REPORT			PLEASE SPECIFY GROUP NO: ZICSS1- 2 3 1 1							
NAME	SURNAME	Temat ćwiczenia zgodny z wykazem tematów:	PLEASE SPECIFY THE DATE OF THE CLASS BELOW:				ROK: 2025r.			
Volha	Hryshkevich	LOGICAL FUNCTORS ATANUA.EXE basics	PN	WT	SR	CZ	PT	SB	ND	
			CLASS START TIME:				___ : ___			
UWAGA !!! Wypełniamy tylko białe pola. W **punkcie 1**, proszę zakreślić odpowiednie pola i podać godzinę w której odbywają się zajęcia, zgodnie z planem zajęć.										

Theoretical Introduction:

COMBINATION SYSTEMS (please write a few words about combination systems
UKŁADY KOMBINACYJNE (proszę napisać kilka słów na temat układów kombinacyjnych)

A **combination system**, more precisely called a **combinational system**, is one of the most fundamental concepts in digital electronics and control engineering. It is a digital system in which the **outputs depend solely on the current input values**. This means that, unlike systems with memory, a combinational system's response is determined entirely by the present state of its inputs and not by any previous conditions or histories.

In mathematical terms, a combinational system can be described as a function:

$$Y = f(A_1, A_2, A_3, \ldots, A_n)$$

where (A_1, A_2, …, A_n) represent the input variables, (Y) is the output or group of outputs, and (f) is a **Boolean function** that defines how the outputs are related to the inputs.

Combinational systems operate without memory elements such as flip-flops or registers. Therefore, they are also referred to as **memoryless systems**. They do not store any information, and their outputs change instantaneously (except for small propagation delays) when any of the inputs change. This makes them extremely fast, predictable, and easy to design, which is why they are widely used in both digital logic circuits and industrial control systems.

Characteristics of Combinational Systems

Combinational systems have several distinctive features that separate them from sequential systems:

1. **No memory:** The outputs depend only on the current inputs, not on previous states.
2. **Instantaneous operation:** Output signals respond as soon as input signals change.
3. **Deterministic behavior:** For every combination of input values, there is one unique output combination.

4. **Boolean description:** The entire system can be expressed using Boolean algebra with logical functors such as AND, OR, and NOT.
   For example, a simple two-input AND gate follows the rule:

$$Y = A \wedge B$$

   This means that the output (Y) will only be true (1) when both (A) and (B) are true.

Basic Example

A simple illustration of a combinational system is a **two-input AND gate**. It has two inputs (A and B) and one output (Y). The output is true only if both inputs are true. The corresponding truth table is as follows:

A	B	Y = A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

This truth table shows that the AND gate implements a logical condition where **all** required inputs must be true to produce a true output. This type of logic is fundamental in safety systems, interlocks, and automatic control decision-making.

Combinational vs. Sequential Systems

It is important to distinguish combinational systems from sequential systems, since both are used in digital control:

Feature	Combinational system	Sequential system
Memory	None	Has memory elements
Output depends on	Present inputs only	Present inputs and past states
Typical components	Logic gates (AND, OR, NOT)	Flip-flops, latches, counters
Representation	Boolean expressions, truth tables	State diagrams, timing charts
Example	Adders, decoders, logic interlocks	Timers, registers, controllers

Thus, combinational systems form the “instant decision-making” part of a digital system, while sequential logic manages timing, counting, and control sequences.

Practical Examples in Industry

Combinational logic is applied extensively in **industrial process control**, often in conjunction with sensors and actuators. Common examples include:

1. **Safety Interlocks:**
   A machine may only start if all safety conditions are met, such as guards closed, pressure within range, and emergency stop not active.
   $$Start = SafetyOK \wedge PressureOK \wedge \neg EmergencyStop$$
2. **Alarm Systems:**
   Alarms can be activated when any of several dangerous conditions occur.
   $$Alarm = HighPressure \vee HighTemperature \vee LowLevel$$
3. **Permission (Permissive) Logic:**
   Pumps, motors, or valves may only operate if a set of permissive signals is true.
   $$PumpEnable = LevelOK \wedge ValveOpen \wedge \neg Trip$$
4. **Code Converters:**
   Circuits that convert binary codes (like Binary-Coded Decimal to 7-segment display) are also combinational systems.
5. **Arithmetic Circuits:**
   Adders, subtractors, and comparators — all of which operate without storing previous data — are typical examples of combinational logic.

Representation of Combinational Systems

There are several methods for representing and analyzing combinational systems:

1. **Truth Tables:**
   These list all possible input combinations and the corresponding outputs.
2. **Boolean Equations:**
   Logical relationships expressed using Boolean algebra (AND, OR, NOT, etc.).
3. **Logic Diagrams:**
   Graphical representations using standard logic gate symbols.
4. **Karnaugh Maps (K-Maps):**
   Visual tools used to simplify complex Boolean expressions.
5. **Hardware Description Languages (HDL):**
   Modern implementations often use languages like VHDL, Verilog, or PLC Structured Text to describe combinational systems.

Logical Functors in Combinational Logic

The fundamental logical functors are the building blocks of combinational systems:

- **NOT (¬A)** — Inverts the input. If A = 1, then ¬A = 0.
- **AND (A ∧ B)** — Produces 1 only if all inputs are 1.

- **OR (A ∨ B)** — Produces 1 if any input is 1.
- **NAND (¬ (A ∧ B))** — The inverse of AND; output is 0 only when all inputs are 1.
- **NOR (¬ (A ∨ B))** — The inverse of OR; output is 1 only when all inputs are 0.
- **XOR (A ⊕ B)** — Output is 1 when inputs are different.
- **XNOR (¬ (A ⊕ B))** — Output is 1 when inputs are equal.

Every complex digital function can be constructed using these functors. In fact, **NAND** and **NOR** are considered *universal gates* because any logical operation can be implemented using only one of them.

For instance:

$$A \wedge B = \neg(\neg(A \wedge B)) = NAND(NAND(A, B), NAND(A, B))$$

This ability makes NAND and NOR very important for hardware design and optimization.

Implementation in Industrial Systems

In industrial control systems, combinational logic is often implemented in **Programmable Logic Controllers (PLCs)**. PLCs allow engineers to express Boolean relationships using languages such as:

- **Ladder Logic:**
  Series (AND) and parallel (OR) contacts represent logical relationships.
- **Structured Text:**
  Similar to programming languages:
- PumpEnable := (PressureOK AND LevelOK) AND NOT EmergencyStop;
- **Function Block Diagram (FBD):**
  Visual blocks for AND, OR, NOT, XOR, etc.

In hardware, combinational systems may be built from:

- **Integrated circuits (ICs)** with logic gates,
- **FPGAs** for high-speed industrial computation,
- **Relays or safety relays**, where wiring itself forms the combinational logic.

Example: Industrial Boiler Control

Consider a boiler system where the heater should only turn on if:

- The water level is high enough,
- The temperature is below a safe limit,
- The emergency stop is not activated.

The logic for this operation is:

$$HeaterOn = LevelOK \wedge TempOK \wedge \neg EmergencyStop$$

If any of these conditions become false, the heater immediately turns off.

This is a classic example of a **combinational control decision** in an industrial process — fast, reliable, and directly dependent on sensor inputs.

Advantages of Combinational Systems

1. **Simplicity:**
   Direct and transparent relationship between inputs and outputs.
2. **High Speed:**
   Since there are no feedback elements, propagation delays are minimal.
3. **Deterministic Operation:**
   The same input always produces the same output, ensuring predictability.
4. **Ease of Testing and Simulation:**
   Truth tables and Boolean equations make validation straightforward.
5. **Reliability:**
   Because there is no memory or timing dependence, fewer failure modes exist.

Limitations

However, combinational systems also have their drawbacks:

1. **No Memory:**
   They cannot remember past events or states; hence not suitable for sequential operations or timing control.
2. **Transient Errors (Glitches):**
   Due to propagation delays, temporary incorrect outputs can appear during input transitions.
3. **Complexity Growth:**
   For a large number of inputs, truth tables and Boolean simplifications become complicated.
4. **Limited Functionality Alone:**
   For more advanced control (e.g., sequences, delays, counting), combinational logic must be combined with sequential elements.

Role of Combinational Systems in Digital Control

In modern industrial automation, combinational logic serves as the **core of decision-making** within larger digital control systems. These systems continuously monitor inputs from sensors — such as temperature, pressure, level, and flow — and evaluate logical conditions to determine which actuators should operate.

For example, in a chemical mixing process, a valve may open only if:

- The tank is not full,
- The mixer is running,
- The material feed is available.

This can be expressed as:

$$ValveOpen = \neg TankFull \land MixerRunning \land MaterialAvailable$$

Such Boolean expressions form the logical foundation of **digital control systems for industrial processes**. They are implemented using PLCs or microcontrollers that constantly evaluate these logical conditions in real-time.

Conclusion

A **combinational system** is therefore the essential building block of all digital and industrial control logic. Its outputs depend solely on current input conditions, described through Boolean functions made up of logical functors such as AND, OR, and NOT. While simple in structure, combinational systems perform the critical “decision-making” tasks that keep industrial processes safe, efficient, and automated.

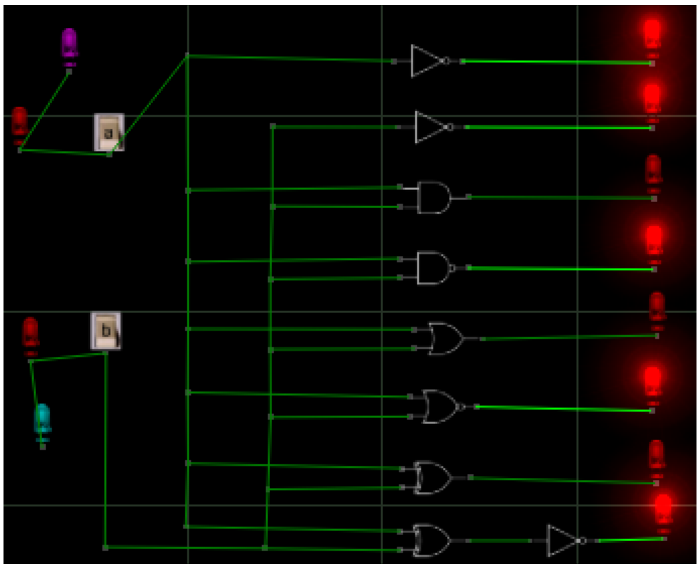
Although they lack memory and timing capabilities, their speed, predictability, and simplicity make them invaluable in applications ranging from safety interlocks to arithmetic circuits and control logic in PLCs. In practical digital control systems, combinational logic is often combined with sequential logic to achieve complete, reliable, and intelligent industrial automation.

Task 1: Zadanie 1

In the program ATANUA. EXE set the logical functors according to the diagram and then fill in all the truth tables:

W programie ATANUA. EXE zestaw funktory logiczne wg. schematu a następnie wypełnij wszystkie tablice prawdy:

Table 1 Basic logical functors. Podstawowe funktory logiczne

Insert a screenshot of the designed layout here Tutaj proszę wstawić zrzut ekranu z zaprojektowanym układem		INPUT		OUTPUT
		a	**b**	**Y**
	**NOT**	**1**	**x**	0
		0	**x**	1
	NOT	**x**	**1**	0
		x	**0**	1
	AND	**1**	**1**	1
		1	**0**	0
		0	**1**	0
		0	**0**	0
	NAND	**1**	**1**	0
		1	**0**	1
		0	**1**	1
		0	**0**	1
	OR	**1**	**1**	1
		1	**0**	1
		0	**1**	1
		0	**0**	0
	NOR	**1**	**1**	0
		1	**0**	0
		0	**1**	0
		0	**0**	1
	EX-OR	**1**	**1**	0
		1	**0**	1
		0	**1**	1
		0	**0**	0
	EX-NOR	**1**	**1**	1
		1	**0**	0
		0	**1**	0
		0	**0**	1

Task 2.

Please find the appropriate ICs in the ATANUA program and simulate their operation using all the logic gates in them. Then, in the tables, include screen shots of the simulations carried out in the ATANUA program and, using the Internet, make their data sheets as done in Table 2. Please remember the circuit symbols, e.g.: ULN 7400 and truth tables and present graphics with the distribution of gates in them. All information can be found in datasheets found on the Internet. NOTE: There is no EX-NOR (XNOR) circuit in the ATANUA program, so use the EX-OR (XOR) circuit and negate the inputs with the NOT logic functor. Proszę w programie ATANUA znaleźć odpowiednie układy scalone oraz przeprowadzić symulację ich działania wykorzystując wszystkie bramki logiczne w nich zawarte. Następnie w tabelach należy zamieścić zrzuty ekranu z symulacji przeprowadzonych w programie ATANUA a korzystając z Internetu wykonać ich karty katalogowe tak jak to zrobiono w Tabeli 2. Proszę pamiętać o symbolach układów np.: ULN 7400 oraz tablicach prawdy i przedstawieniu grafiki z rozkładem bramek w nich zawartych. Wszystkie informacje można znaleźć w kartach katalogowych znajdujących się w Internecie. UWAGA! W programie ATANUA nie ma układu EX-NOR (XNOR) dlatego należy wykorzystać układ EX-OR (XOR) i zanegować wejścia funktorem logicznym NOT.

Table 2 Functor NOT Example , Funktor NOT Przykład

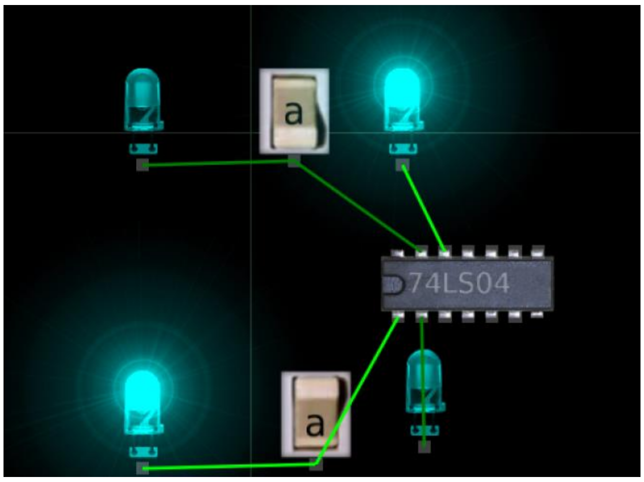
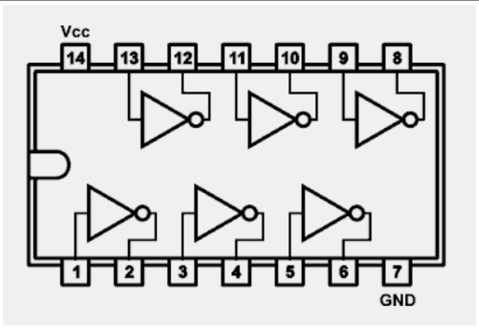
Tutaj proszę wstawić zrzut ekranu z zaprojektowanym układem	Wejście		Wyjście
	a	b	Y
	**1**	**x**	**0**
	0	**x**	**1**
	Nazwa układu scalonego		
	ULN 7404		
	 Vcc 14 13 12 11 10 9 8 1 2 3 4 5 6 7 GND		

Table 3 Logical functor AND, Funktor logiczny AND

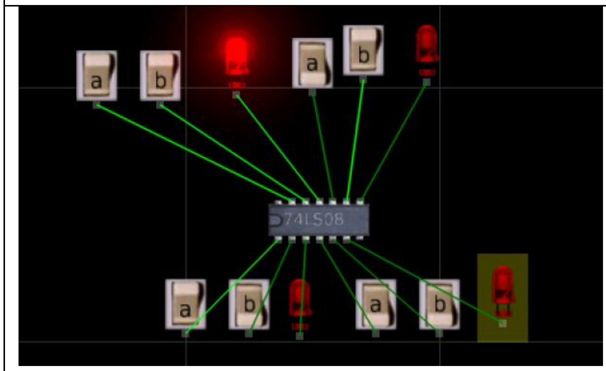
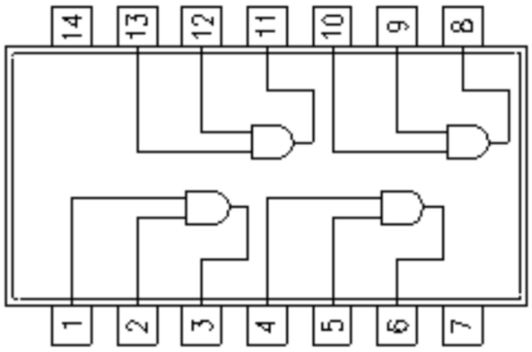
Tutaj proszę wstawić zrzut ekranu z zaprojektowanym układem	Wejście		Wyjście
	a	b	Y
	1	**1**	**1**
	1	**0**	**0**
	0	**1**	**0**
	0	**0**	**0**
	Nazwa układu scalonego		
	ULN 7408		
	 14 13 12 11 10 9 8 1 2 3 4 5 6 7		

Table 4 NAND logic function, Funktor logiczny NAND

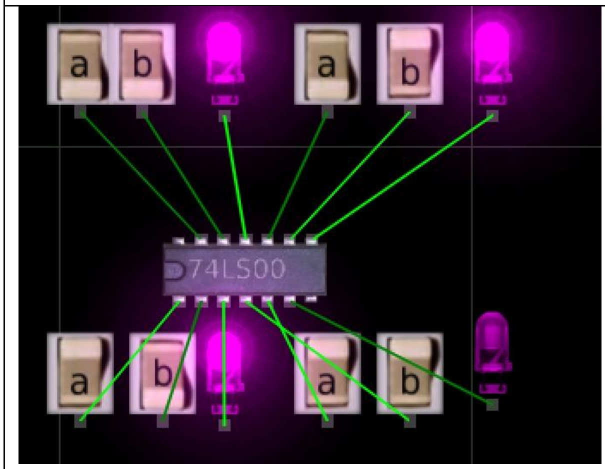
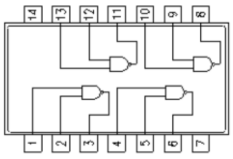
Tutaj proszę wstawić zrzut ekranu z zaprojektowanym układem	Wejście		Wyjście
	a	b	Y
	1	**1**	**0**
	1	**0**	**1**
	0	**1**	**1**
	0	**0**	**1**
	Nazwa układu scalonego		
	7400		
	 14 13 12 11 10 9 8 1 2 3 4 5 6 7		

Table 5 OR logical functor, Funktor logiczny OR

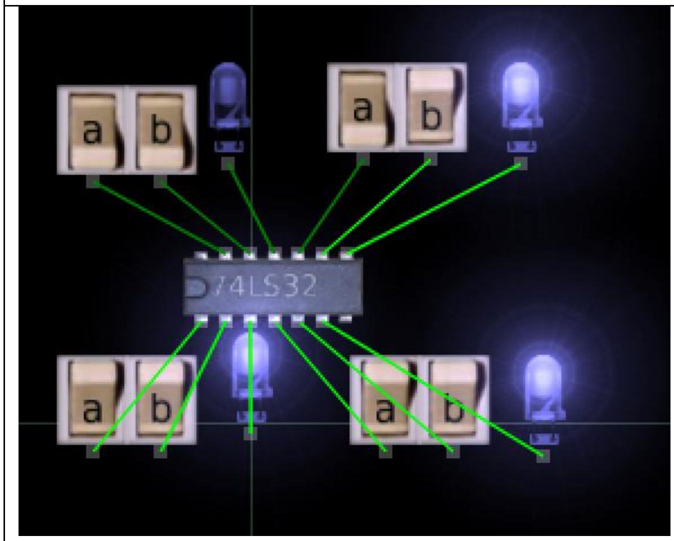
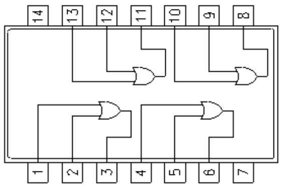
Tutaj proszę wstawić zrzut ekranu z zaprojektowanym układem	Wejście		Wyjście
	a	b	Y
	1	**1**	**1**
	1	**0**	**1**
	0	**1**	**1**
	0	**0**	**0**
	Nazwa układu scalonego		
	7432		
	 14 13 12 11 10 9 8 1 2 3 4 5 6 7		

Table 6 NOR logical functor, Funktor logiczny NOR

Tutaj proszę wstawić zrzut ekranu z zaprojektowanym układem	Wejście		Wyjście
	a	b	Y
	1	**1**	**0**
	1	**0**	**0**
	0	**1**	**0**
	0	**0**	**1**
	Nazwa układu scalonego		
	7402		

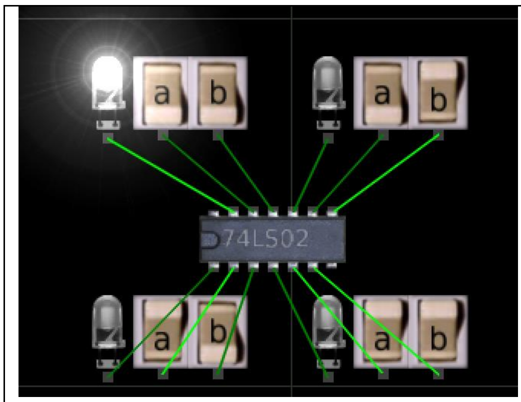
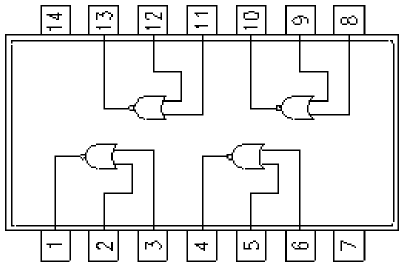
	

Tabela 7 EX-OR (XOR) logic functor, Funktor logiczny EX-OR (XOR)

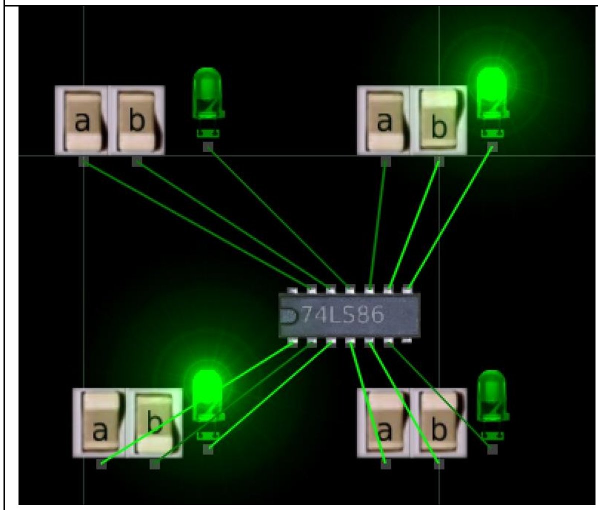
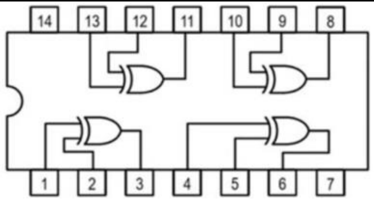
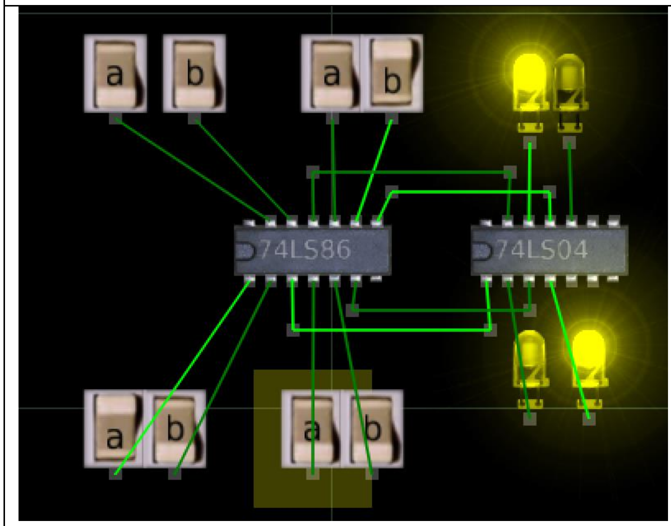
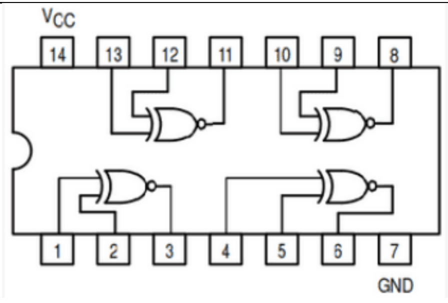
Tutaj proszę wstawić zrzut ekranu z zaprojektowanym układem	Wejście		Wyjście
	a	b	Y
	1	**1**	**0**
	1	**0**	**1**
	0	**1**	**1**
	0	**0**	**0**
	Nazwa układu scalonego		
	7486		
			

Table 8 EX-NOR (XNOR) logic function. Funktor logiczny EX-NOR (XNOR)

Tutaj proszę wstawić zrzut ekranu z zaprojektowanym układem	Wejście		Wyjście
	a	b	Y
	1	**1**	**1**
	1	**0**	**0**
	0	**1**	**0**
	0	**0**	**1**
	Nazwa układu scalonego		
	74266		
	 VCC 14 13 12 11 10 9 8 1 2 3 4 5 6 7 GND		