# Fundamentals of Logic Circuit Design

## Part 3: Digital Systems

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

Project is co-financed by European Union within European Social Fund

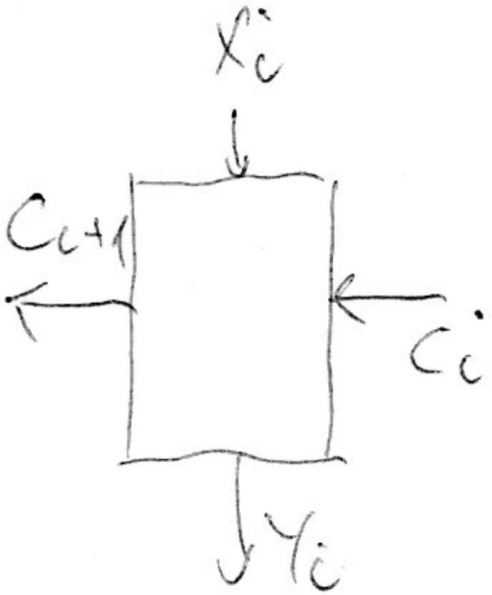
# Hierarchical Design

Design blocks of the half-adder for i = 0 and i > 0 bits. Half adder calculates y = x +1.

Inputs $x_i$,$c_i$ , outputs $y_i$, $c_{i+1}$

Result: [$c_{i+1}$ ,$y_i$] = $x_i$, + $c_i$

# Half-adder




| $x_i$ $c_i$ | $\Sigma$ | $c_{i+1}$ $y_i$ |
|---|---|---|
| 0 0 | 0 | 0 0 |
| 0 1 | 1 | 0 1 |
| 1 1 | 2 | 1 0 |
| 1 0 | 1 | 0 1 |

$$c_{i+1} = x_i c_i$$

$$y_i = \bar{x}_i c_i + x_i \bar{c}_i = x \oplus c_i$$

# Number codes

Homogeneous positional representation of numbers:

$$N = d_{n-1}d_{n-2}\ldots d_1 d_0 \centerdot d_{-1}\ldots d_{-m}$$

- $N$ – the number (value)
- $n$ – number of digits in the integer part
- $m$ – number of digits in the fractional part
- $b$ – base or radix
- $d_i$ – digit of the number $d_i \in \{0, 1, \ldots, b-1\}$

Value:

$$N = d_{n-1}b^{n-1} + \ldots + d_1 b^1 + d_0 b^0 + d_{-1}b^{-1} + \ldots + d_{-m}b^{-m}$$

Short form notation:

$$N = \sum_{i=-m}^{n-1} d_i b^i$$

Example: decimal numbers

$$N = 359.1_{10} \quad n = 3 \quad m = 1 \quad b = 10 \quad d_i \in \{0, 1, \ldots, 9\}$$

$$N = 3 * 10^2 \; + \; 5 * 10^1 \; + \; 9 * 10^0 \; + \; 1 * 10^{-1}$$

$$N = \sum_{i=-1}^{2} d_i * 10^i$$

$$d_2 = 3 \quad d_1 = 5 \quad d_0 = 9 \quad d_{-1} = 1$$

## Example: binary numbers

$$N = 1100.01_2 \quad n = 4 \quad m = 2 \quad b = 2 \quad d_i \in \{0, 1\}$$

$$N = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2}$$

$$N = \sum_{i=-2}^{3} d_i * 2^i$$

$$d_3 = 1 \quad d_2 = 1 \quad d_1 = 0 \quad d_0 = 0 \quad d_{-1} = 0 \quad d_{-2} = 1$$

## Addition: positive decimal numbers (i.e. radix $b = 10$)

| position | | $2$ | $1$ | $0$ | $-1$ |
|---|---|---|---|---|---|
| position value | | $10^2$ | $10^1$ | $10^0$ | $10^{-1}$ |
| operand 1 | | $3$ | $5$ | $9$ . | $1$ |
| operand 2 | $+$ | $2$ | $7$ | $1$ . | $0$ |
| | | $5$ | $12$ | $10$ . | $1$ |
| | | $5$ | $10+2$ | $10+0$ . | $1$ |
| | | $5$ | $10+3$ | $0$ . | $1$ |
| | | $5+1$ | $3$ | $0$ . | $1$ |
| result | | $6$ | $3$ | $0$ . | $1$ |

Addition: positive binary numbers (i.e. radix $b = 2$)

| position | | $3$ | $2$ | $1$ | $0$ | $-1$ |
|---|---|---|---|---|---|---|
| position value | | $2^3$ | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ |
| operand 1 | | $0$ | $0$ | $1$ | $1$ . | $1$ |
| operand 2 | $+$ | $0$ | $1$ | $1$ | $0$ . | $0$ |
| | | $0$ | $1$ | $10$ | $1$ . | $1$ |
| | | $0$ | $1$ | $10+0$ | $1$ . | $1$ |
| | | $0$ | $1+1$ | $0$ | $1$ . | $1$ |
| | | $0$ | $10$ | $0$ | $1$ . | $1$ |
| | | $0$ | $10+0$ | $0$ | $1$ . | $1$ |
| result | | $1$ | $0$ | $0$ | $1$ . | $1$ |

## Multiplication: positive decimal numbers (i.e. $b = 10$)

| | | | |
|---|---|---|---|
| operand 1 | | 3 | 5 |
| operand 2 | $*$ | 2 | 7 |
| | | 21 | 35 |
| | 6 | 10 | |
| | | 21 | $30 + 5$ |
| | 6 | $10 + 0$ | |
| | | $21 + 3$ | 5 |
| | $6 + 1$ | 0 | |
| | | $20 + 4$ | 5 |
| | 7 | 0 | |
| | 2 | 4 | 5 |
| | 7 | 0 | |
| result | 9 | 4 | 5 |

Multiplication: positive binary numbers (i.e. $b = 2$)

| | | | | | | |
|---|---|---|---|---|---|---|
| operand 1 | | | | 1 | 1 | 1 |
| operand 2 | $*$ | | | 1 | 0 | 1 |
| | | | | 1 | 1 | 1 |
| | | | 0 | 0 | 0 | |
| | | 1 | 1 | 1 | | |
| | | 1 | 1 | 10 | 1 | 1 |
| | | 1 | 1 | $10+0$ | 1 | 1 |
| | | 1 | $1+1$ | 0 | 1 | 1 |
| | | 1 | 10 | 0 | 1 | 1 |
| | | 1 | $10+0$ | 0 | 1 | 1 |
| | | $1+1$ | 0 | 0 | 1 | 1 |
| | | 10 | 0 | 0 | 1 | 1 |
| | | $10+0$ | 0 | 0 | 1 | 1 |
| result | 1 | 0 | 0 | 0 | 1 | 1 |

# Multiplication by radix

## Multiplication by 10 of decimal numbers

| | |
|---|---|
| operand 1 | 3 5 |
| operand 2 | $*$ 1 0 |
| | 0 0 |
| | 3 5 |
| result | 3 5 0 |

$$shl(35) = 350$$

# Multiplication by 2 of binary numbers

| | |
|---|---|
| operand 1 | $1\ 1 \rightarrow 3_{10}$ |
| operand 2 | $*\ 1\ 0 \rightarrow 2_{10}$ |
| | $0\ 0$ |
| | $1\ 1$ |
| result | $1\ 1\ 0 \rightarrow 6_{10}$ |

$$shl(11) = 110$$

# Division by radix

## Division by 10 of decimal numbers

$$350 \div 10 = 35.0$$

$$shr(350) = 35$$

$$352 \div 10 = 35.2$$

$$shr(352) = 35$$

## Division by 2 of binary numbers

$$1110 \div 10 = 111.0 \qquad \rightarrow 14_{10} \div 2_{10} = 7_{10}$$

$$shr(1110) = 111 \qquad \rightarrow 7_{10}$$

$$1111 \div 10 = 111.1 \qquad \rightarrow 15_{10} \div 2_{10} = 7.5_{10}$$

$$shr(1111) = 111 \qquad \rightarrow 7_{10}$$

# Signed-Magnitude number representation

## Any radix representation

$$N = z_n d_{n-1} d_{n-2} \ldots d_1 d_0 \blacksquare d_{-1} \ldots d_{-m}$$

$$z_n \in \{+, -\} \quad \rightarrow \quad z_n \in \{0, b-1\}$$

$$d_i \in \{0, 1, \ldots, b-1\}$$

Example

$$+30_{10} = 011110_2$$
$$-30_{10} = 111110_2$$
$$+3_{10} = 000011_2$$
$$-3_{10} = 100011_2$$

## Binary numbers

$$N = z_n d_{n-1} d_{n-2} \ldots d_1 d_0 \blacksquare d_{-1} \ldots d_{-m}$$

$$z_n \in \{+, -\} \quad \rightarrow \quad z_n \in \{0, 1\}$$

$$d_i \in \{0, 1\}$$

# Multiplication

1. Hide the signs (produce absolute values)
2. Multiply the numbers
3. Adjust the sign of the result

# Addition

1. If both numbers are positive then add them
2. If both numbers are negative:
    (a) hide the signs (produce absolute values)
    (b) add the numbers
    (c) produce negative sign
3. If one of the numbers is negative:
    (a) hide the signs (produce absolute values)
    (b) subtract the smaller number from the larger one
    (c) adjust the sign of the result

# Properties of signed-magnitude representation:

- Very simple sign change
- Difficult addition/subtraction
- Simple multiplication
- Double representation of 0 (+0 = 0000, -0 = 1000)

# Diminished radix complement representation

If radix is *b* and numbers have *n* digits (excluding sign):

$$N = z_n d_{n-1} d_{n-2} \dots d_1 d_0$$

$$z_n \in \{0, b'\}$$

$$d_i \in \{0, 1, \dots, b'\}$$

$$b' = b - 1 \quad \rightarrow \quad \underline{\text{diminished radix}}$$

$$\boxed{-N = (b^n - 1) - |N|}$$

$$b^n = 1\underbrace{00\dots00}_{\times n}$$

$$(b^n - 1) = \underbrace{b'b'\dots b'b'}_{\times n}$$

$$-N = \overline{z}_n \overline{d}_{n-1} \overline{d}_{n-2} \dots \overline{d}_1 \overline{d}_0$$

where:

$$\overline{z}_n = b' - z_n$$

$$\overline{d}_i = b' - d_i \qquad i = 0 \dots n-1$$

If numbers with an $m$-bit fractional part are considered

$$N = z_n d_{n-1} d_{n-2} \ldots d_1 d_0 \cdot d_{-1} \ldots d_{-m}$$

then

$$\boxed{-N = (b^n - b^{-m}) - |N|}$$

Example: (decimal numbers)

$$\begin{aligned} N &= \quad 3\ 5\ 7 \\ +N &= 0\ 3\ 5\ 7 \\ -N &= \overline{0}\ \overline{3}\ \overline{5}\ \overline{7} \end{aligned}$$

where

$$\overline{z}_3 = \overline{0} = (10-1) - 0 = 9 - 0 = 9$$
$$\overline{d}_2 = \overline{3} = (10-1) - 3 = 9 - 3 = 6$$
$$\overline{d}_1 = \overline{5} = (10-1) - 5 = 9 - 5 = 4$$
$$\overline{d}_0 = \overline{7} = (10-1) - 7 = 9 - 7 = 2$$

$$\Rightarrow \quad -N = 9\ 6\ 4\ 2$$

# Addition/subtraction

$$\begin{array}{lrr} N^1 \rightarrow & 0357 & 0000 \\ N^2 \rightarrow & +\ 9642 & +\ 0000 \\ \hline N^r \rightarrow & 9999 & 0000 \end{array}$$

$\Rightarrow$ Two representations of $0$: $0000$, $9999$

$$\begin{array}{rrrr} 0357 & 9642 & 0512 & 9487 \\ +\ 0357 & +\ 9642 & +\ 0512 & +\ 9487 \\ \hline 0714 & 1\ 9284 & \underline{1}024 & 1\ \underline{8}97\underline{4} \end{array}$$

$\Rightarrow$ Overflow results if: $z_n^1 = z_n^2 \neq z_n^r$

$$
\begin{array}{rr}
 & 0357 \\
- & 0295 \\
\hline
 & 0357 \\
+ & 9704 \\
\hline
1 & 0061 \\
+ & \quad 1 \\
\hline
 & 0062
\end{array}
\qquad\qquad
\begin{array}{rr}
 & 0295 \\
- & 0357 \\
\hline
 & 0295 \\
+ & 9642 \\
\hline
 & 9937 \\
\hline
- & 0062
\end{array}
$$

$\Rightarrow$ If there is a carry out of the sign position adjust the result by adding $1$ to it (end-arround carry).

This is due to two representations of 0. If the addition requires counting past the 0 boundary, what can be detected by watching the carry out of the sign position, we need to add 1 to the result. We start counting with the lower (i.e. negative number) and add increments of 1 until the count reaches the other number. If in the process of counting the 0 boundary is crossed we must add 1 to fix the count.

| | |
|---|---|
| $\dots$ | $\dots$ |
| $-2$ | $9997$ |
| $-1$ | $9998$ |
| $-0$ | $9999$ |
| $+0$ | $0000$ |
| $+1$ | $0001$ |
| $+2$ | $0002$ |
| $\dots$ | $\dots$ |

$$\left.\begin{array}{r} -295 \\ +1 \\ \vdots \\ +1 \\ \hline 61 \\ +1 \\ \hline 62 \end{array}\right\} \times 357$$

$$\left.\begin{array}{r} -357 \\ +1 \\ \vdots \\ +1 \\ \hline -62 \end{array}\right\} \times 295$$

# Example (binary numbers):

$$N = \quad 1\ 0\ 1$$

$$+N = 0\ 1\ 0\ 1$$

$$-N = \overline{0}\ \overline{1}\ \overline{0}\ \overline{1}$$

where

$$\overline{z}_3 = \overline{0} = (2-1) - 0 = 1 - 0 = 1$$

$$\overline{d}_2 = \overline{1} = (2-1) - 1 = 1 - 1 = 0$$

$$\overline{d}_1 = \overline{0} = (2-1) - 0 = 1 - 0 = 1$$

$$\overline{d}_0 = \overline{1} = (2-1) - 1 = 1 - 1 = 0$$

$$\Rightarrow\ -N = 1\ 0\ 1\ 0$$

# Addition/subtraction

$$\begin{array}{lrr} N^1 \rightarrow & 0101 & 0000 \\ N^2 \rightarrow & +\ 1010 & +\ 0000 \\ \hline N^r \rightarrow & 1111 & 0000 \end{array}$$

$\Rightarrow$ Two representations of $0$: $0000$, $1111$

$$\begin{array}{rrrr} 0011 & 1100 & 0101 & 1010 \\ +\ 0011 & +\ 1100 & +\ 0101 & +\ 1010 \\ \hline 0110 & 1\ 1000 & \underline{1}010 & 1\ \underline{0}10\underline{0} \end{array}$$

$\Rightarrow$ Overflow results if: $z_n^1 = z_n^2 \neq z_n^r$

$$
\begin{array}{r}
0101 \\
-\ 0011 \\
\hline
0101 \\
+\ 1100 \\
\hline
1\ 0001 \\
+\quad 1 \\
\hline
0010
\end{array}
\qquad
\begin{array}{r}
0011 \\
-\ 0101 \\
\hline
0011 \\
+\ 1010 \\
\hline
1101 \\
\hline
-\ 0010
\end{array}
$$

$\Rightarrow$ If there is a carry out of the sign position adjust the result by adding $1$ to it (end-arround carry).

# Properties of diminished radix complement representation:

- Fairly simple sign change
- Fairly simple addition/subtraction
- Fairly simple multiplication
- Double representation of 0 (+0 = 0000, -0 = 1111)

# Radix complement representation

If radix is *b* and numbers have *n* digits (excluding sign):

$$N = z_n d_{n-1} d_{n-2} \dots d_1 d_0$$

$$z_n \in \{0, b'\}$$

$$d_i \in \{0, 1, \dots, b'\}$$

$$b' = b - 1$$

$$\boxed{-N = b^n - |N|}$$

$$-N = (b^n - 1) - |N| + 1$$

$$b^n = 1\underbrace{00\dots00}_{\times n}$$

$$(b^n - 1) = \underbrace{b'b'\dots b'b'}_{\times n}$$

$$-N = \overline{z}_n \overline{d}_{n-1} \overline{d}_{n-2} \dots \overline{d}_1 \overline{d}_0 + 1$$

where:

$$\overline{z}_n = b' - z_n$$

$$\overline{d}_i = b' - d_i \qquad i = 0 \dots n-1$$

If numbers with an $m$-bit fractional part are considered

$$N = z_n d_{n-1} d_{n-2} \ldots d_1 d_0 \blacksquare d_{-1} \ldots d_{-m}$$

then

$$\boxed{-N = b^n - |N|}$$

# Example (decimal numbers):

$$N = \quad 3\ 5\ 7$$

$$+N = 0\ 3\ 5\ 7$$

$$-N = \overline{0}\ \overline{3}\ \overline{5}\ \overline{7} + 1$$

where

$$\overline{z}_3 = \overline{0} = (10-1) - 0 = 9 - 0 = 9$$

$$\overline{d}_2 = \overline{3} = (10-1) - 3 = 9 - 3 = 6$$

$$\overline{d}_1 = \overline{5} = (10-1) - 5 = 9 - 5 = 4$$

$$\overline{d}_0 = \overline{7} = (10-1) - 7 = 9 - 7 = 2$$

$$\Rightarrow -N = 9\ 6\ 4\ 2\ +1$$

$$= 9\ 6\ 4\ 3$$

| | | |
|---|---|---|
| $N^1 \rightarrow$ | $0357$ | $0000$ |
| $N^2 \rightarrow$ | $+\ 9643$ | $+\ 0000$ |
| $N^r \rightarrow$ | $1\ 0000$ | $0000$ |

$\Rightarrow$ One representation of $0$: $0000$

| | | | |
|---|---|---|---|
| $0357$ | $9643$ | $0512$ | $9488$ |
| $+\ 0357$ | $+\ 9643$ | $+\ 0512$ | $+\ 9488$ |
| $0714$ | $1\ 9286$ | $\underline{1}024$ | $1\ \underline{8}976$ |

$\Rightarrow$ Overflow results if: $z_n^1 = z_n^2 \neq z_n^r$

# Example: (binary numbers)

$$N = \quad 1\ 0\ 1$$
$$+N = 0\ 1\ 0\ 1$$
$$-N = \overline{0}\ \overline{1}\ \overline{0}\ \overline{1}\ +1$$

where

$$\overline{z}_3 = \overline{0} = (2-1) - 0 = 1 - 0 = 1$$
$$\overline{d}_2 = \overline{1} = (2-1) - 1 = 1 - 1 = 0$$
$$\overline{d}_1 = \overline{0} = (2-1) - 0 = 1 - 0 = 1$$
$$\overline{d}_0 = \overline{1} = (2-1) - 1 = 1 - 1 = 0$$

$$\begin{array}{r} -N = 1\ 0\ 1\ \ 0 \\ +\ 1 \\ \hline = 1\ 0\ 1\ \ 1 \end{array}$$

$$-5_{10} = 1\ 0\ 1\ \ 1$$
$$-(-5_{10}) = \overline{1}\ \overline{0}\ \overline{1}\ \ \overline{1}\ +1$$
$$= 0\ 1\ 0\ \ 0$$
$$\begin{array}{r} +\ 1 \\ \hline = 0\ 1\ 0\ \ 1 = 5_{10} \end{array}$$

$$\Rightarrow\ -(-5_{10}) = 5_{10}$$
$$-(-N) = N$$

# Addition/subtraction

$$\begin{array}{rrr} N^1 \rightarrow & 0101 & 0000 \\ N^2 \rightarrow & +\ 1011 & +\ 0000 \\ \hline N^r \rightarrow & 1\ 0000 & 0000 \end{array}$$

$\Rightarrow$ One representation of $0$: $0000$

$$\begin{array}{rrrr} 0011 & 1101 & 0101 & 1011 \\ +\ 0011 & +\ 1101 & +\ 0101 & +\ 1011 \\ \hline 0110 & 1\ 1010 & \underline{1}010 & 1\ \underline{0}110 \end{array}$$

$\Rightarrow$ Overflow results if: $z_n^1 = z_n^2 \neq z_n^r$

# Properties of radix complement representation:

- Slightly complex sign change
- Simple addition/subtraction
- Fairly simple multiplication
- Single representation of 0

# Multiplication by 2 of 2's complement numbers

| | |
|---|---|
| operand 1 | $0\ 0\ 1\ 1 \rightarrow 3_{10}$ |
| operand 2 | $*\ 1\ 0 \rightarrow 2_{10}$ |
| | $0\ 0\ 0\ 0$ |
| | $0\ 0\ 1\ 1$ |
| result | $0\ 1\ 1\ 0 \rightarrow 6_{10}$ |

$shl_l(0011) = 0110 \quad \rightarrow \quad$ logical shift left

| | |
|---|---|
| operand 1 | $1\ 1\ 0\ 1 \rightarrow -3_{10}$ |
| operand 2 | $*\ 1\ 0 \rightarrow 2_{10}$ |
| | $0\ 0\ 0\ 0$ |
| | $1\ 1\ 0\ 1$ |
| result | $1\ 0\ 1\ 0 \rightarrow -6_{10}$ |

$shl_l(1101) = 1010 \quad \rightarrow \quad$ logical shift left

| | |
|---|---|
| operand 1 | $0\ 1\ 0\ 1 \rightarrow 5_{10}$ |
| operand 2 | $*\ 1\ 0 \rightarrow 2_{10}$ |
| | $0\ 0\ 0\ 0$ |
| | $0\ 1\ 0\ 1$ |
| result | $1\ 0\ 1\ 0 \rightarrow -6_{10} \neq 10_{10}$ |

$shl_l(0101) = 1010 \quad \rightarrow \quad$ logical shift left

| | |
|---|---|
| operand 1 | $1\ 0\ 1\ 0 \rightarrow -6_{10}$ |
| operand 2 | $*\ 1\ 0 \rightarrow 2_{10}$ |
| | $0\ 0\ 0\ 0$ |
| | $1\ 0\ 1\ 0$ |
| result | $0\ 1\ 0\ 0 \rightarrow 4_{10} \neq -12_{10}$ |

$shl_l(1010) = 0100 \quad \rightarrow$ logical shift left

$\Rightarrow$ The result is correct if $z_n = d_{n-1}$

$\Rightarrow$ Adequate word length must be ensured

# Points to remember

- Number of bits must be at least one more bit than is required for the natural binary representation of the numerically larger operand
- In multiplication we need twice as many bits as we have in original operands.
- Decide which operand will be the multiplier and which will be the multiplicand

# Booth's Alorithm

1. Add 0 to the right side of multipliplier and clear product.
2. If two least signifcant bits of multipliplier are:
   1. 01 – add multiplicand to product,
   2. 10 – substract multiplicand from product.
3. Shift multiplier to the left and multiplicand to the right.
4. Repeat steps 2-3 n times

- Numbers must be coded in compliments two representation using n bits.
- Booth's algorithm performs an addition when it encounters the change from 0 to 1 and a subtraction when it encounters the change from 1 to 0.
- Booth's algorithm performs less operations than the normal multiplication algorithm if ones or zeros in a multiplier are grouped into long blocks,

# Multiplication 3 by 3 Example

3 : 011 -3 : 101

| | | | |
|---|---|---|---|
| 000000 | 011_0 | | stage 1 |
| +101 | | sub | |
| 101000 | | | |
| 110100 | 001_1 | shr | |
| 111010 | 000_1 | shr | stage 2 |
| +011 | | add | stage 3 |
| 110010 | | | |
| xx1001 | | shr | |

# Algorithm Explanation

$$F = (a_{n-1}, \ldots, a_1, a_0) * b$$

$$F = (0 - a_0) * b * 2^0 +$$
$$(a_0 - a_1) * b * 2^1 +$$
$$\ldots$$
$$(a_{n-2} - a_{n-1}) * b * 2^{n-1} +$$
$$a_{n-1} * b * 2^n$$

$$2^i a_i = 2^{i+1} a_i - 2^i a_i$$

# Integer Division

Dividend = Quotient × Divisor + Remainder

16 = 5 x 3 + 1 10001 / 11

| | | | |
|---|---|---|---|
| 10001 | | 11000 | shift right divisor |
| -01100 | 001 | 01100 | substract |
| 00100 | 010 | 00110 | shift right divisor |
| -00011 | 101 | 00011 | substract |
| 00001 | | | remainder |

# Division by 2 of 2's complement numbers

$$0110 \div 10 = 0011.0 \quad \rightarrow 6_{10} \div 2_{10} = 3_{10}$$

$$shr_a(0110) = 0011 \quad \rightarrow 3_{10}$$

shift right arithmetic $\Rightarrow (z'_n \leftarrow z_n)$

$$0111 \div 10 = 0011.1 \quad \rightarrow 7_{10} \div 2_{10} = 3.5_{10}$$

$$shr_a(0111) = 0011 \quad \rightarrow 3_{10}$$

$$1110 \div 10 = 1111.0 \quad \rightarrow -2_{10} \div 2_{10} = -1_{10}$$

$$shr_a(1110) = 1111 \quad \rightarrow -1_{10}$$

$$1111 \div 10 = 1111.1 \quad \rightarrow -1_{10} \div 2_{10} = -0.5_{10}$$

$$shr_a(1111) = 1111 \quad \rightarrow -1_{10}$$

The result is rounded downwards by truncating the fractional part

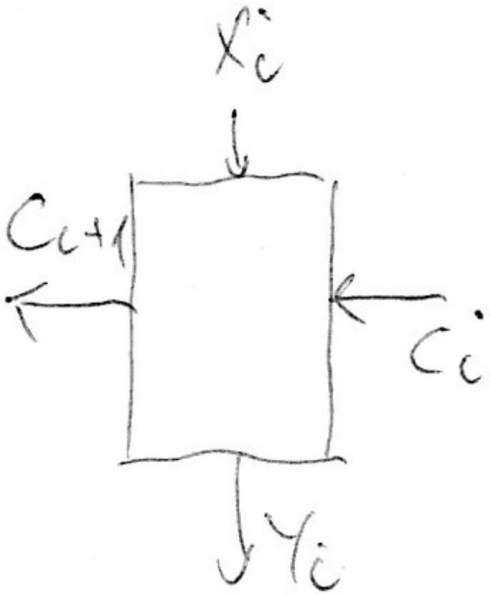
# Hierarchical Design

Design blocks of the half-adder for i = 0 and i > 0 bits. Half adder calculates y = x +1.

Inputs $x_i$,$c_i$ , outputs $y_i$, $c_{i+1}$

Result: [$c_{i+1}$ ,$y_i$] = $x_i$, + $c_i$

# Half-adder




| $x_i$ $c_i$ | $\Sigma$ | $c_{i+1}$ $y_i$ |
|---|---|---|
| 0 0 | 0 | 0 0 |
| 0 1 | 1 | 0 1 |
| 1 1 | 2 | 1 0 |
| 1 0 | 1 | 0 1 |

$$c_{i+1} = x_i c_i$$

$$y_i = \bar{x}_i c_i + x_i \bar{c}_i = x \oplus c_i$$

# Functional blocks

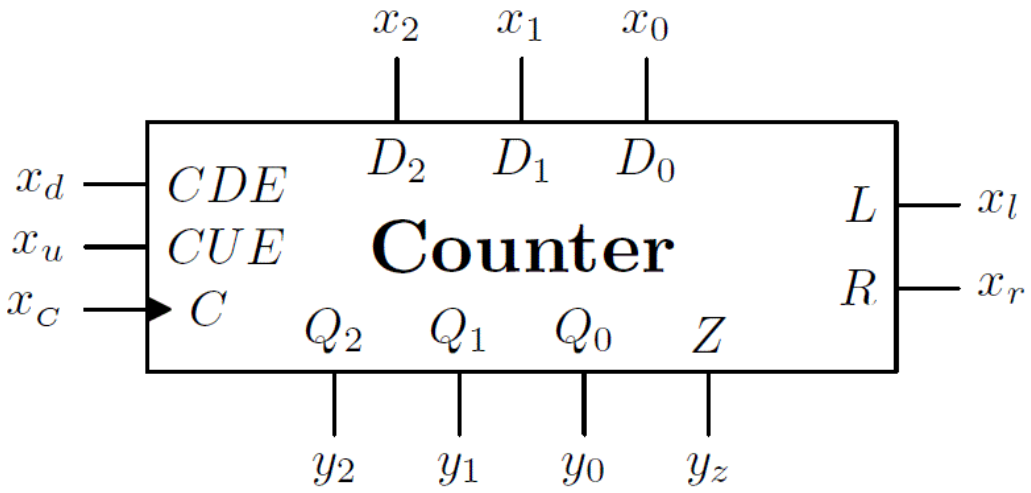
**Design:** 3-bit reversible synchronous counter with static loading and resetting and zero contents detection.

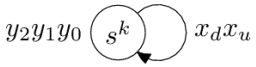
| $x_d$ | $x_u$ | function |
|---|---|---|
| 0 | 0 | stop |
| 0 | 1 | count up |
| 1 | 0 | count down |
| 1 | 1 | forbidden |

| $x_l$ | $x_r$ | function |
|---|---|---|
| 0 | 0 | synchronous operation |
| 0 | 1 | reset: $\Rightarrow y_i = 0, \quad i = 0, 1, 2$ |
| 1 | 0 | load: $\Rightarrow y_i = x_i, \quad i = 0, 1, 2$ |
| 1 | 1 | forbidden |

# Design steps:

- design the synchronous automaton, i.e. reversible synchronous counter (use D or JK type flip-flops)
- design the asynchronous automaton, i.e. static loading and resetting (use static RS type flip-flops)
- design the combinational automaton, i.e. zero contents detection

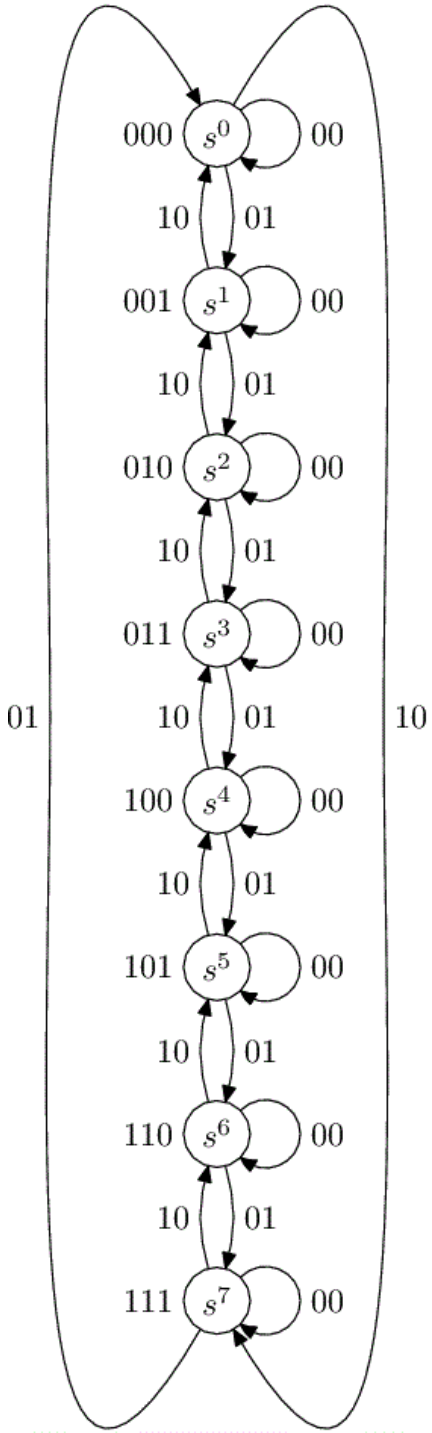





| $s^i$ | $\longrightarrow$ | $q_2$ | $q_1$ | $q_0$ |
|---|---|---|---|---|
| $s^0$ | $\longrightarrow$ | 0 | 0 | 0 |
| $s^1$ | $\longrightarrow$ | 0 | 0 | 1 |
| $s^2$ | $\longrightarrow$ | 0 | 1 | 0 |
| $s^3$ | $\longrightarrow$ | 0 | 1 | 1 |
| $s^4$ | $\longrightarrow$ | 1 | 0 | 0 |
| $s^5$ | $\longrightarrow$ | 1 | 0 | 1 |
| $s^6$ | $\longrightarrow$ | 1 | 1 | 0 |
| $s^7$ | $\longrightarrow$ | 1 | 1 | 1 |

$$\Downarrow$$

$$[y_2, y_1, y_0] = [q_2, q_1, q_0]$$

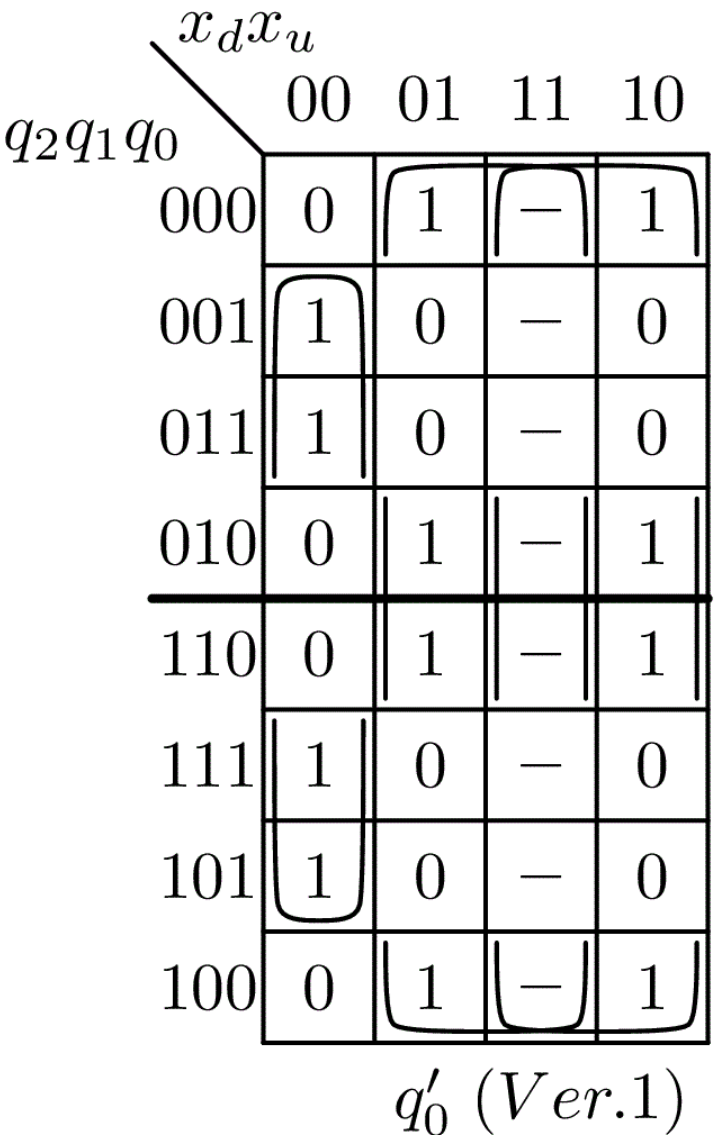


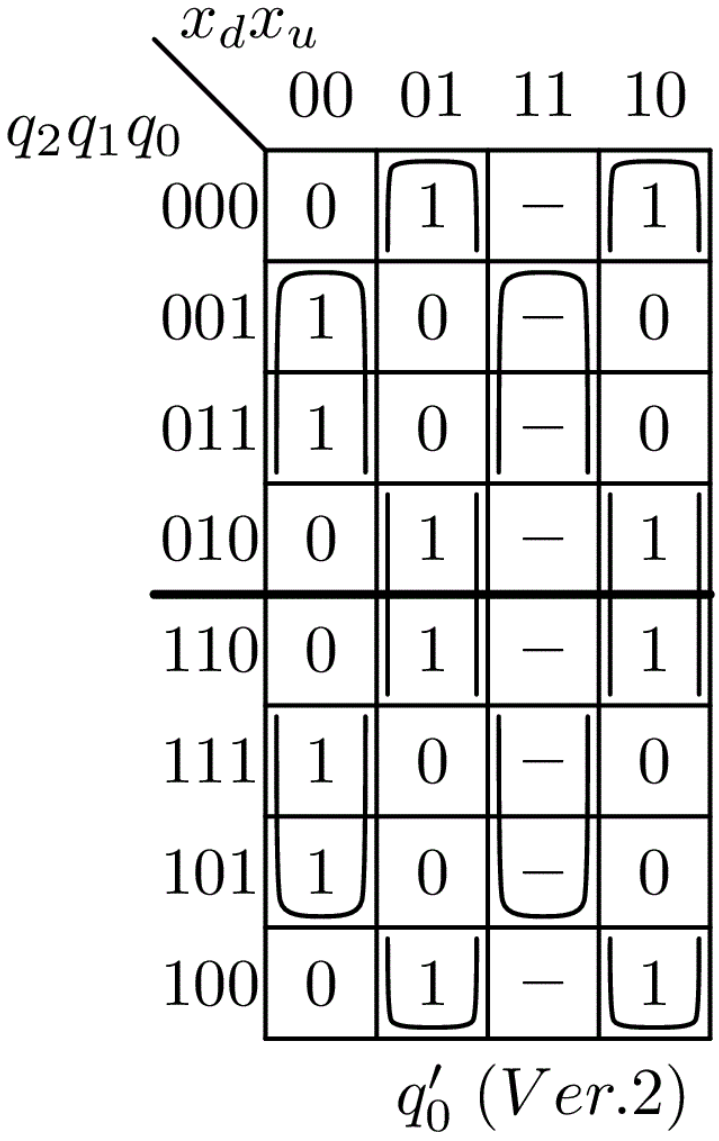

| $s$ \ $x_dx_u$ | 00 | 01 | 11 | 10 | $y$ |
|---|---|---|---|---|---|
| $s^0$ | $s^0$ | $s^1$ | – | $s^7$ | 0 |
| $s^1$ | $s^1$ | $s^2$ | – | $s^0$ | 1 |
| $s^2$ | $s^2$ | $s^3$ | – | $s^1$ | 2 |
| $s^3$ | $s^3$ | $s^4$ | – | $s^2$ | 3 |
| $s^4$ | $s^4$ | $s^5$ | – | $s^3$ | 4 |
| $s^5$ | $s^5$ | $s^6$ | – | $s^4$ | 5 |
| $s^6$ | $s^6$ | $s^7$ | – | $s^5$ | 6 |
| $s^7$ | $s^7$ | $s^0$ | – | $s^6$ | 7 |

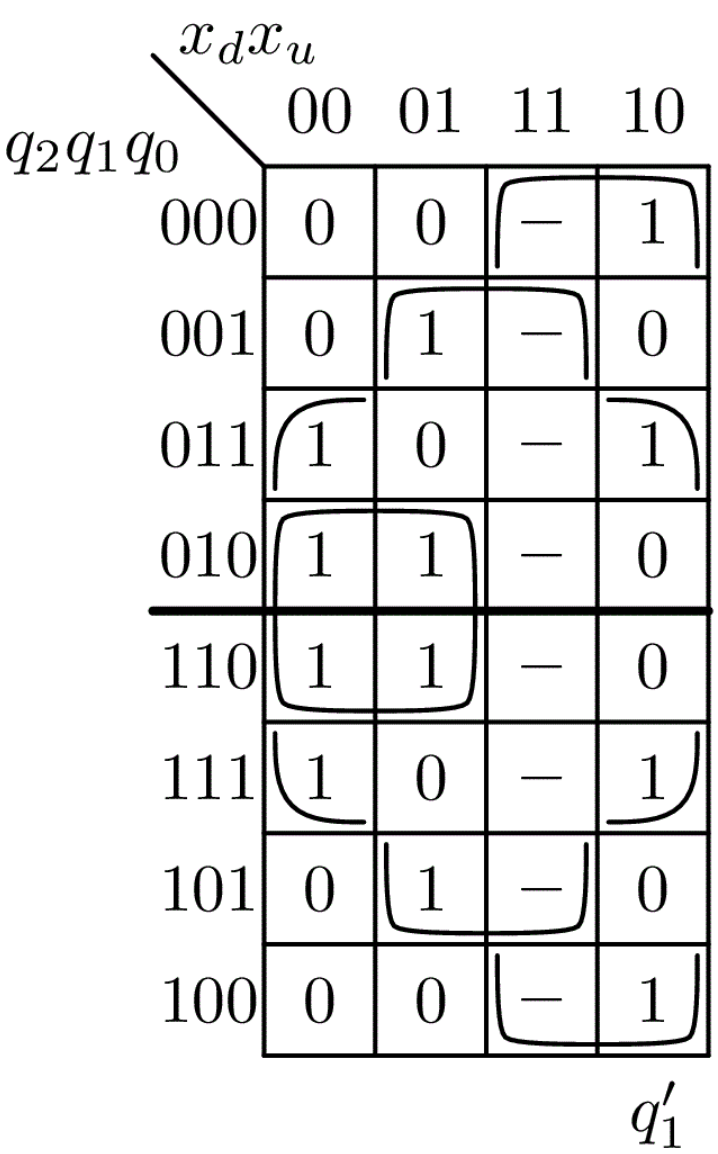
$s'/y$

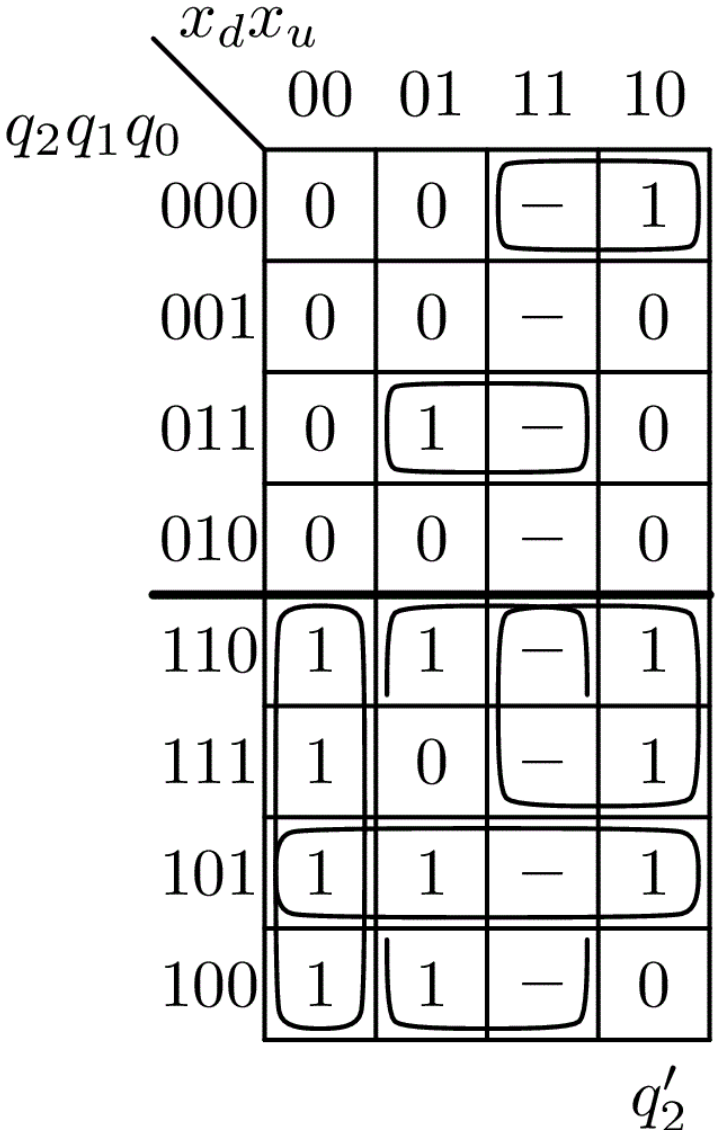
| $q_2q_1q_0$ \ $x_dx_u$ | 00 | 01 | 11 | 10 | $y_2y_1y_0$ |
|---|---|---|---|---|---|
| 000 | 000 | 001 | – | 111 | 000 |
| 001 | 001 | 010 | – | 000 | 001 |
| 011 | 011 | 100 | – | 010 | 011 |
| 010 | 010 | 011 | – | 001 | 010 |
| 110 | 110 | 111 | – | 101 | 110 |
| 111 | 111 | 000 | – | 110 | 111 |
| 101 | 101 | 110 | – | 100 | 101 |
| 100 | 100 | 101 | – | 011 | 100 |

$q'_2q'_1q'_0/y$


$x_d x_u$
$q_2 q_1 q_0$
00 01 11 10
000 0 1 – 1
001 1 0 – 0
011 1 0 – 0
010 0 1 – 1
110 0 1 – 1
111 1 0 – 0
101 1 0 – 0
100 0 1 – 1
$q_0'$ $(Ver.1)$


$x_d x_u$
$q_2 q_1 q_0$
00 01 11 10
000 0 1 – 1
001 1 0 – 0
011 1 0 – 0
010 0 1 – 1
110 0 1 – 1
111 1 0 – 0
101 1 0 – 0
100 0 1 – 1
$q_0'$ $(Ver.2)$


$x_d x_u$
$q_2 q_1 q_0$
00 01 11 10
000 0 0 – 1
001 0 1 – 0
011 1 0 – 1
010 1 1 – 0
110 1 1 – 0
111 1 0 – 1
101 0 1 – 0
100 0 0 – 1
$q_1'$


$x_d x_u$
$q_2 q_1 q_0$
00 01 11 10
000 0 0 – 1
001 0 0 – 0
011 0 1 – 0
010 0 0 – 0
110 1 1 – 1
111 1 0 – 1
101 1 1 – 1
100 1 1 – 0
$q_2'$

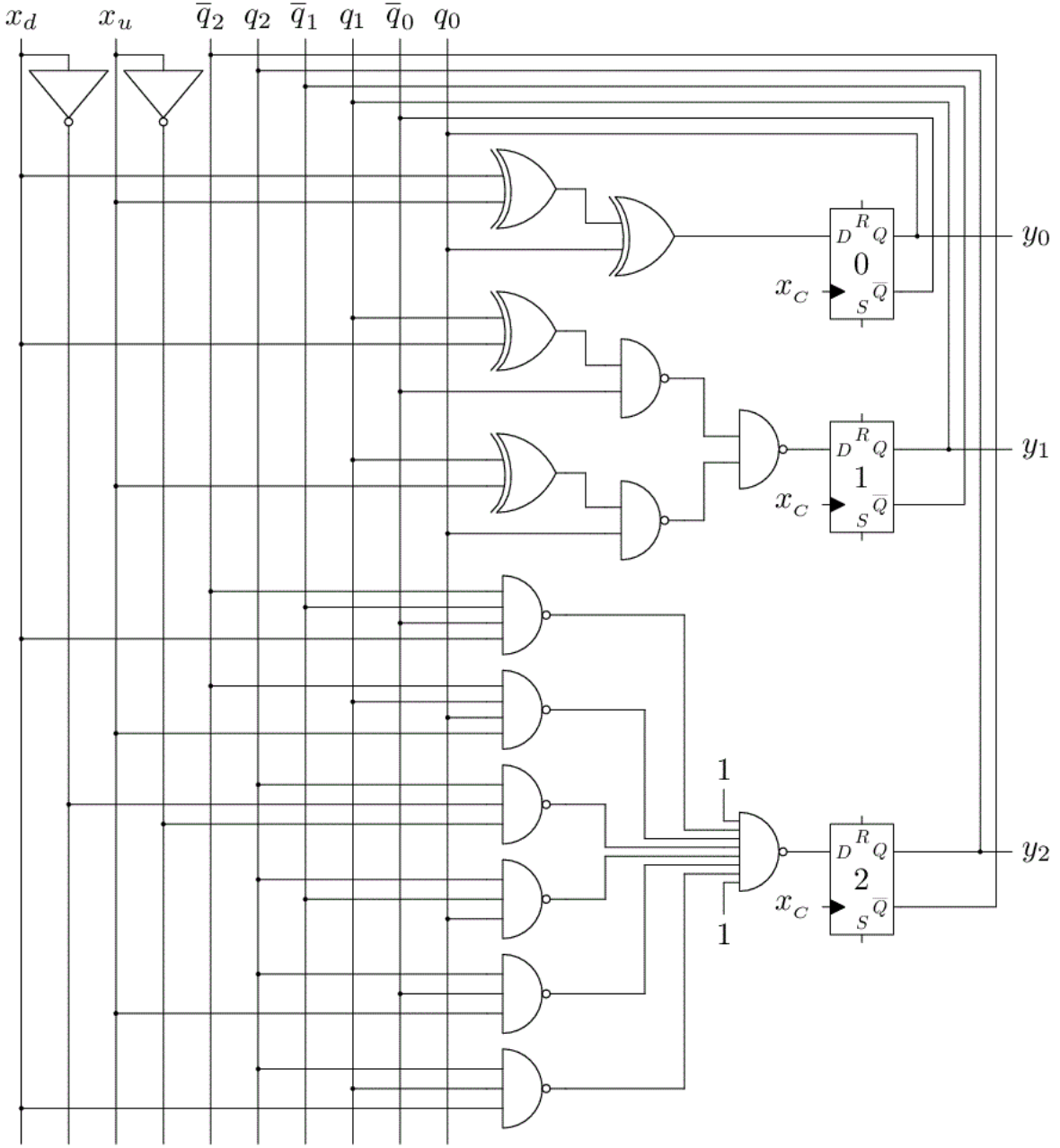
$$q_2 = \overline{q}_2\overline{q}_1\overline{q}_0 x_d + \overline{q}_2 q_1 q_0 x_u + q_2\overline{x}_d\overline{x}_u + q_2\overline{q}_1 q_0 + q_2\overline{q}_0 x_u + q_2 q_1 x_d$$

$$\begin{aligned} q_1 &= \overline{q}_1\overline{q}_0 x_d + q_1\overline{q}_0\overline{x}_d + \overline{q}_1 q_0 x_u + q_1 q_0\overline{x}_u \\ &= \overline{q}_0\left(\overline{q}_1 x_d + q_1\overline{x}_d\right) + q_0\left(\overline{q}_1 x_u + q_1\overline{x}_u\right) \\ &= \overline{q}_0\left(q_1 \oplus x_d\right) + q_0\left(q_1 \oplus x_u\right) \end{aligned}$$

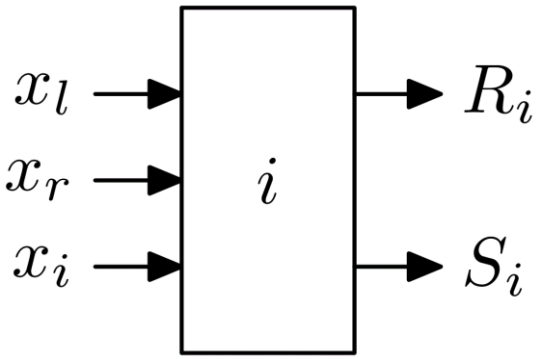
$$q_0 = \overline{q}_0 x_d + \overline{q}_0 x_u + q_0\overline{x}_d\overline{x}_u \qquad (Ver.1)$$

$$\begin{aligned} q_0 &= q_0\overline{x}_d\overline{x}_u + q_0 x_d x_u + \overline{q}_0\overline{x}_d x_u + \overline{q}_0 x_d\overline{x}_u \qquad (Ver.2) \\ &= q_0\left(\overline{x_d \oplus x_u}\right) + \overline{q}_0\left(x_d \oplus x_u\right) \\ &= q_0 \oplus \left(x_d \oplus x_u\right) \end{aligned}$$

**Reversible synchronous counter circuit diagram**



# Static loading and resetting



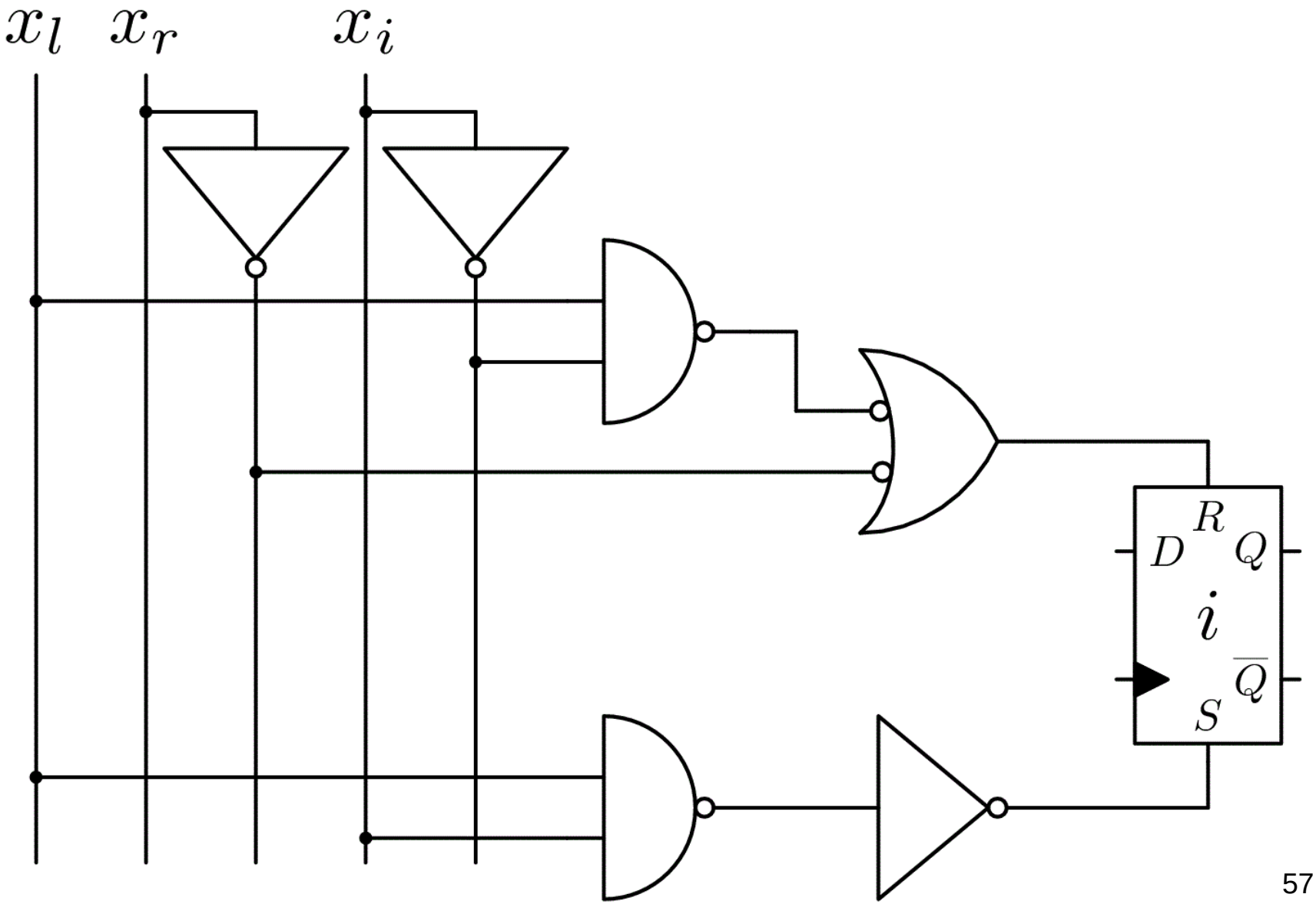

| $x_l x_r$ \ $x_i$ | 0 | 1 |
|---|---|---|
| 00 | 0 | 0 |
| 01 | 0 | 0 |
| 11 | – | – |
| 10 | 0 | 1 |

$S_i$

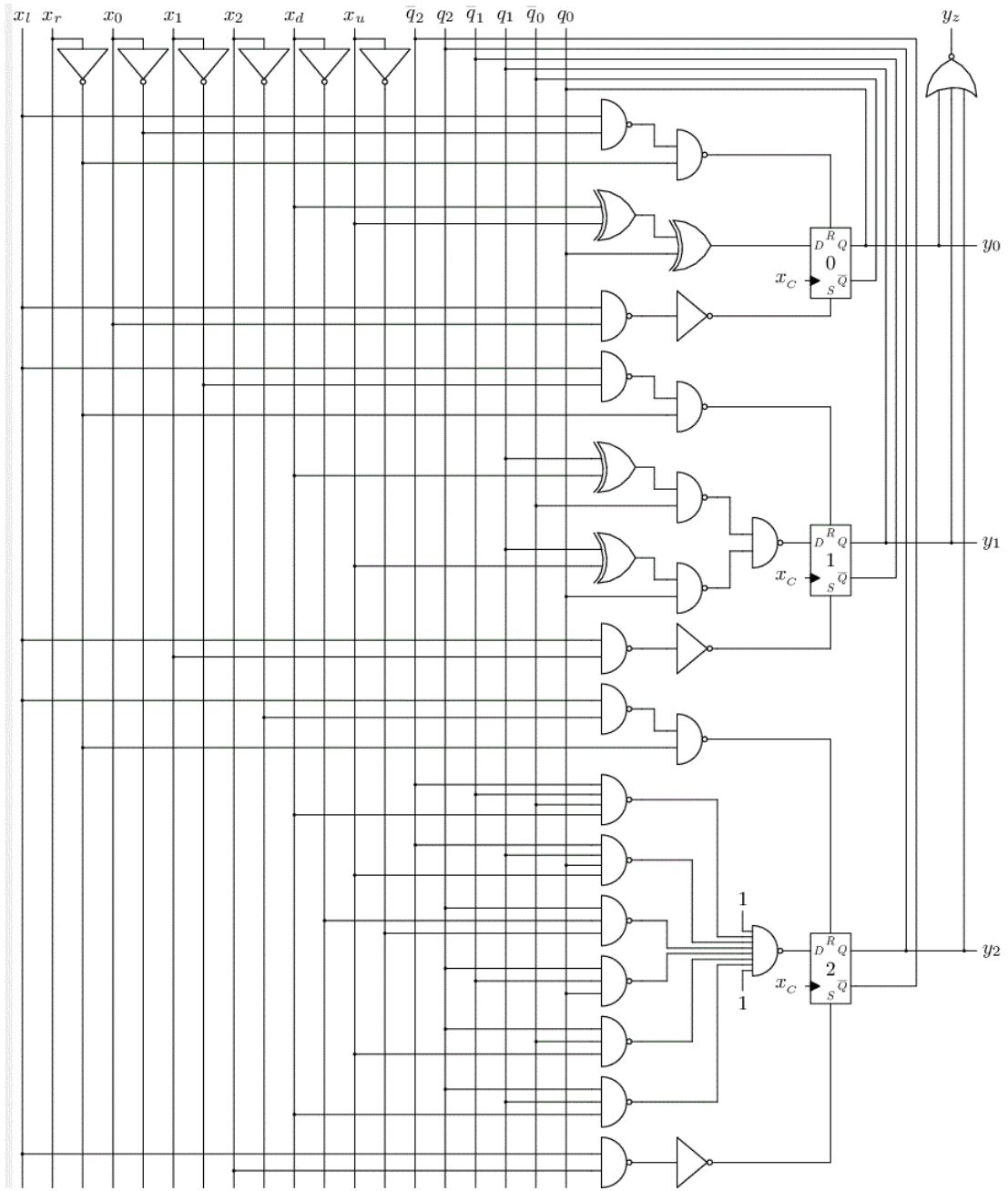
| $x_l x_r$ \ $x_i$ | 0 | 1 |
|---|---|---|
| 00 | 0 | 0 |
| 01 | 1 | 1 |
| 11 | – | – |
| 10 | 1 | 0 |

$R_i$

$$\begin{cases} R_i = x_r + x_l \overline{x}_i \\ S_i = x_l x_i \end{cases} \qquad i = 0, 1, 2$$


$x_l$
$x_r$
$x_i$
D
R
Q
i
S
$\overline{Q}$

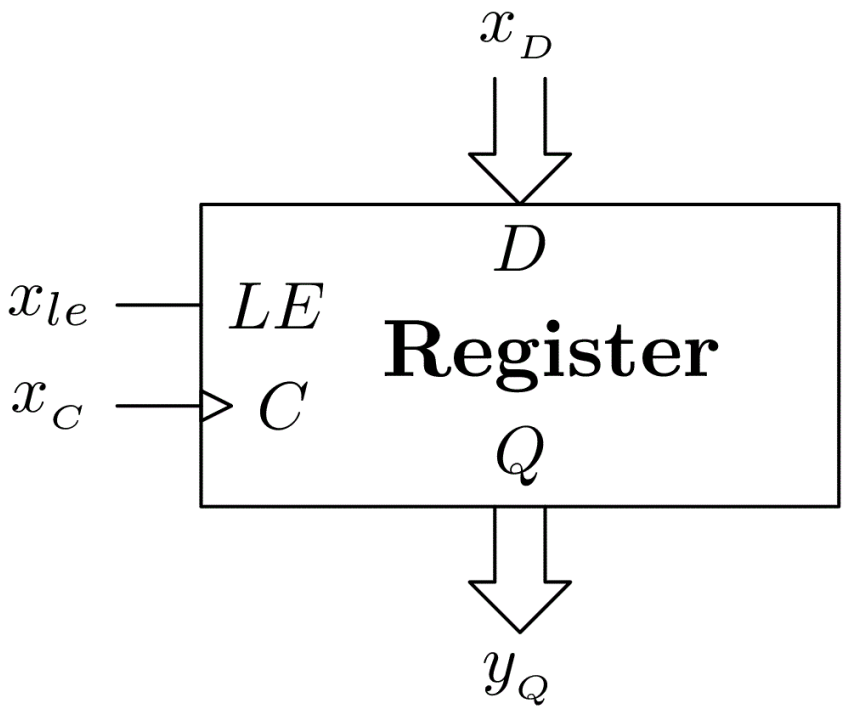
# Reversible synchronous counter Full circuit diagram



# Functional blocks

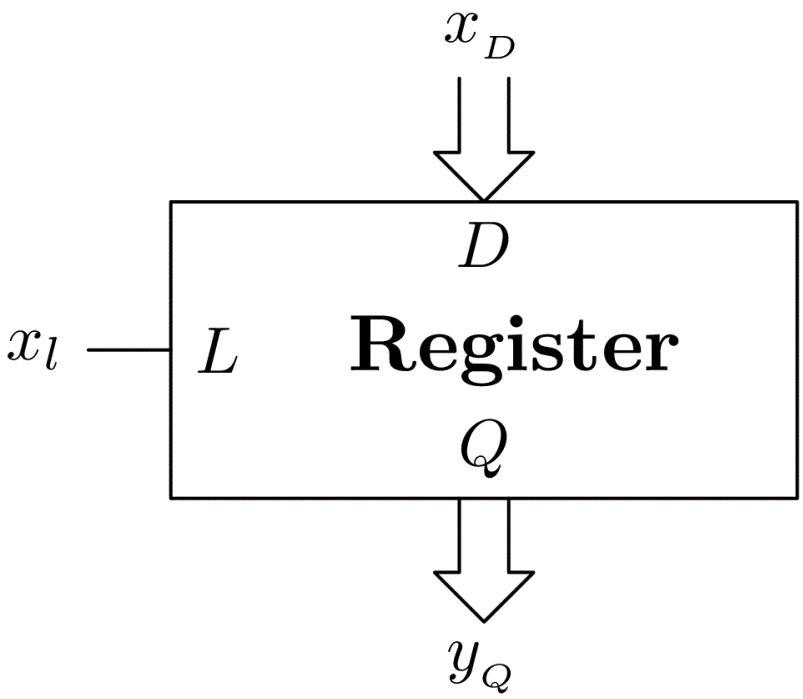
- **Input types:**
  - Data
  - Control
- **Control actions:**
  - Synchronous
  - Static
  - Dynamic

# Synchronous load



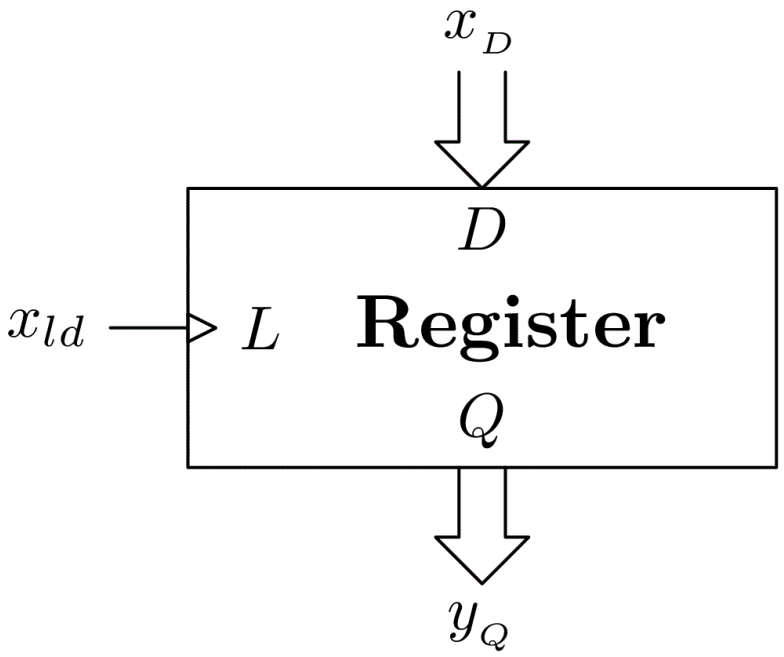

| Cause | Microoperation | Result |
|---|---|---|
| $x_{le} = 1 \bigwedge x_C = (0 \nearrow 1)$ | $Q \leftarrow x_D$ | $y_Q = Q$ |

# Static load




| Cause | Microoperation | Result |
|---|---|---|
| $x_l = 1$ | $Q \leftarrow x_D$ | $y_Q = Q = x_D$ |

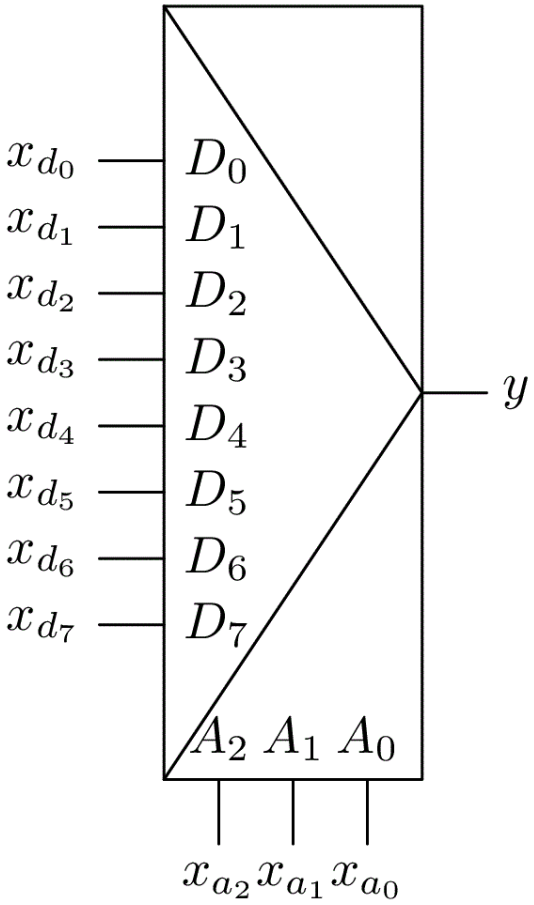
# Dynamic load




| Cause | Microoperation | Result |
|---|---|---|
| $x_{ld} = (0 \nearrow 1)$ | $Q \twoheadleftarrow x_D$ | $y_Q = Q$ |

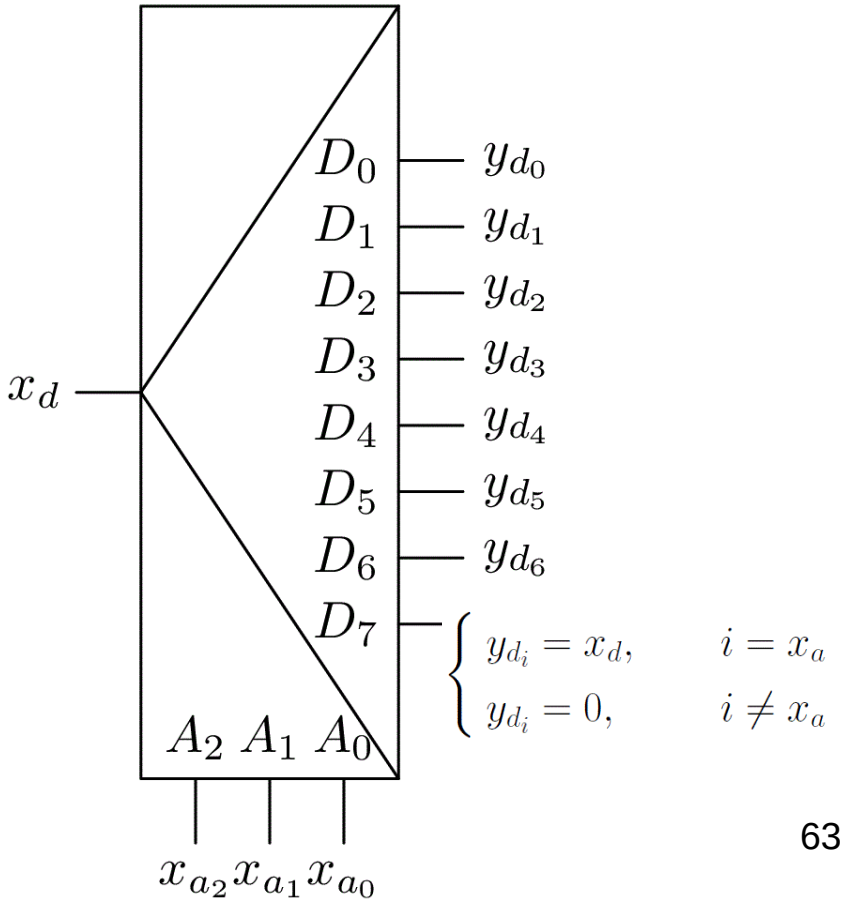
# Combinational functional blocks:

## Multiplexer



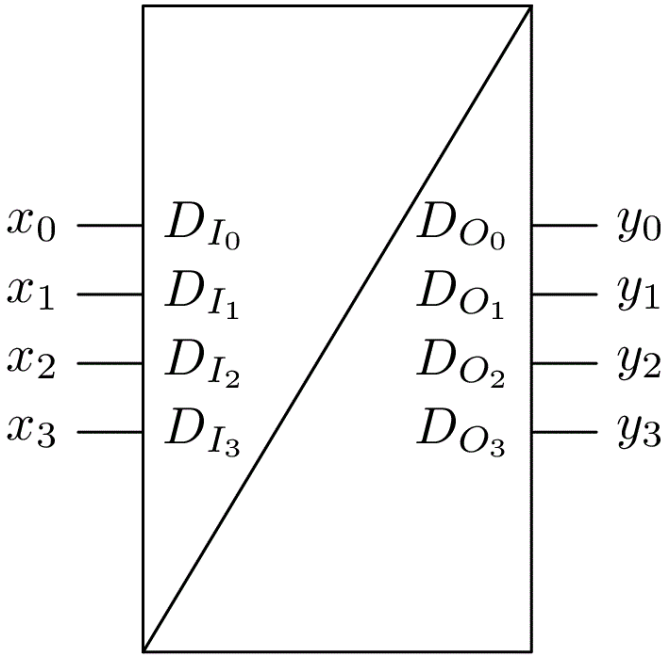

$$y = x_{d_i}, \quad i = x_a$$

## Demultiplexer



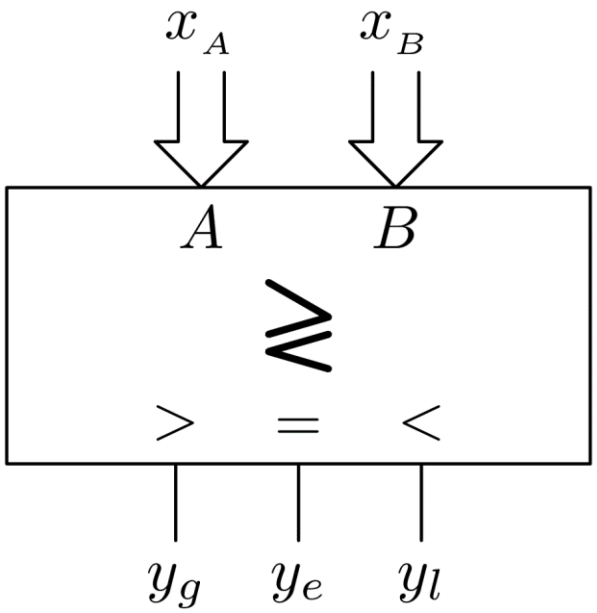

$$\begin{cases} y_{d_i} = x_d, & i = x_a \\ y_{d_i} = 0, & i \neq x_a \end{cases}$$

# Decoder



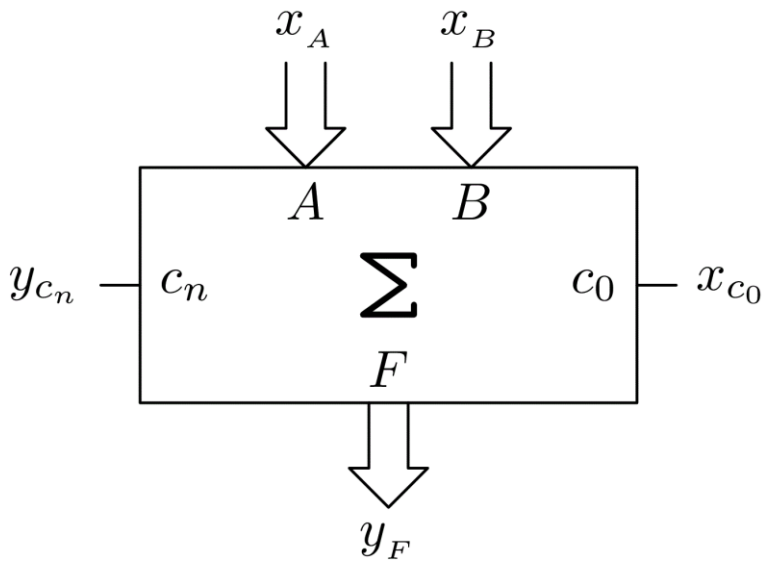

$$y = f(x)$$

# Comparator



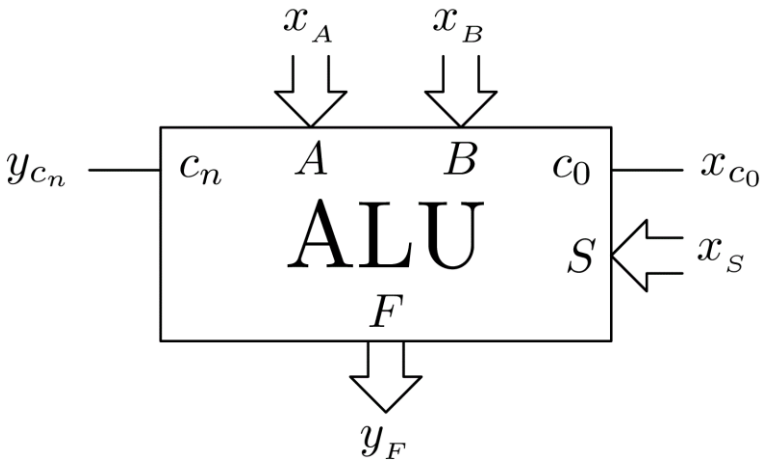

$$\begin{cases} y_g = 1 \Leftarrow x_A > x_B \\ y_e = 1 \Leftarrow x_A = x_B \\ y_l = 1 \Leftarrow x_A < x_B \end{cases}$$

# Adder



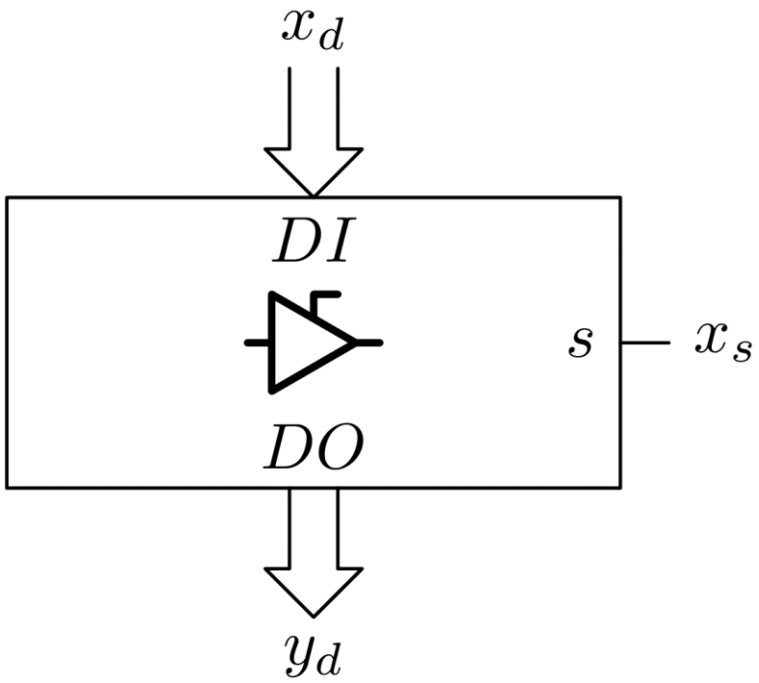

$$y_{c_n} \circ y_F = x_A + x_B + x_{c_0}$$

# Arithmetic Logic Unit




$$y_{c_n} \circ y_F = f(x_S, x_A, x_B, x_{c_0})$$

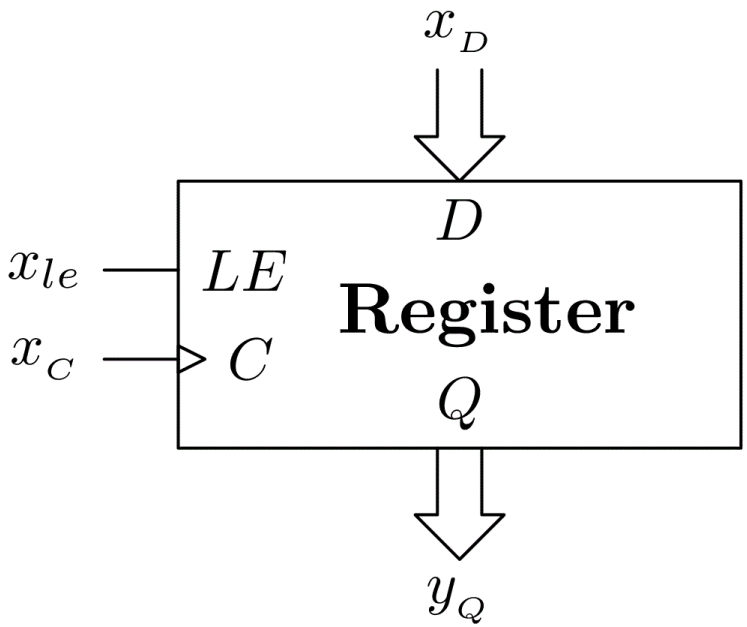
# Three state buffer




$$\begin{cases} y_{d_i} = x_{d_i} \Leftarrow x_s = 1 \\ y_{d_i} = \infty \Leftarrow x_s = 0 \end{cases}$$
$$i = 0, \ldots, n-1$$

# Basic register actions:

P - parallel S - serial I - in O - out




## PIPO

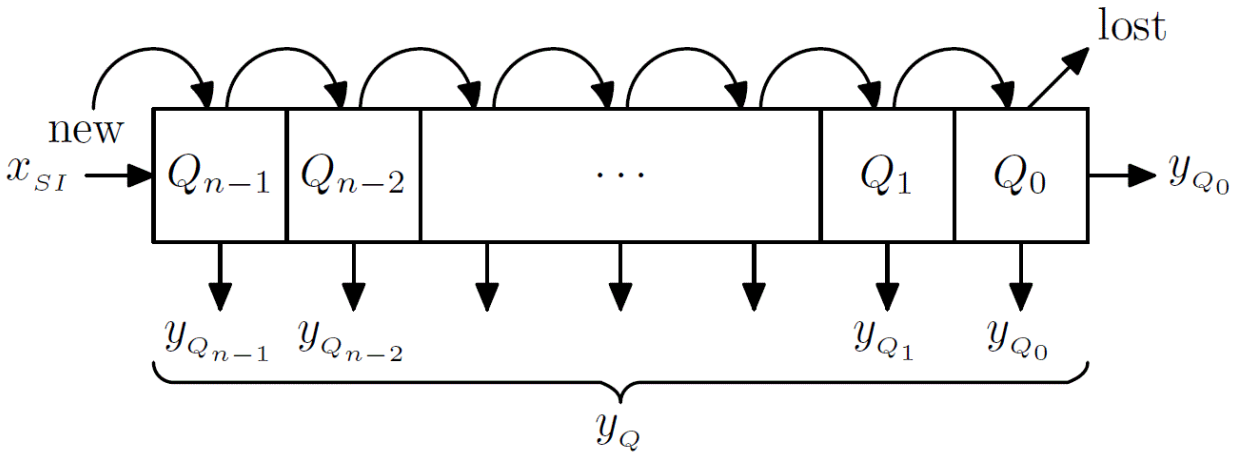
$$\begin{cases} Q \leftarrow x_D \\ y_Q = Q \end{cases}$$

$$x_{le} = 1 \bigwedge x_C = 0 \nearrow 1$$

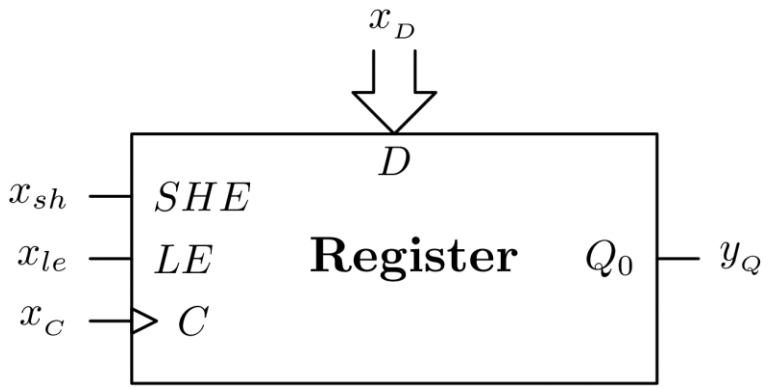
# **SISO** (shift right)


$$\begin{cases} \begin{cases} Q_{n-1} \leftarrow x_{SI} \\ Q_i \leftarrow Q_{i+1} \end{cases} \\ \quad i = 0, \ldots, n-2 \\ y_Q = Q_0 \end{cases} \qquad x_{sh} = 1 \ \bigwedge \ x_C = 0 \nearrow 1$$

## Shift register principle of operation

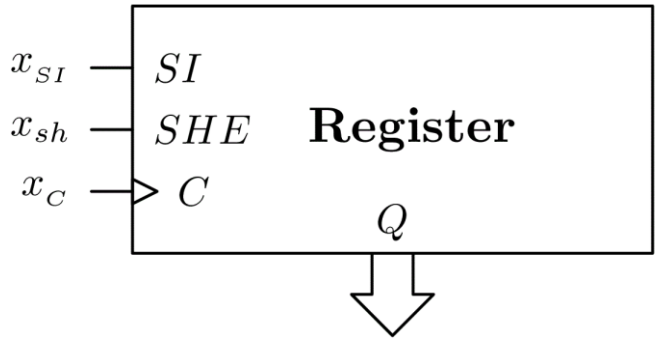


# PISO (shift right)



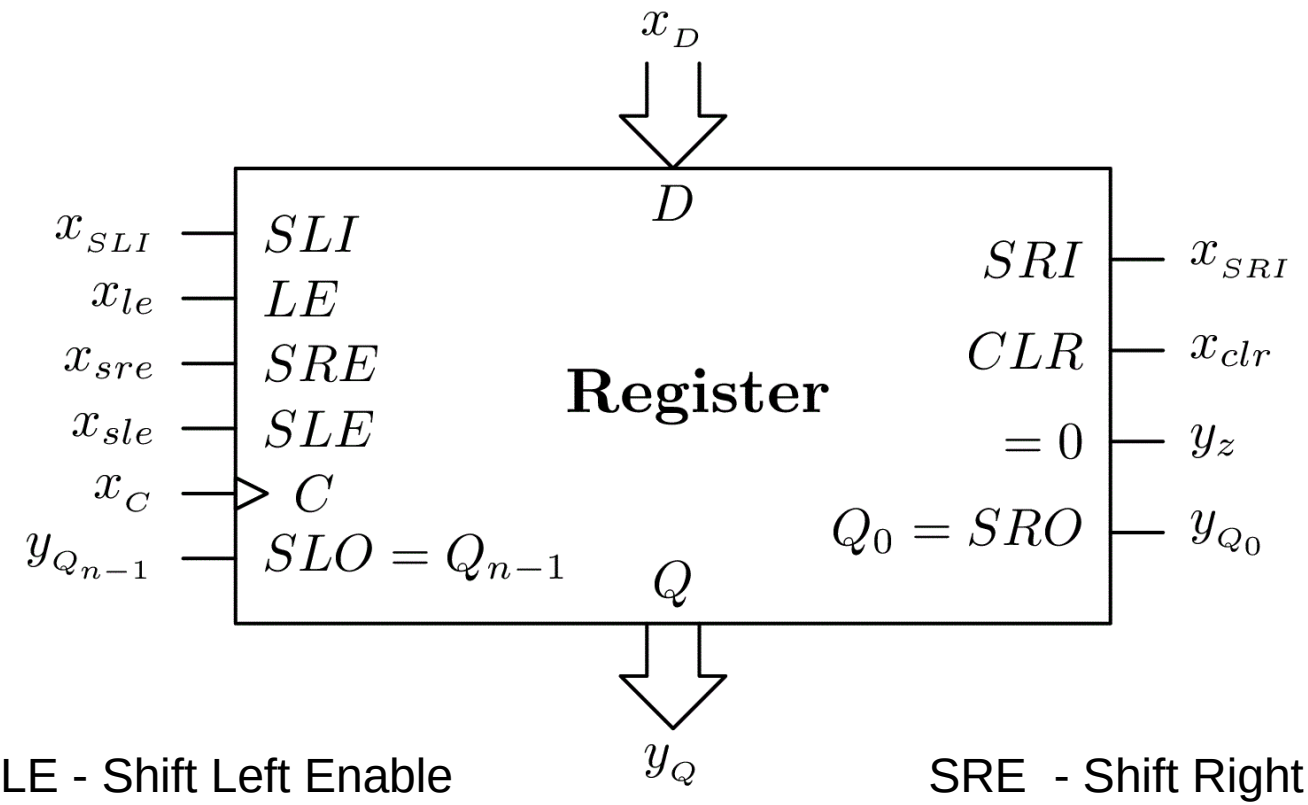

$$\begin{cases} \begin{cases} Q \leftarrow x_D \\ y_Q = Q_0 \end{cases} & x_{le} = 1 \bigwedge x_C = 0 \nearrow 1 \\ \begin{cases} \begin{cases} Q_{n-1} \leftarrow 0 \\ Q_i \leftarrow Q_{i+1} \end{cases} \\ \quad i = 0, \ldots, n-2 \\ y_Q = Q_0 \end{cases} & x_{sh} = 1 \bigwedge x_C = 0 \nearrow 1 \end{cases}$$

# SIPO (shift right)




$$\begin{cases} \begin{cases} Q_{n-1} \leftarrow x_{SI} \\ Q_i \leftarrow Q_{i+1} \end{cases} & x_{sh} = 1 \bigwedge x_C = 0 \nearrow 1 \\ \quad i = 0, \ldots, n-2 \\ y_Q = Q \end{cases}$$

# Universal reversible shift register

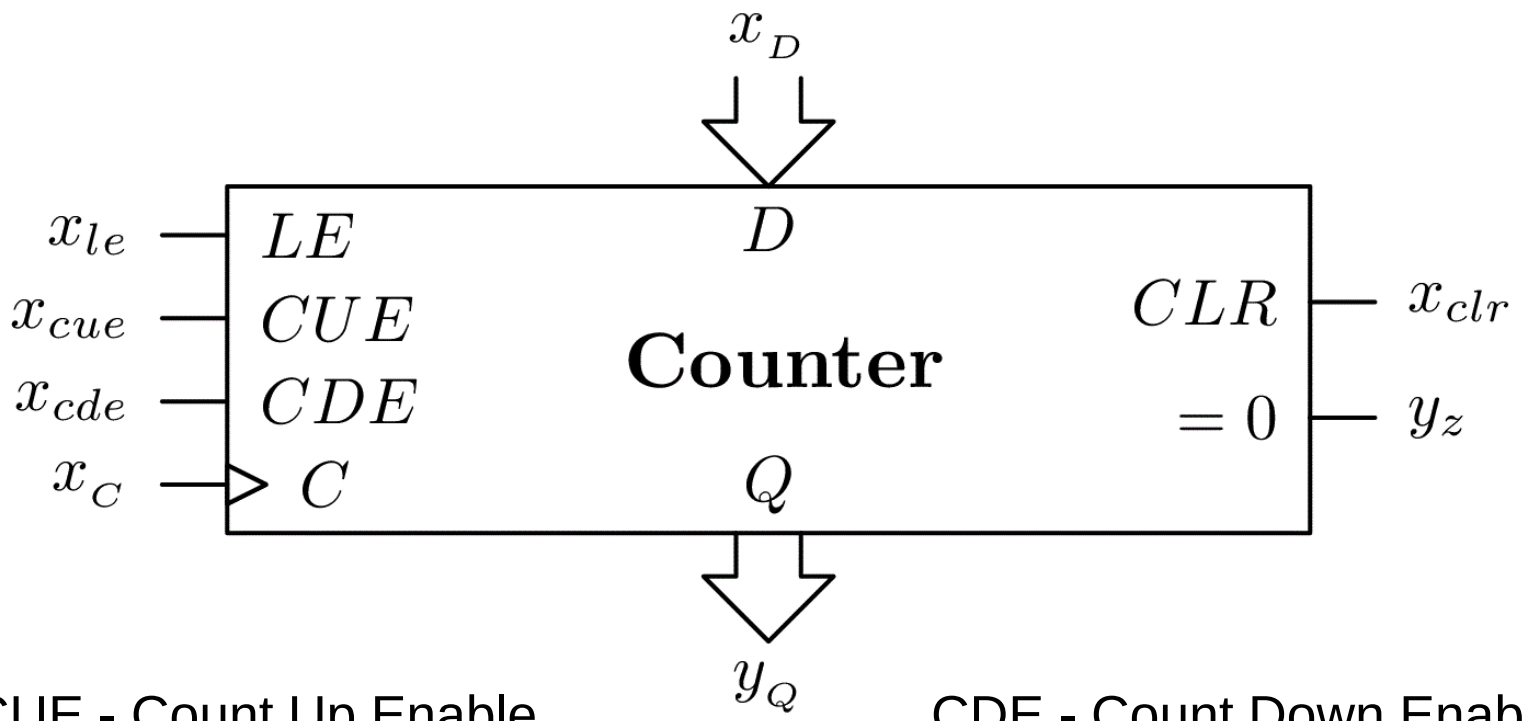



SLE - Shift Left Enable
SLI - Serial Left Input
SLO - Serial Left Output
LE - Load Enable

SRE - Shift Right Enable
SRI - Serial Right Input
SRO - Serial Right Output
CLR - Clear

$$
\begin{cases}
\begin{cases} Q_i \leftarrow 0 \\ \quad i = 0, \ldots, n-1 \end{cases} & x_{clr} = 1 \\
& \text{(other inputs irrelevant)} \\
\begin{cases} Q_i \leftarrow x_{D_i} \\ \quad i = 0, \ldots, n-1 \end{cases} & x_{le} = 1 \ \bigwedge \ x_C = 0 \nearrow 1 \\
& \text{(other inputs} = 0) \\
\begin{cases} Q_{n-1} \leftarrow x_{SRI} \\ Q_i \leftarrow Q_{i+1} \\ \quad i = 0, \ldots, n-2 \end{cases} & x_{sre} = 1 \ \bigwedge \ x_C = 0 \nearrow 1 \\
& \text{(other inputs} = 0) \\
\begin{cases} Q_0 \leftarrow x_{SLI} \\ Q_{i+1} \leftarrow Q_i \\ \quad i = 0, \ldots, n-2 \end{cases} & x_{sle} = 1 \ \bigwedge \ x_C = 0 \nearrow 1 \\
& \text{(other inputs} = 0) \\
y_Q = Q & \\
y_z = \begin{cases} 1 \Leftrightarrow \forall i \ \ Q_i = 0, & i = 0, \ldots, n-1 \\ 0 \Leftrightarrow \exists i \ \ Q_i \neq 0, & i = 0, \ldots, n-1 \end{cases} &
\end{cases}
$$

# Universal up-down counter




CUE - Count Up Enable

CDE - Count Down Enable

LE - Load Enable

CLR - Clear

mod$^m$ - binary $m = 2^n$, where $n$ - number of bits

decimal $m = 10$

$$
\begin{cases}
Q \leftarrowtail [0,\ldots,0] & \begin{cases} x_{clr} = 1 \\ \text{(other inputs irrelevant)} \end{cases} \\
Q \leftarrow x_D & \begin{cases} x_{le} = 1 \ \bigwedge \ x_C = 0 \nearrow 1 \\ \text{(other inputs} = 0) \end{cases} \\
Q \leftarrow (Q+1)_{mod_m} & \begin{cases} x_{cue} = 1 \ \bigwedge \ x_C = 0 \nearrow 1 \\ \text{(other inputs} = 0) \end{cases} \\
Q \leftarrow (Q-1)_{mod_m} & \begin{cases} x_{cde} = 1 \ \bigwedge \ x_C = 0 \nearrow 1 \\ \text{(other inputs} = 0) \end{cases} \\
y_Q = Q & \\
y_z = \begin{cases} 1 \Leftrightarrow \forall i \ \ Q_i = 0, \quad i = 0,\ldots,n-1 \\ 0 \Leftrightarrow \exists i \ \ Q_i \neq 0, \quad i = 0,\ldots,n-1 \end{cases} &
\end{cases}
$$

# Digital control and data processing systems

**Example:** Multiplication of positive binary numbers

$$y_R = x_A * x_B$$

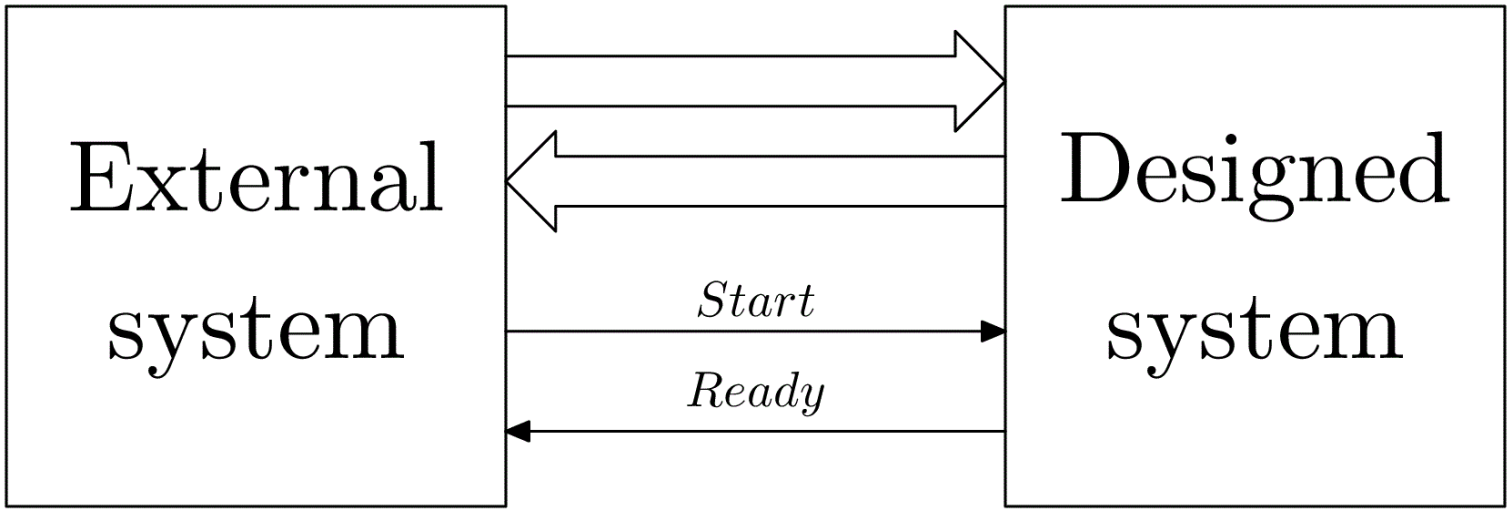
e.g.: $6_{10} * 5_{10} = 30_{10} \iff 110_2 * 101_2 = 11110_2$

## Algorithm — a very simple one:

$$\left.\begin{array}{r} 6 \\ +6 \\ +6 \\ +6 \\ +6 \end{array}\right\} \times 5$$
$$30$$

$$\begin{array}{rcl} 0 & & \\ +6 & \longrightarrow & 1^{st} \\ \hline 6 & & \\ +6 & \longrightarrow & 2^{nd} \\ \hline 12 & & \\ +6 & \longrightarrow & 3^{rd} \\ \hline 18 & & \\ +6 & \longrightarrow & 4^{th} \\ \hline 24 & & \\ +6 & \longrightarrow & 5^{th} \\ \hline 30 & & \end{array}$$

# Communication between systems

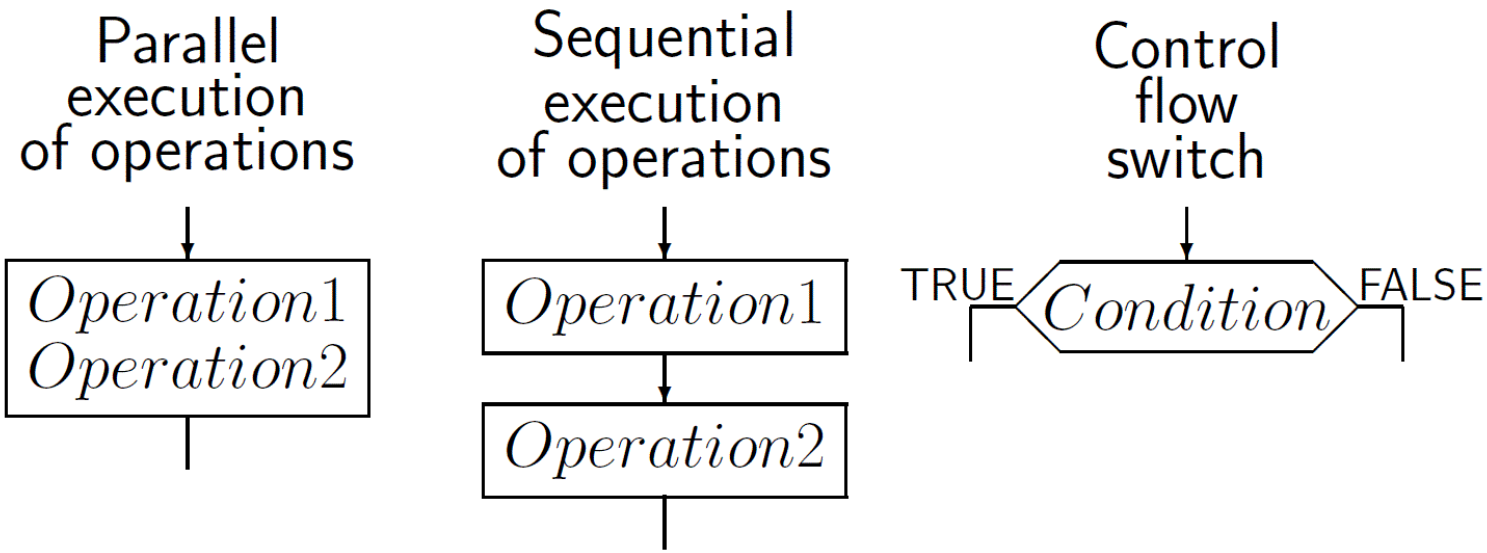



Handshaking is implemented by two signals:

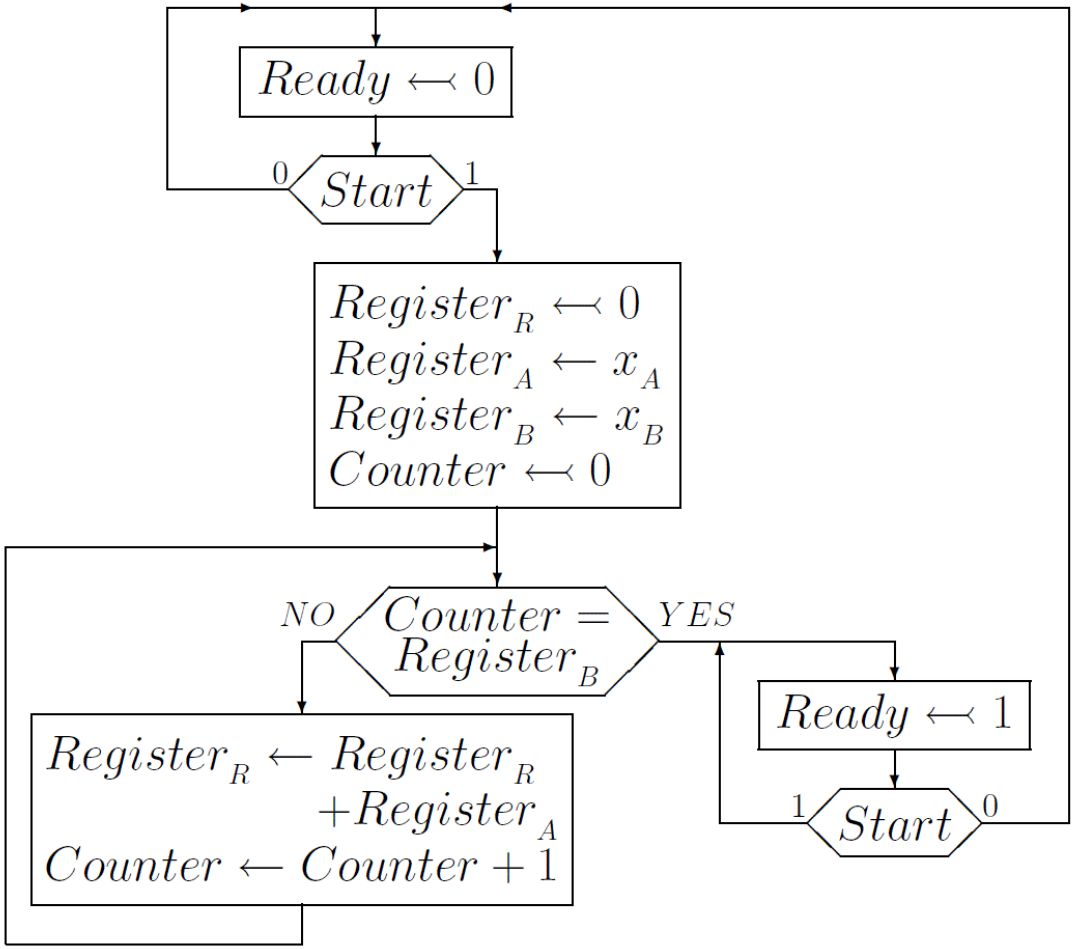
*Start* - external control signal
*Ready* - internal reply signal

| | Initial state | Data ready | Data processed | Result ready | Result withdrawn | Final state |
|---|---|---|---|---|---|---|
| *Start* | 0 | 1 | 1 | 1 | 0 | 0 |
| *Ready* | 0 | 0 | 0 | 1 | 1 | 0 |

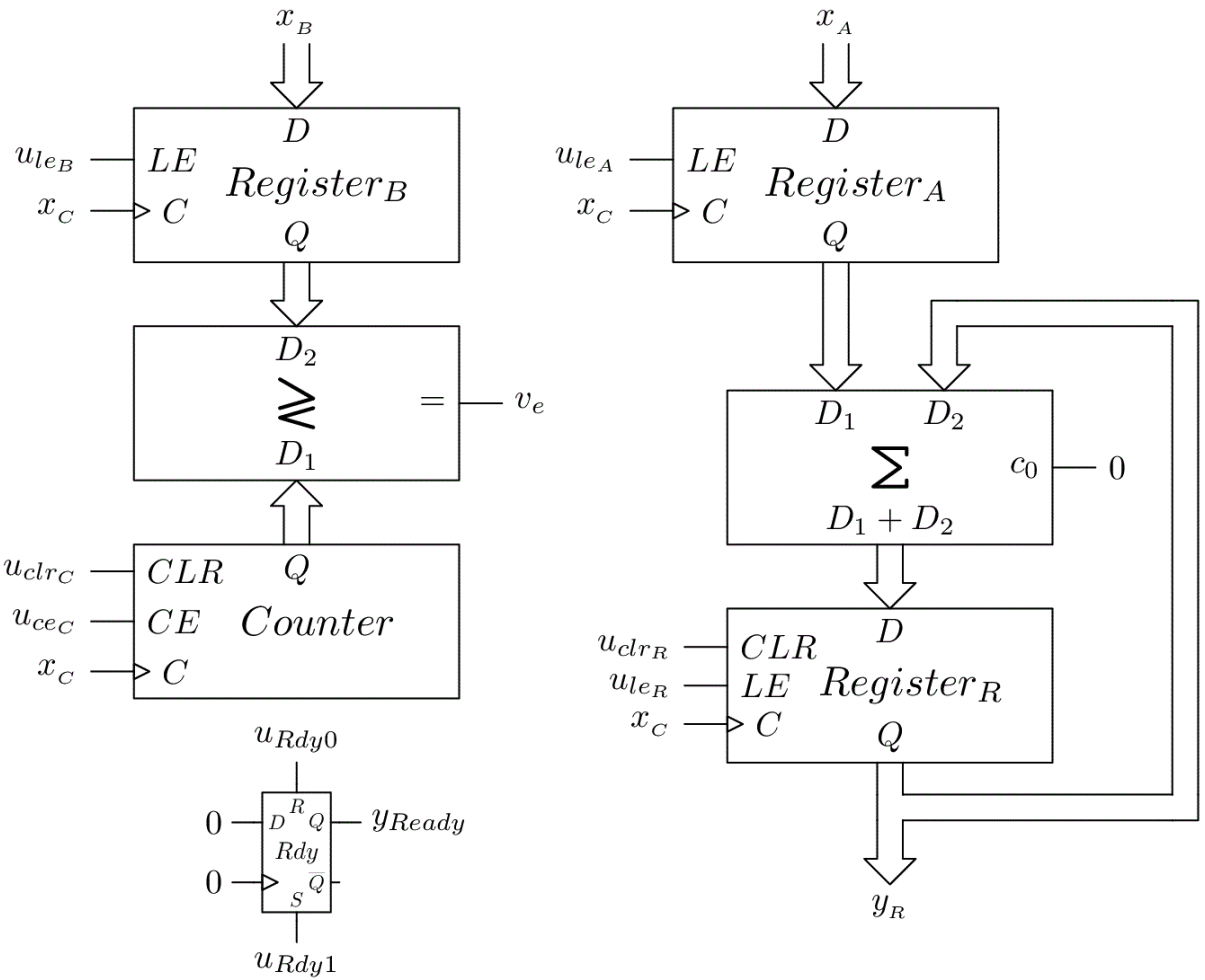
# Flow chart blocks



# Flow chart of multiplication of positive binary numbers



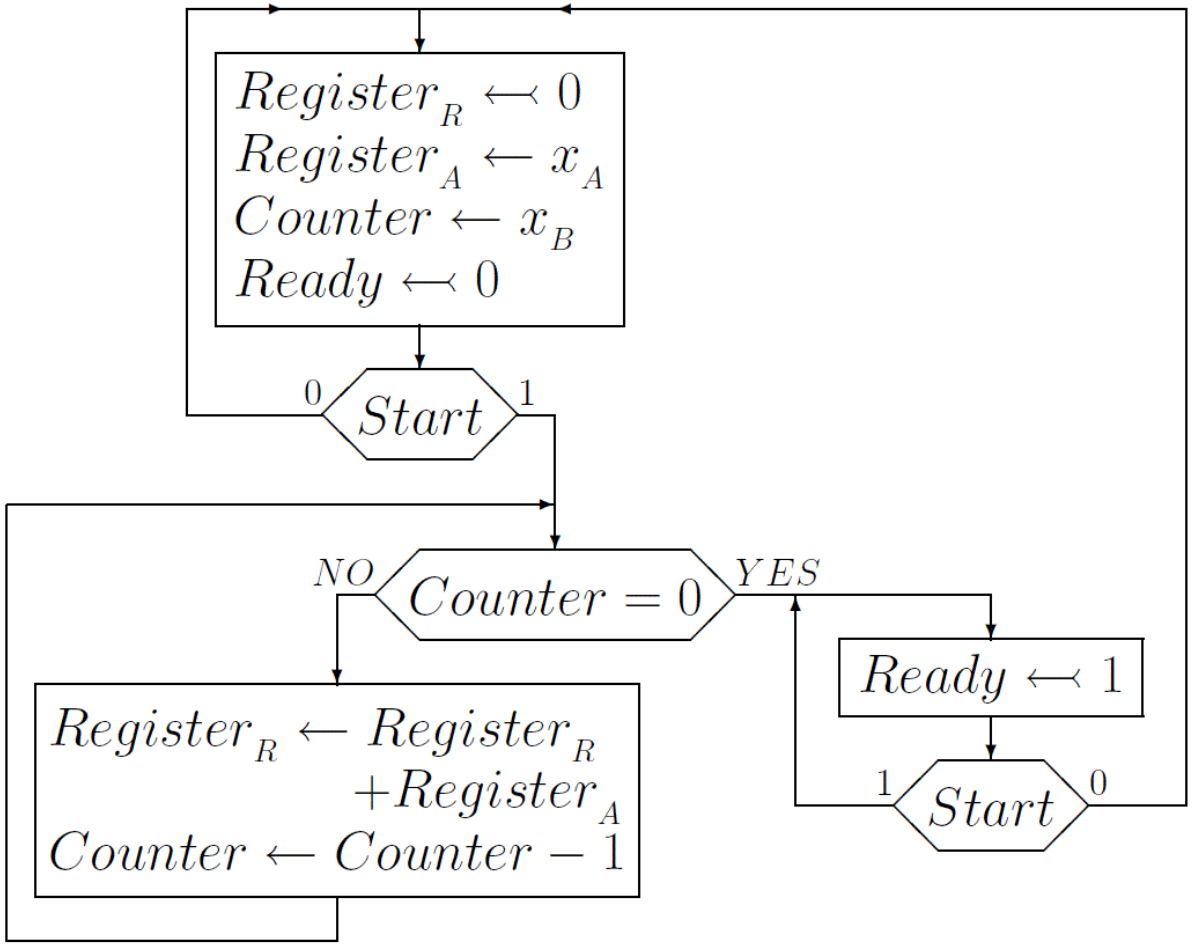
# Handshaking flip-flop and data path subsystem for the multiplication of positive binary numbers - initial version



# Necessary resources:

- 3 registers for storing $x_A$, $x_B$ and the result counter
- comparator
- adder
- flip-flop implementing handshaking (*Ready*)

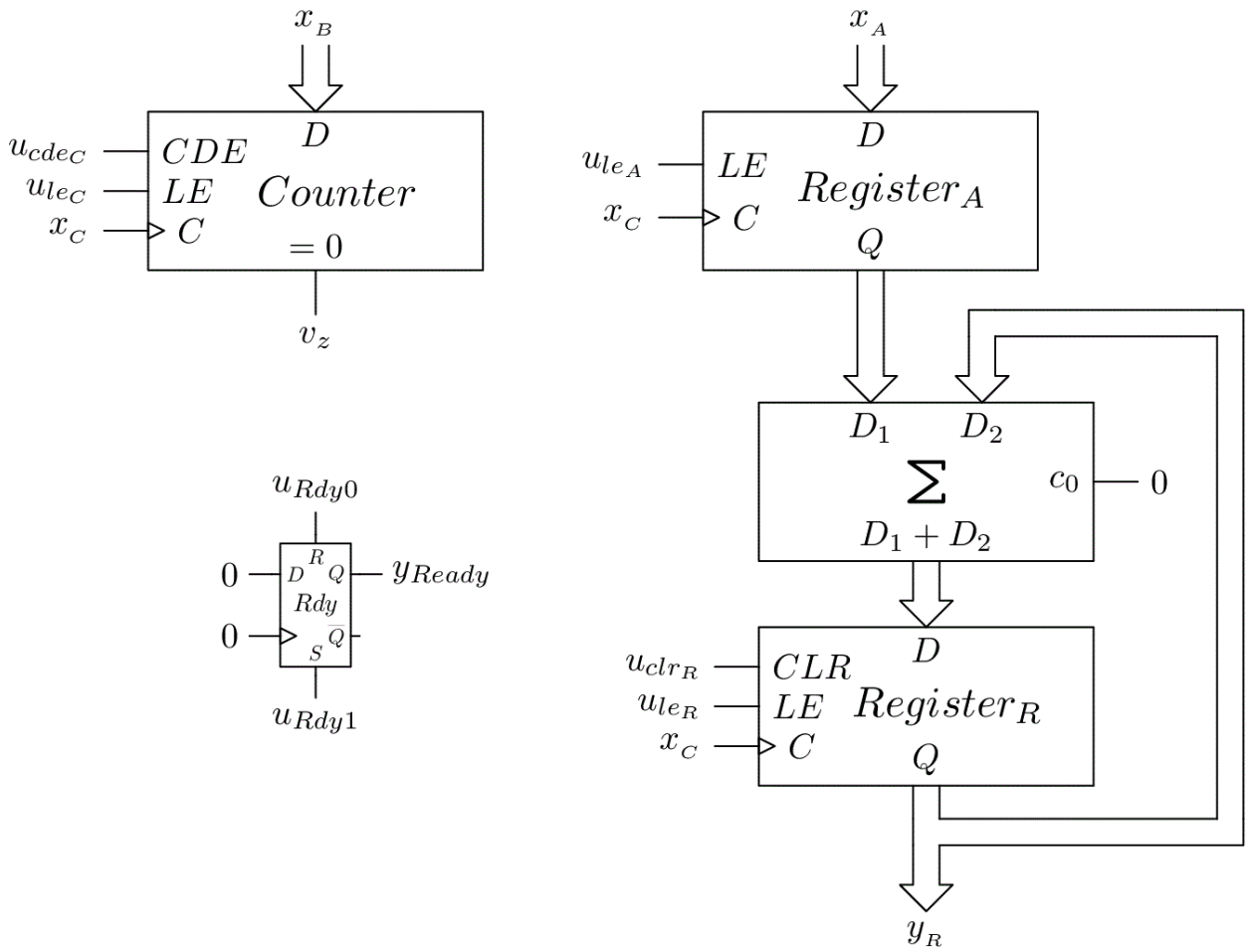
# Modified flow chart of multiplication of positive binary numbers




Drawback:

Execution time is proportional to $x_B$

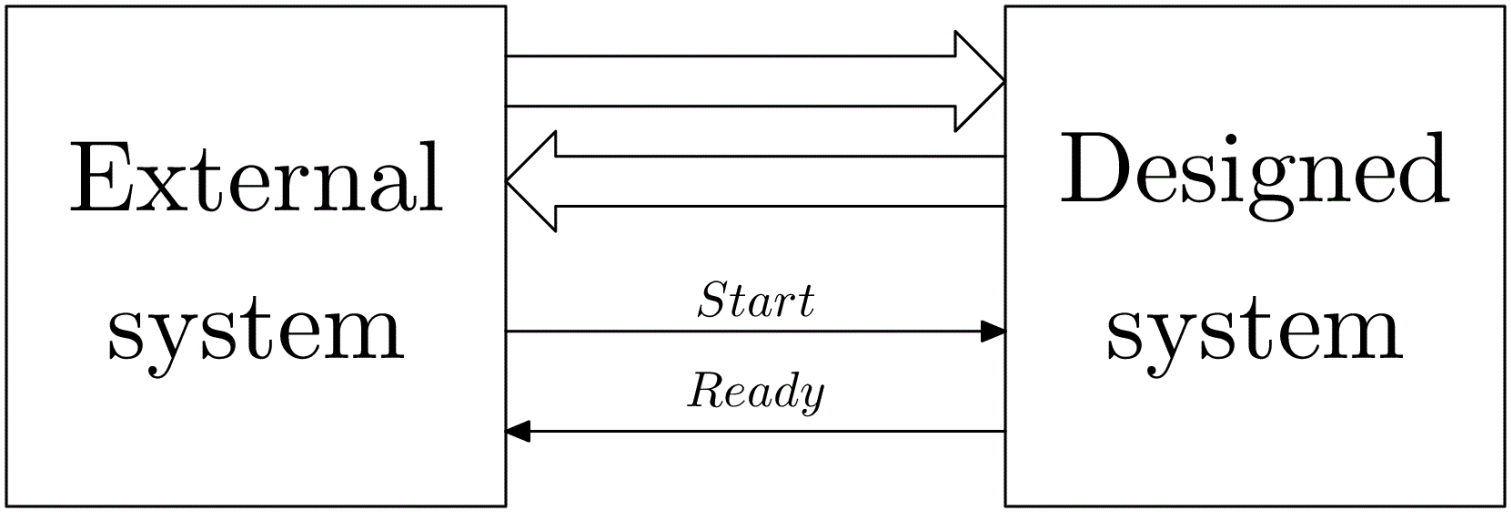
# Handshaking flip-flop and data path subsystem for the multiplication of positive binary numbers - modified version



# Necessary resources:

- 2 registers for storing $x_A$ and the result
- counter
- adder
- flip-flop implementing handshaking (*Ready*)

# Communication between systems




Handshaking is implemented by two signals:

*Start* - external control signal
*Ready* - internal reply signal

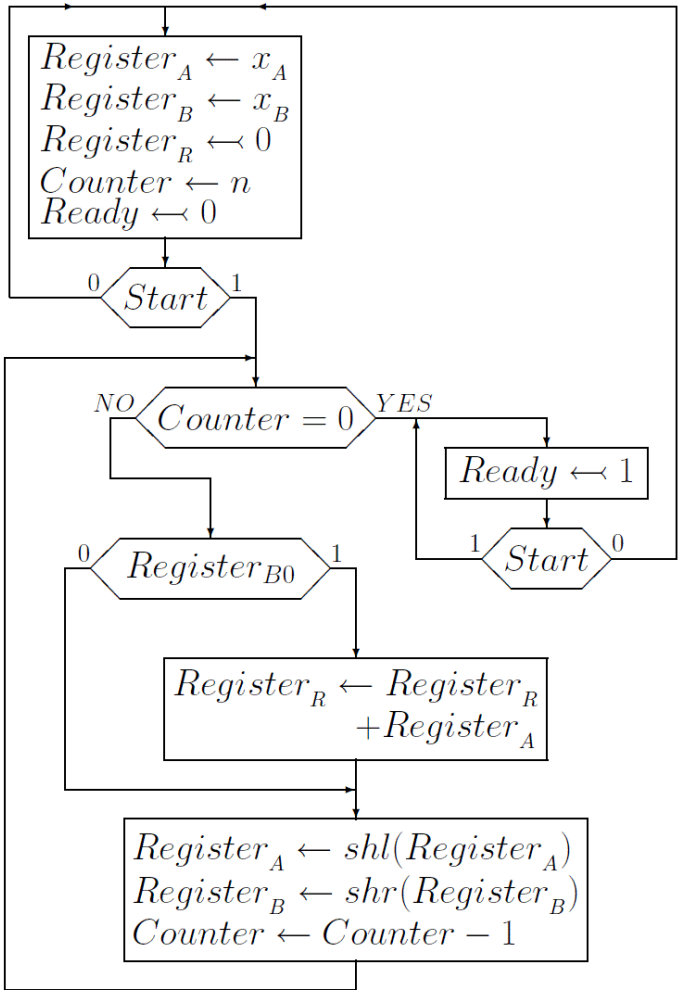
| | Initial state | Data ready | Data processed | Result ready | Result withdrawn | Final state |
|---|---|---|---|---|---|---|
| *Start* | 0 | 1 | 1 | 1 | 0 | 0 |
| *Ready* | 0 | 0 | 0 | 1 | 1 | 0 |

# Improved algorithm of positive binary number multiplication

$$
\begin{array}{rrrrl}
 & 0 & 0 & 0 & \rightarrow \text{ result initially} \\
+ & 1 & 1 & 0 & \rightarrow shl^0(x_A) * x_{B0} \Rightarrow \quad 110 * \boxed{1} \\
\hline
 & 1 & 1 & 0 & \rightarrow \text{ result after } 1^{st} \text{ operation} \\
+ & 0 & 0 & 0 & \rightarrow shl^1(x_A) * x_{B1} \Rightarrow \quad 1100 * \boxed{0} \\
\hline
 & 0 & 1 & 1 & 0 \quad \rightarrow \text{ result after } 2^{nd} \text{ operation} \\
+ & 1 & 1 & 0 & \rightarrow shl^2(x_A) * x_{B2} \Rightarrow 11000 * \boxed{1} \\
\hline
= & 1 & 1 & 1 & 1 \quad 0 \quad \rightarrow \text{ result after } 3^{rd} \text{ operation}
\end{array}
$$

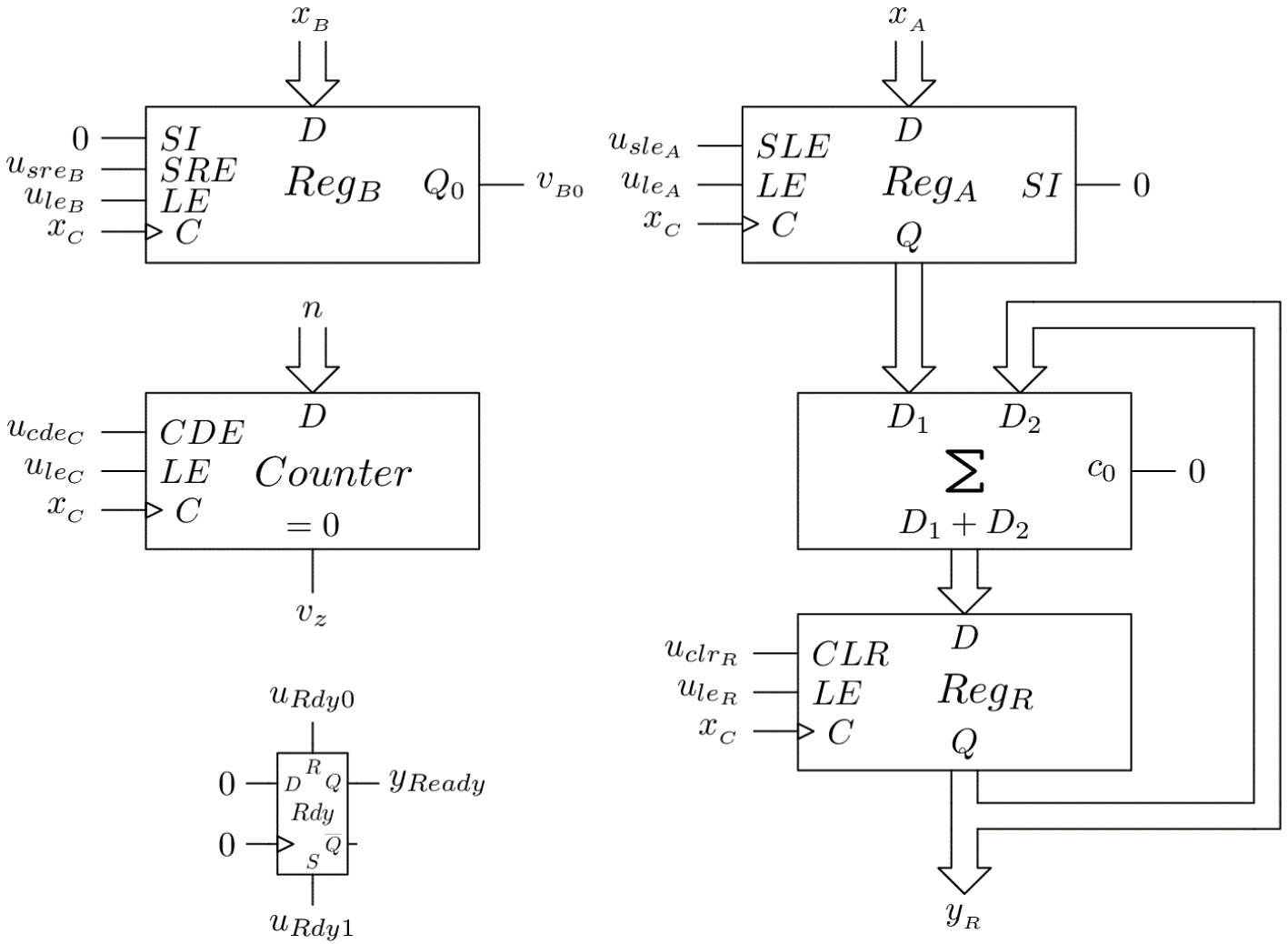
$$
\begin{array}{rrrrrl}
 & & 1 & 1 & 0 & \rightarrow \quad 6_{10} \\
 & * & 1 & 0 & 1 & \rightarrow \quad 5_{10} \\
\hline
 & & 1 & 1 & 0 & \\
 & 0 & 0 & 0 & & \\
1 & 1 & 0 & & & \\
\hline
1 & 1 & 1 & 1 & 0 & \rightarrow \quad 30_{10}
\end{array}
$$

# Improved flow chart of positive binary number multiplication




- Execution time is proportional to *n* regardless of the value of $x_B$
- If $x_B$ is small, e.g. 0, the multiplication takes unnecessarily long time

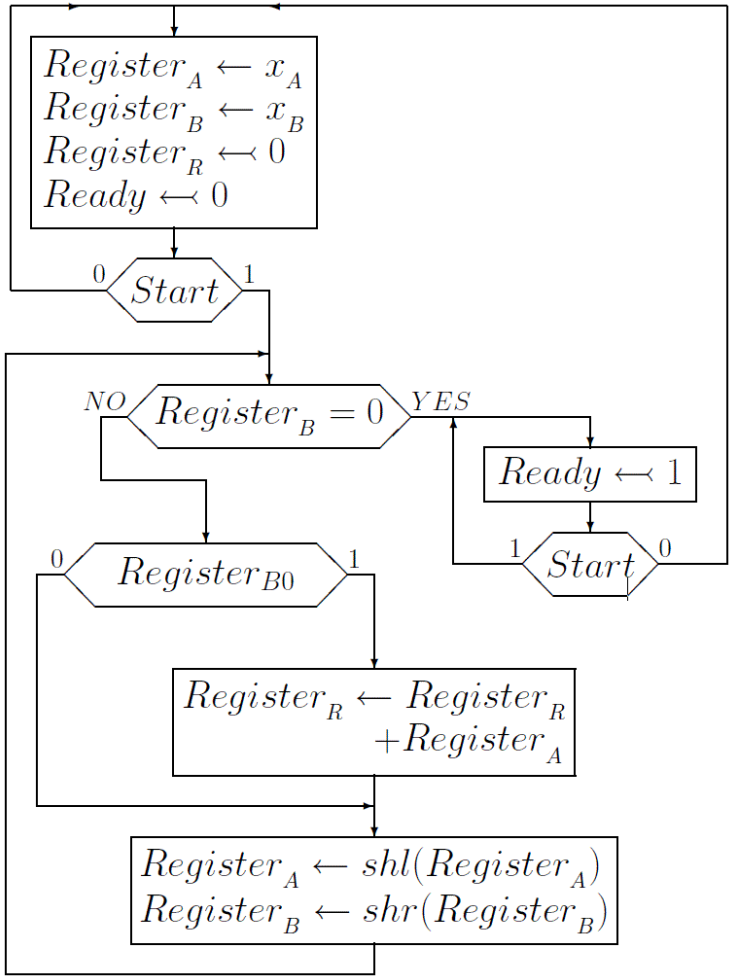
# Handshaking flip-flop and data path subsystem for the multiplication of positive binary numbers - improved version



# Necessary resources:

- 2 shift registers for storing and shifting $x_A$ and $x_B$ register for storing the result
- counter
- adder
- flip-flop implementing handshaking (*Ready*)

# Final flow chart of positive binary number multiplication




$n$ – number of bits in the multiplied numbers
$\Rightarrow$ the result has $2n$ bits at the most

Example:

$$\begin{array}{rrrrrrl} & & 1 & 1 & 1 & \rightarrow & 7_{10} \\ & * & 1 & 1 & 1 & \rightarrow & 7_{10} \\ \hline & & 1 & 1 & 1 & & \\ & 1 & 1 & 1 & & & \\ 1 & 1 & 1 & & & & \\ \hline 1 & 1 & 0 & 0 & 0 & 1 & \rightarrow 49_{10} \end{array}$$

Proof:

$n$ – number of bits of $x_A$ and $x_B$

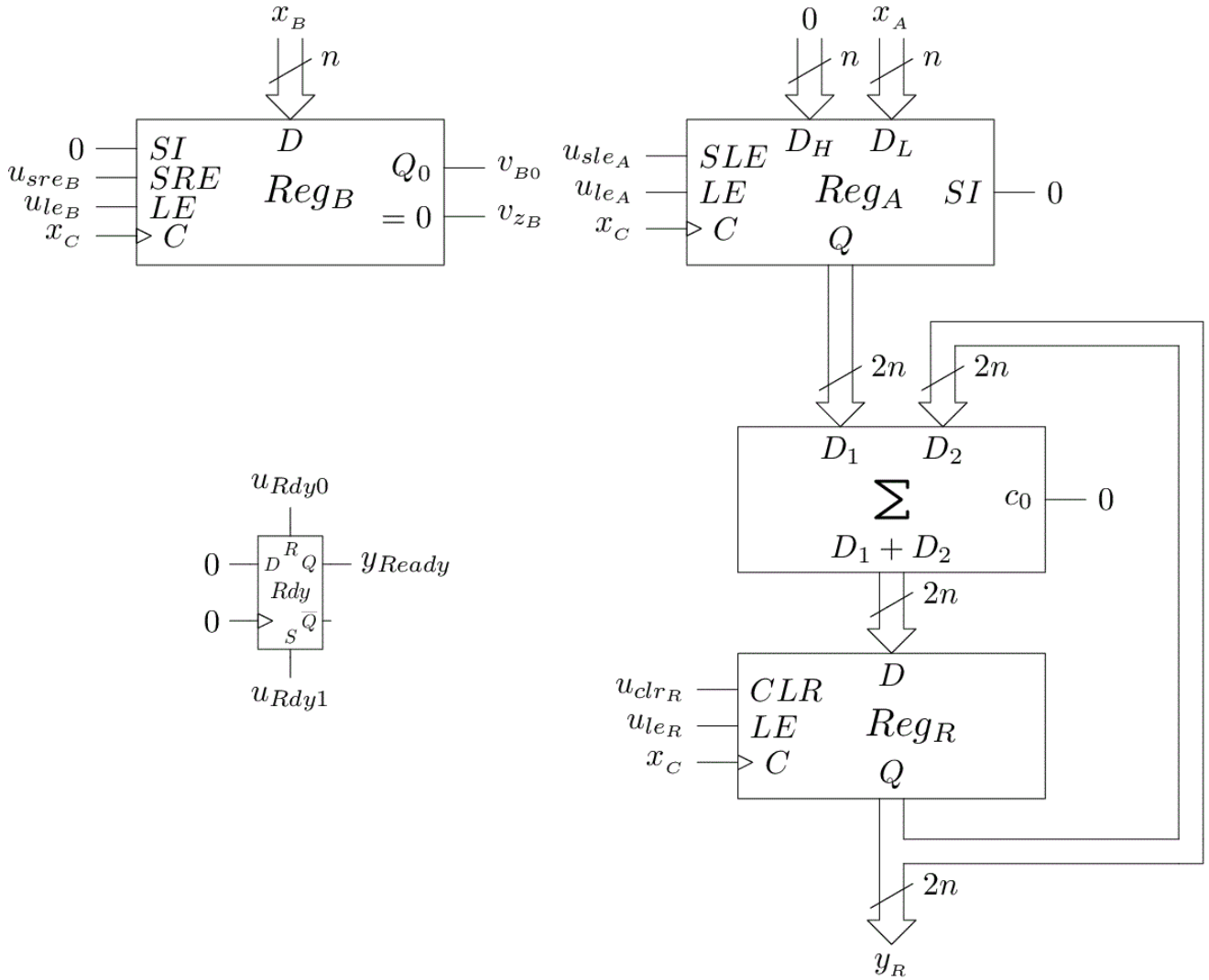
$x_A < 2^n$ and $x_B < 2^n \Rightarrow$

$$x_A * x_B < 2^n * 2^n = 2^{n+n} = 2^{2n} = 1\underbrace{0\ldots0}_{\times 2n}$$

$$\Rightarrow max(x_A * x_B) \leqslant 2^{2n} - 1$$

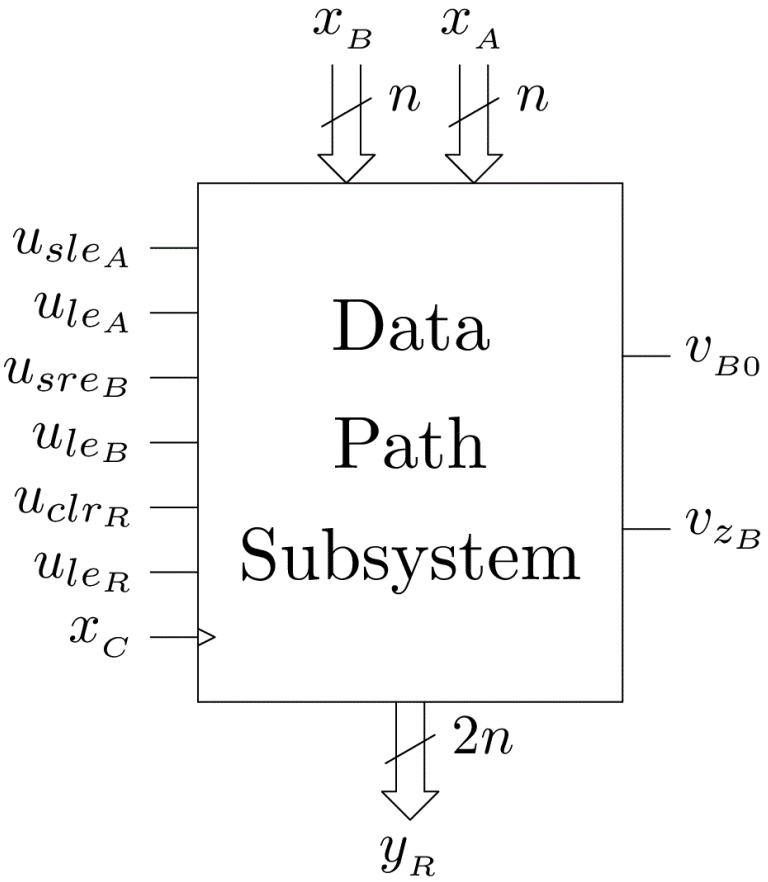
$$\Rightarrow max(x_A * x_B) \leqslant \underbrace{1\ldots1}_{\times 2n}$$

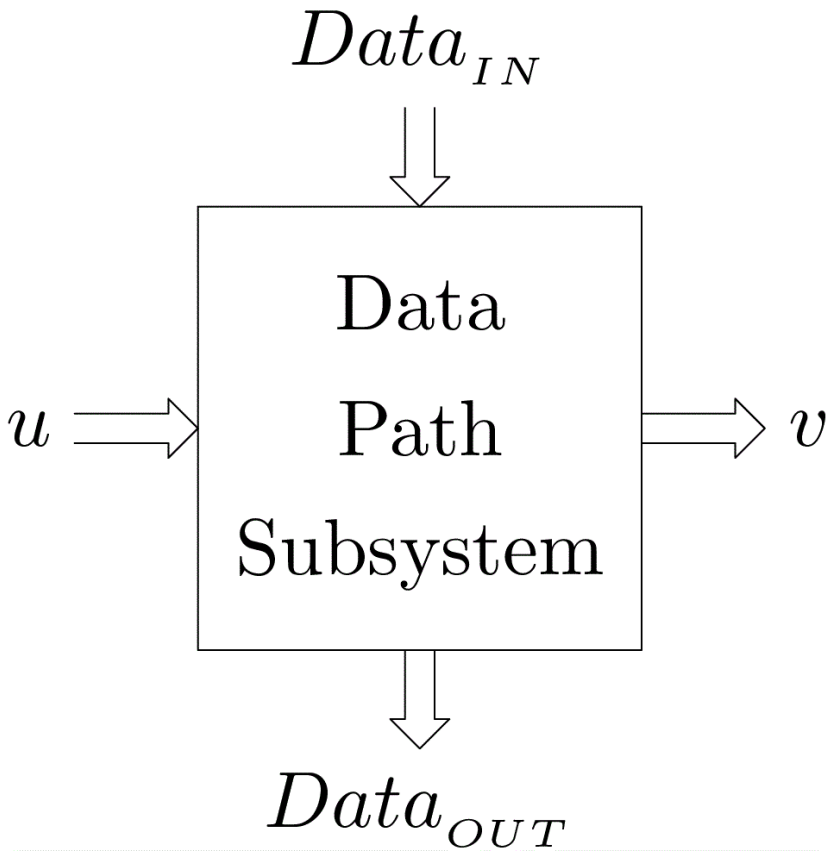
# Handshaking flip-flop and data path subsystem for the multiplication of positive binary numbers - final version

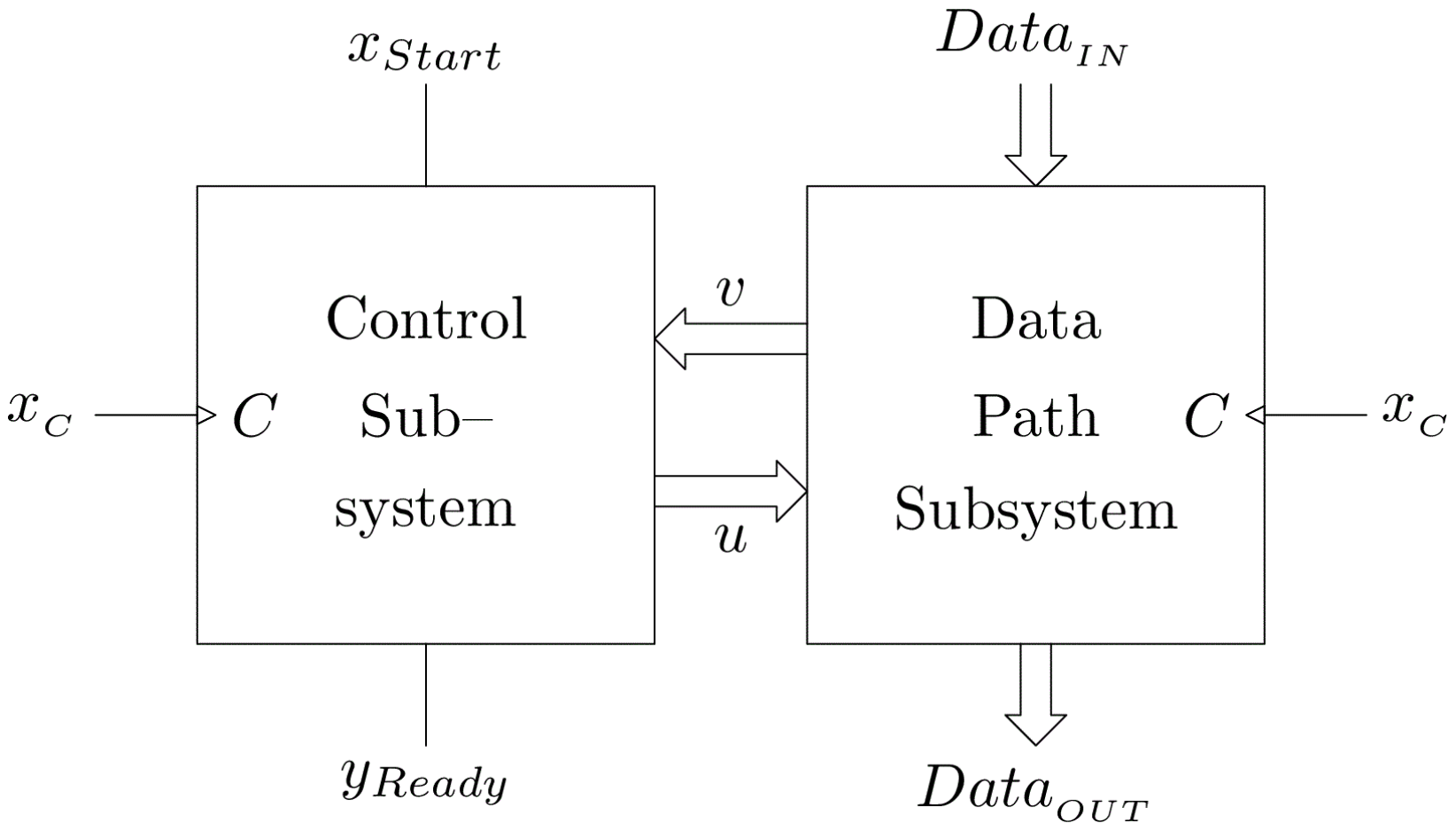



Necessary resources:

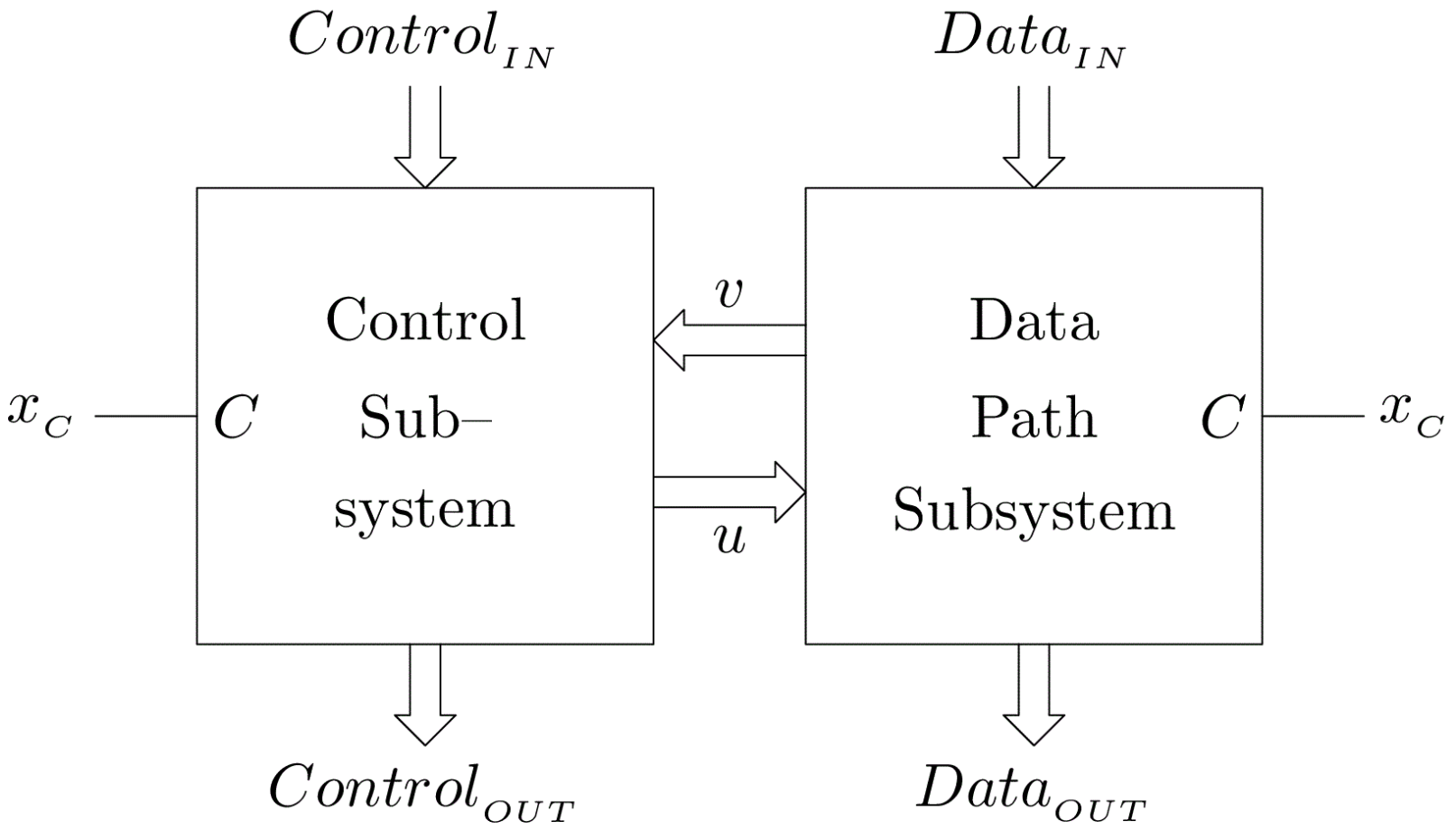
- 2 shift registers for storing and shifting $x_A$ and $x_B$
- register for storing the result
- adder
- flip-flop implementing handshaking (*Ready*)


$x_B$
$x_A$
$n$
$n$
$u_{sle_A}$
$u_{le_A}$
$u_{sre_B}$
$u_{le_B}$
$u_{clr_R}$
$u_{le_R}$
$x_C$
Data
Path
Subsystem
$v_{B0}$
$v_{z_B}$
$2n$
$y_R$

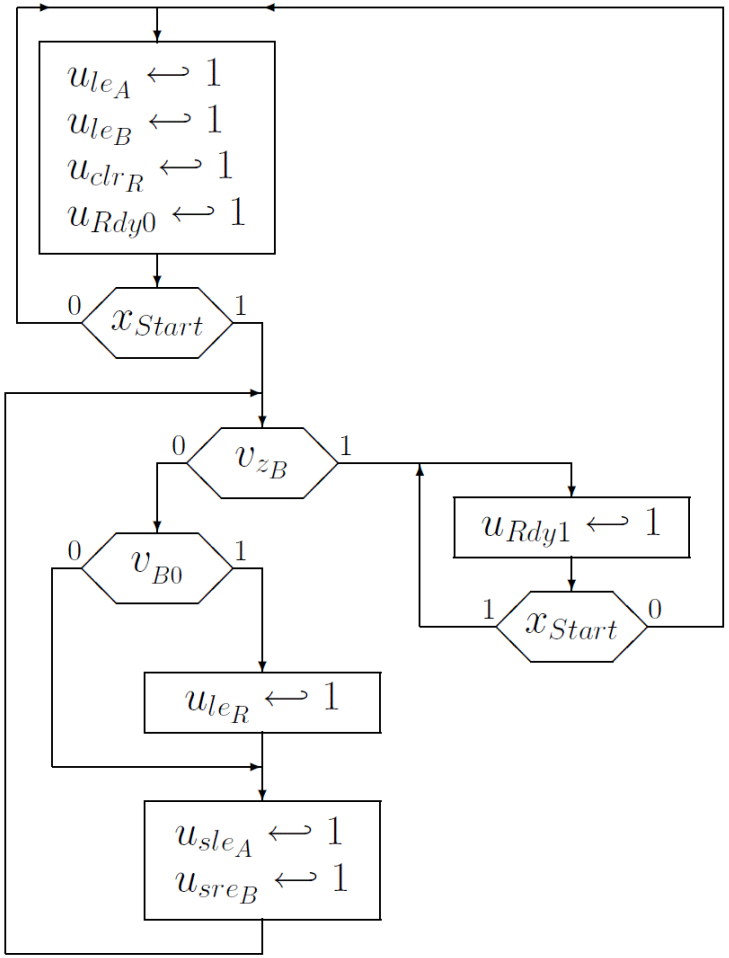

$Data_{IN}$
$u$
Data
Path
Subsystem
$v$
$Data_{OUT}$


$x_{Start}$
$Data_{IN}$
Control
Sub–
system
Data
Path
Subsystem
$v$
$u$
$x_C$
$C$
$C$
$x_C$
$y_{Ready}$
$Data_{OUT}$

# General structure of a digital control and data processing system



# Signal assignment flow chart for positive binary number multiplication system




For the purpose of brevity let:

$$x_{Start} \longrightarrow x_s$$
$$v_{z_B} \longrightarrow v_z$$
$$v_{B0} \longrightarrow v_0$$

# Algorithms transforming flow charts into state transition graphs

## For a Moore automaton

1.Place the graph nodes (states) prior to operational blocks

2.If there still exists a loop in the flow chart which does not contain any graph nodes (states) then introduce into that loop a graph node (state) prior to the conditional block within that loop

## For a Mealy automaton

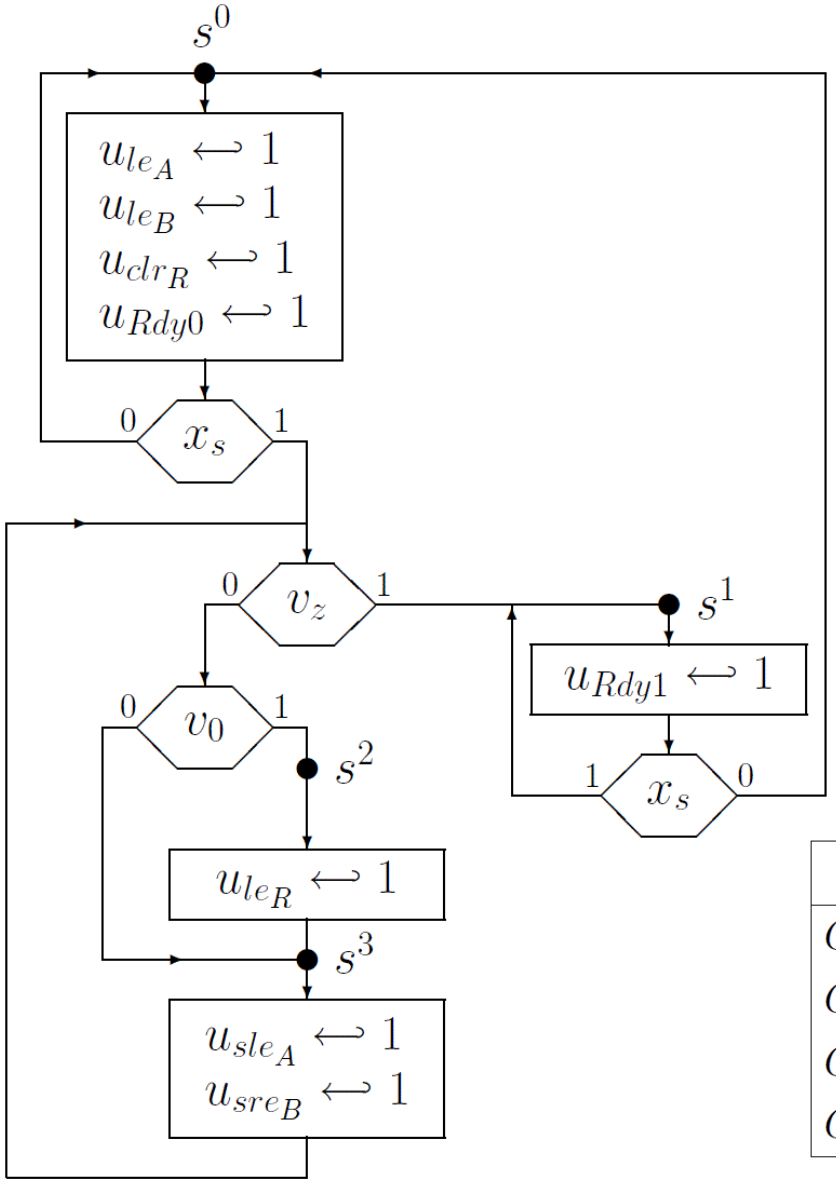
1.Place one graph node (state) prior to a conditional block within each loop of the flow chart

2.If there exists a sequence of operational blocks without conditional blocks in-between separate the operational blocks by introducing extra graph nodes (states)

# Denitions:

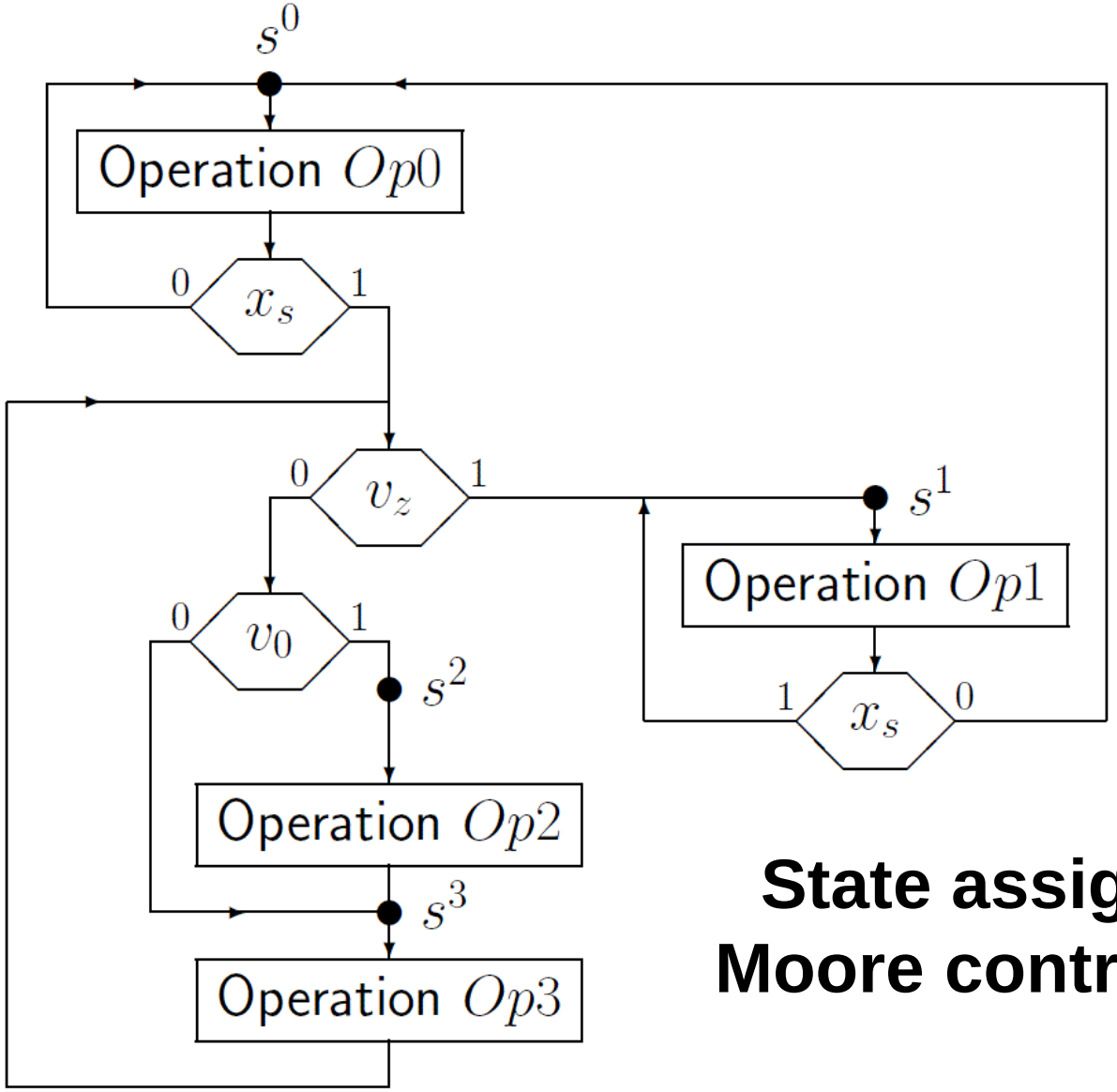
**Microoperation** - an action performed by a single functional block in one clock cycle

**Operation** - a set of microoperations concurrently executed in one clock cycle

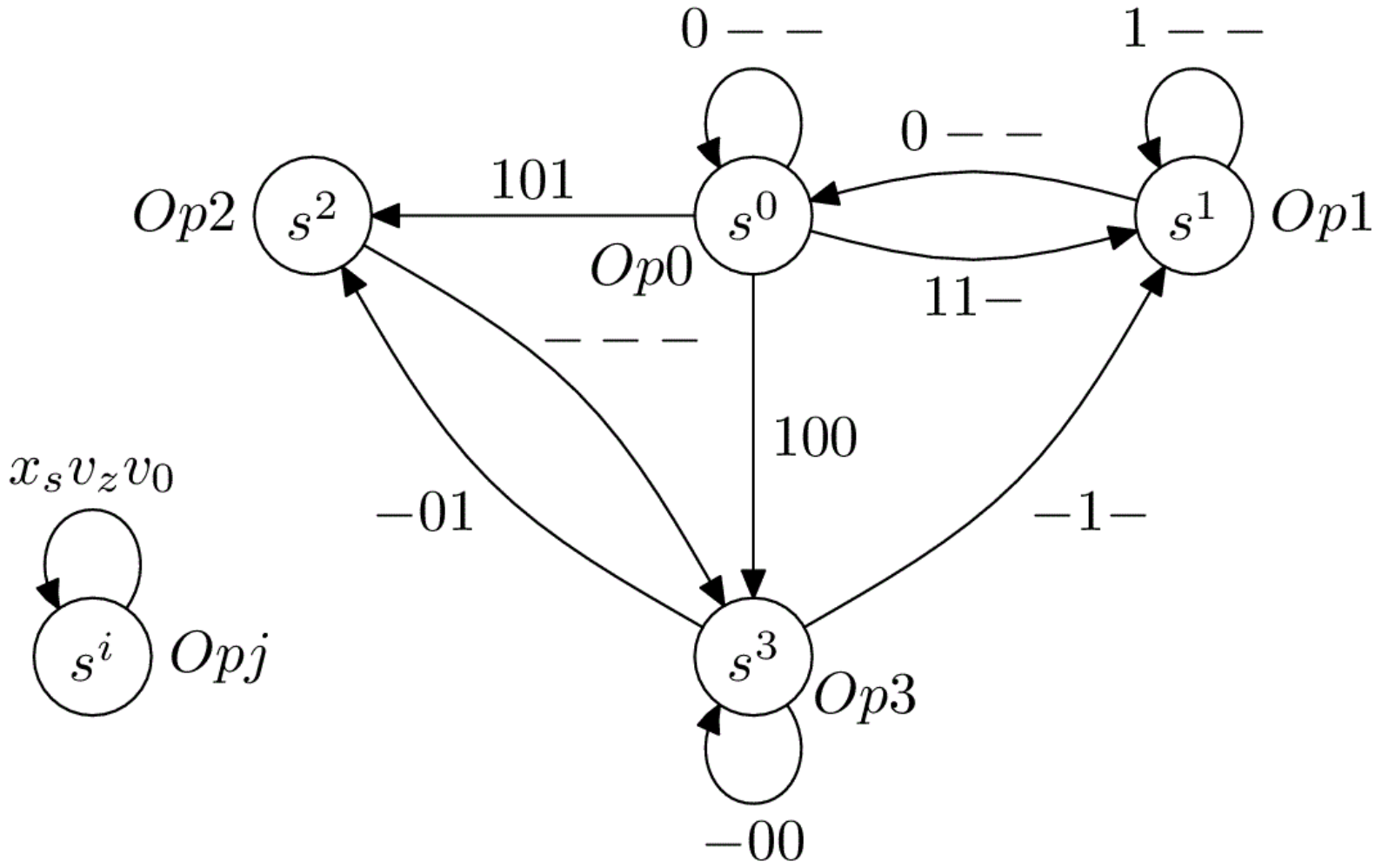



# State assignment for a Moore control automaton

| | $u_{le_A}$ | $u_{le_B}$ | $u_{clr_R}$ | $u_{Rdy0}$ | $u_{Rdy1}$ | $u_{le_R}$ | $u_{sle_A}$ | $u_{sre_B}$ |
|---|---|---|---|---|---|---|---|---|
| $Op0$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| $Op1$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $Op2$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $Op3$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |




# State assignment for a Moore control automaton


0 – –
1 – –
0 – –
101
Op2
$s^2$
$s^0$
$s^1$
Op1
Op0
11–
– – –
100
$x_s v_z v_0$
–01
–1–
$s^i$
Opj
$s^3$
Op3
–00

| $x_s v_z v_0$ / $s$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 | $Op$ |
|---|---|---|---|---|---|---|---|---|---|
| $s^0$ | $s^0$ | $s^0$ | – | $s^0$ | $s^1$ | – | $s^2$ | $s^3$ | $Op0$ |
| $s^1$ | $s^0$ | $s^0$ | – | $s^0$ | $s^1$ | – | $s^1$ | $s^1$ | $Op1$ |
| $s^2$ | $s^3$ | $s^3$ | – | $s^3$ | $s^3$ | – | $s^3$ | $s^3$ | $Op2$ |
| $s^3$ | $s^3$ | $s^2$ | – | $s^1$ | $s^1$ | – | $s^2$ | $s^3$ | $Op3$ |
| | | | | | | | | | $s'/Op$ |

State assignment:

| $s$ | $\rightarrow$ | $q_1$ | $q_0$ |
|---|---|---|---|
| $s^0$ | $\rightarrow$ | 0 | 0 |
| $s^1$ | $\rightarrow$ | 0 | 1 |
| $s^2$ | $\rightarrow$ | 1 | 1 |
| $s^3$ | $\rightarrow$ | 1 | 0 |

$$\Rightarrow \quad Op0 = \overline{q}_1\overline{q}_0 = u_{le_A}, u_{le_B}, u_{clr_R}, u_{Rdy0}$$
$$Op1 = \overline{q}_1 q_0 = u_{Rdy1}$$
$$Op2 = q_1 q_0 = u_{le_R}$$
$$Op3 = q_1\overline{q}_0 = u_{sle_A}, u_{sre_B}$$

| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 00 | 00 | – | 00 | 01 | – | 11 | 10 |
| 01 | 00 | 00 | – | 00 | 01 | – | 01 | 01 |
| 11 | 10 | 10 | – | 10 | 10 | – | 10 | 10 |
| 10 | 10 | 11 | – | 01 | 01 | – | 11 | 10 |

$q_1'q_0'$

| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | – | 0 | 1 | – | 1 | 0 |
| 01 | 0 | 0 | – | 0 | 1 | – | 1 | 1 |
| 11 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 10 | 0 | 1 | – | 1 | 1 | – | 1 | 0 |

$q_0'$

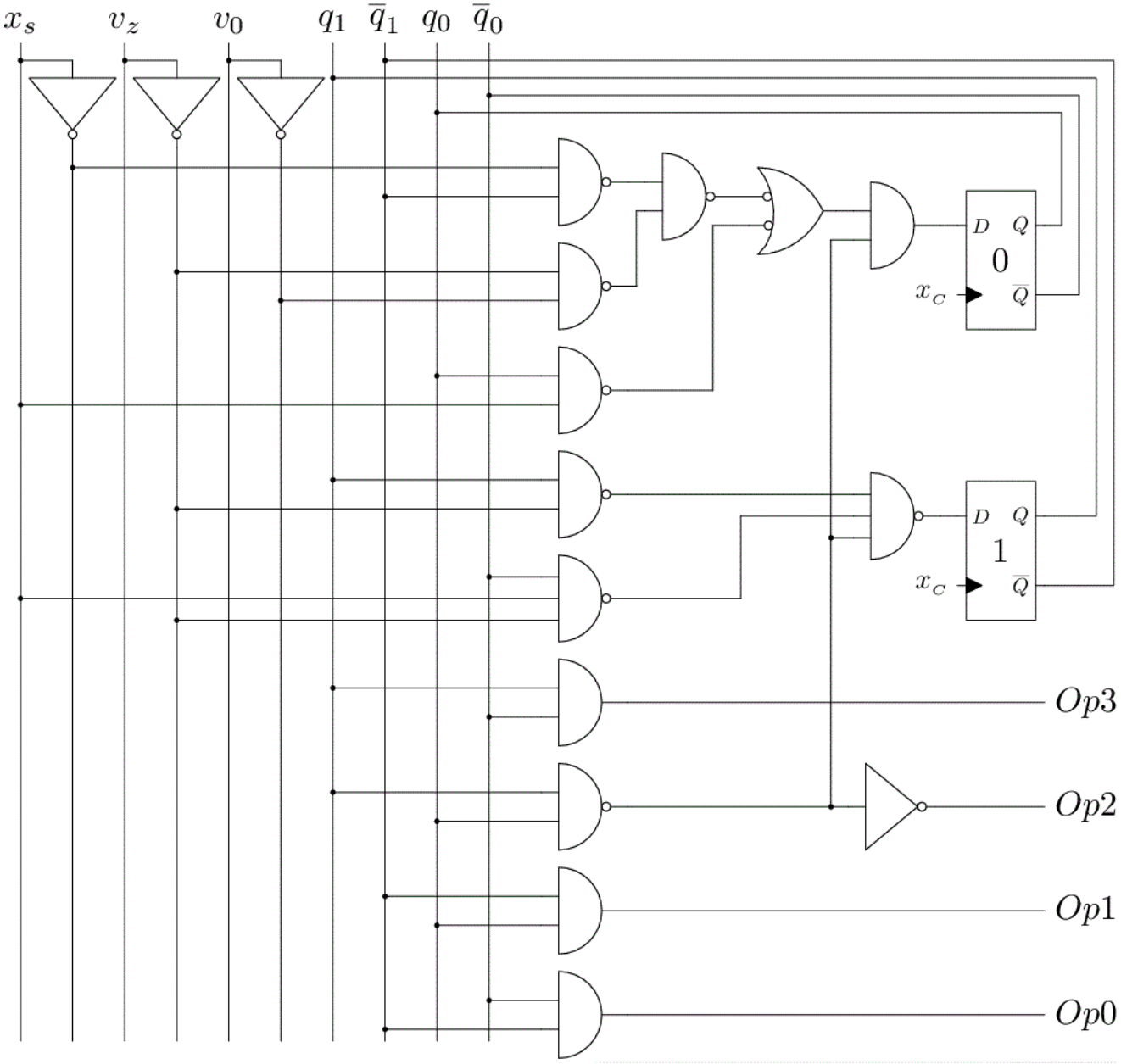
| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | – | 0 | 0 | – | 1 | 1 |
| 01 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 11 | 1 | 1 | – | 1 | 1 | – | 1 | 1 |
| 10 | 1 | 1 | – | 0 | 0 | – | 1 | 1 |

$q_1'$

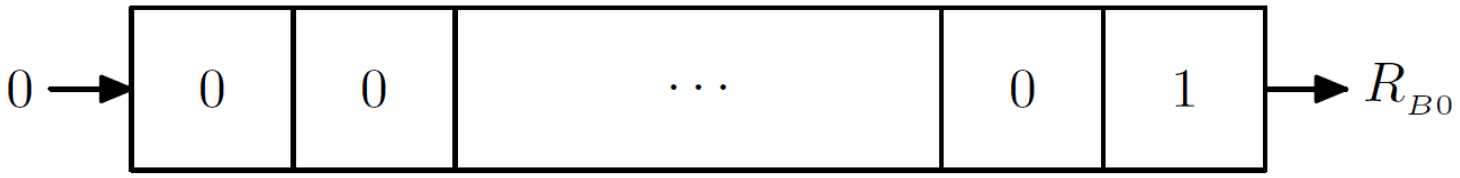
$$\begin{cases} q_1' &= q_1q_0 + q_1\overline{v}_z + \overline{q}_0x_s\overline{v}_z \\ q_0' &= x_sv_z\overline{q_0q_1} + x_sv_0\overline{q_0q_1} + x_sq_0\overline{q_0q_1} + q_1v_z\overline{q_0q_1} + q_1v_0\overline{q_0q_1} \end{cases}$$

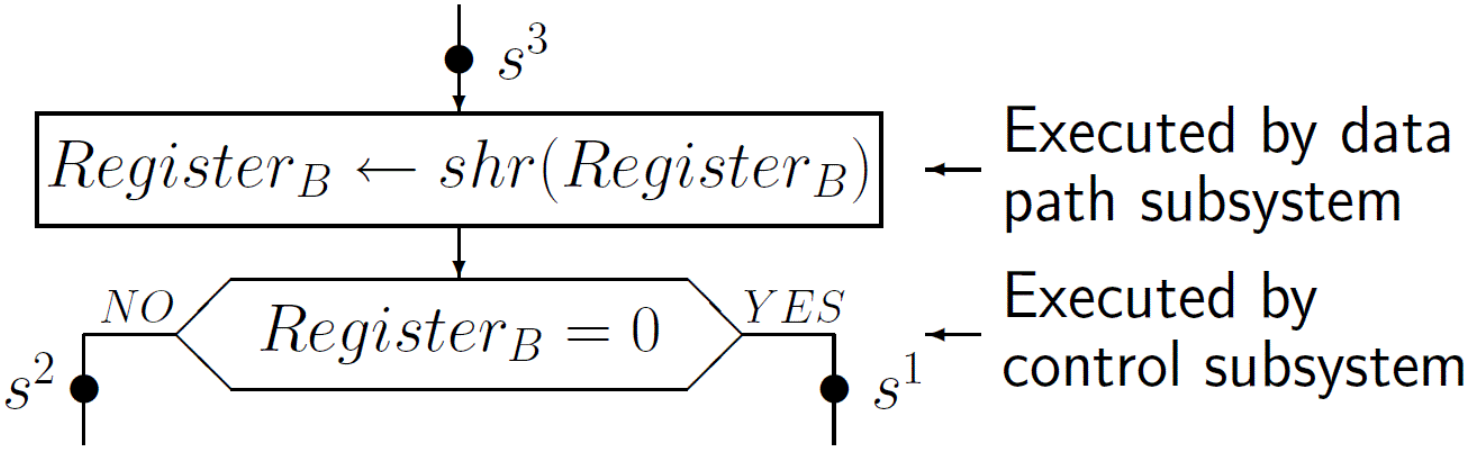
$$\begin{cases} q_1' &= q_1 q_0 + q_1 \overline{v}_z + \overline{q}_0 x_s \overline{v}_z \\ q_0' &= x_s v_z \overline{q_0 q_1} + x_s v_0 \overline{q_0 q_1} + x_s q_0 \overline{q_0 q_1} + q_1 v_z \overline{q_0 q_1} + q_1 v_0 \overline{q_0 q_1} \end{cases}$$

$$\begin{aligned} q_0' &= \Big[x_s v_z + x_s v_0 + x_s q_0 + q_1 v_z + q_1 v_0\Big] \overline{q_0 q_1} \\ &= \Big[x_s (v_z + v_0) + x_s q_0 + q_1 (v_z + v_0)\Big] \overline{q_0 q_1} \\ &= \Big[x_s (v_z + v_0) + q_1 (v_z + v_0) + x_s q_0\Big] \overline{q_0 q_1} \\ &= \Big[(x_s + q_1)(v_z + v_0) + x_s q_0\Big] \overline{q_0 q_1} \\ &= \Big[(\overline{\overline{x}_s \overline{q}_1})(\overline{\overline{v}_z \overline{v}_0}) + x_s q_0\Big] \overline{q_0 q_1} \\ &= \Big[\overline{\overline{x}_s \overline{q}_1}\ \overline{\overline{v}_z \overline{v}_0} + x_s q_0\Big] \overline{q_0 q_1} \end{aligned}$$


$x_s$
$v_z$
$v_0$
$q_1$
$\overline{q}_1$
$q_0$
$\overline{q}_0$
D Q
0
$x_C$
$\overline{Q}$
D Q
1
$x_C$
$\overline{Q}$
$Op3$
$Op2$
$Op1$
$Op0$

# Problem with the Moore automaton



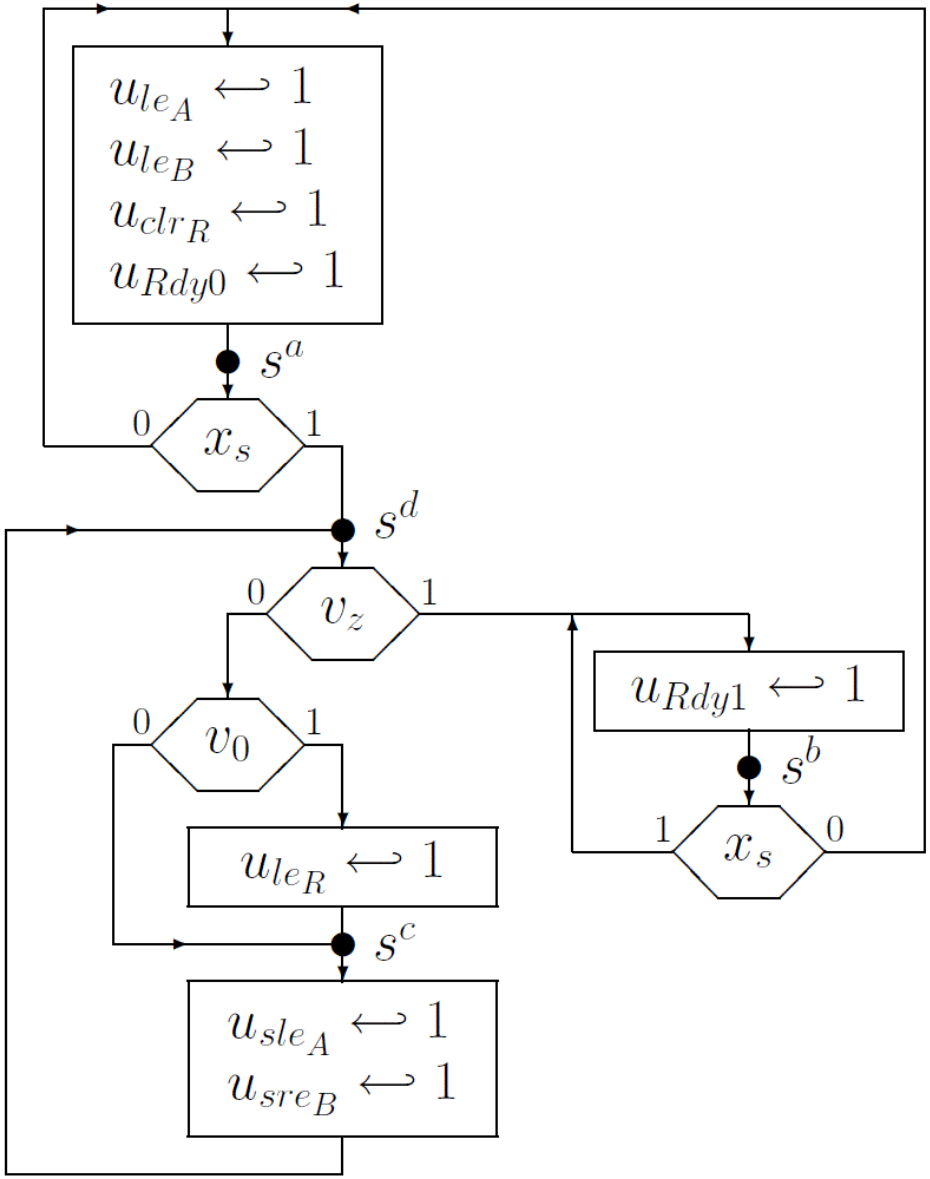



- Initially control automaton is in *s*3 and $Register_B = 1$.
- When the clock ticks we expect *RegisterB* to become 0 and the control automaton to go to *s*1.
- But when the clock ticks the automaton will reach $s^2$ instead of $s^1$, because shifting and testing of the register contents is executed in parallel by the two subsystems.

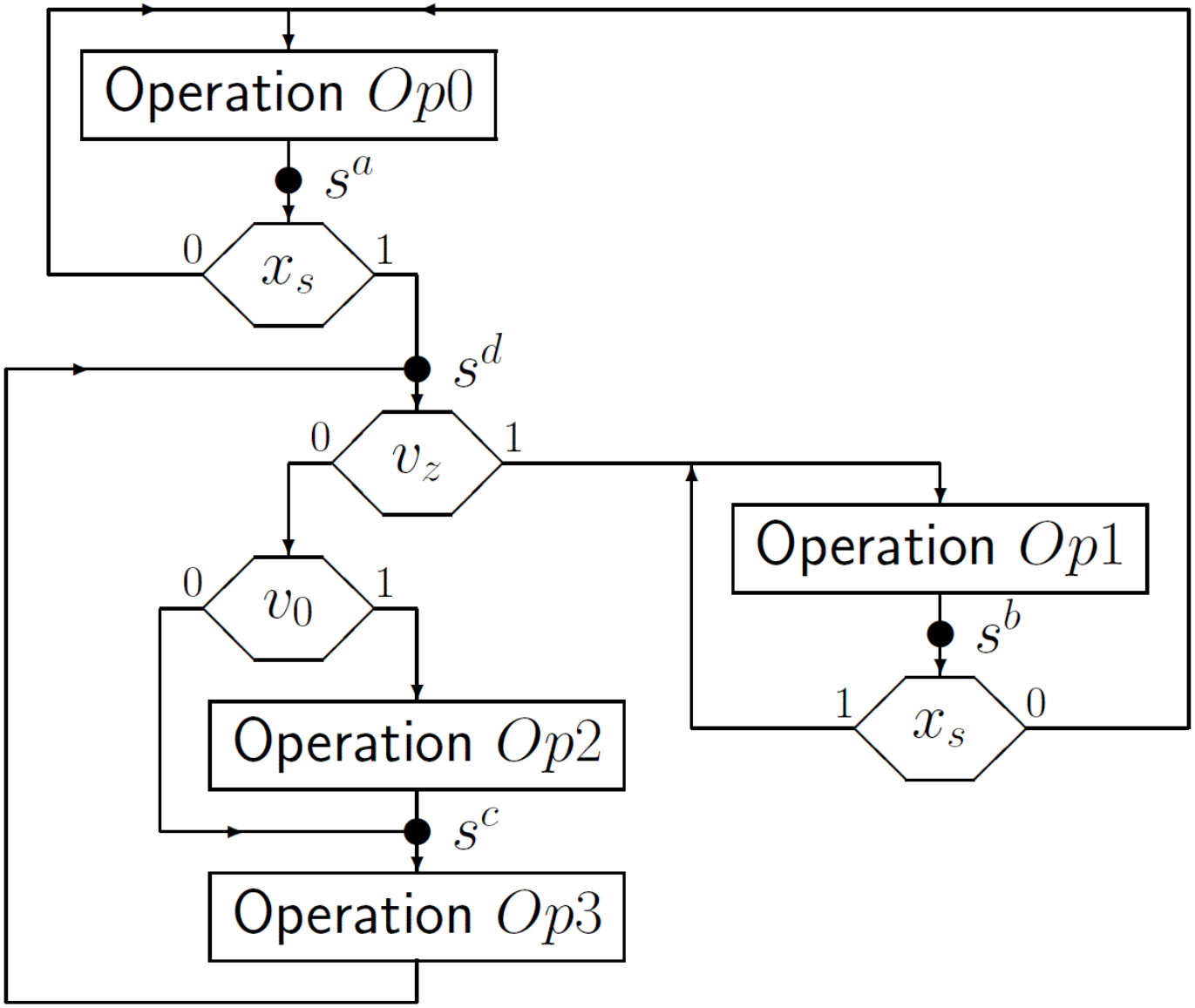
Remedy: clock the two subsystems by a two phase clock (i.e. the positive and negative clock pulse edges) or introduce an extra state in between the operational and conditional blocks.

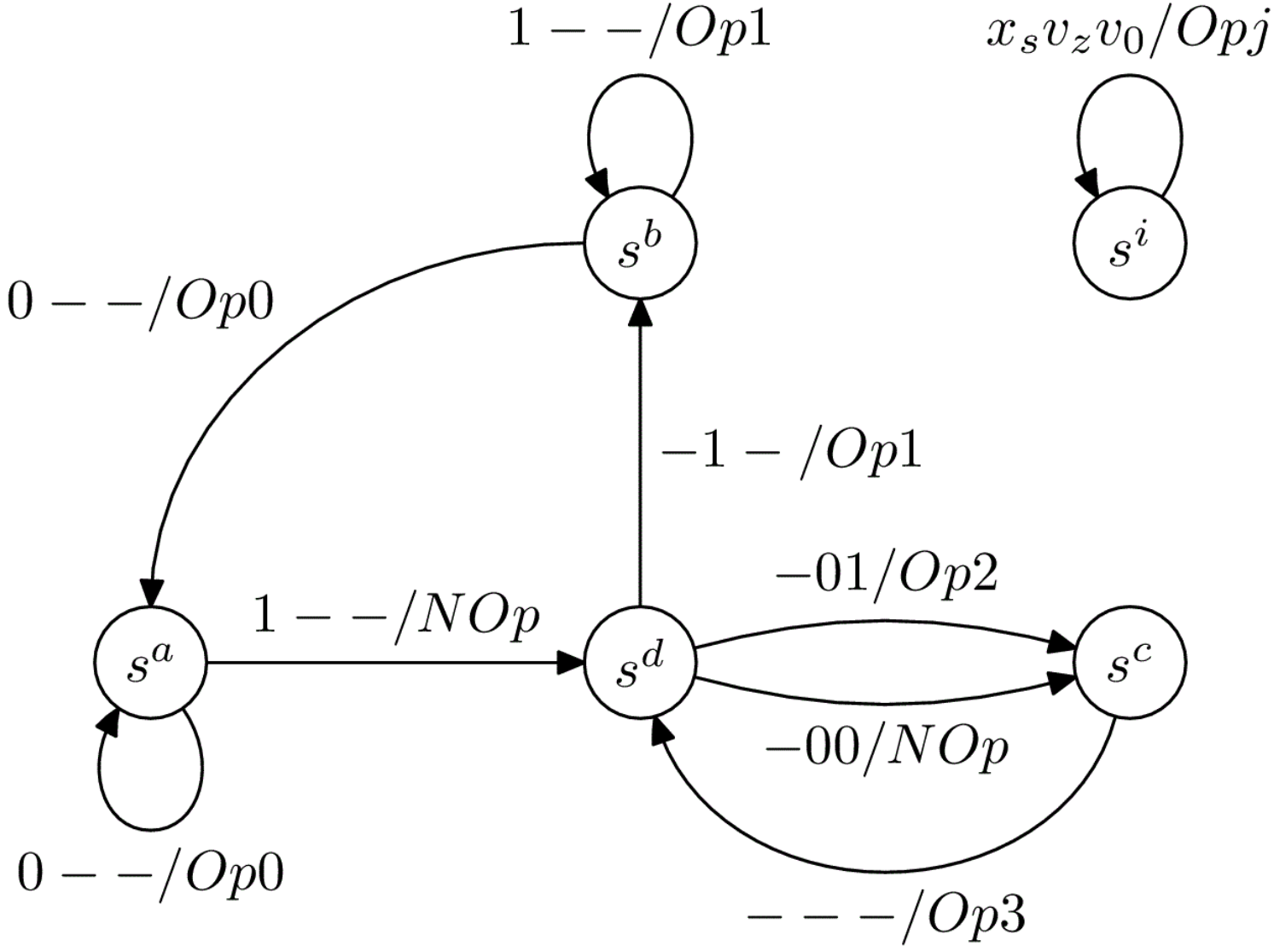
In general, the above should be done for all situations in which the tested condition depends on the operation just prior to it, but only for microoperations that are executed synchronously and not those that are executed statically or dynamically (on the positive signal edge).

# State assignment for a Mealy control automaton



| | $u_{le_A}$ | $u_{le_B}$ | $u_{clr_R}$ | $u_{Rdy0}$ | $u_{Rdy1}$ | $u_{le_R}$ | $u_{sle_A}$ | $u_{sre_B}$ |
|---|---|---|---|---|---|---|---|---|
| $Op0$ | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| $Op1$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $Op2$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $Op3$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| $NOp$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |


Operation $Op0$
$s^a$
0
$x_s$
1
$s^d$
0
$v_z$
1
0
$v_0$
1
Operation $Op2$
$s^c$
Operation $Op3$
Operation $Op1$
$s^b$
1
$x_s$
0


1 − −/Op1
x_s v_z v_0/Opj
s^b
s^i
0 − −/Op0
−1 − /Op1
−01/Op2
1 − −/NOp
s^a
s^d
s^c
−00/NOp
0 − −/Op0
− − −/Op3

| $s$ \ $x_s v_z v_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| $s^a$ | $s^a/Op0$ | $s^a/Op0$ | – | $s^a/Op0$ | $s^d/NOp$ | – | $s^d/NOp$ | $s^d/NOp$ |
| $s^b$ | $s^a/Op0$ | $s^a/Op0$ | – | $s^a/Op0$ | $s^b/Op1$ | – | $s^b/Op1$ | $s^b/Op1$ |
| $s^c$ | $s^d/Op3$ | $s^d/Op3$ | – | $s^d/Op3$ | $s^d/Op3$ | – | $s^d/Op3$ | $s^d/Op3$ |
| $s^d$ | $s^c/NOp$ | $s^c/Op2$ | – | $s^b/Op1$ | $s^b/Op1$ | – | $s^c/Op2$ | $s^c/NOp$ |

$s'/Op$

State assignment:

| $s$ | $\longrightarrow$ | $q_1$ | $q_0$ |
|---|---|---|---|
| $s^a$ | $\longrightarrow$ | 0 | 0 |
| $s^b$ | $\longrightarrow$ | 0 | 1 |
| $s^c$ | $\longrightarrow$ | 1 | 1 |
| $s^d$ | $\longrightarrow$ | 1 | 0 |

| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | $00/Op0$ | $00/Op0$ | – | $00/Op0$ | $10/NOp$ | – | $10/NOp$ | $10/NOp$ |
| 01 | $00/Op0$ | $00/Op0$ | – | $00/Op0$ | $01/Op1$ | – | $01/Op1$ | $01/Op1$ |
| 11 | $10/Op3$ | $10/Op3$ | – | $10/Op3$ | $10/Op3$ | – | $10/Op3$ | $10/Op3$ |
| 10 | $11/NOp$ | $11/Op2$ | – | $01/Op1$ | $01/Op1$ | – | $11/Op2$ | $11/NOp$ |

$q_1'q_0'/Op$

| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | – | 0 | 1 | – | 1 | 1 |
| 01 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 11 | 1 | 1 | – | 1 | 1 | – | 1 | 1 |
| 10 | 1 | 1 | – | 0 | 0 | – | 1 | 1 |

$q_1'$

| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 01 | 0 | 0 | – | 0 | 1 | – | 1 | 1 |
| 11 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 10 | 1 | 1 | – | 1 | 1 | – | 1 | 1 |

$q_0'$

| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | 1 | – | 1 | 0 | – | 0 | 0 |
| 01 | 1 | 1 | – | 1 | 0 | – | 0 | 0 |
| 11 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 10 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |

$Op0$

| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 01 | 0 | 0 | – | 0 | 1 | – | 1 | 1 |
| 11 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 10 | 0 | 0 | – | 1 | 1 | – | 0 | 0 |

$Op1$

| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 01 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 11 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 10 | 0 | 1 | – | 0 | 0 | – | 1 | 0 |

$Op2$

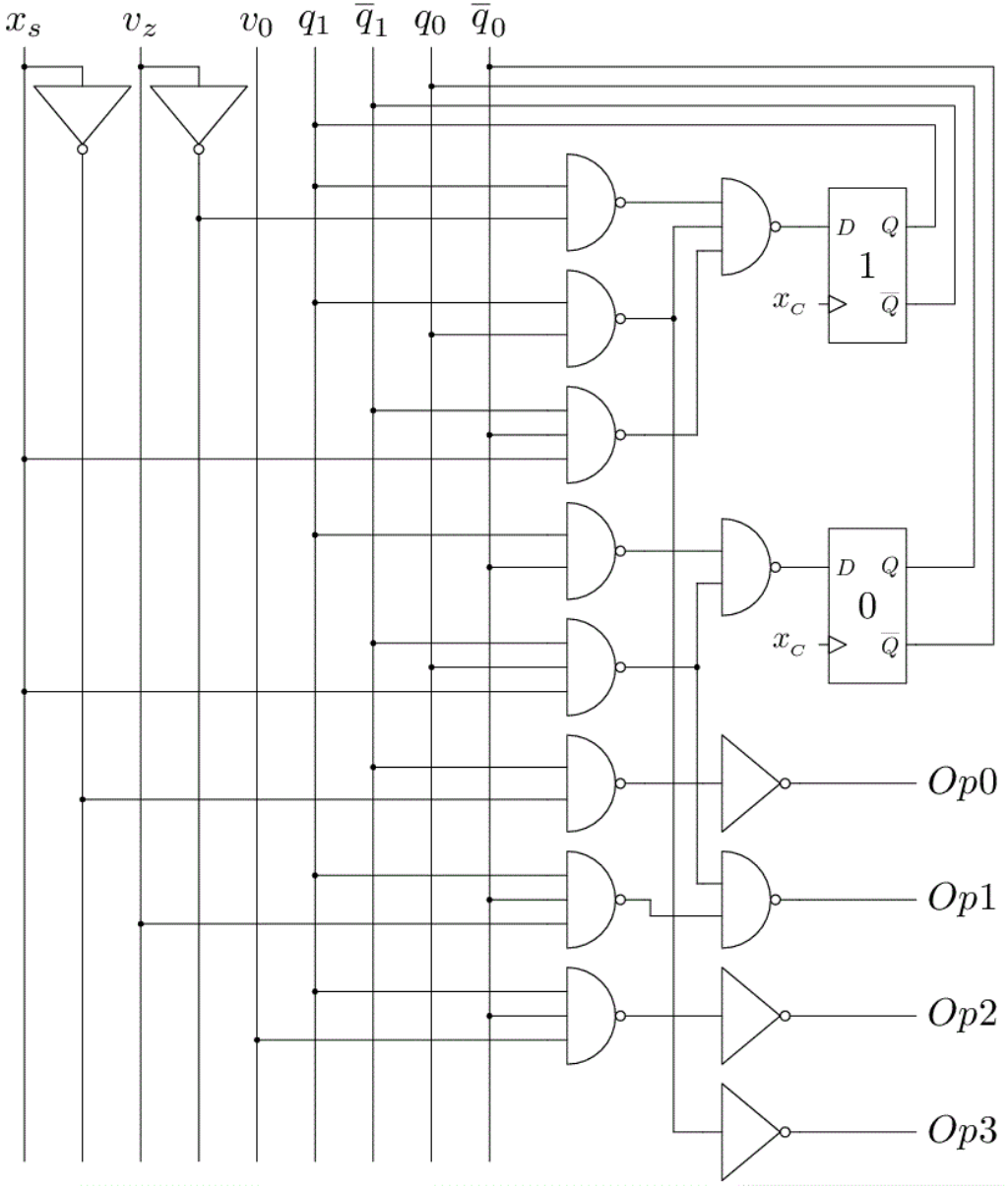
| $q_1q_0$ \ $x_sv_zv_0$ | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 01 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |
| 11 | 1 | 1 | – | 1 | 1 | – | 1 | 1 |
| 10 | 0 | 0 | – | 0 | 0 | – | 0 | 0 |

$Op3$

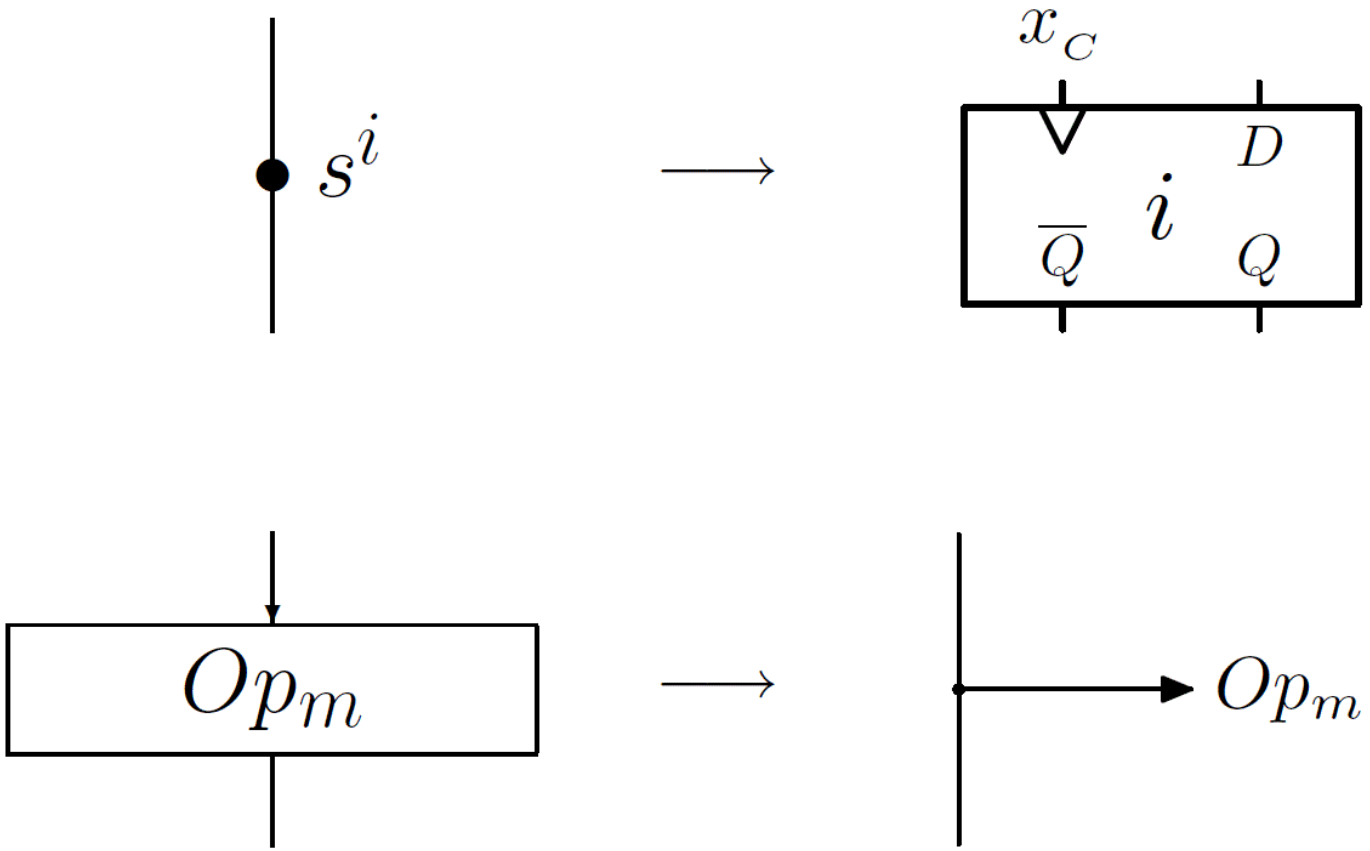
$$\begin{cases} q_1' = q_1\overline{v}_z + q_1q_0 + \overline{q}_1\overline{q}_0x_s \\ q_0' = q_1\overline{q}_0 + \overline{q}_1q_0x_s \\ Op0 = \overline{q}_1\overline{x}_s \\ Op1 = q_1\overline{q}_0v_z + \overline{q}_1q_0x_s \\ Op2 = q_1\overline{q}_0v_0 \\ Op3 = q_1q_0 \end{cases}$$

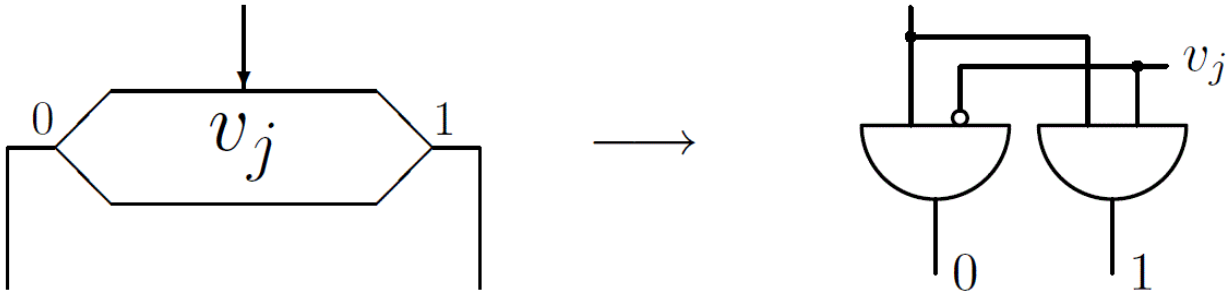
$$\begin{cases} u_{le_A}, u_{le_B}, u_{clr_R}, u_{Rdy0} = \overline{q}_1\overline{x}_s \\ u_{Rdy1} = q_1\overline{q}_0v_z + \overline{q}_1q_0x_s \\ u_{le_R} = q_1\overline{q}_0v_0 \\ u_{sle_A}, u_{sre_B} = q_1q_0 \end{cases}$$

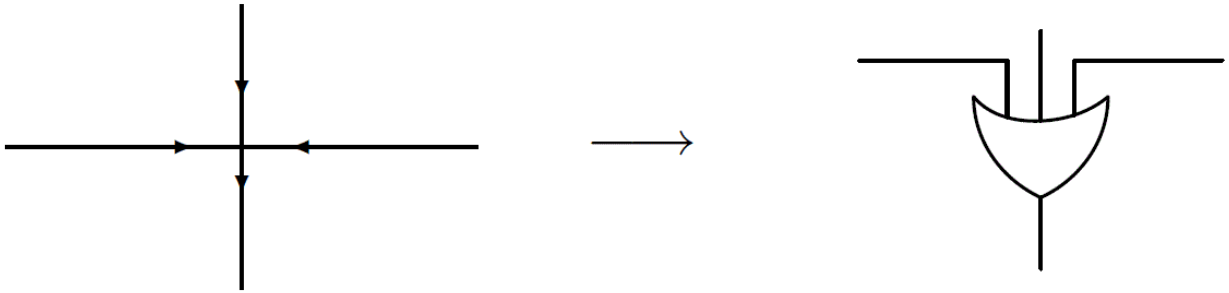
# Circuit diagram of a minimal Mealy control automaton



# Transformation of a flow chart into a sequencer

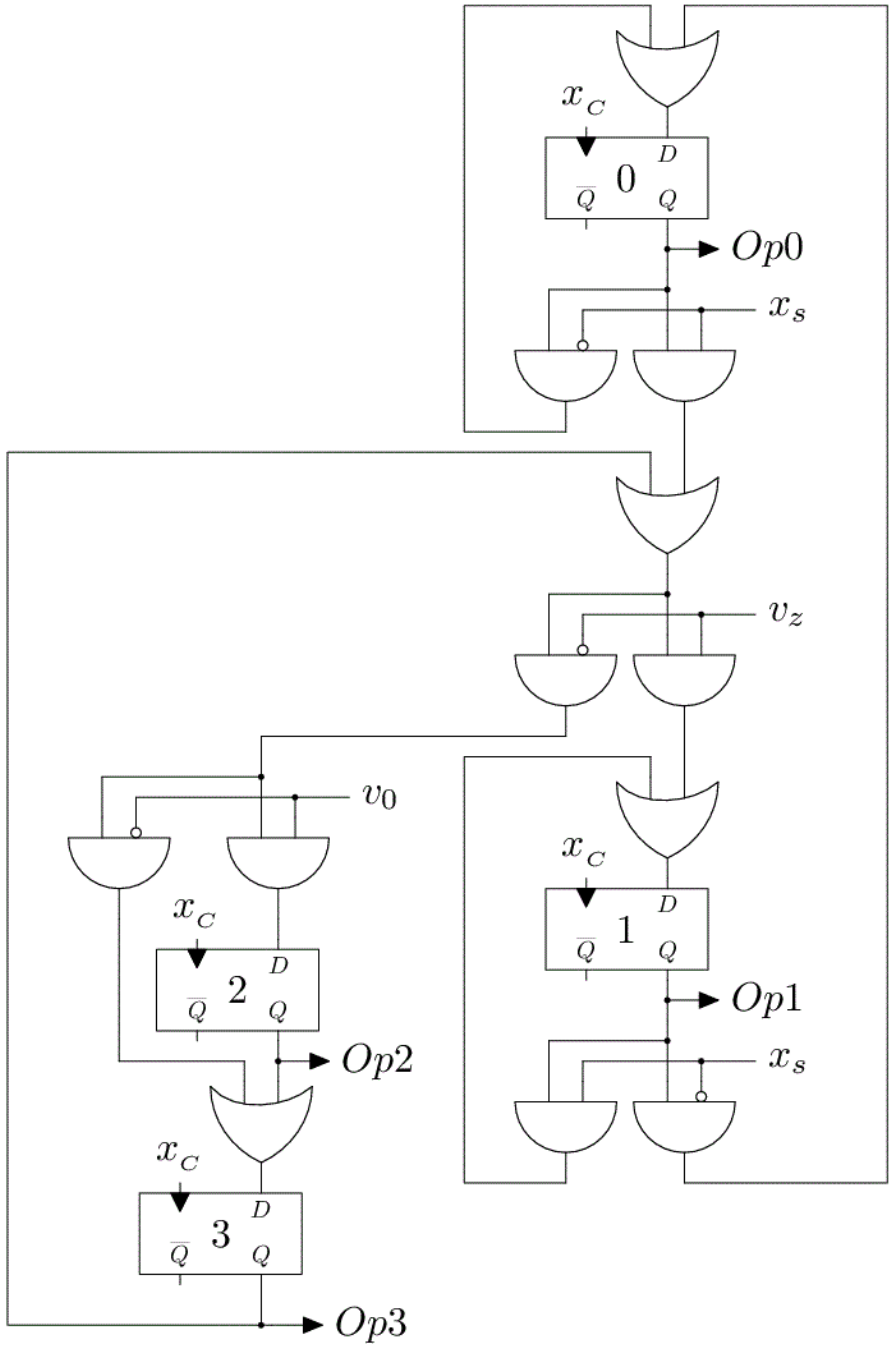




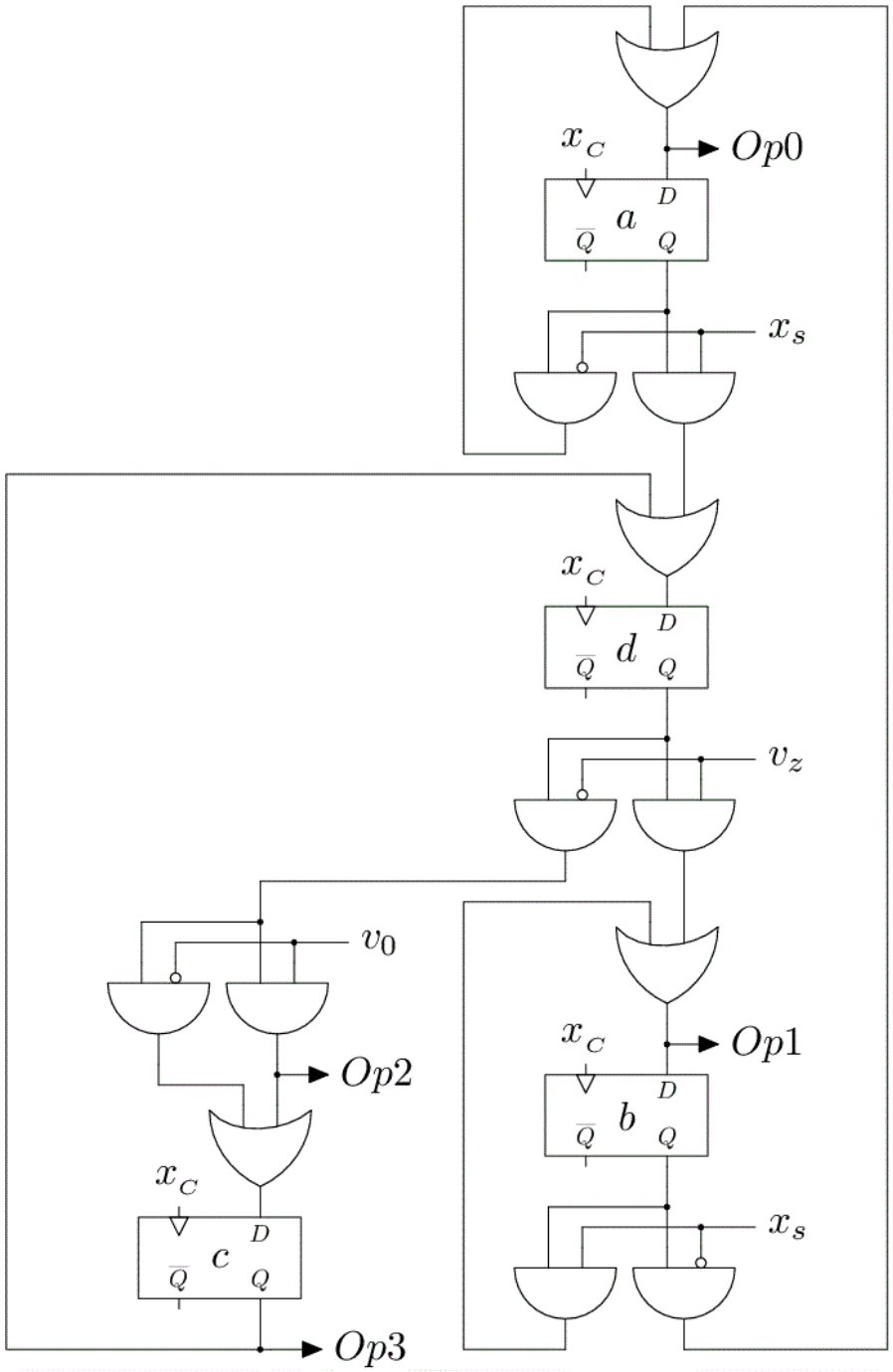



$$\begin{array}{rcccccccccc} & & q_{n-1} & \dots & q_{i+1} & q_i & q_{i-1} & \dots & q_0 \\ s^i & \rightarrow & 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{array}$$

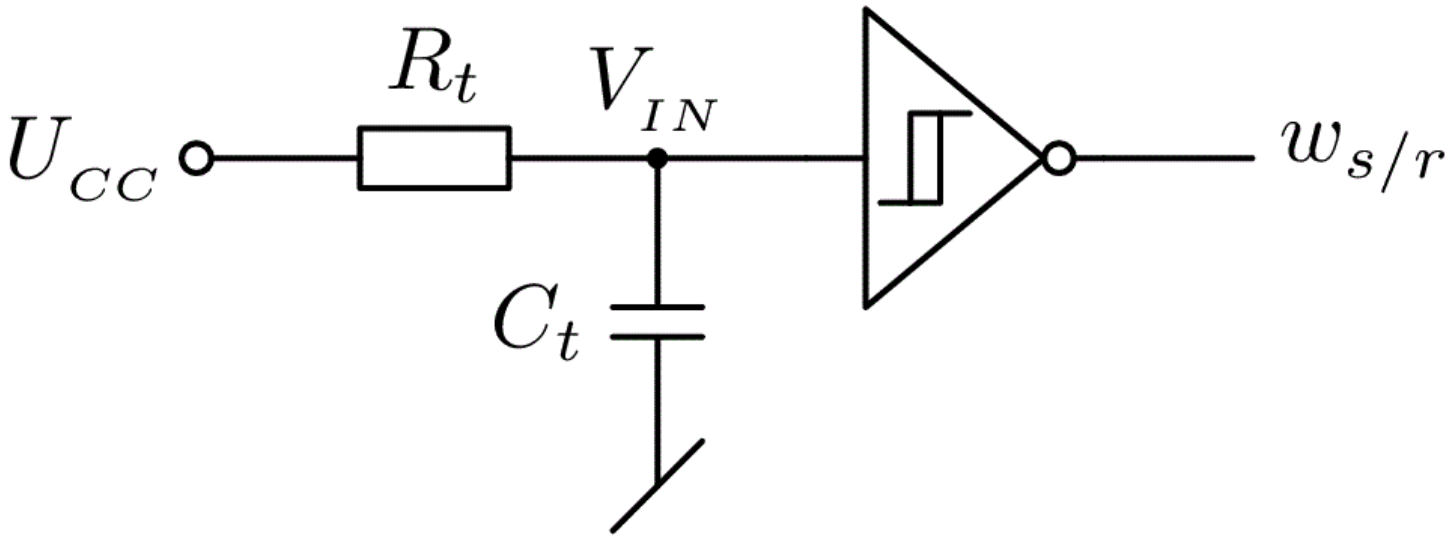
# Circuit diagram of a Moore sequencer



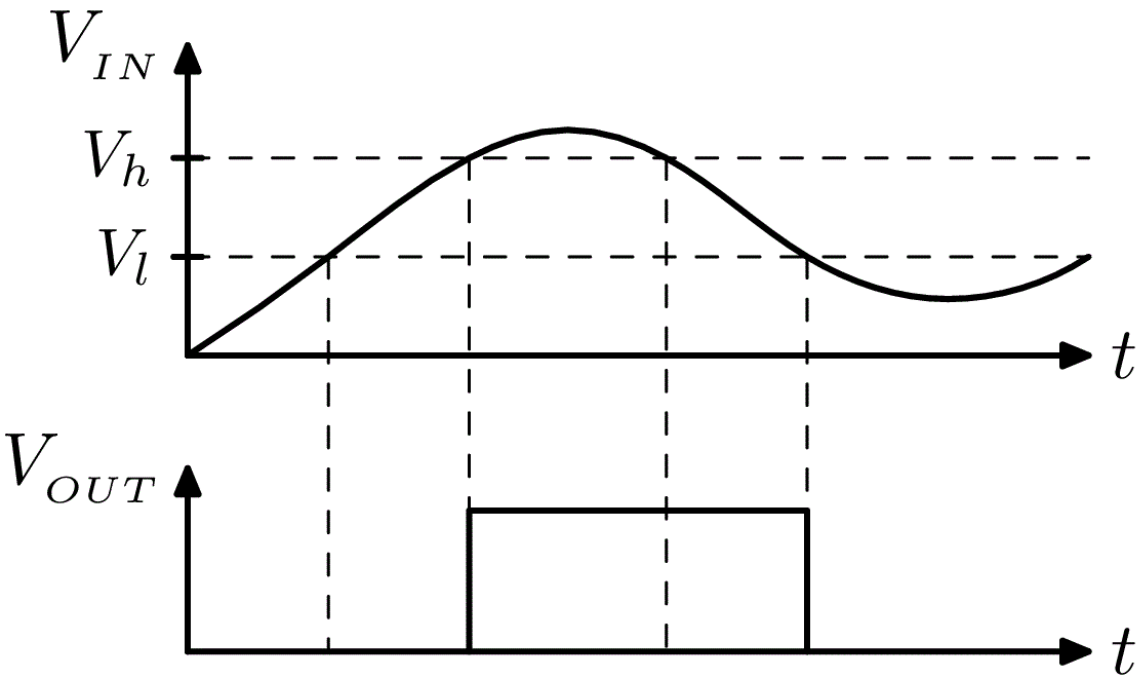
# Circuit diagram of a Mealy sequencer



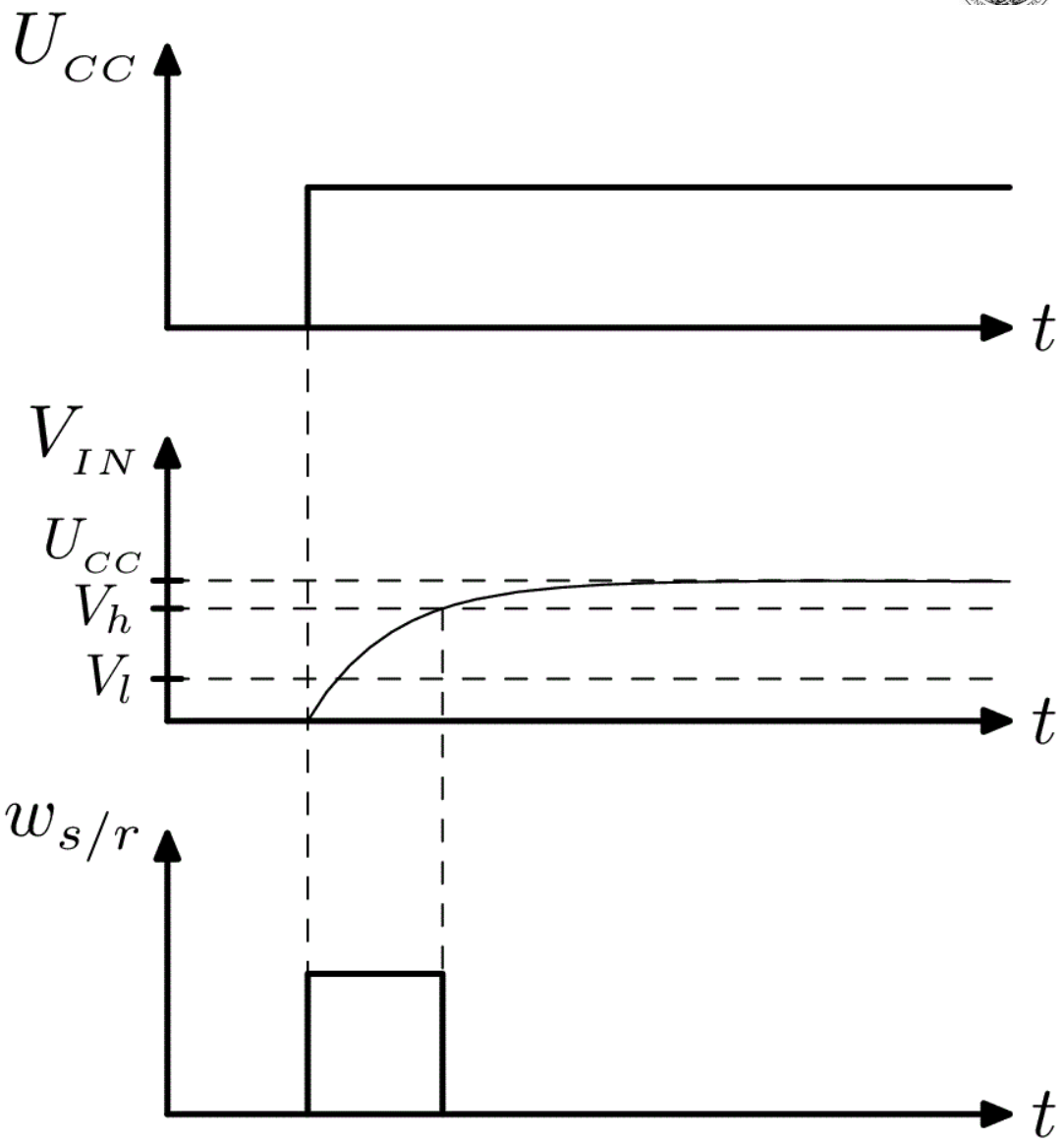
# Initialization circuitry



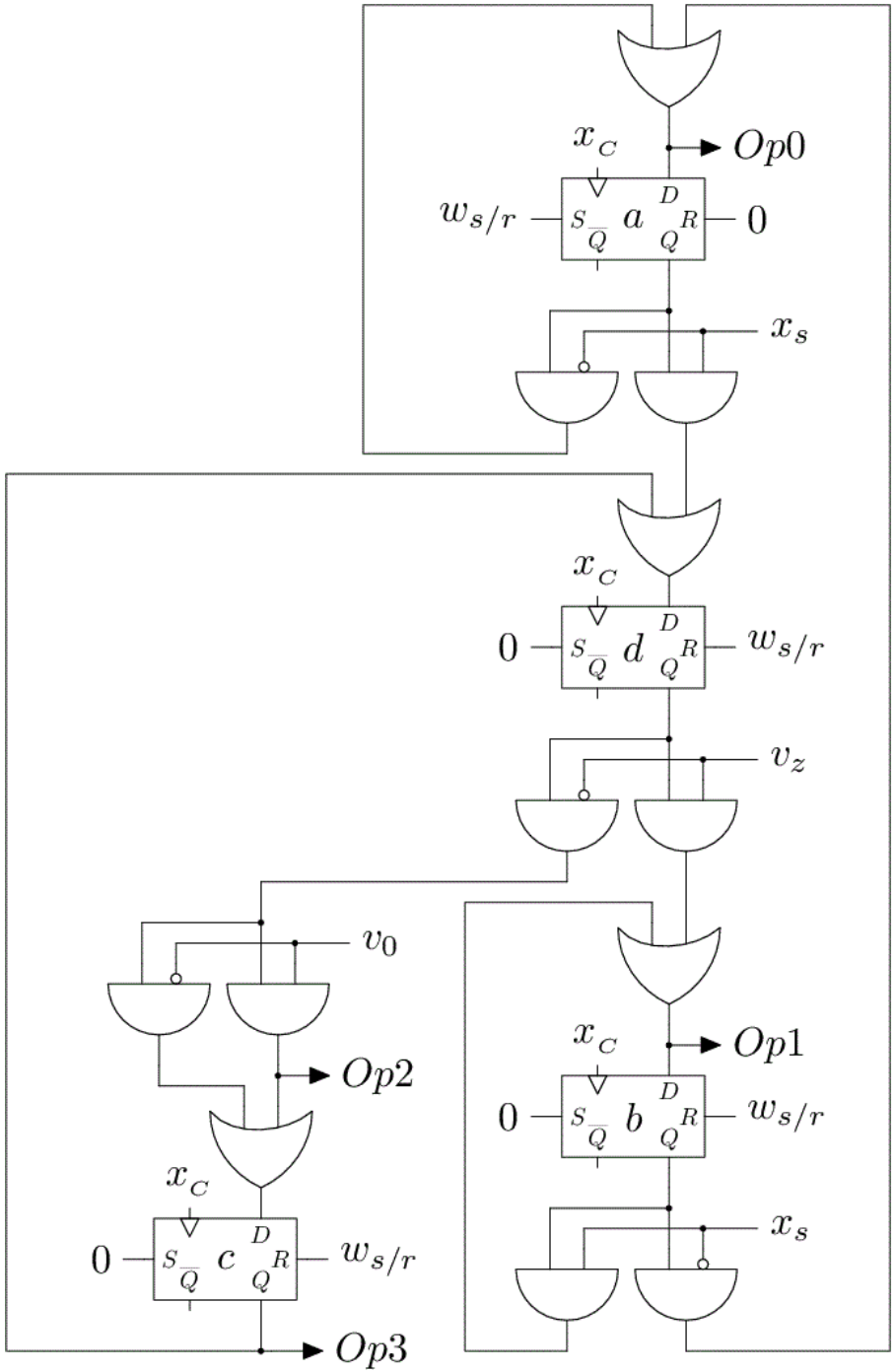
# Schmitt gate operation



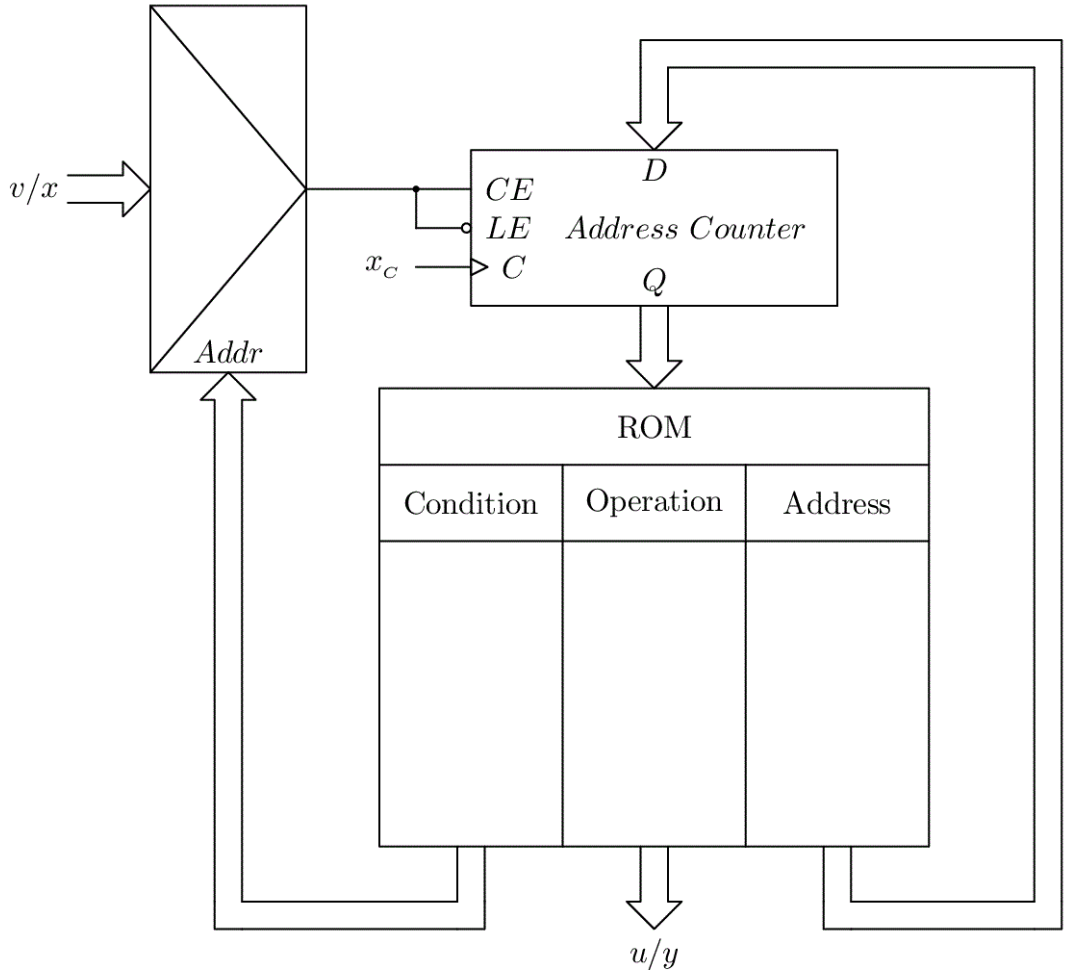
Time plots of the initialization circuitry



# Mealy sequencer with initial reset



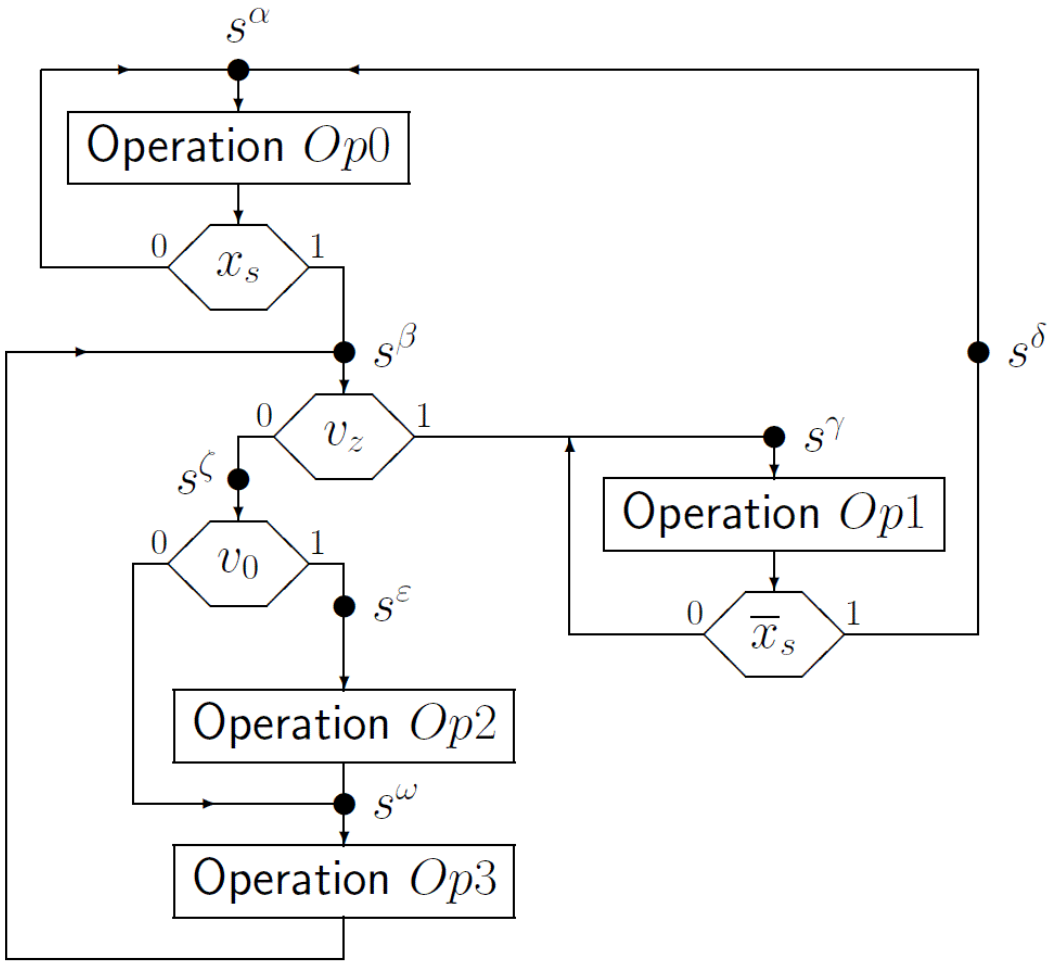
# Microprogrammed Moore control automaton




Microinstruction:

*Address*: *Operation*;
if Condition then
goto $Address_{New}$;
else
goto *Address*+1;
fi;

# State assignment for the microprogrammed Moore control automaton




## State- address assignment:

$$
\begin{aligned}
s^\alpha &\rightarrow Address_0 = 000 \\
s^\beta &\rightarrow Address_1 = 001 \\
s^\gamma &\rightarrow Address_2 = 010 \\
s^\delta &\rightarrow Address_3 = 011 \\
s^\zeta &\rightarrow Address_4 = 100 \\
s^\varepsilon &\rightarrow Address_5 = 101 \\
s^\omega &\rightarrow Address_6 = 110
\end{aligned}
$$

For brevity: $Address_i = Ad_i$

# Microprogram of the Moore control automaton

$Ad_0$: $Op0$; if $x_s$ then goto $Ad_1$; else goto $Ad_0$; fi;

$Ad_1$: $NOp$; if $v_z$ then goto $Ad_2$; else goto $Ad_4$; fi;

$Ad_2$: $Op1$; if $\overline{x}_s$ then goto $Ad_3$; else goto $Ad_2$; fi;

$Ad_3$: $NOp$; if $false$ then goto $-$; else goto $Ad_0$; fi;

$Ad_4$: $NOp$; if $v_0$ then goto $Ad_5$; else goto $Ad_6$; fi;

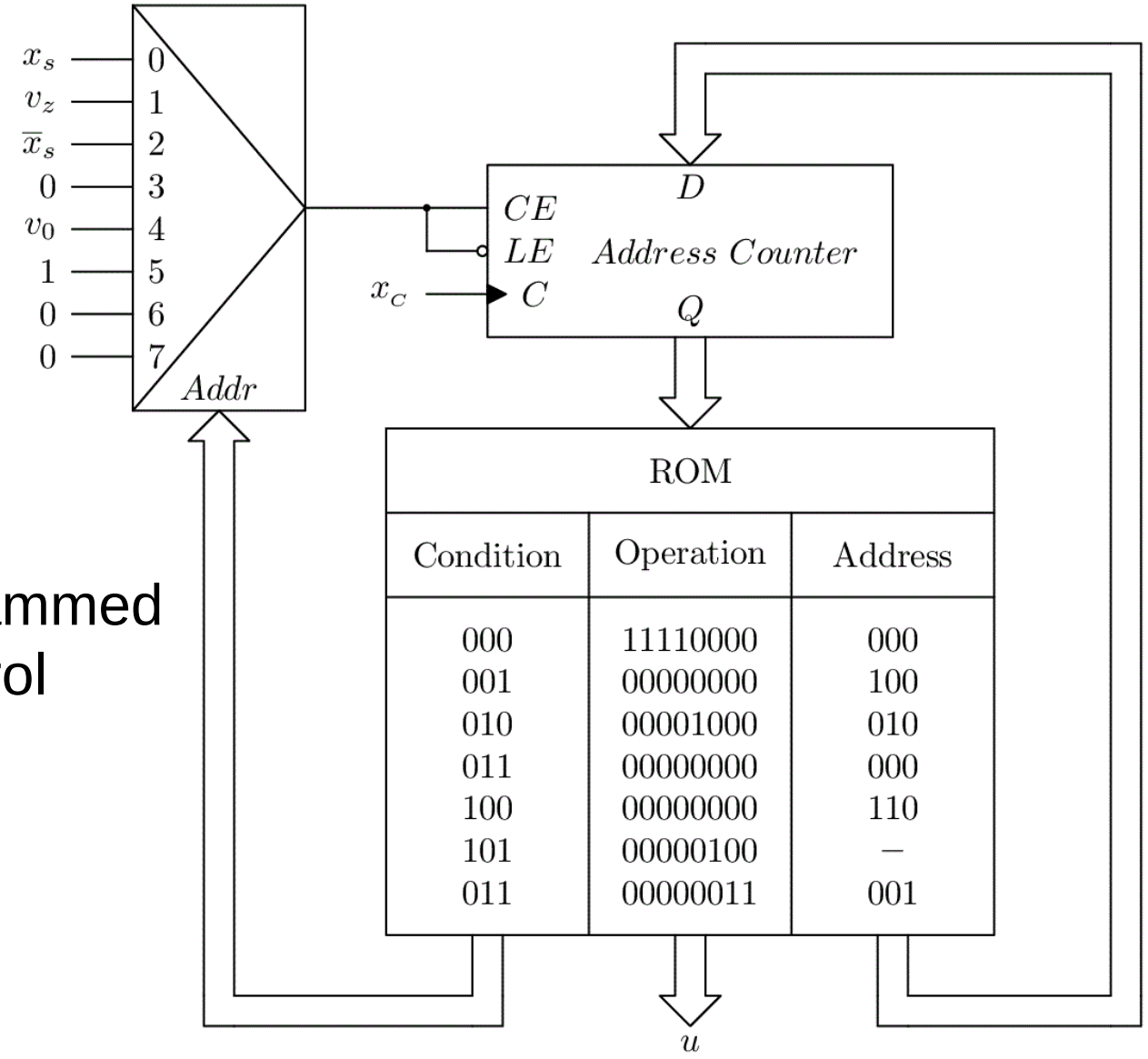
$Ad_5$: $Op2$; if $true$ then goto $Ad_6$; else goto $-$; fi;

$Ad_6$: $Op3$; if $false$ then goto $-$; else goto $Ad_1$; fi;

# ROM contents

| $Ad_i$ | $C_j$ | $u_{le_A}$ | $u_{le_B}$ | $u_{clr_R}$ | $u_{Rdy0}$ | $u_{Rdy1}$ | $u_{le_R}$ | $u_{sle_A}$ | $u_{sre_B}$ | $Ad_k$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 000 | 000 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 000 |
| 001 | 001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 |
| 010 | 010 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 010 |
| 011 | 011 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 000 |
| 100 | 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 110 |
| 101 | 101 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | – |
| 110 | 011 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 001 |

Complete microprogrammed Moore control automaton

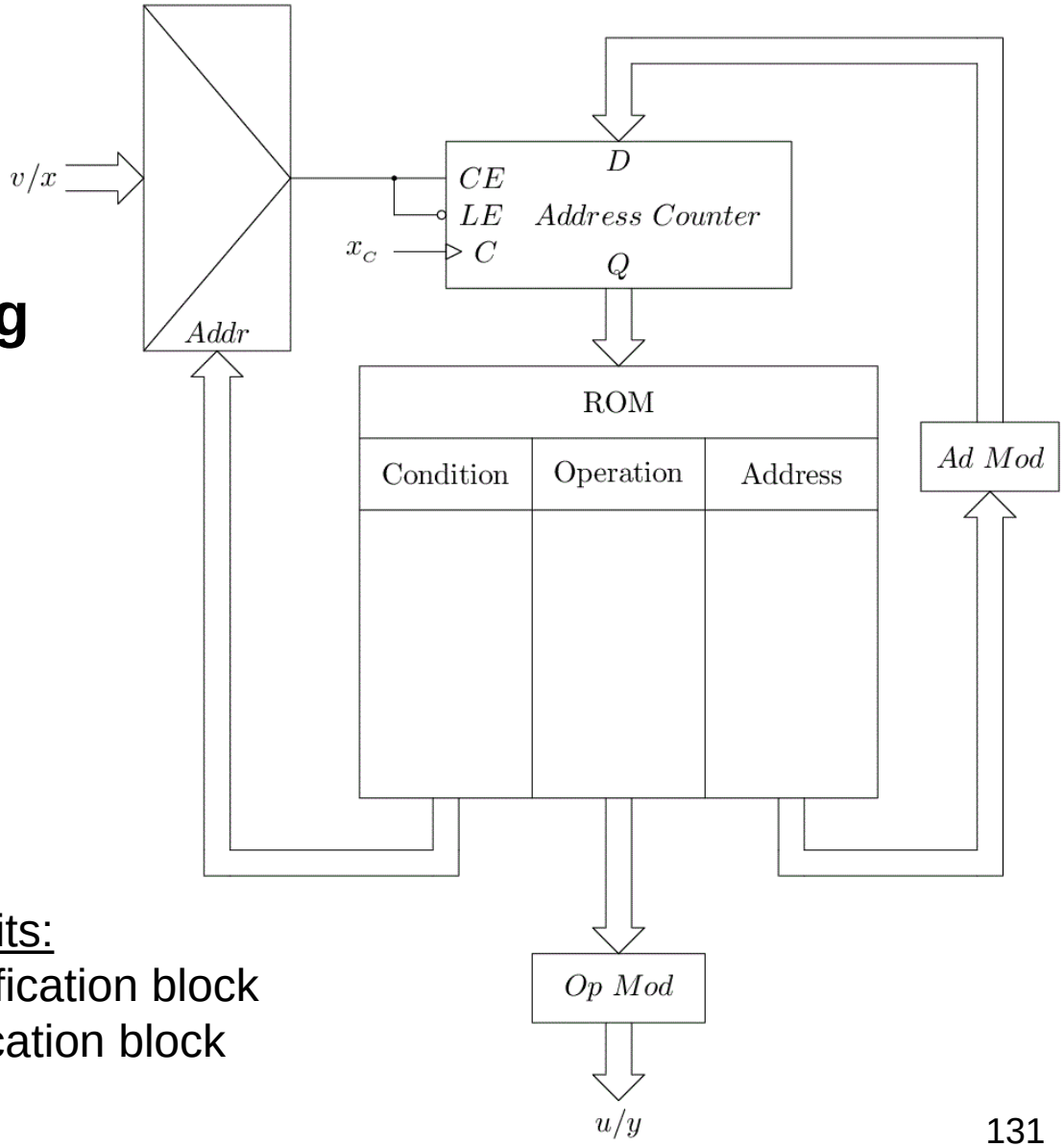


# Reduction of the ROM size

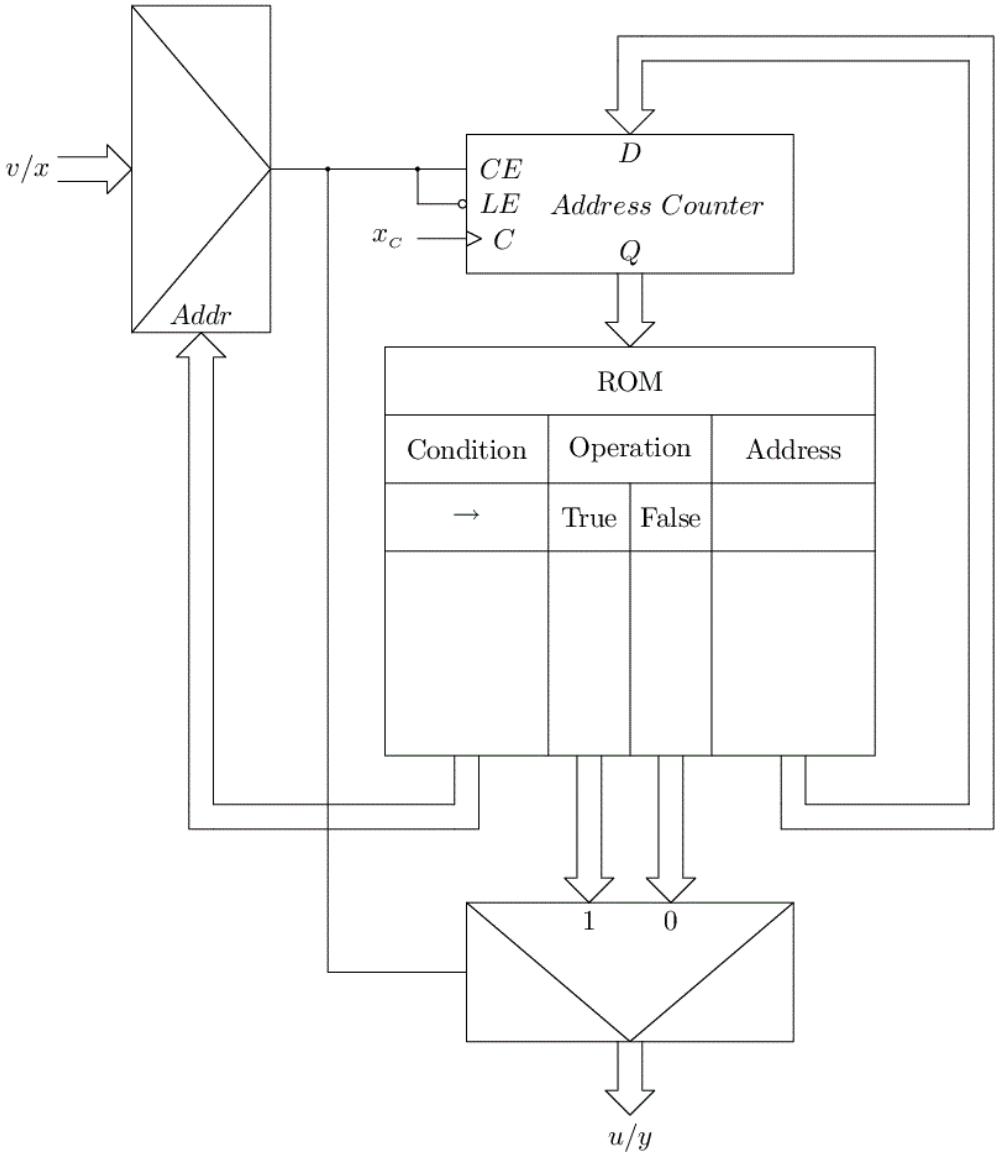
| $Ad_i$ | $C_j$ | $Op0$ | $Op1$ | $Op2$ | $Op3$ | $Ad_k$ |
|---|---|---|---|---|---|---|
| 000 | 000 | 1 | 0 | 0 | 0 | 000 |
| 001 | 001 | 0 | 0 | 0 | 0 | 100 |
| 010 | 010 | 0 | 1 | 0 | 0 | 010 |
| 011 | 011 | 0 | 0 | 0 | 0 | 000 |
| 100 | 100 | 0 | 0 | 0 | 0 | 110 |
| 101 | 101 | 0 | 0 | 1 | 0 | – |
| 110 | 011 | 0 | 0 | 0 | 1 | 001 |
| | 6 cases | 5 cases | | | | 5 cases |

$$
\begin{aligned}
Op0 &= u_{le_A}, u_{le_B}, u_{clr_R}, u_{Rdy0} \\
Op1 &= u_{Rdy1} \\
Op2 &= u_{le_R} \\
Op3 &= u_{sle_A}, u_{sre_B}
\end{aligned}
$$

# ROM space saving modifications




Combinational logic circuits:
*Op Mod* - operation modification block
*Ad Mod* - address modification block

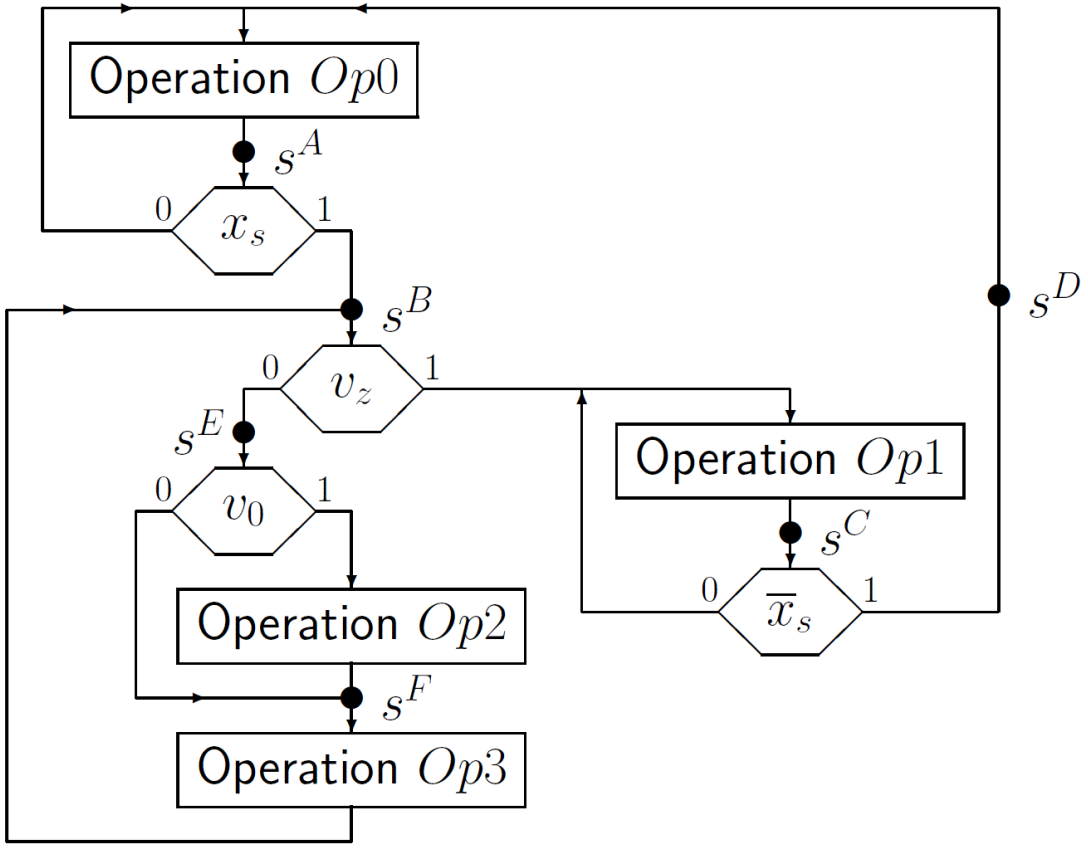



# Microprogrammed Mealy control automaton

Microinstruction:

*Address*:
    if *Condition* then
        *OperationT* ;
        goto $Address_{New}$;
    else
        *OperationF*;
        goto *Address*++;
    fi;

# State assignment for the microprogrammed Mealy control automaton




State-address assignment:

$$
\begin{aligned}
s^A &\rightarrow Address_0 = 000 \\
s^B &\rightarrow Address_1 = 001 \\
s^C &\rightarrow Address_2 = 010 \\
s^D &\rightarrow Address_3 = 011 \\
s^E &\rightarrow Address_4 = 100 \\
s^F &\rightarrow Address_5 = 101
\end{aligned}
$$

For brevity: $Address_i = Ad_i$

# Microprogram of the Mealy control automaton

$Ad_0$: if $x_s$ then $NOp$; goto $Ad_1$; else $Op0$; goto $Ad_0$; fi;

$Ad_1$: if $v_z$ then $Op1$; goto $Ad_2$; else $NOp$; goto $Ad_4$; fi;

$Ad_2$: if $\overline{x}_s$ then $NOp$; goto $Ad_3$; else $Op1$; goto $Ad_2$; fi;

$Ad_3$: if $false$ then $NOp$; goto $-$; else $Op0$; goto $Ad_0$; fi;

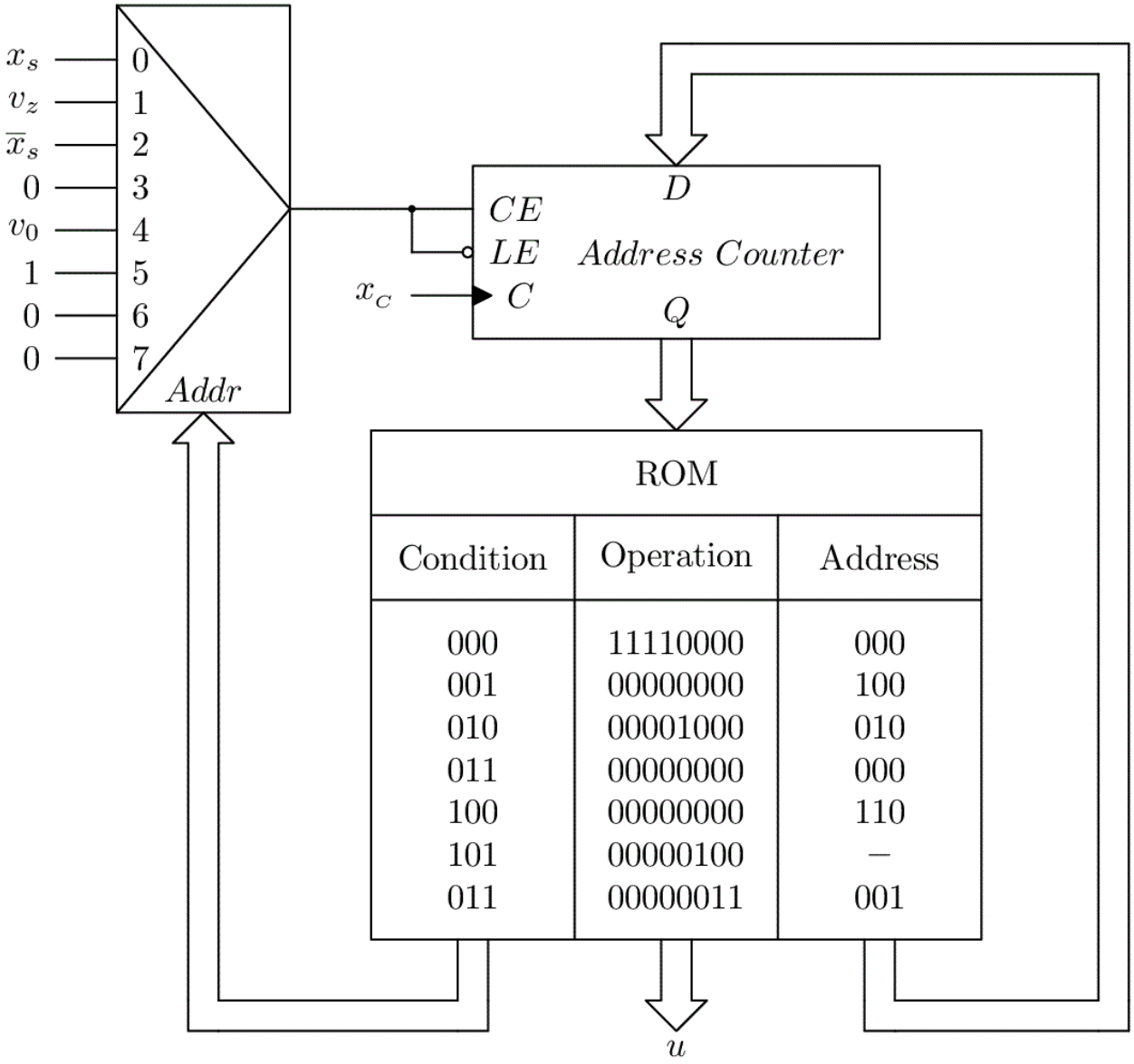
$Ad_4$: if $v_0$ then $Op2$; goto $Ad_5$; else $NOp$; goto $Ad_5$; fi;

$Ad_5$: if $false$ then $NOp$; goto $-$; else $Op3$; goto $Ad_1$; fi;

# ROM contents

| $Ad_i$ | $C_j$ | $u_{le_A}$ | $u_{le_B}$ | $u_{clr_R}$ | $u_{Rdy0}$ | $u_{Rdy1}$ | $u_{le_R}$ | $u_{sle_A}$ | $u_{sre_B}$ |
|---|---|---|---|---|---|---|---|---|---|
| 000 | 000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 001 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 010 | 010 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 011 | 011 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 100 | 100 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 101 | 011 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| $Ad_i$ | $u_{le_A}$ | $u_{le_B}$ | $u_{clr_R}$ | $u_{Rdy0}$ | $u_{Rdy1}$ | $u_{le_R}$ | $u_{sle_A}$ | $u_{sre_B}$ | $Ad_k$ |
|---|---|---|---|---|---|---|---|---|---|
| 000 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 000 |
| 001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 |
| 010 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 010 |
| 011 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 000 |
| 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 101 |
| 101 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 001 |



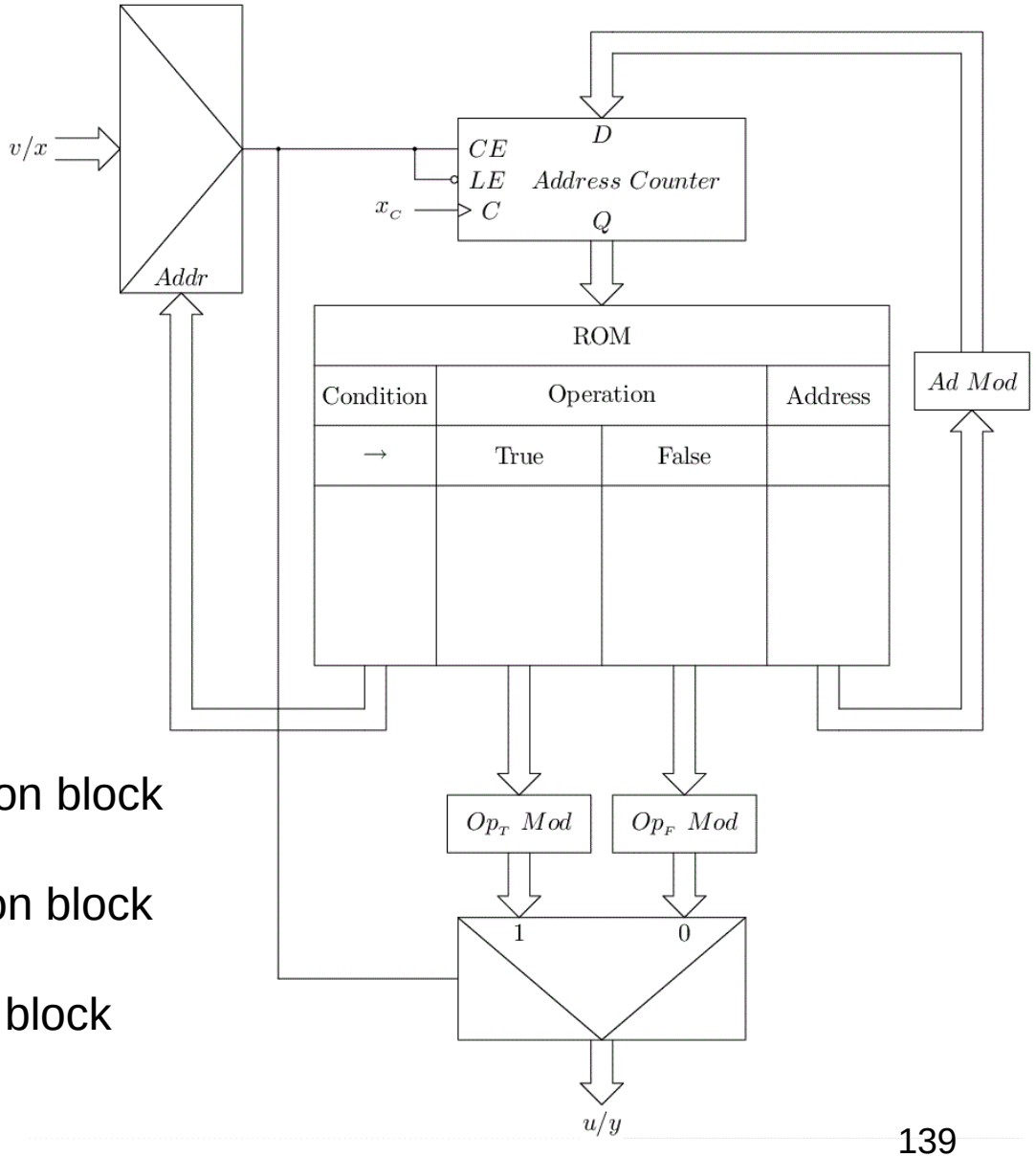

**Complete microprogrammed Mealy control automaton**

# Reduction of the ROM size

$$
\begin{aligned}
Op0 &= u_{le_A}, u_{le_B}, u_{clr_R}, u_{Rdy0} \\
Op1 &= u_{Rdy1} \\
Op2 &= u_{le_R} \\
Op3 &= u_{sle_A}, u_{sre_B}
\end{aligned}
$$

| | | $Operation_T$ | | | | $Operation_F$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $Ad_i$ | $C_j$ | $Op0$ | $Op1$ | $Op2$ | $Op3$ | $Op0$ | $Op1$ | $Op2$ | $Op3$ | $Ad_k$ |
| 000 | 000 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 000 |
| 001 | 001 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 100 |
| 010 | 010 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 010 |
| 011 | 011 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 000 |
| 100 | 100 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 101 |
| 101 | 011 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 001 |
| | | 3 cases | | | | 4 cases | | | | |
| | | 5 different cases (3 bits) in all | | | | | | | | |

# ROM space saving modifications (V.1)



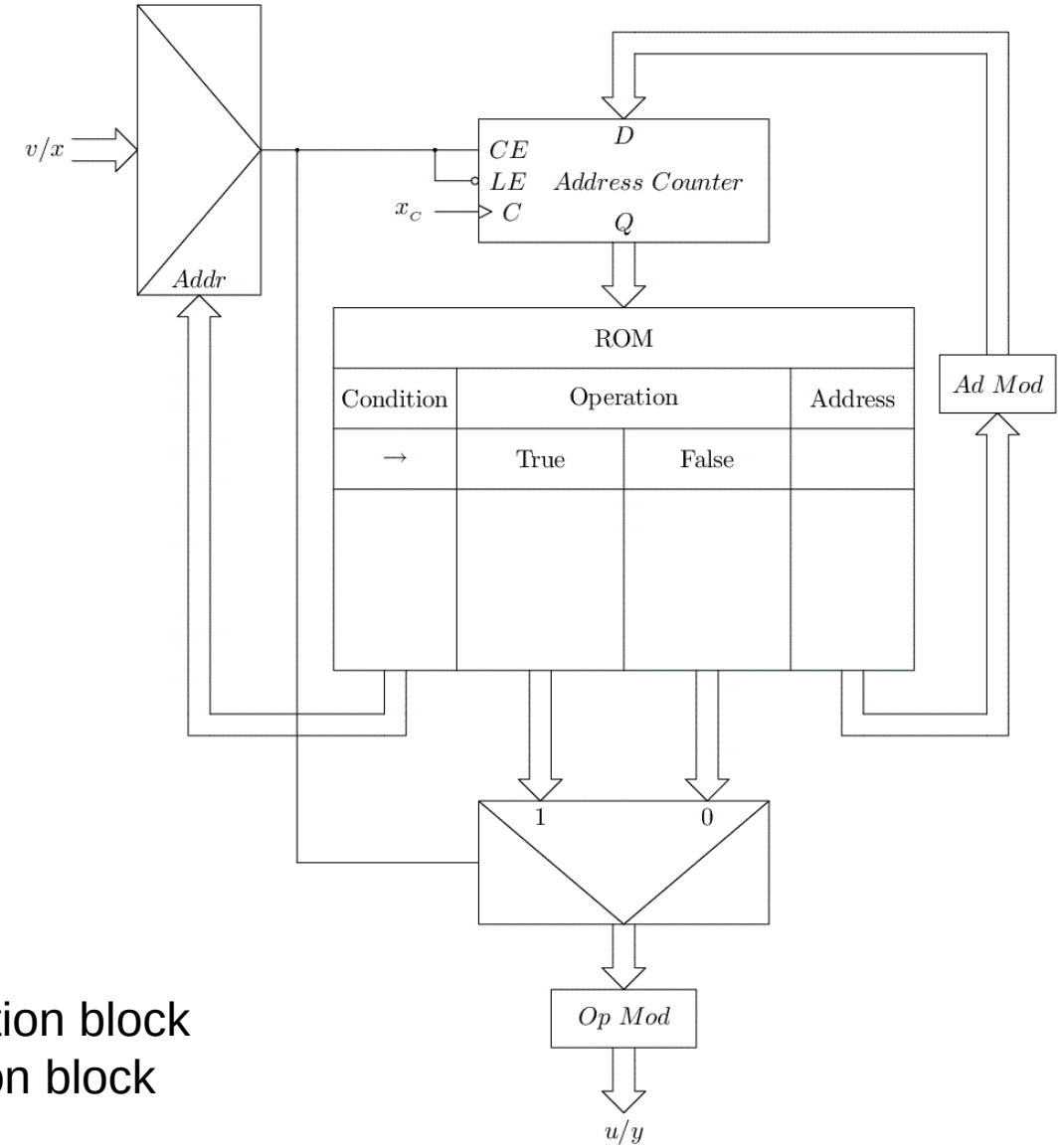

Combinational logic circuits:

$Op_T$ Mod - operation modification block (for condition true)

*$Op_F$ Mod* - operation modfication block (for condition false)

*Ad Mod* - address modification block

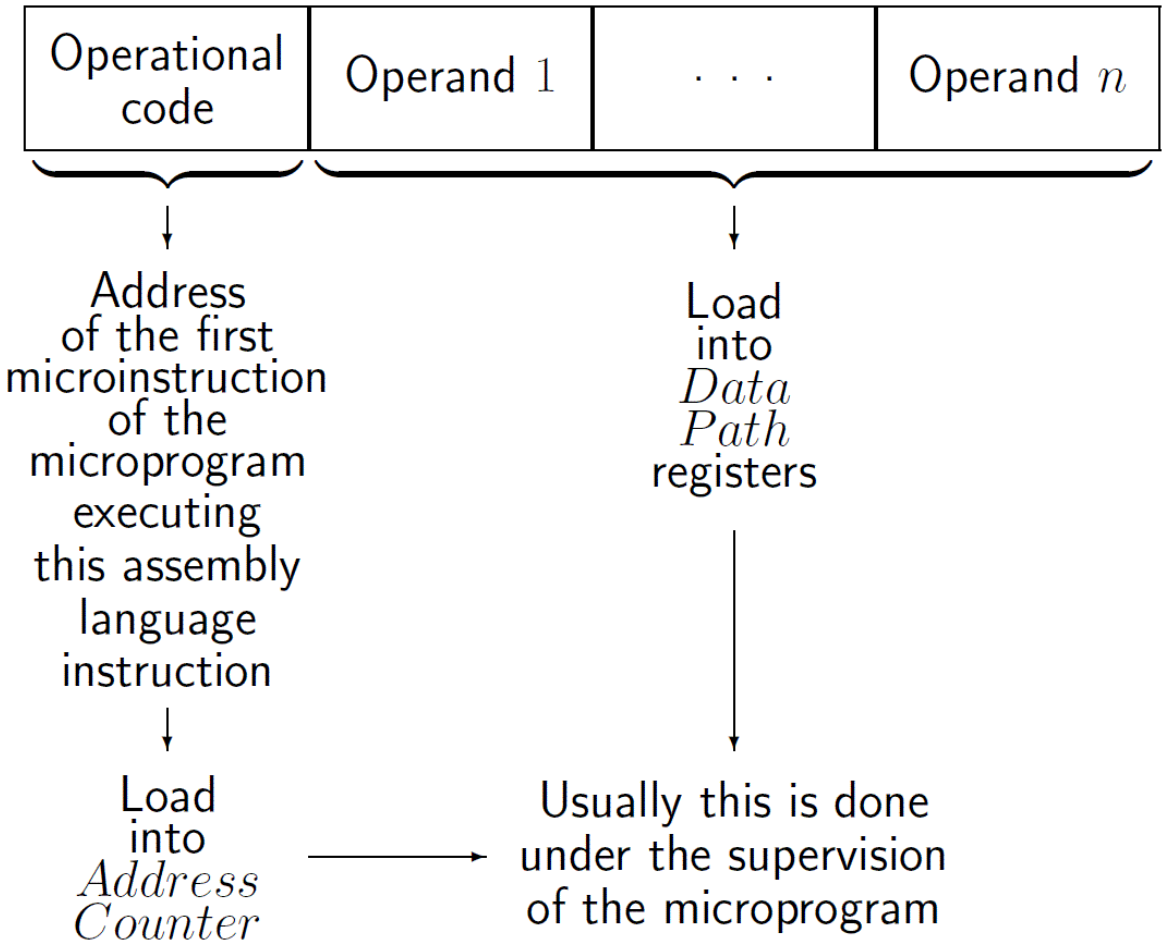
# ROM space saving modifications (V.2)




Combinational logic circuits:

Op Mod - operation modification block
*Ad Mod* - address modification block

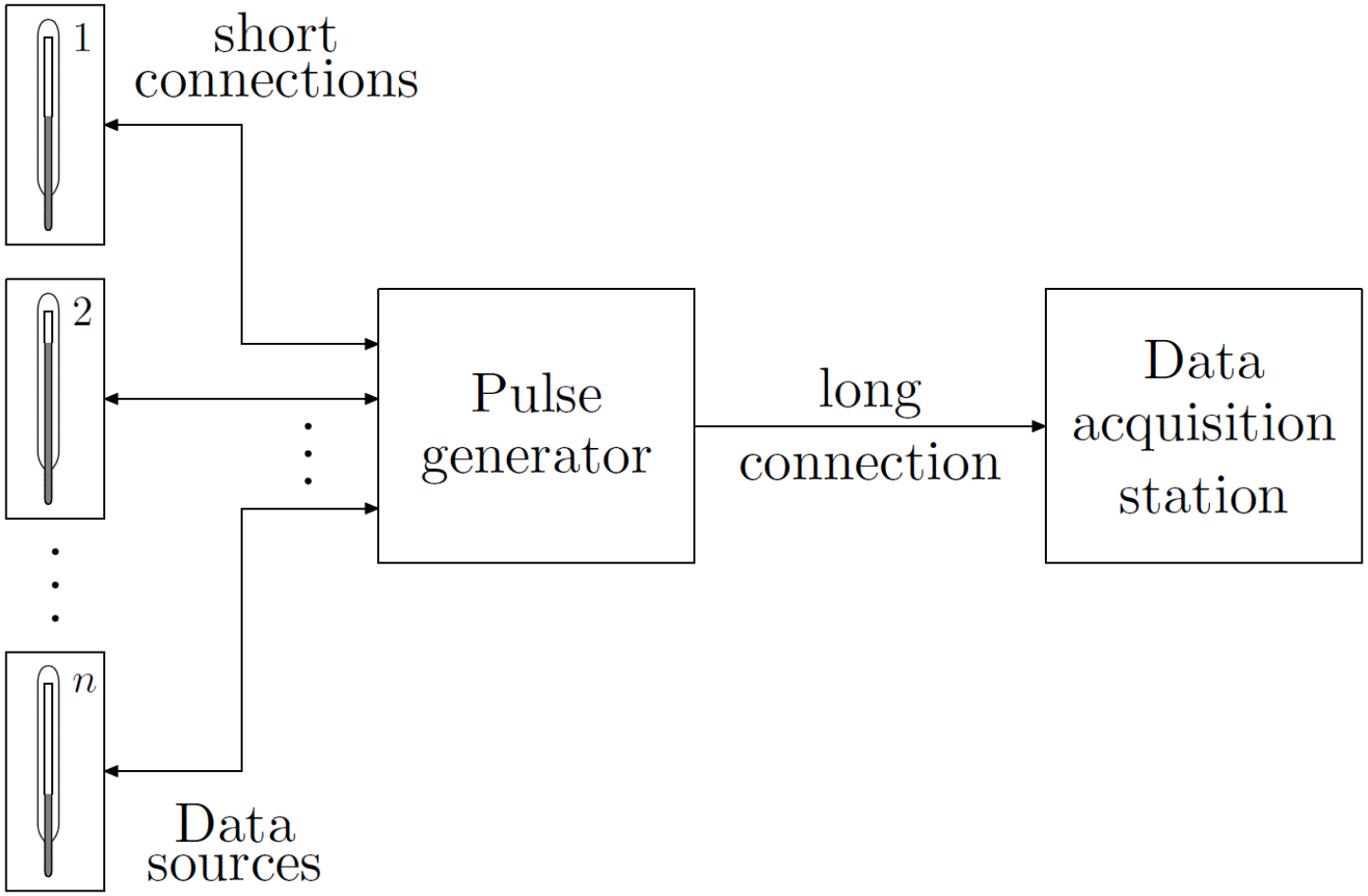
# Microprogrammed machines

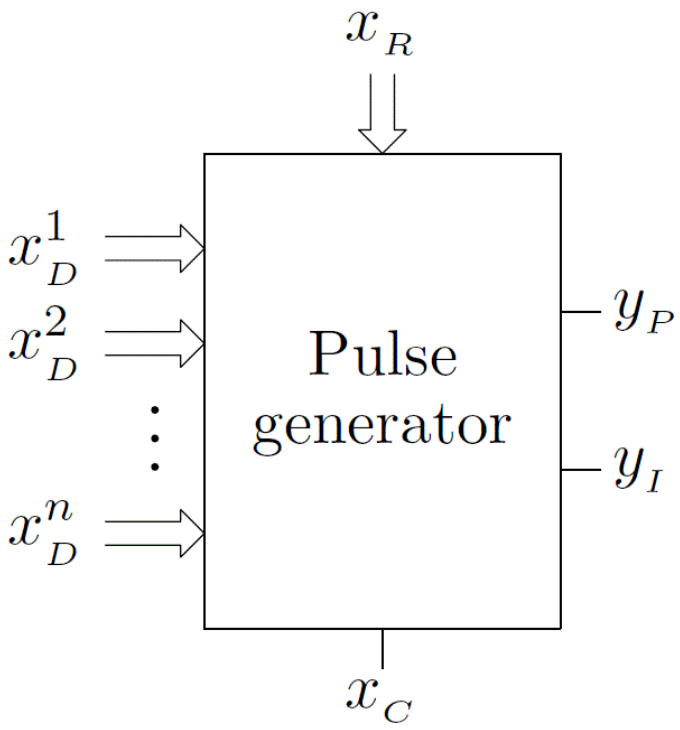


# What else needs to be covered:

- assembly languages (computer architectures)
- operating systems (virtual machines)
- real time operating systems (real time systems)
- high level programming languages (programming concepts)
  - procedural approach
  - object oriented approach
- programming real time systems

# Multi-source parallel-to-serial data converter






$x_D^i$ – data inputs $i = 1, \ldots, n$, ($n$ – number of sources)
(data to be transmitted in parallel binary format)

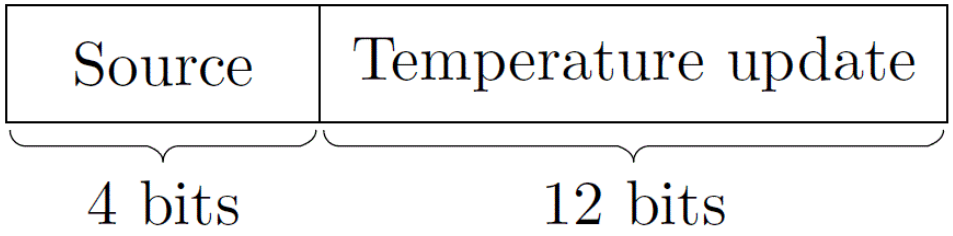
$x_R$ – data ready (one line per source)

$x_C$ – clock input

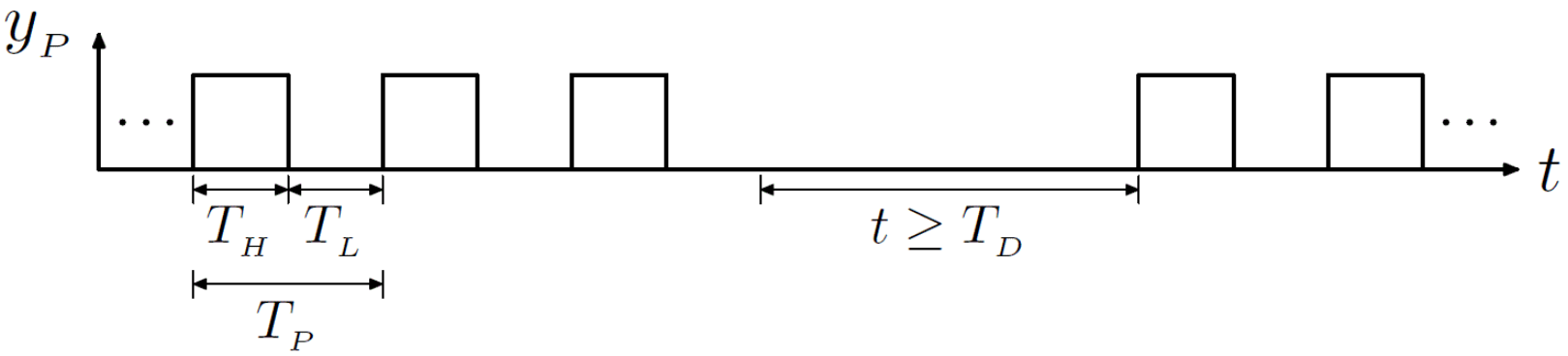
$y_P$ – pulse train output
(transmitted data in serial pulse format)

$y_I$ – source data inhibiting output

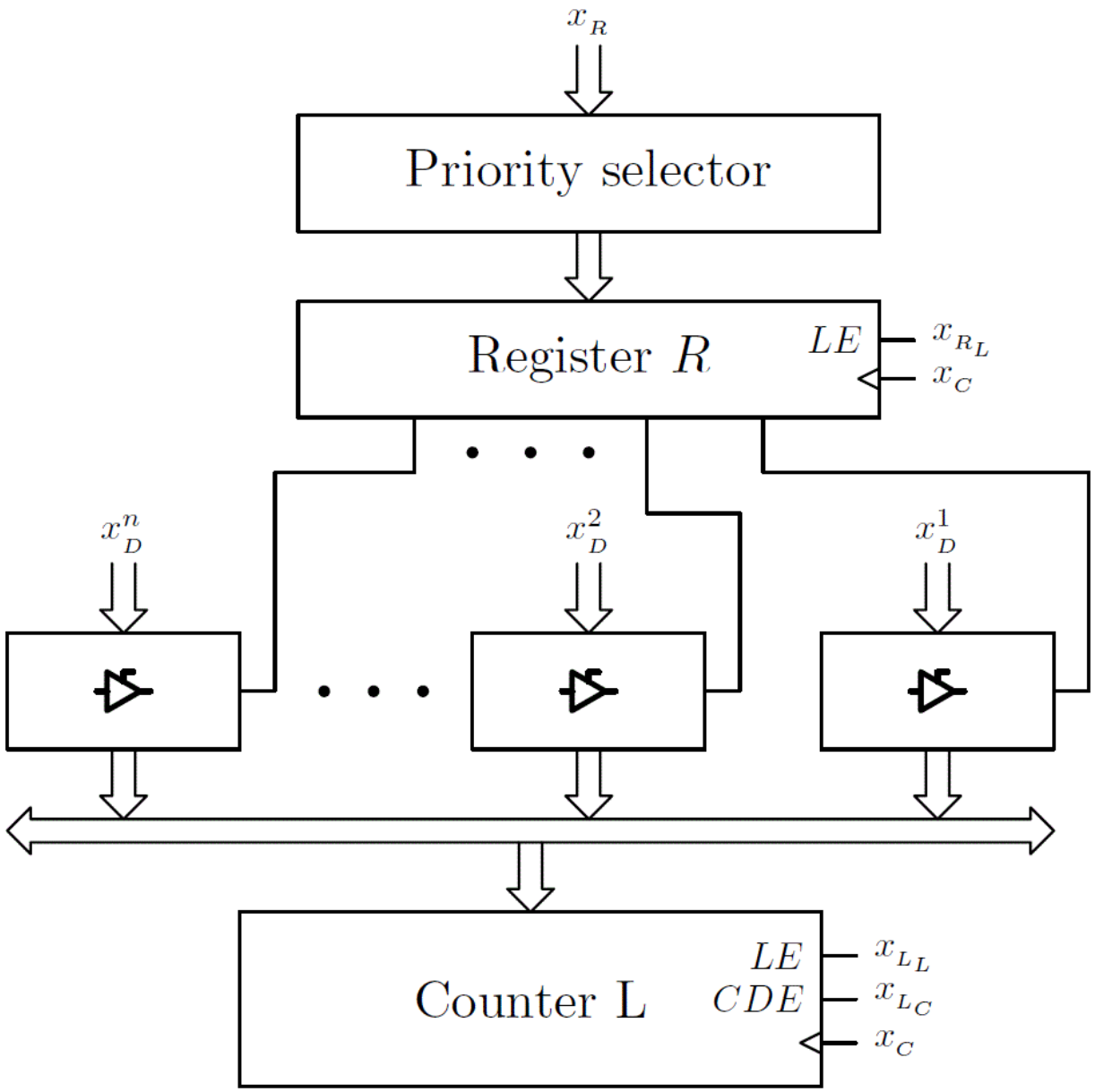
Data format of $x_D^i, \quad i = 1, \ldots, n, \quad n \leq 16$

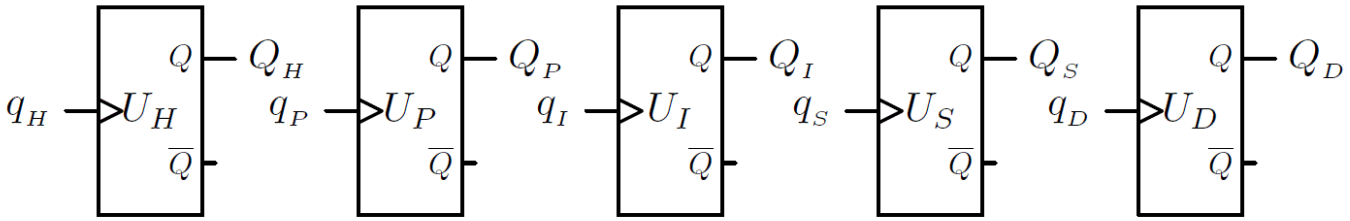



Parameters of the pulse trains



- Sources produce data infrequently and independently of each other
- Each source has a priority associated with it (the priorities of each two sources differ)
- The pulse generator transmits data in accordance with the source priorities
- When $y_I = 1$ sources are not allowed to broadcast new data to the pulse generator. The pulse generator has to take into account that there might be a small delay of $t < T_S$ before the sources will react to the inhibition.
- The pulse generator converts data $x^i_D$, treated as binary numbers, into pulse trains, where the number of pulses is equal to $x^i_D$.
- The data acquisition station, which acts as the receiver of the data, distinguishes the consecutive numbers by detecting a long pause between the pulse trains.


$x_R$
Priority selector
Register $R$
$LE$
$x_{R_L}$
$x_C$
$x_D^n$
$x_D^2$
$x_D^1$
Counter L
$LE$
$CDE$
$x_{L_L}$
$x_{L_C}$
$x_C$




$$Q_H = 1 \text{ for } T_H, \text{ triggered by } q_H = 0 \nearrow 1$$
$$Q_P = 1 \text{ for } T_P, \text{ triggered by } q_P = 0 \nearrow 1$$
$$Q_I = 1 \text{ for } T_I, \text{ triggered by } q_I = 0 \nearrow 1$$
$$Q_S = 1 \text{ for } T_S, \text{ triggered by } q_S = 0 \nearrow 1$$
$$Q_D = 1 \text{ for } T_D, \text{ triggered by } q_D = 0 \nearrow 1$$

$$T_S = 0.9T_I \qquad T_D = 2T_P \qquad T_H, T_P, T_I, T_S, T_D \gg T_{x_C}$$

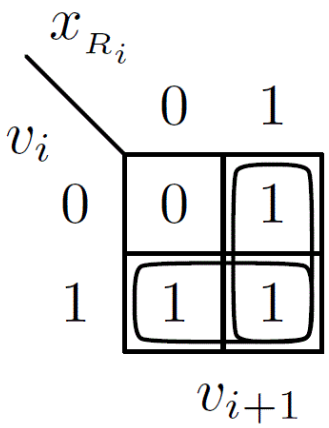
# Priority selector

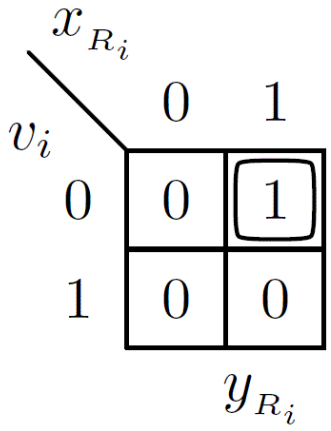
Highest priority line: $i = 1$

$x_{R_i}$ – data source service request line $i = 1, \ldots, n$

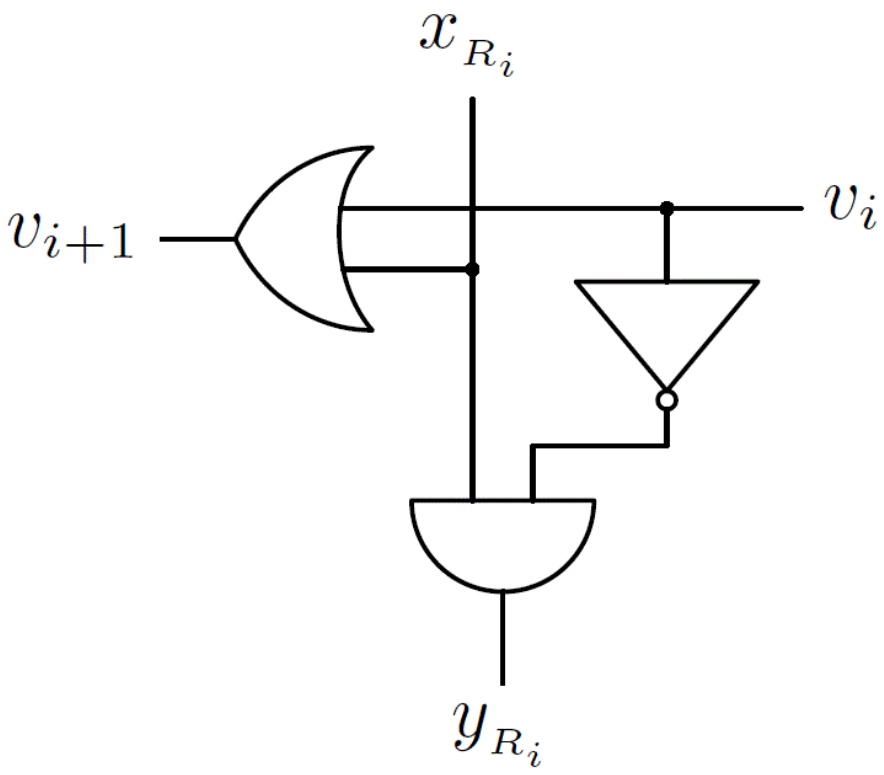
$y_R$ – priority selector output (either: 0 or *1 out of n* code)

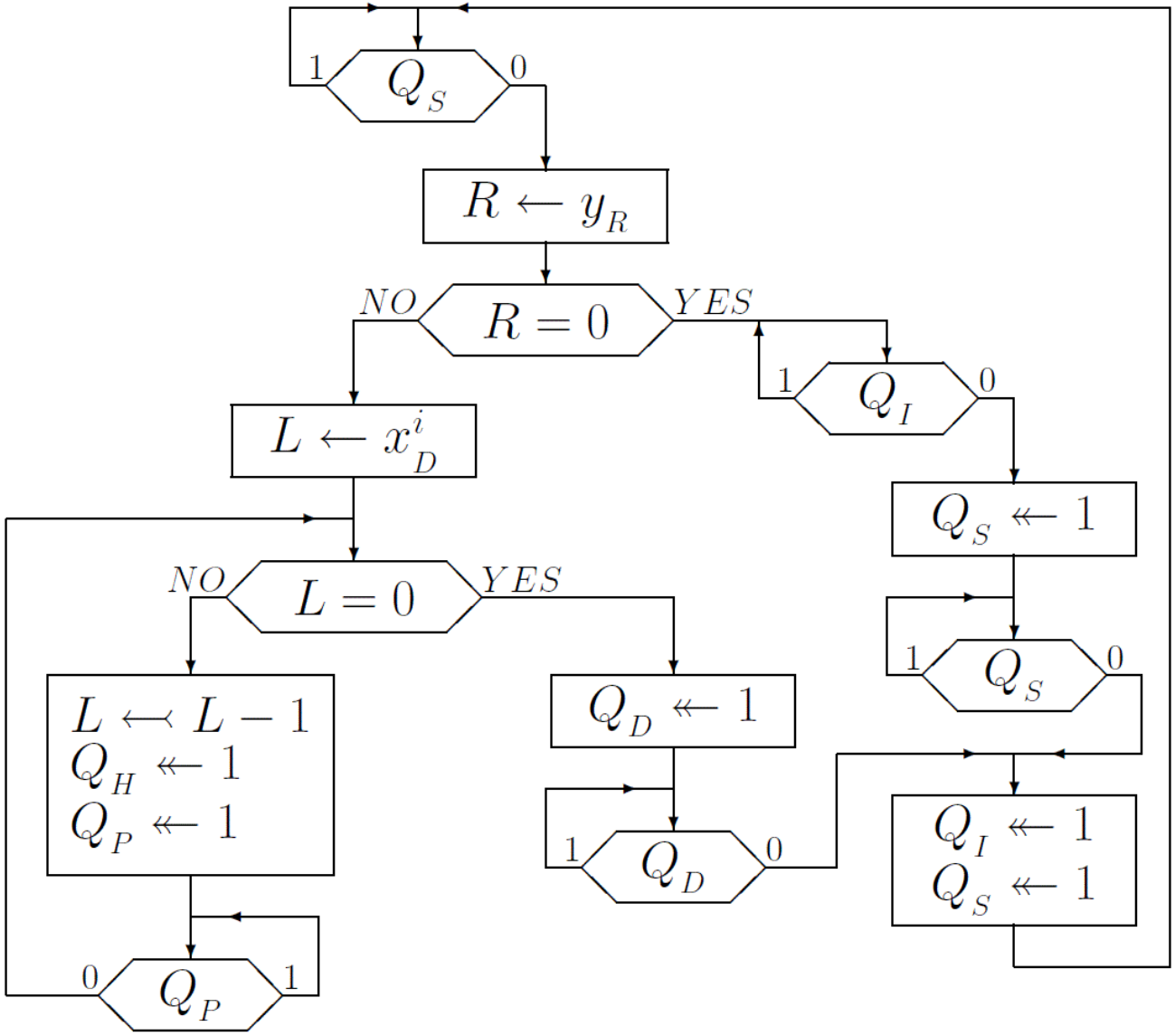
$$v_i = \begin{cases} 0 & - \text{ no higher priority request as yet} \\ 1 & - \text{ there was a higher priority request} \end{cases}$$



$$\begin{cases} v_{i+1} = v_i + x_{R_i} \\ y_{R_i} = \overline{v}_i x_{R_i} \end{cases}$$




1
$Q_S$
0
$R \leftarrow y_R$
NO
$R = 0$
YES
1
$Q_I$
0
$L \leftarrow x_D^i$
$Q_S \Leftarrow 1$
NO
$L = 0$
YES
1
$Q_S$
0
$L \leftarrow L - 1$
$Q_H \Leftarrow 1$
$Q_P \Leftarrow 1$
$Q_D \Leftarrow 1$
1
$Q_D$
0
$Q_I \Leftarrow 1$
$Q_S \Leftarrow 1$
0
$Q_P$
1

# Data throughput analysis

If the pulse train frequency is $50$kHz, then $T_P = 20\mu$s.

Transmission of a $16$ bit word requires $2^{16} + 2$ pulses at the most ($2$ – inter-train pause).

The time needed to transmit data from 16 sources:

$16 \times (65536+2) \times 20\mu\text{s} = 20972160\mu\text{s} < 21s$ at the most.

$\Rightarrow$ Each source can transmit its measurements every 30s.

HUMAN CAPITAL
HUMAN – BEST INVESTMENT!

EUROPEAN UNION
EUROPEAN
SOCIAL FUND

Project is co-financed by European Union within European Social Fund