Topics for Lecture #10

- General procedure for state machine design
- Counters (concluded)
- Shift registers
- Video display basics
- EE 121 VGA module: care and feeding

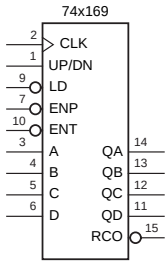
Reading assignment: DDPP 8.1, 8.2, 8.4, 8.5, 8.7, 8.8

State machine design

0. Specify desired behavior of machine unambiguously (often hardest step).
1. Construct state/output table (or draw a state diagram, often preferable).
2. Minimize the number of states (optional).
3. State assignment: choose state variables and assign values to named states.
4. Form transition/output table from state/output table using state values.
5. Choose flip-flop type. Answer: D flip-flops
6. Construct excitation table (not needed for D flip-flops)
7. Derive excitation equations from excitation table.
8. Derive output equations from transition/output table.
9. Draw logic diagram of next-state logic (or provide equations to CAD tools).

Up/down counters

The next-state logic of a binary counter can be augmented to allow it to count down as well. The traditional MSI part was the 74x169 up/down counter. Xilinx Spartan2 library part is CB4CLED.



The 74x163 CLR input has been replaced by a direction control UP/DN. Next-state logic and output logic for the 74x169:

```
[QD,QC,CB,QA] := if LD then [D,C,B,A]
                 else if ENP * ENT
                   if UP then [QD,QC,CB,QA] + [0,0,0,1]
                     else [QD,QC,CB,QA] - [0,0,0,1]
                   else [QD,QC,CB,QA];
                 RCO = UP*ENT*QD*QC*QB*QA + /UP*ENT*/QD*/QC*/QB*/QA
```


Shift registers

Shift registers are the second most important sequential circuit building block. (Counters are first; registers don't count.)

Applications of shift registers:

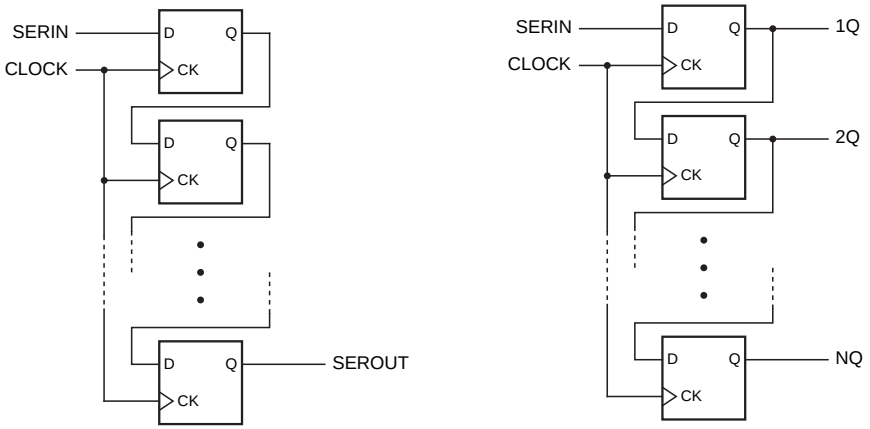
- serial-to-parallel converters
- parallel-to-serial converters
- delay
- counters

Classifications of shift registers:

- serial-in vs. parallel-in
- serial-out vs. parallel-out
- unidirection vs. bidirectional

"Universal" shift register = parallel-in, parallel-out, bidirectional.

Serial-in shift registers

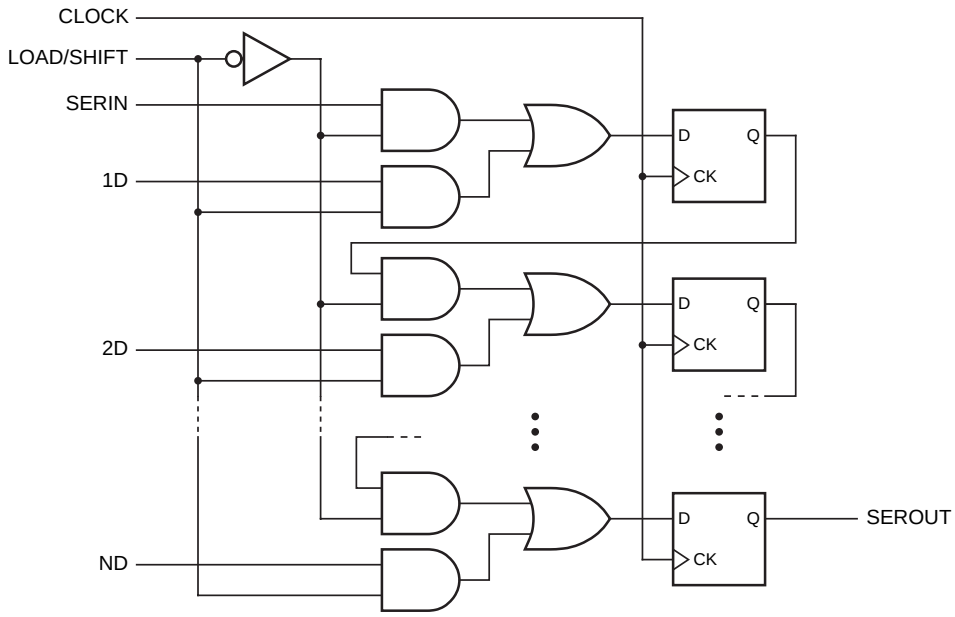


Next state logic is trivial: $Q_i^* = Q_{i-1}$.

Not so trivial: how to name the output (and inputs) of shift registers.
Several conventions are used: letters, numbers starting from 0, etc.

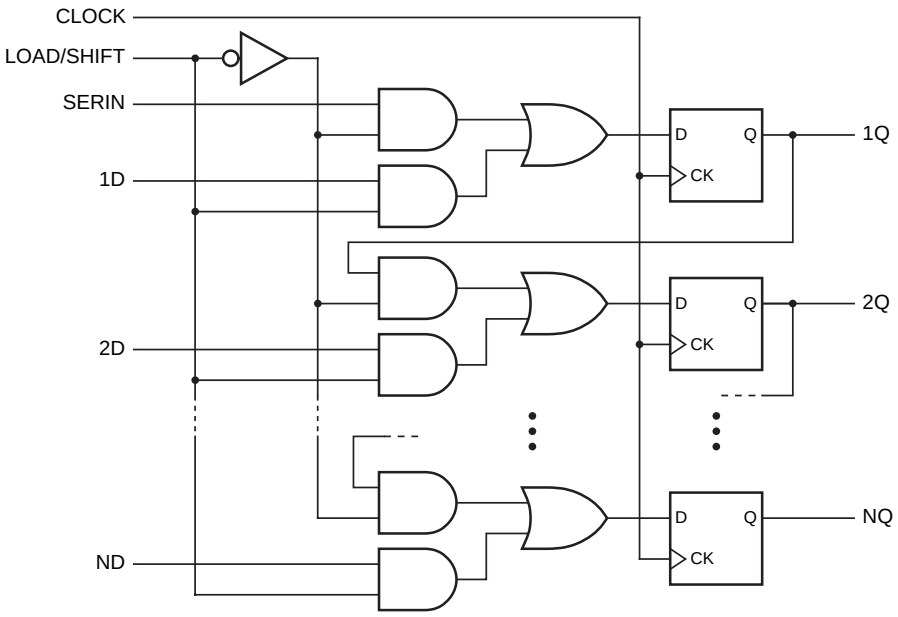
Only difference between serial-out and parallel-out is number of output pins.

Parallel-in serial-out shift register



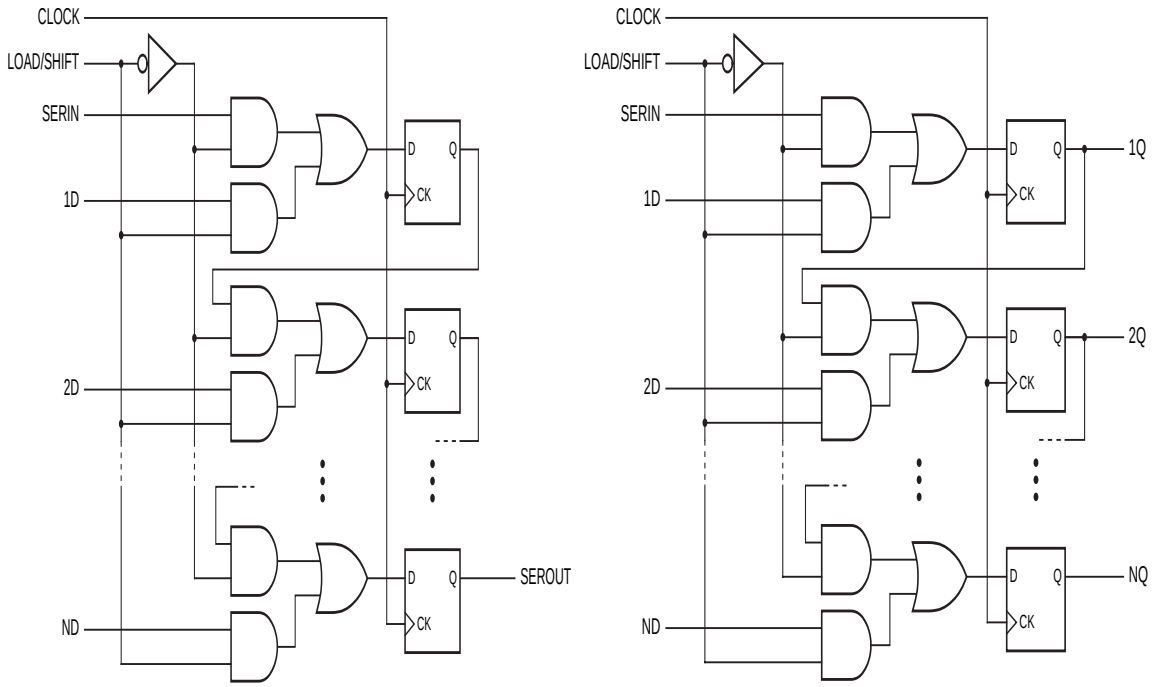
Used for parallel-to-serial conversion: load n bits, then shift for $n-1$ clocks.

Parallel-in parallel-out shift register



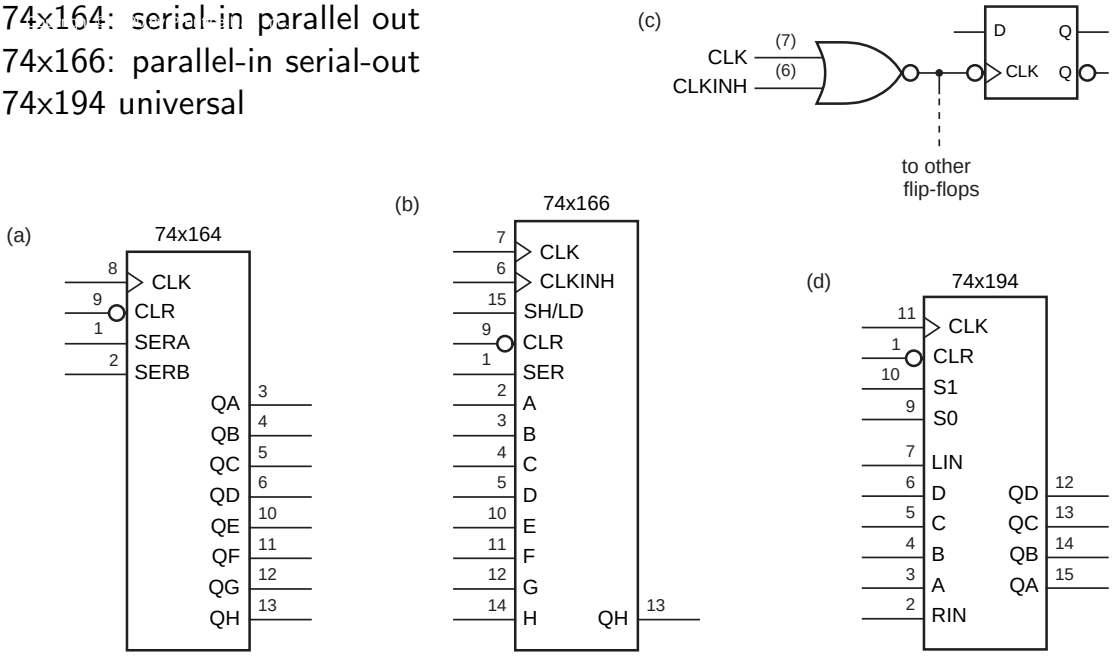
More output pins. Pin count for PI-PI shift register is $2n + c$, where $c \approx 3$.

Serial-out vs. parallel-out

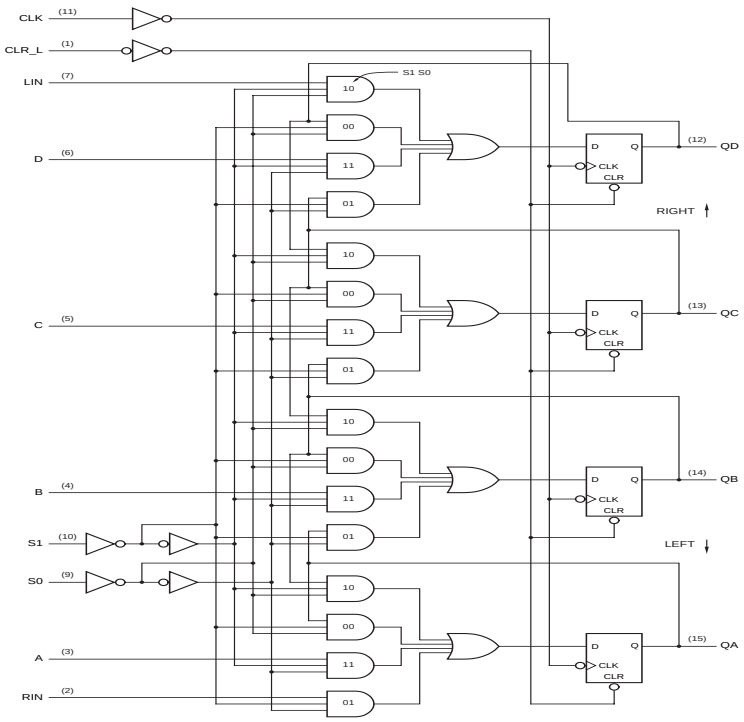


Traditional shift registers

74x164: serial-in parallel out
74x166: parallel-in serial-out
74x194 universal



74x194 universal shift register



Control signals for 74x194 universal shift register

Only two control inputs (compared to four for 74x163) requires that four actions be encoded into two bits.

Table 8-18
Function table for the 74x194 4-bit universal shift register.

	Inputs		*Next state*			
Function	***S1***	***S0***	***QA****	***QB****	***QC****	***QD****
Hold	0	0	QA	QB	QC	QD
Shift right	0	1	RIN	QA	QB	QC
Shift left	1	0	QB	QC	QD	LIN
Load	1	1	A	B	C	D

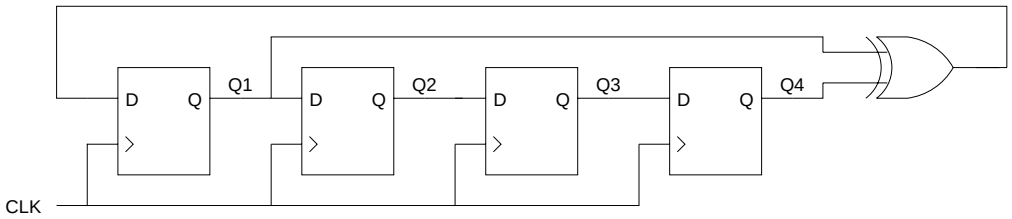
An arbitrary encoding for shift controls could be used. The chosen encoding is somewhat mnemonic.

Data signals are named with letters (A, B, C, D, QA, QB, QC, QD). Serial inputs are LIN, RIN.

Convention used: D is on the right, LIN is input when shifting left.

Linear feedback shift register counters

Cheap counters can be built using shift registers and a small amount of logic.



State sequence:

1000 1100 1110 1111 0111 1011 0101 1010
1101 0110 0011 1001 0100 0010 0001 1001

Self-correcting version of this LFSR can be built with a few more gates.