CSU22022 Computer Architecture I

Sixteenth Lecture - Binary Multiplier

Michael Manzke

2023-2024

Trinity College Dublin

Shift-and-Add Multiply

We will consider a shift-and-add multiply circuit as an example datapath.

$$P = A \times B$$

Product = Multiplier × Multiplicand

Figure 1: A, B and P are n-bit unsigned integers.

Bit Products

$$P_i \quad i=0,\ n-1$$

$$P_i = a_i \times B = \left\{ \begin{array}{l} 0 \text{ if } a_i=0 \\ \\ B \text{ if } a_i=1 \end{array} \right\} = a_i \wedge B$$

$$PP_j = \sum_{i=0}^{j} P_i \times 2^i = P_j \times 2^j + PP_{j-1}$$

$$P = PP_{n-1}$$

Figure 2: We can generate the bit products.

Multiplication Example

23	10111	Multiplicand	
19	10011	Multiplier	
	10111	$\leftarrow P_0$	
	10111	$\leftarrow P_1 \times 2^1$	
	00000	$\leftarrow P_2 \times 2^2$	
	00000	$\leftarrow P_3 \times 2^3$	
	10111	$\leftarrow P_4 \times 2^4$	
437	110110101	Product	

Shift-and-Add Multiply Example

					Mutiplicand						Multiplier							
					B4	B3	B2	B1	B0		Q4	Q3	Q2	Q1	Q0			
					1	0	1	1	1	(23)	1	0	0	1	1	(19)		
					1	0	1	1	1		1	0	0	1	1		P0	Q0 = 1 -> Add, Shift
				1	0	1	1	1			1	0	0	1	1		P1 x2^1	Q1 = 1 -> Add, Shift
			0	0	0	0	0				1	0	0	1	1		P2 x2^2	Q2 = 0 -> Shift
		0	0	0	0	0					1	0	0	1	1		P3 x2^3	Q3 = 0 -> Shift
	1	0	1	1	1						1	0	0	1	1		P4 x2^4	Q1 = 1 -> Add, Shift
	1	1	0	1	1	0	1	0	1								P0+P1 x2^1+P2 x2^2+P3 x2^3+P1 x2^4	
A9	A8	A7	A6	A5	A4	A3	A2	A1	A0									
	Product																	

Shift-and-Add Multiply Example

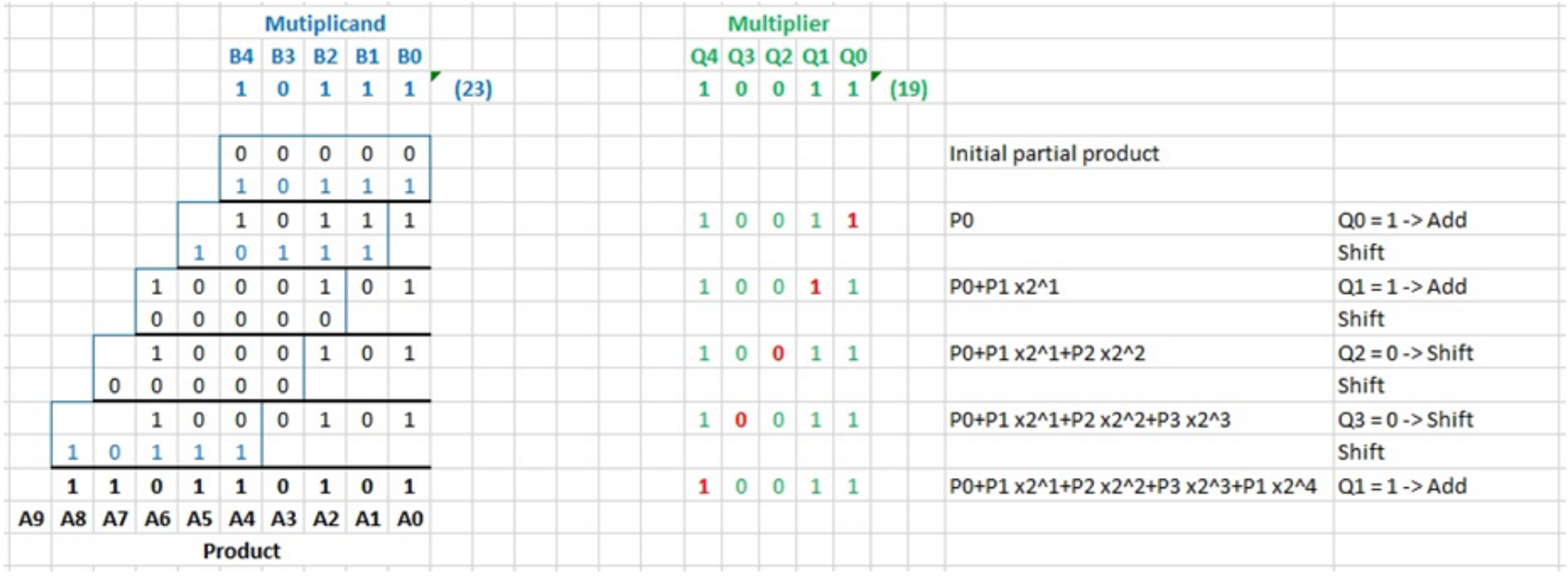


Figure 3: We add the multiplicand if the relevante Q-bit is '1'.

Shift-and-Add Multiply Example

	Mutiplicand										Multiplier							
C	B4	B3	B2	B1	B0						Q4	Q3	Q2	Q1	Q0			
	1	0	1	1	1	(23)					1	0	0	1	1	(19)		
	0	0	0	0	0												Initial partial product	
	1	0	1	1	1													
	1	0	1	1	1						1	0	0	1	1		P0	Q0 = 1 -> Add
	0	1	0	1	1	1												Shift
	1	0	1	1	1													
1	0	0	0	1	0	1					1	0	0	1	1		P0+P1 x2^1	Q1 = 1 -> Add
	1	0	0	0	1	0	1											Shift
	0	1	0	0	0	1	0	1			1	0	0	1	1		P0+P1 x2^1+P2 x2^2	Q2 = 0 -> Shift
	0	0	1	0	0	0	1	0	1		1	0	0	1	1		P0+P1 x2^1+P2 x2^2+P3 x2^3	Q3 = 0 -> Shift
	1	0	1	1	1													
	1	1	0	1	1	0	1	0	1		1	0	0	1	1		P0+P1 x2^1+P2 x2^2+P3 x2^3+P1 x2^4	Q1 = 1 -> Add
	0	1	1	0	1	1	0	1	0	1								Shift
	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0								
					Product													

Figure 4: Reorganise the rows. The 5 MSB are added only.

Hardware Multiplication

23	10111	Multiplicand
19	10011	Multiplier
	00000	Initial partial product
	10111	Add multiplicand, since multiplier bit is 1
	10111	Partial product after add and before shift
	010111	Partial product after shift
	10111	Add multiplicand, since multiplier bit is 1
	1000101	Partial product after add and before shift[a]
	1000101	Partial product after shift
	01000101	Partial product after shift
	001000101	Partial product after shift
	10111	Add multiplicand, since multiplier bit is 1
	110110101	Partial product after add and before shift
437	0110110101	Product after final shift

Figure 5: With overflow.

Shift-and-Add Multiply Example

Clock			Mutiplicand										Multiplier							
Cycle			B4	B3	B2	B1	B0						Q4	Q3	Q2	Q1	Q0			
			1	0	1	1	1	(23)					1	0	0	1	1	(19)		
		C	A4	A3	A2	A1	A0	Q4	Q3	Q2	Q1	Q0								
		0	0	0	0	0	0	1	0	0	1	1							Initial partial product	
1	+		1	0	1	1	1						1	0	0	1	1			Q0 = 1 -> Add
		C	A4	A3	A2	A1	A0	Q4	Q3	Q2	Q1	Q0								
		0	1	0	1	1	1	1	0	0	1	1								
		C	A4	A3	A2	A1	A0	Q4	Q3	Q2	Q1	Q0								
2	sr	0	0	1	0	1	1	1	1	0	0	1							P0	Shift
3	+		1	0	1	1	1						1	0	0	1	1			Q1 = 1 -> Add
		C	A4	A3	A2	A1	A0	Q4	Q3	Q2	Q1	Q0								
		1	0	0	0	1	0	1	1	0	0	1								
		C	A4	A3	A2	A1	A0	Q4	Q3	Q2	Q1	Q0								
4	sr	0	1	0	0	0	1	0	1	1	0	0	1	0	0	1	1		P0+P1 x2^1	Shift
		C	A4	A3	A2	A1	A0	Q4	Q3	Q2	Q1	Q0								
5	sr	0	0	1	0	0	0	1	0	1	1	0	1	0	0	1	1		P0+P1 x2^1+P2 x2^2	Q2 = 0 -> Shift
		C	A4	A3	A2	A1	A0	Q4	Q3	Q2	Q1	Q0								
6	sr	0	0	0	1	0	0	0	1	0	1	1							P0+P1 x2^1+P2 x2^2+P3 x2^3	Q3 = 0 -> Shift
7	+		1	0	1	1	1						1	0	0	1	1			Q4 = 1 -> Add
		C	A4	A3	A2	A1	A0	Q4	Q3	Q2	Q1	Q0								
		0	1	1	0	1	1	0	1	0	1	1								
		C	A4	A3	A2	A1	A0	Q4	Q3	Q2	Q1	Q0								
8	sr	0	0	1	1	0	1	1	0	1	0	1							P0+P1 x2^1+P2 x2^2+P3 x2^3+P1 x2^4	Shift
								Product												

Figure 6: Shows the Shift-Registers C-A-Q content

Binary Multiplier Schematic

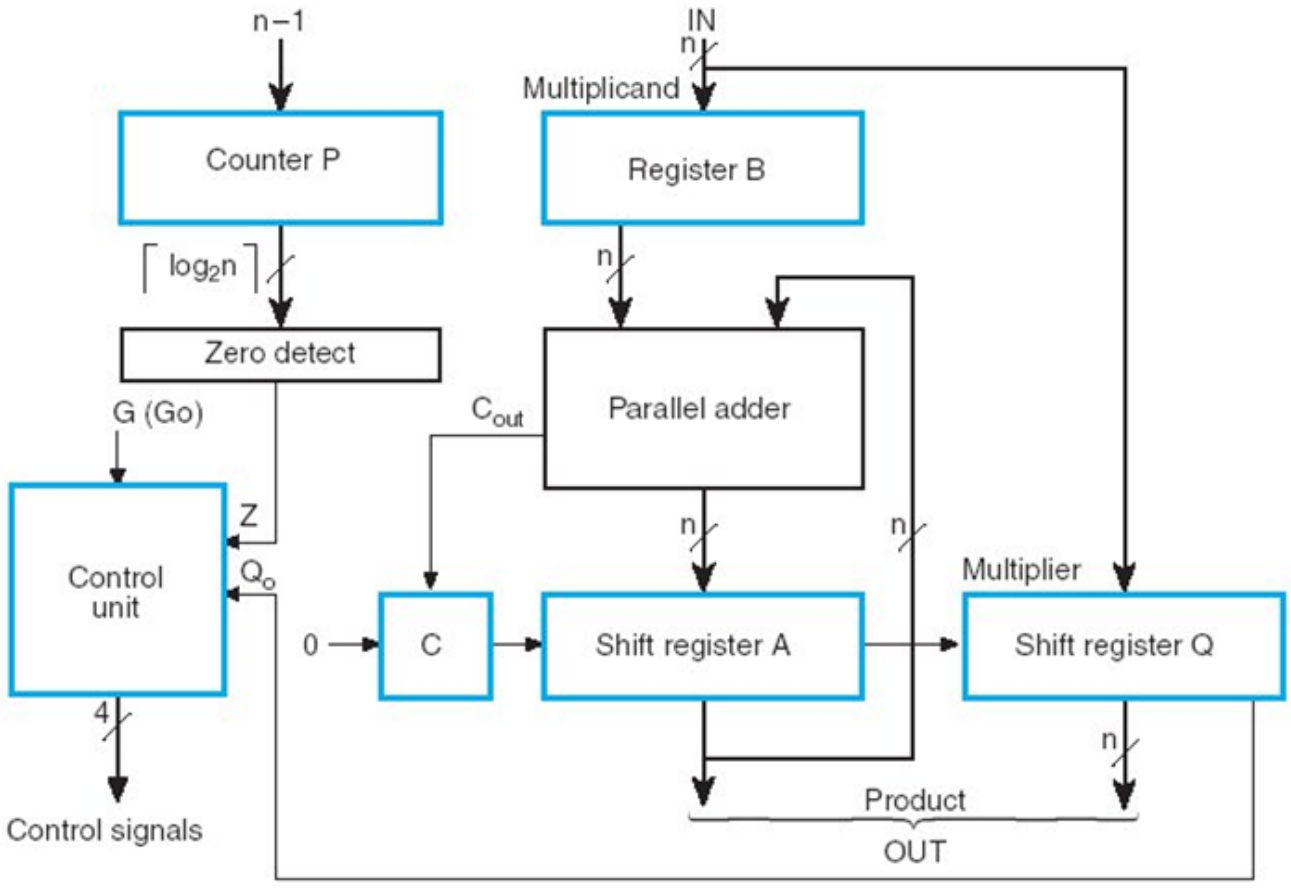


Figure 7: Including control.

Binary Multiplier Schematic

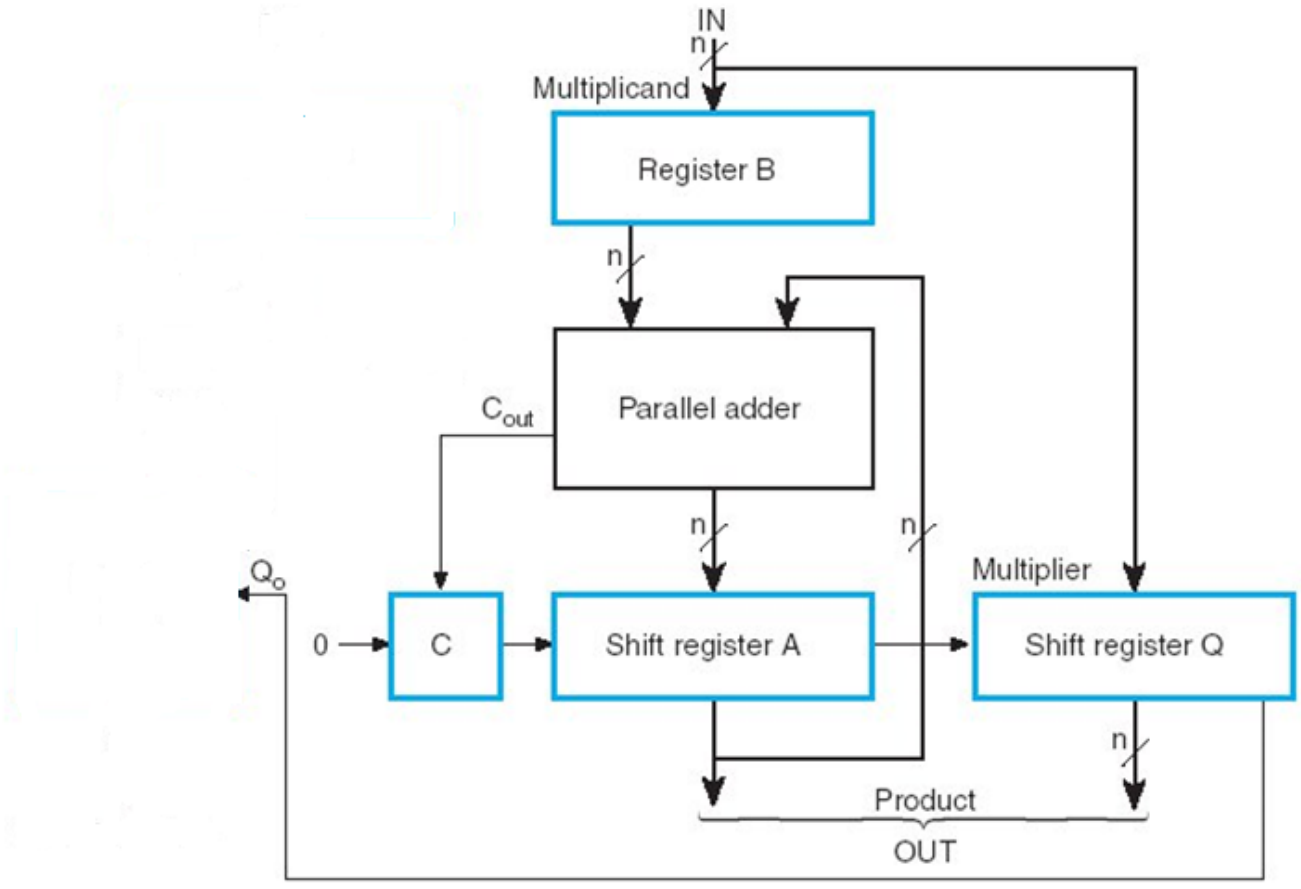


Figure 8: Without control.

Binary Multiplier Datapath Schematic

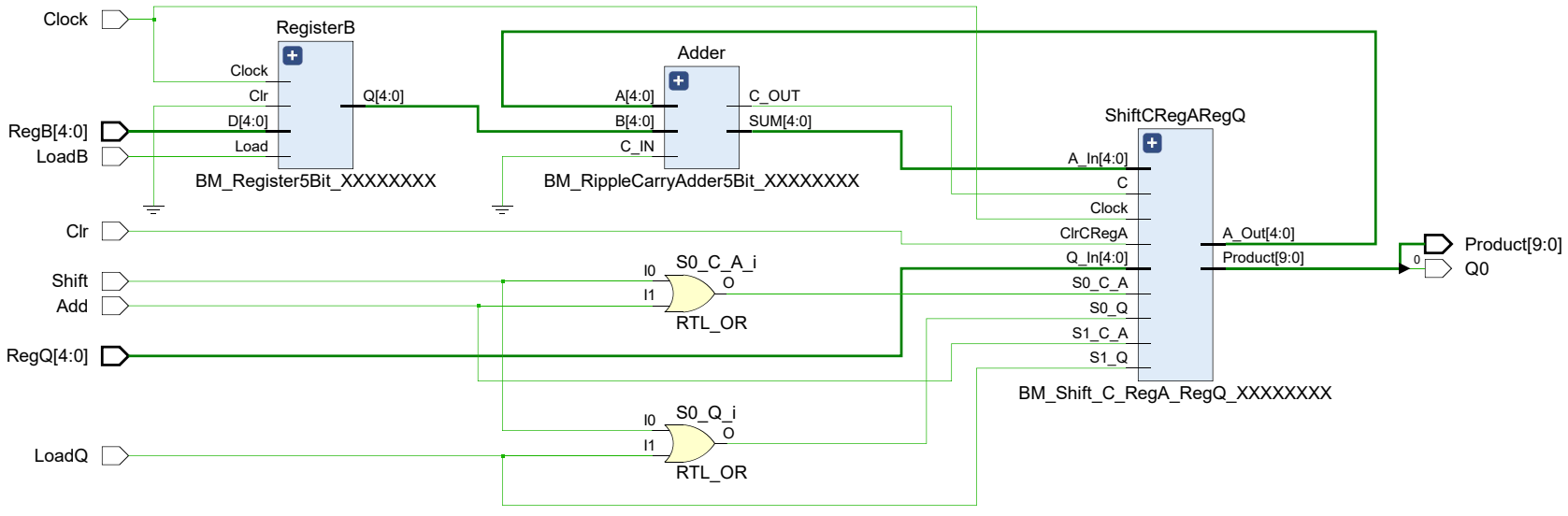


Figure 9: VHDL implementation.

Register B

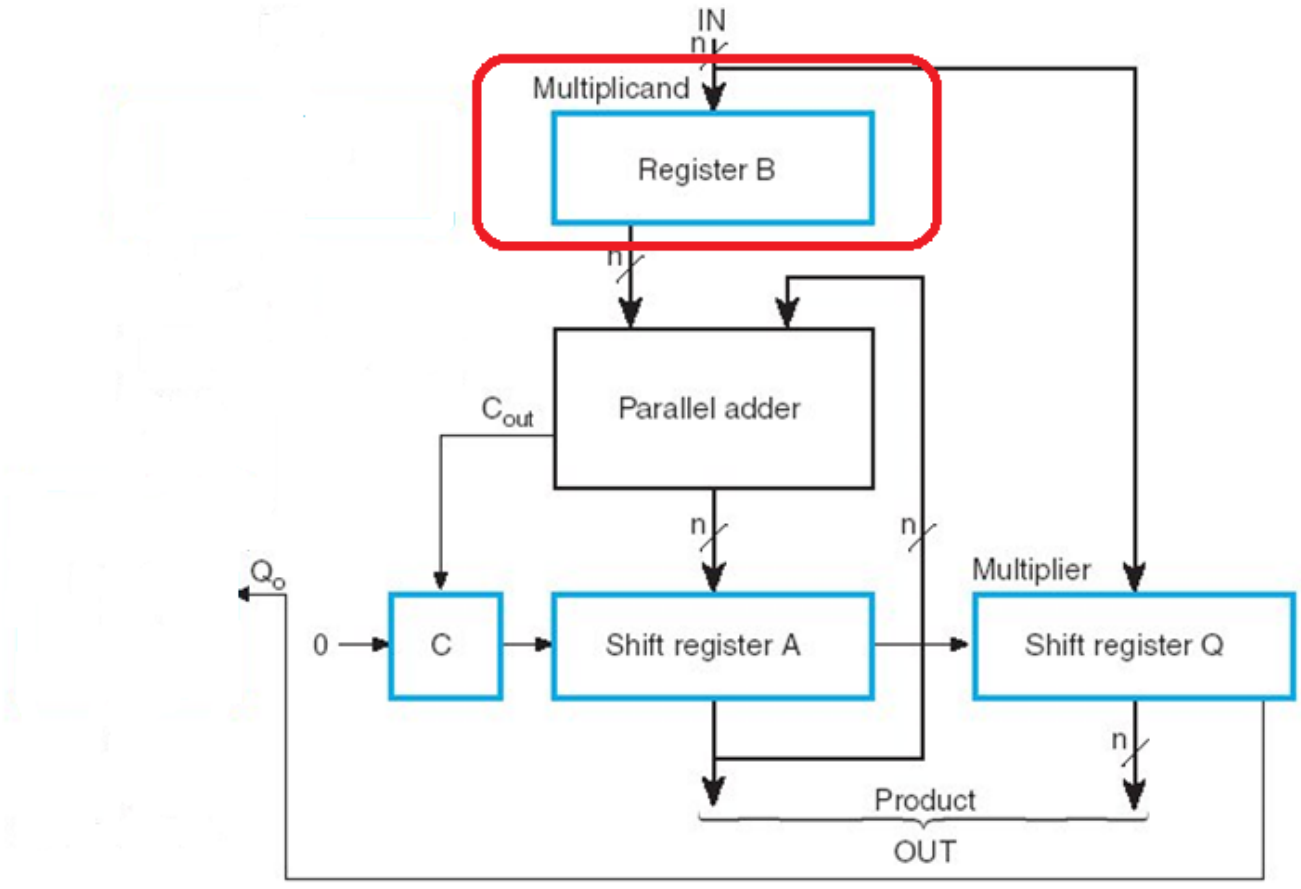


Figure 10: Register B holds the multiplicand.

Register B

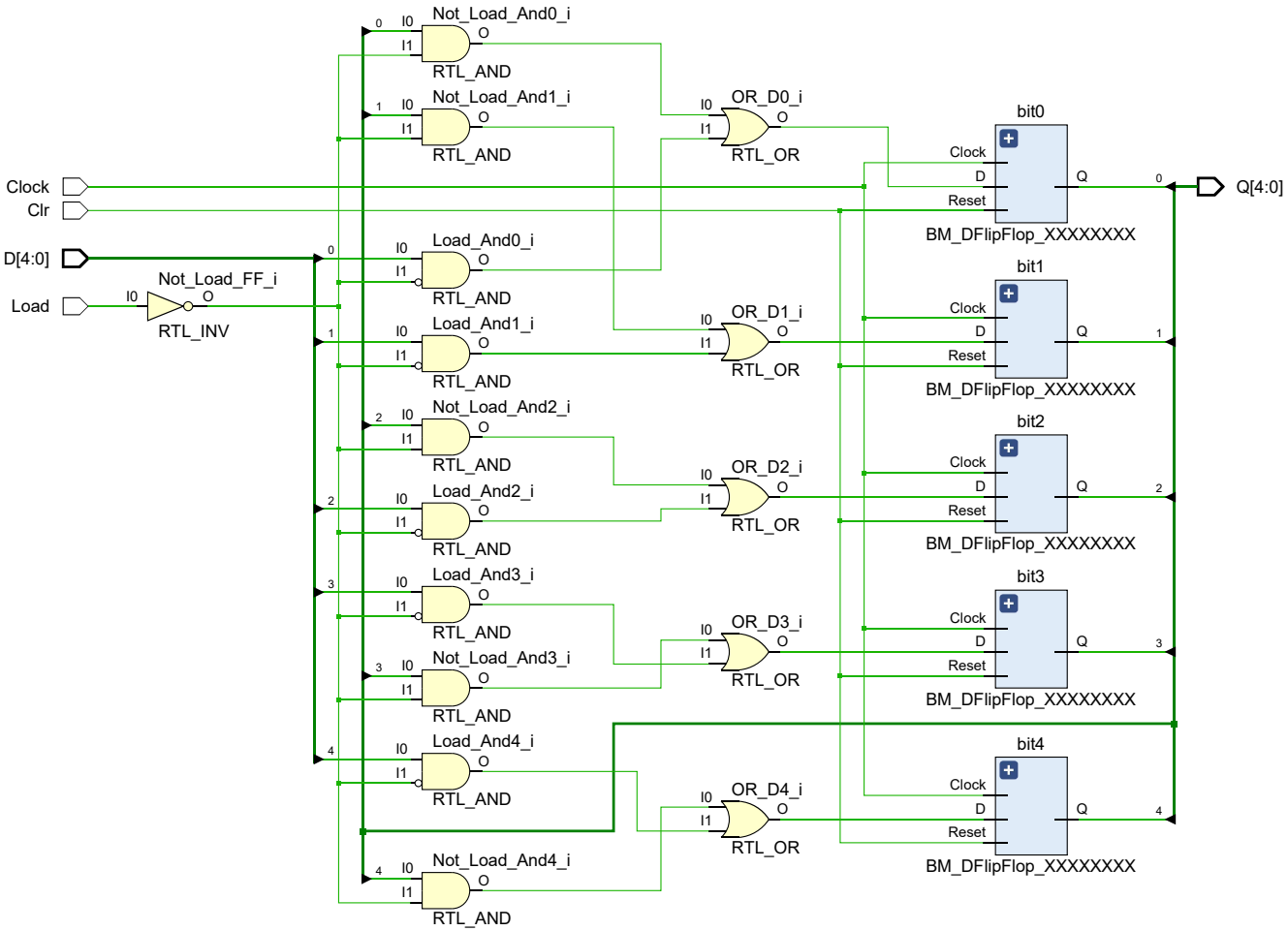


Figure 11: A 5-bit VHDL implementation.

Shift Registers A and B

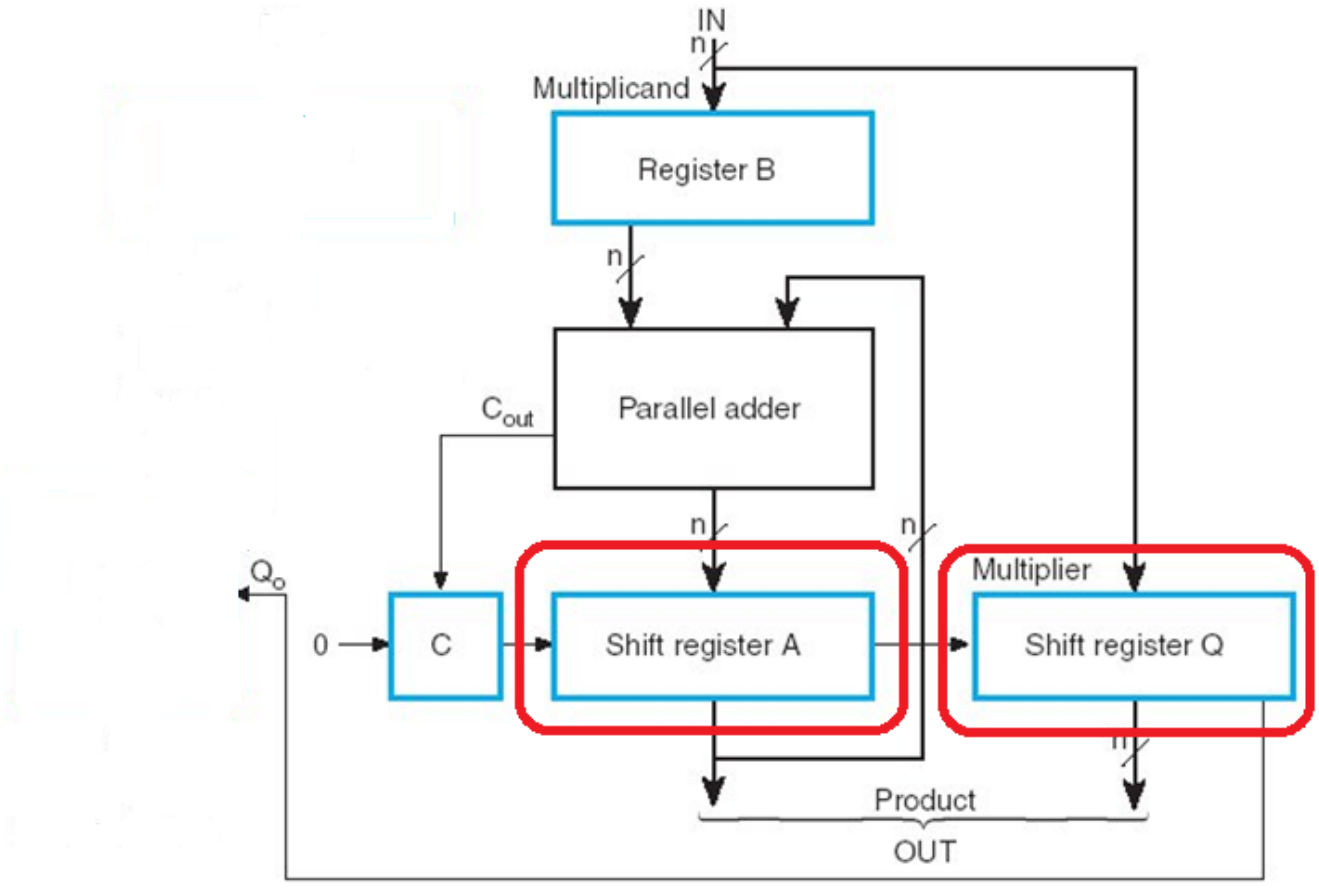


Figure 12: Shift Registers A needs to be cleared and Shift Registers Q holds the multiplier.

4-bit Shift Register Schematic

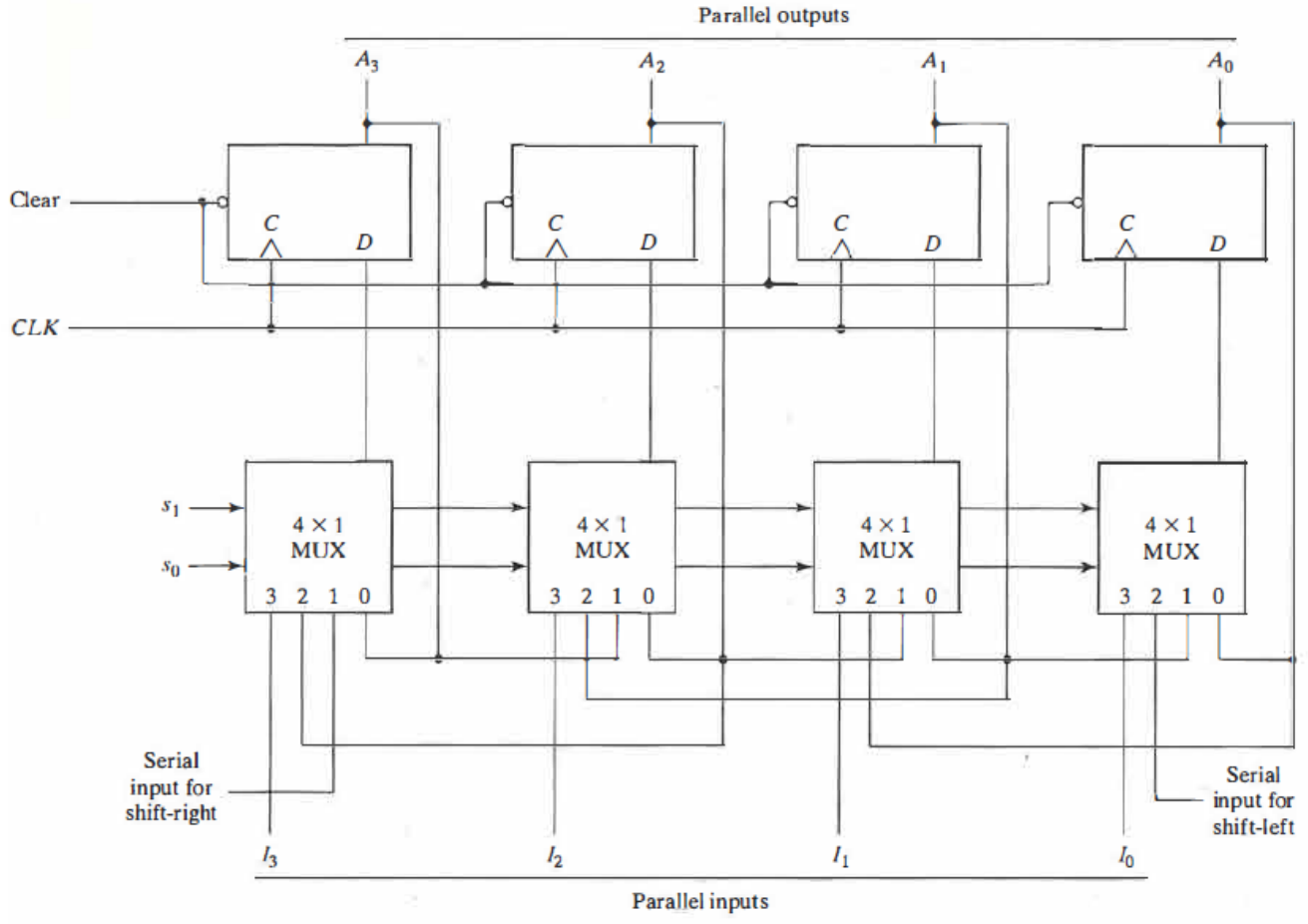
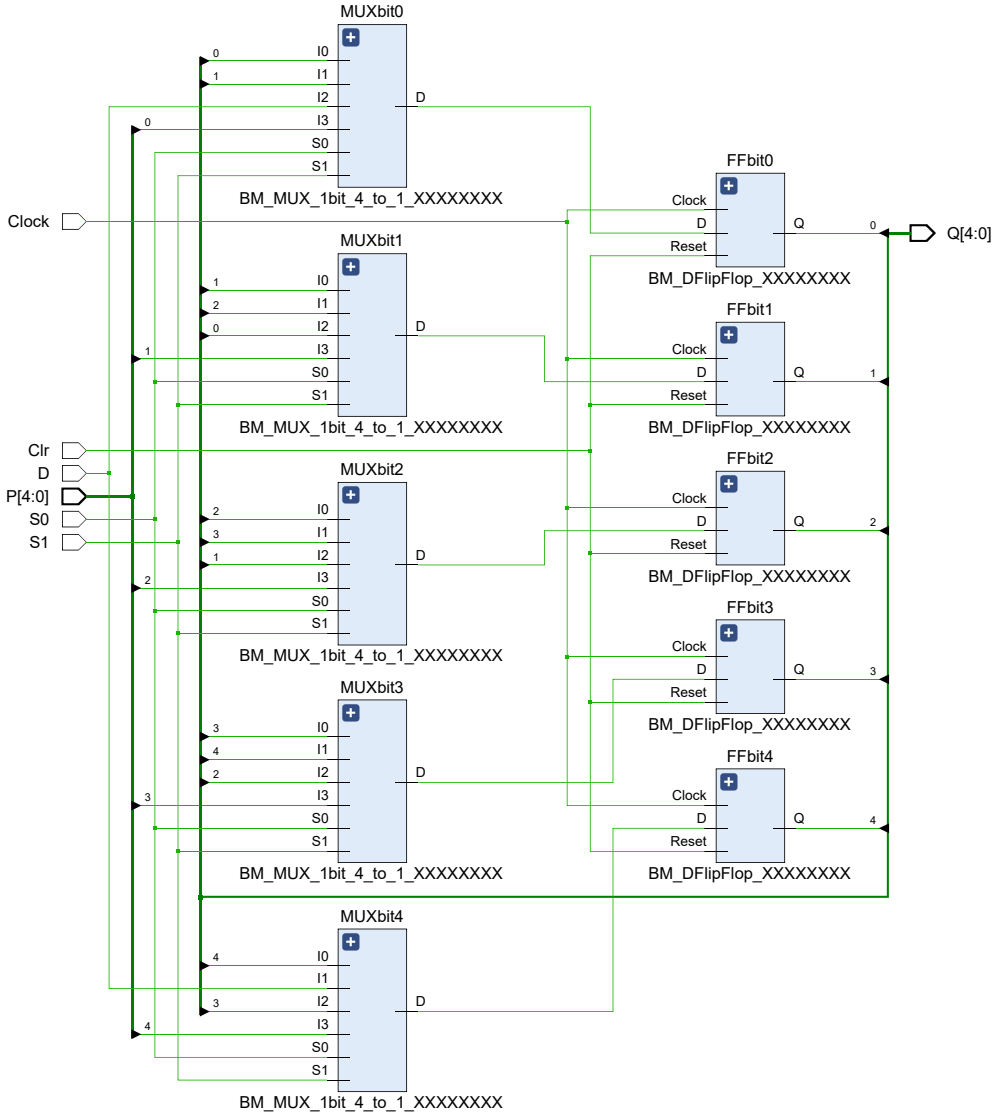
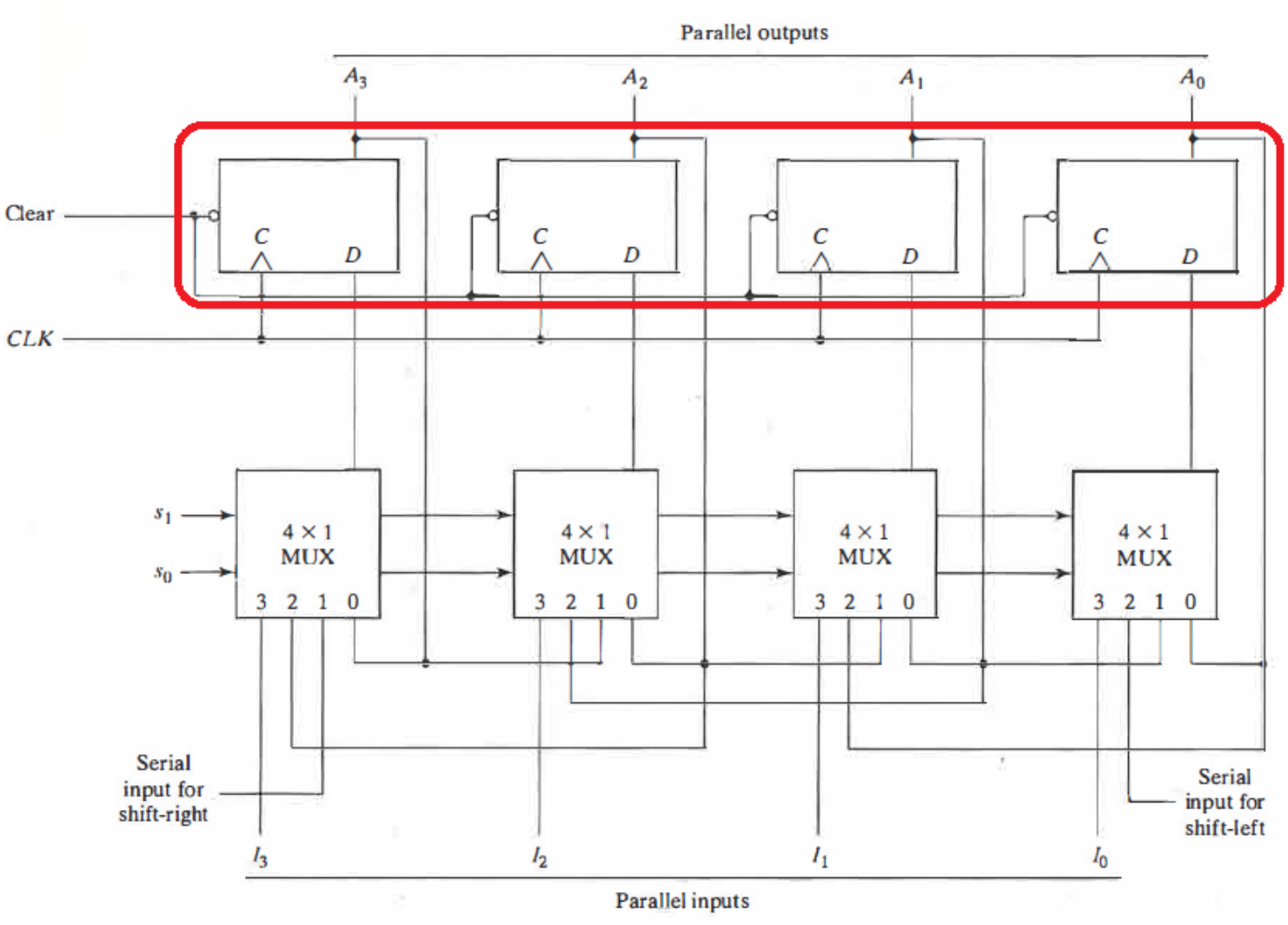


Figure 13: We can keep the content of the shift register, shift to the right, shift to the left, and parallel load the shift register.

5-bit VHDL Shift Register Implementation



Shift Register's D-Flip-Flops



D-Flip-Flops without Reset

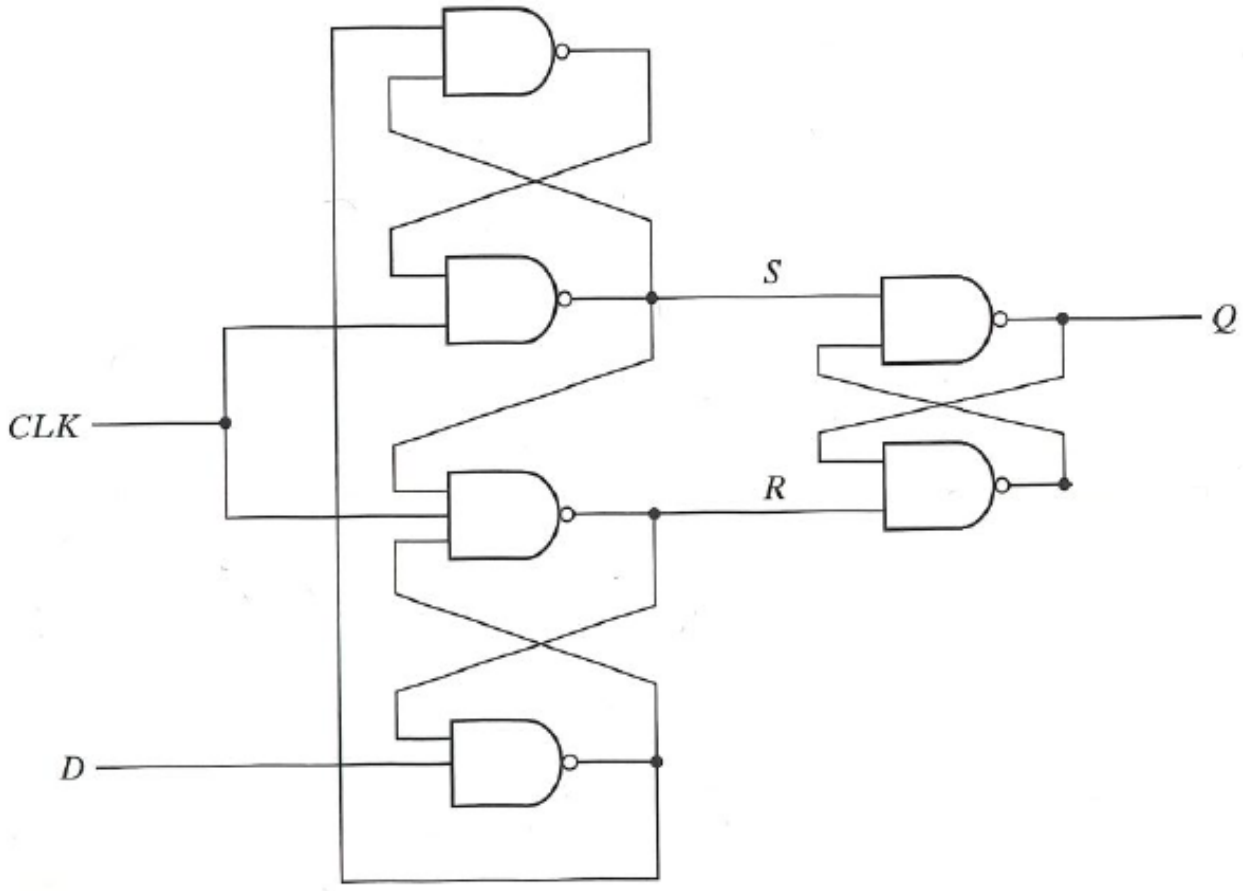


Figure 14: We implemented these so far.

D-Flip-Flops with Reset

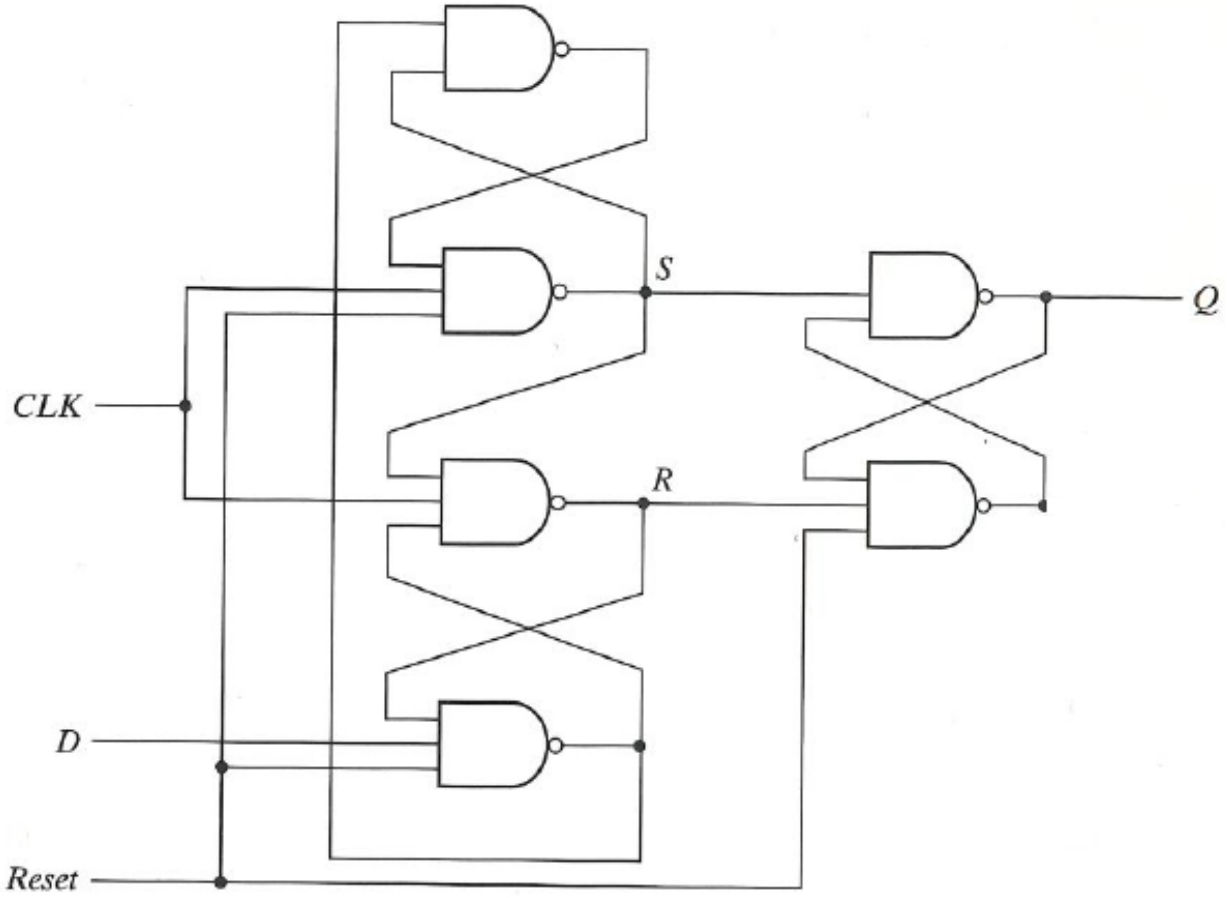


Figure 15: The shift registers in this design require D-Flip-Flops with reset.

VHDL Implementation of a D-Flip-Flops with Reset

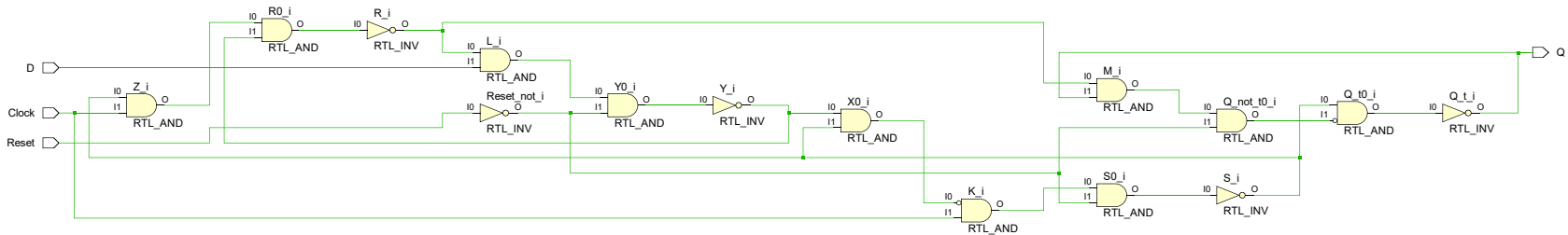
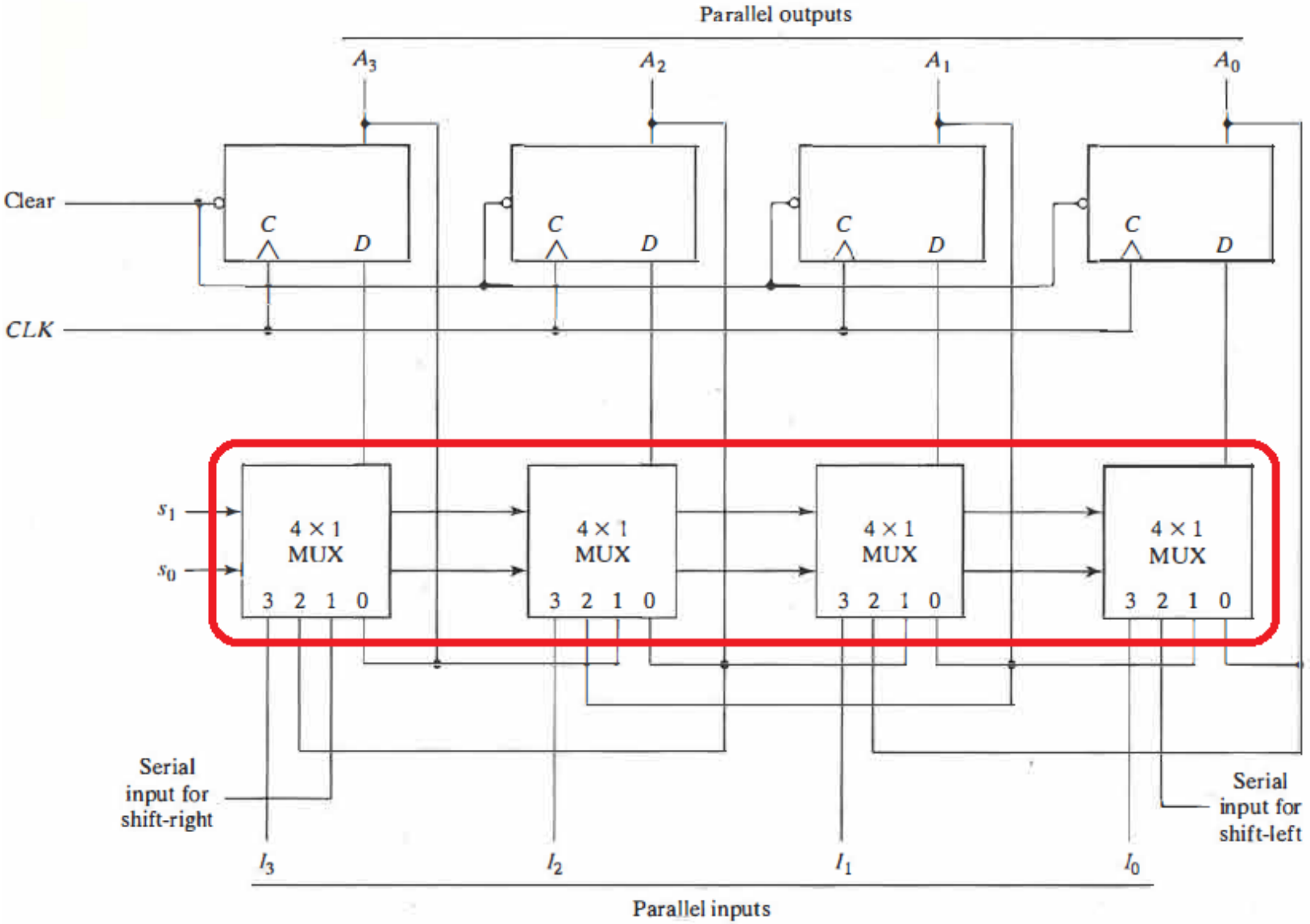
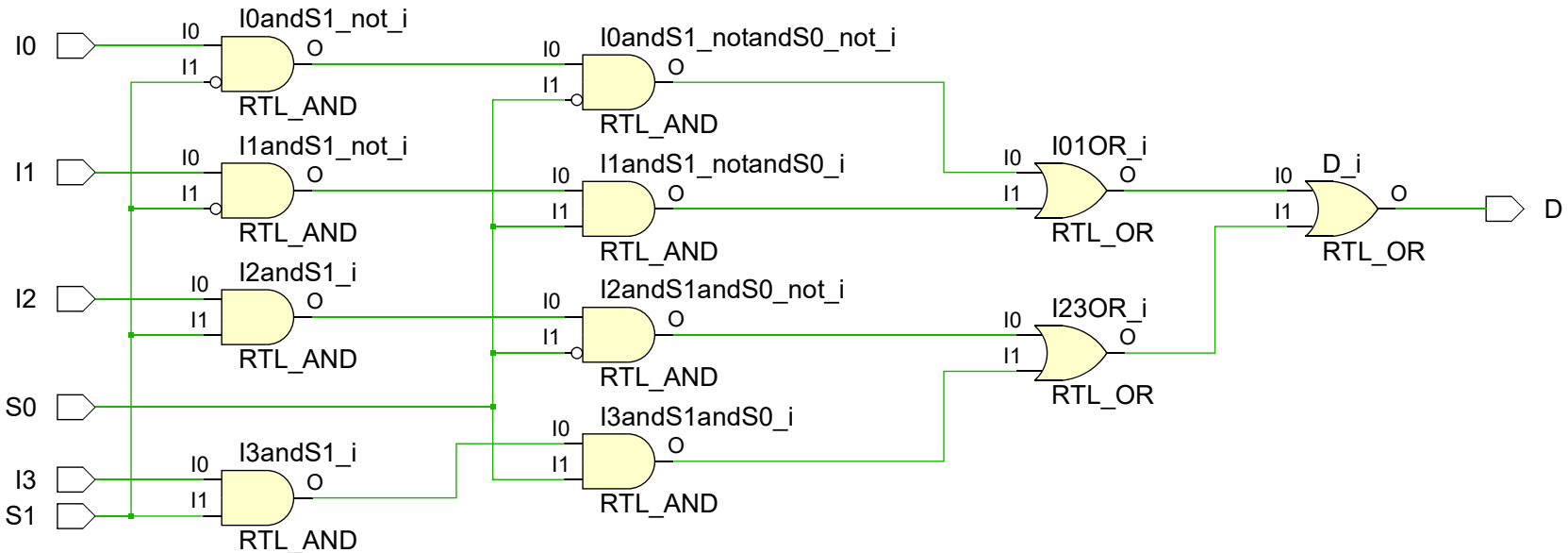


Figure 16: We require an AND-gate and a NAND-gate for every NAND-gate with three inputs. See figure 15.

Shift Register's 4 to 1 Multiplexer



VHDL Implementation of the 4 to 1 Multiplexer



Shift Flip-Flop

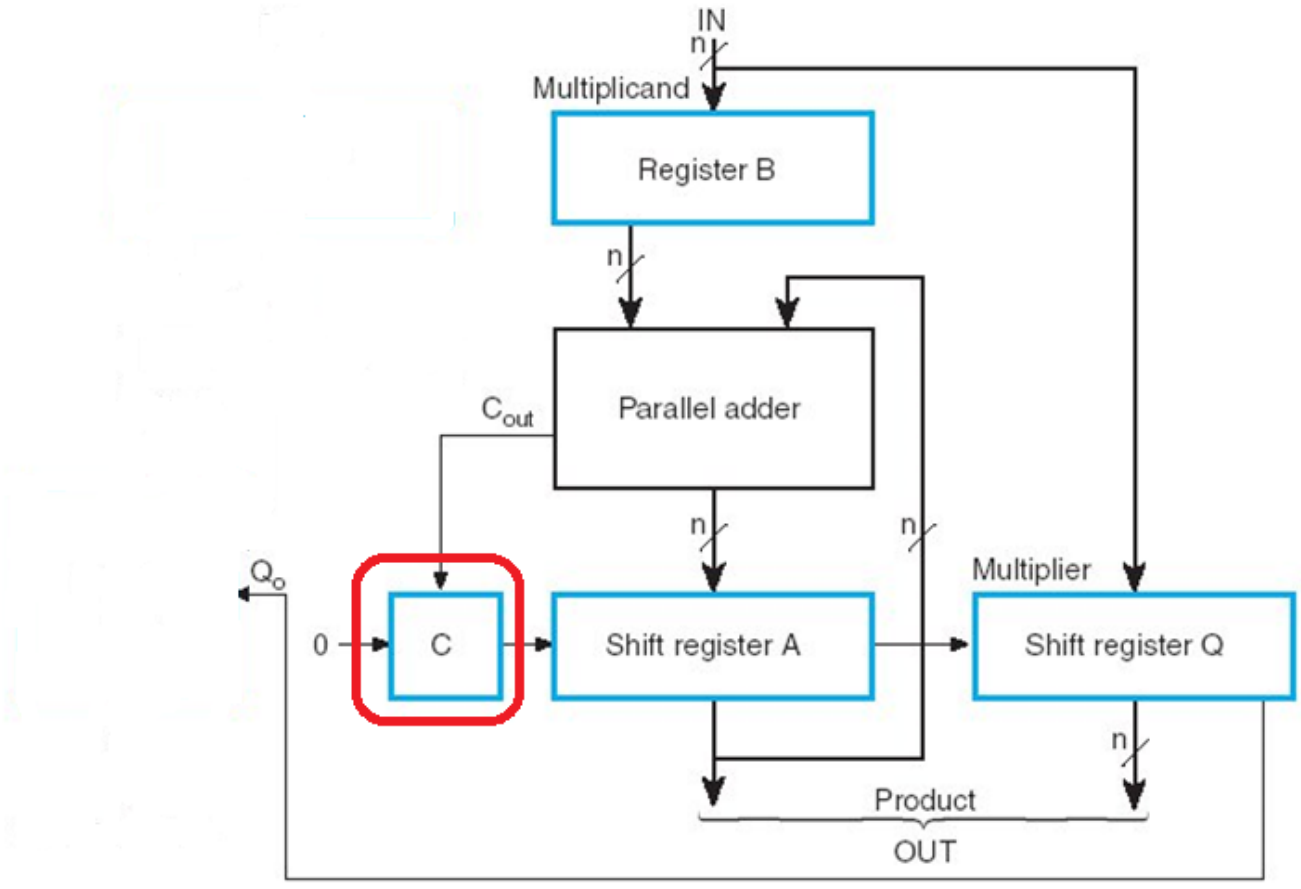
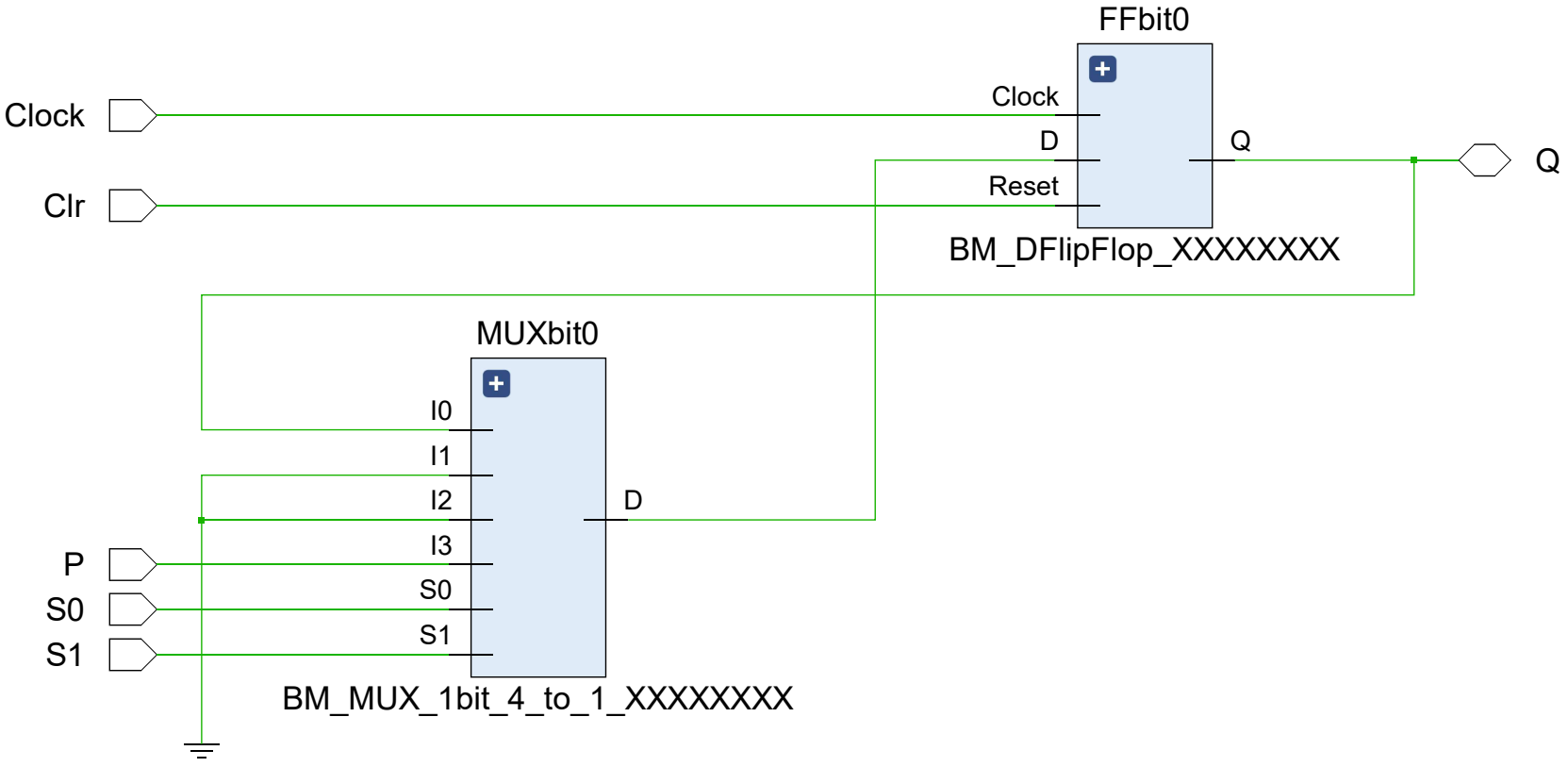


Figure 18: Similar to the Shift Register design with just one bit.

VHDL Implementation of the Shift Flip-Flop



Shift Flip-Flop C and the Shift Registers A and Q

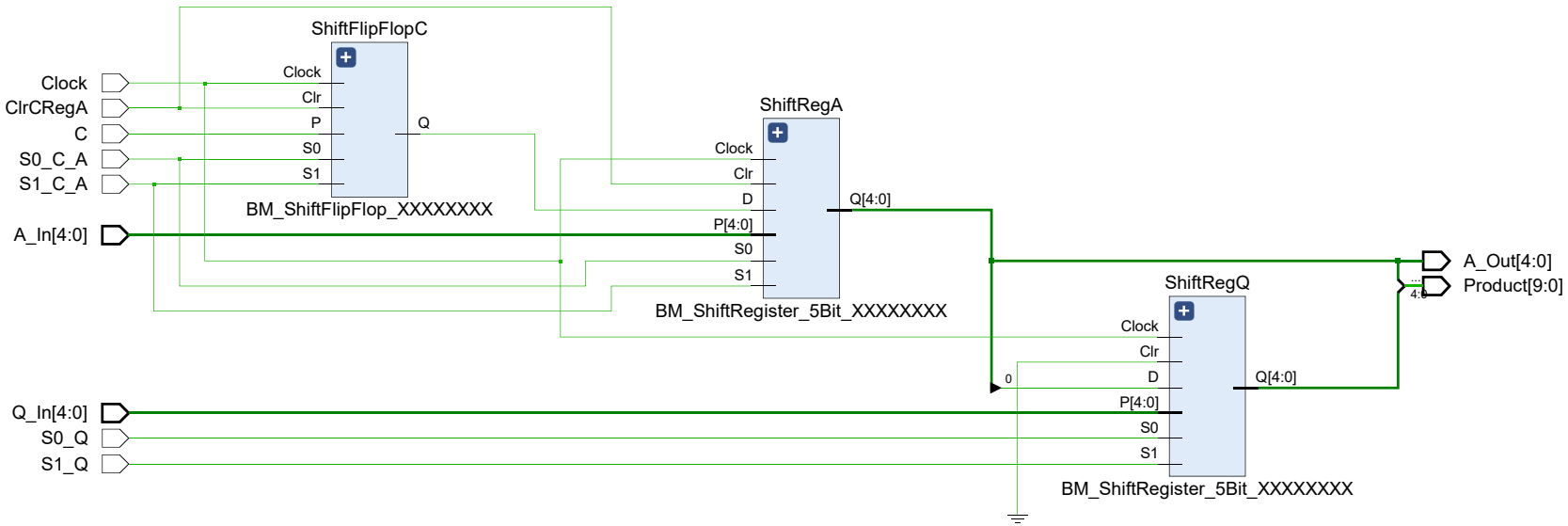
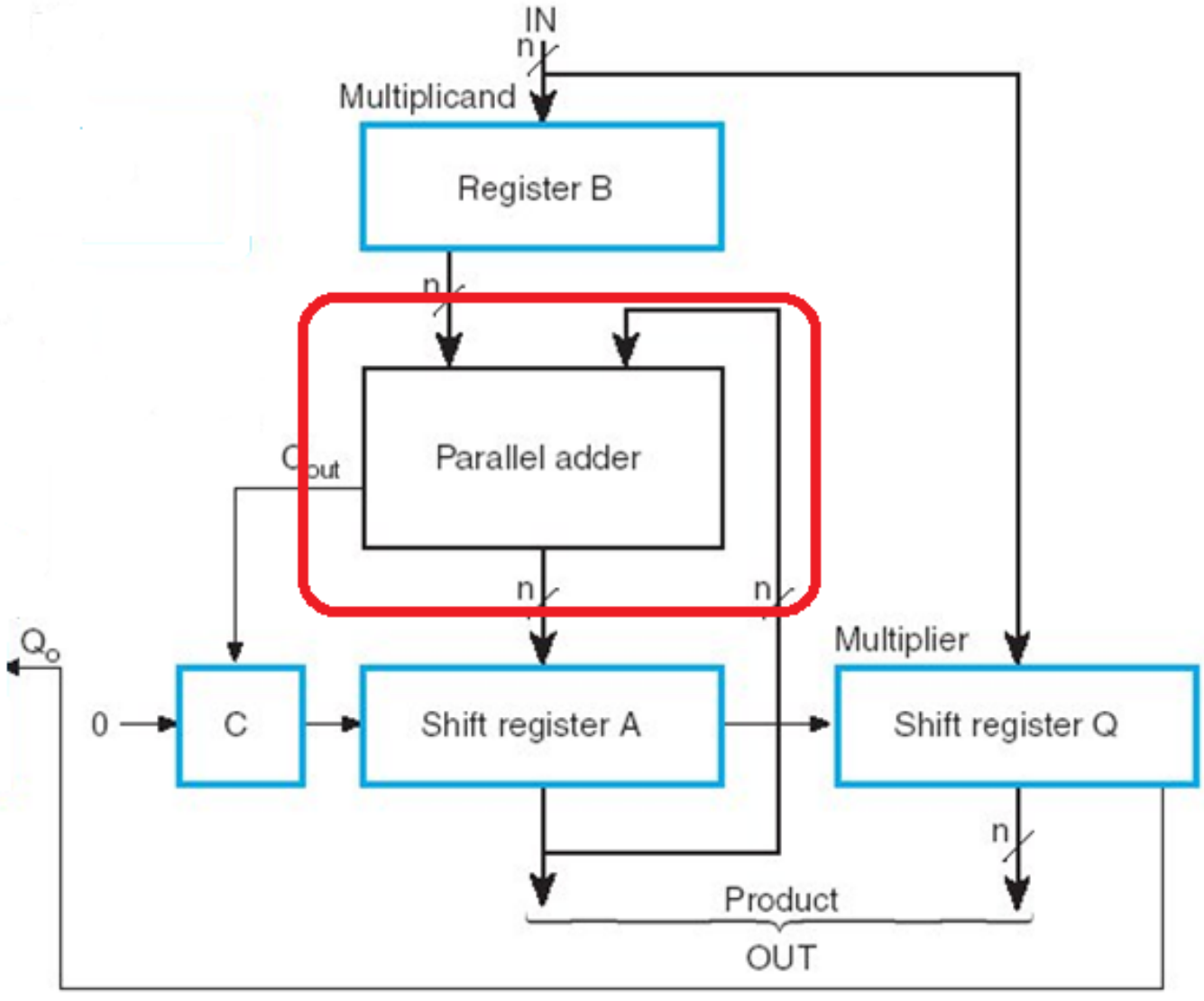
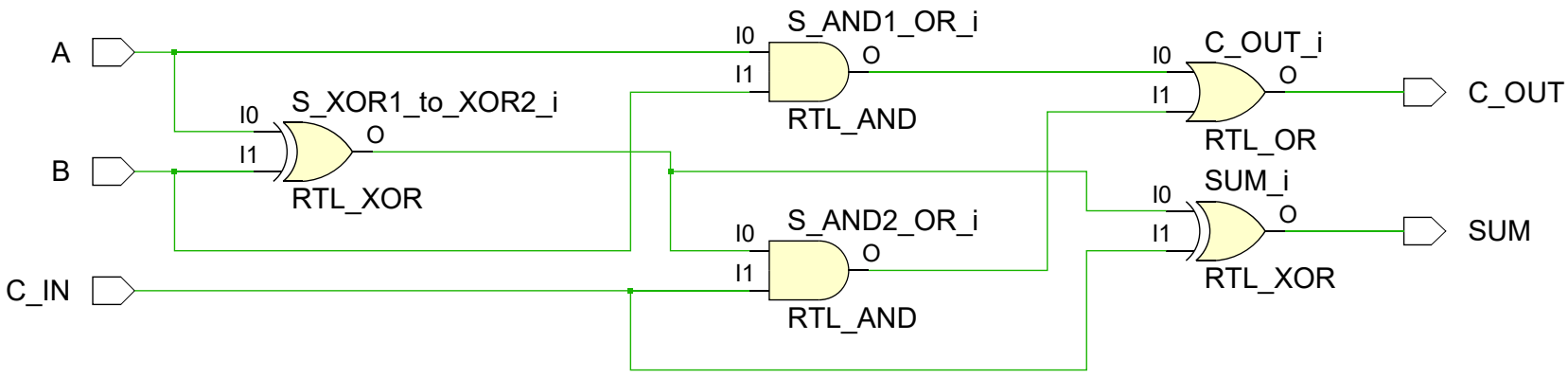


Figure 19: This design allows us to shift the Shift Flip-Flop C and the Shift Registers A and Q at the same time.

Parallel Adder



Full Adder


A
B
C_IN
S_XOR1_to_XOR2_i
I0
I1
O
RTL_XOR
S_AND1_OR_i
RTL_AND
S_AND2_OR_i
RTL_AND
C_OUT_i
RTL_OR
C_OUT
SUM_i
RTL_XOR
SUM

VHDL Implementation of the 5-bit Ripple Carry Adder