signals. This defines the set-up time for the POP command. The settling time for the shift-control logic is one CLB delay per two words of FIFO depth. Longlines should be used to distribute the shift-enable signals.

Asynchronous Input Stage

Asynchronous data may be entered into the FIFO using the circuit shown in Figure 3. An additional input holding register is provided to facilitate edge-triggered input. If appropriate, this can be implemented in IOB registers.

Data may only be entered when the RDY flag signals that the input register is available to accept it. The input clock (PUSH) also asserts the PUSH INP signal which removes the RDY flag. On the next internal clock, PUSH INT is asserted and PUSH INP cleared. When shift is enabled into the first register of the FIFO, data is transferred out of the holding register, PUSH INT is cleared and RDY is re-asserted.

If data is being input from a synchronous system that is not synchronized to the FIFO internal clock, the circuit shown in Figure 4 should be used. Again, an input holding register is provided. However, it is enabled by PUSH, instead of being clocked by it (an IOB register cannot be used). As before, PUSH causes PUSH INP to be asserted. Feedback around the flip-flop sustains PUSH INP until it is recognized by the internal clock, permitting the PUSH command to be removed after the one input clock.

The entry of data into the FIFO proceeds as in the previous scheme. RDY is registered to synchronize it to the input clock. The negative clock edge is used for this, so

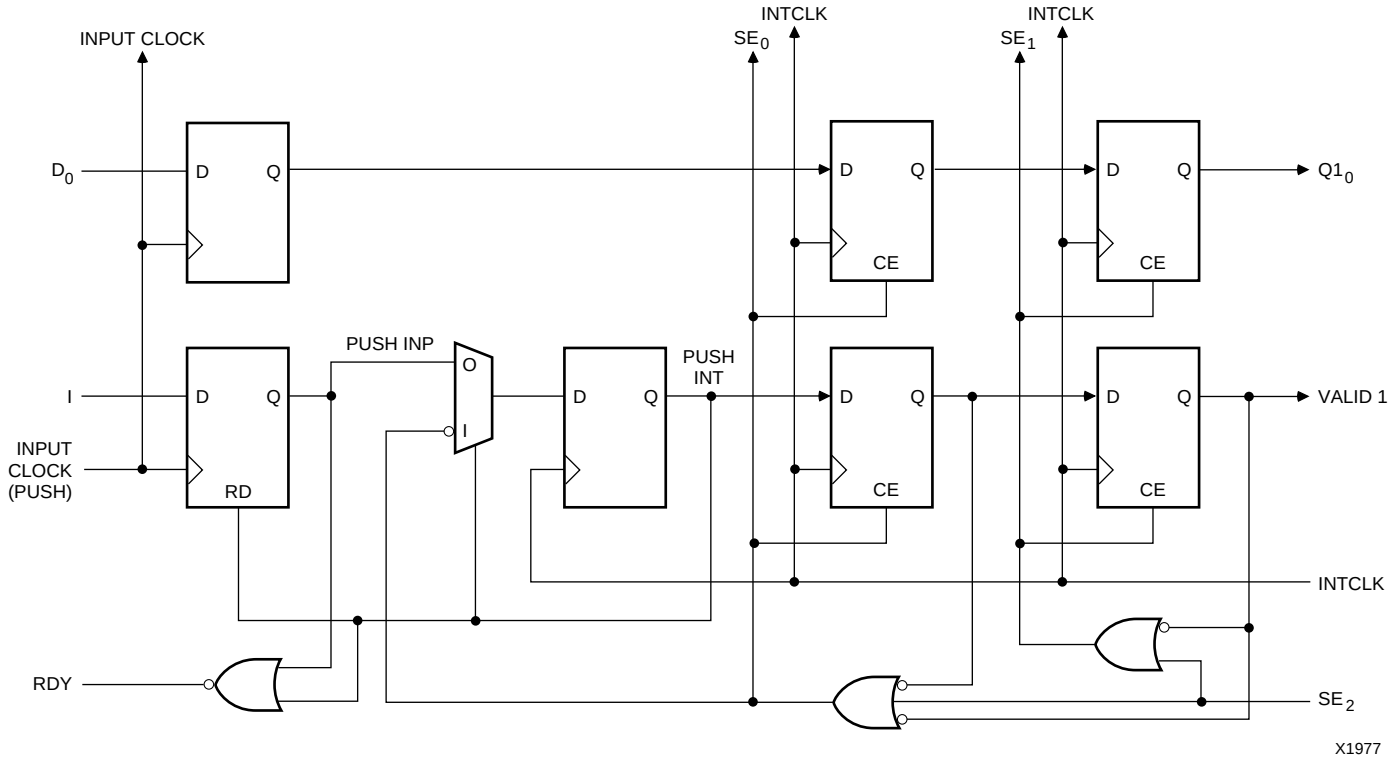

Figure 3. Asynchronous Input Stage

that, if the FIFO is sufficiently fast and is not full, the RDY flag will remain set, and data can be entered on successive input clocks. If the positive clock edge had been used, RDY would always be Low for at least one clock. At best, this would only permit data to be entered on alternate input clocks, no matter how slow.

Asynchronous Output Stage

The circuit shown in Figure 5 should be used, if an asynchronous output is required. For an immediate, edge-triggered output, a holding register is provided, which is clocked by the output clock (POP). IOB flip-flops may be used for this register.

The output register may only be clocked when the RDY flag signals that data is available in the last register of the FIFO. The output clock causes data to be transferred out of the FIFO, and asserts POP OUT. This removes the RDY flag. On the next internal clock, POP INT is asserted and POP OUT is cleared. POP INT is held, and the FIFO shifts, until the last register again contains valid data. It is then cleared, and the RDY flag is re-asserted.

If data is being output to a synchronous system that is not synchronized to the FIFO internal clock, the circuit shown in Figure 6 should be used. The output register, which cannot be implemented in IOBs, is enabled by POP. POP also causes POP OUT to be asserted. Feedback around the register sustains POP OUT until it is recognized by the internal clock, even if POP is removed and another output clock occurs.

The transfer of data out of the FIFO proceeds as in the previous scheme. RDY is synchronized with the negative edge of the output clock. As a result, data can be output on successive clocks if the FIFO is fast enough and data is available.

Implementation Notes

The obvious organization for the FIFO is as a rectangular array of CLBs, with the control logic in the bottom row. The flip-flops may be partitioned into CLBs in two ways. If adjacent bits of the same word are combined, the result is a FIFO that is twice as wide as it is tall (assuming equal numbers of bits and words).

Alternatively, two bits of equal rank from adjacent words may be combined. This gives a FIFO that is twice as tall as it is wide and is potentially faster. The critical path through the control logic passes through a chain of half as many gates as there are words. The tall, narrow organization allows these gates to be implemented in adjacent CLBs with zero-delay direct interconnects.

Both forms of the FIFO are available as macros, using CLBMAPs.

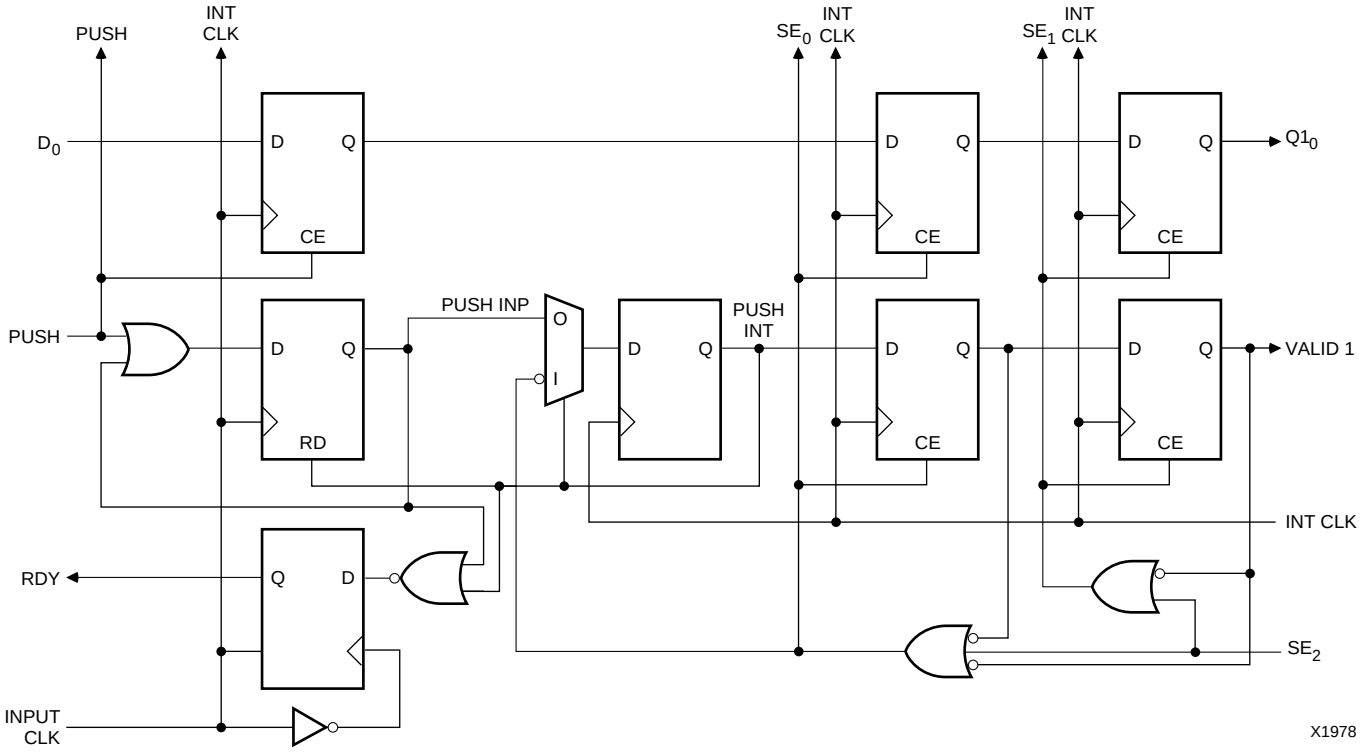

Figure 4. Asynchronous Input Stage (From Synchronous System)

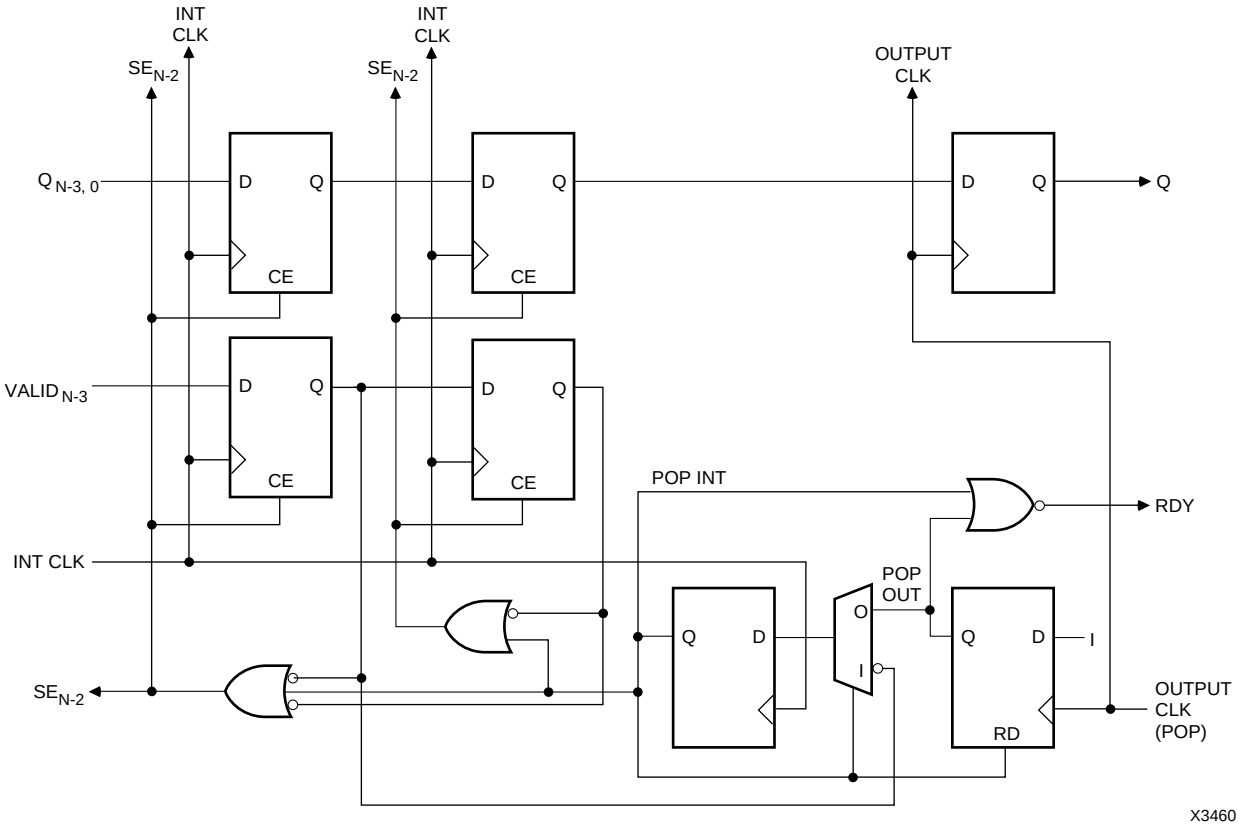

Figure 5. Asynchronous Output Stage

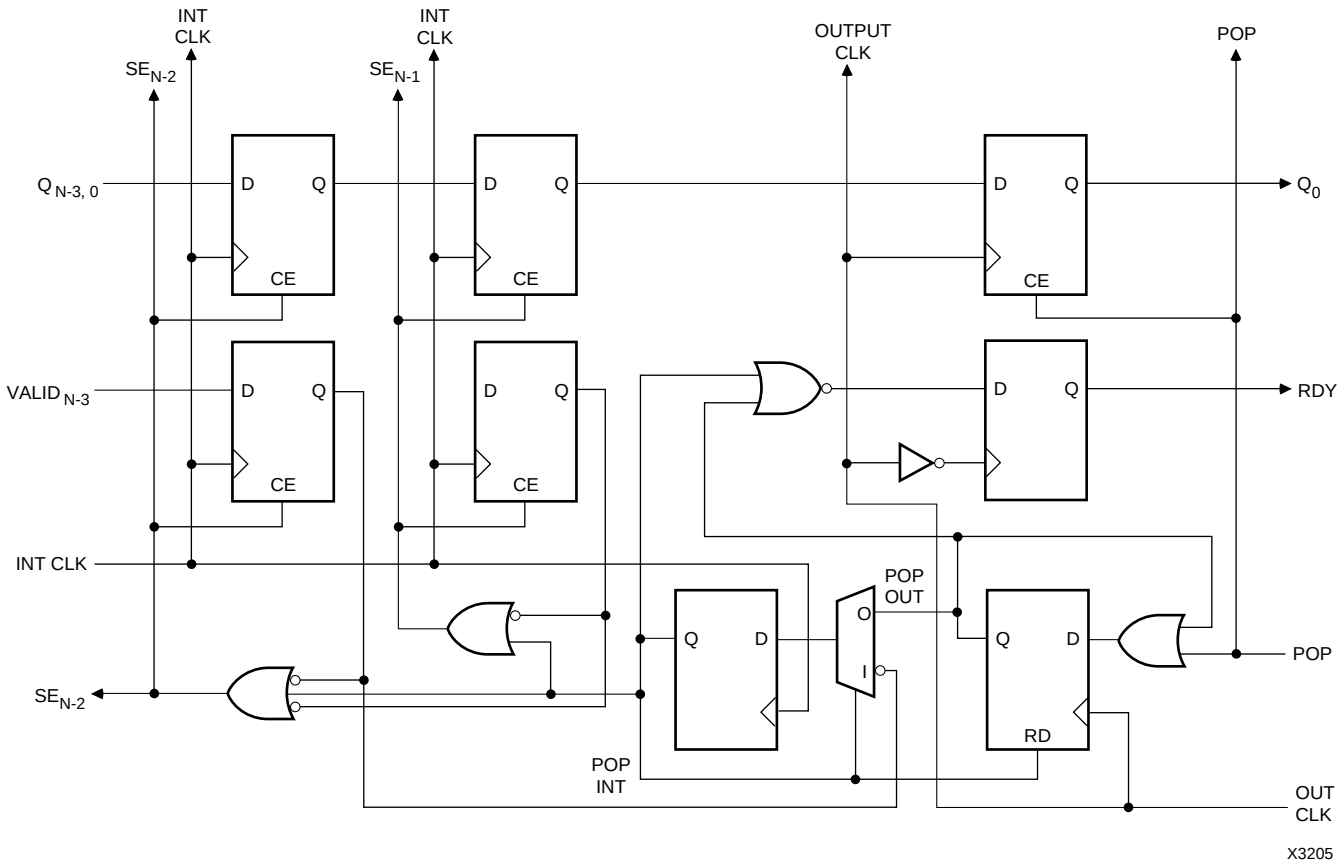

Figure 6. Asynchronous Output Stage (To Synchronous System)