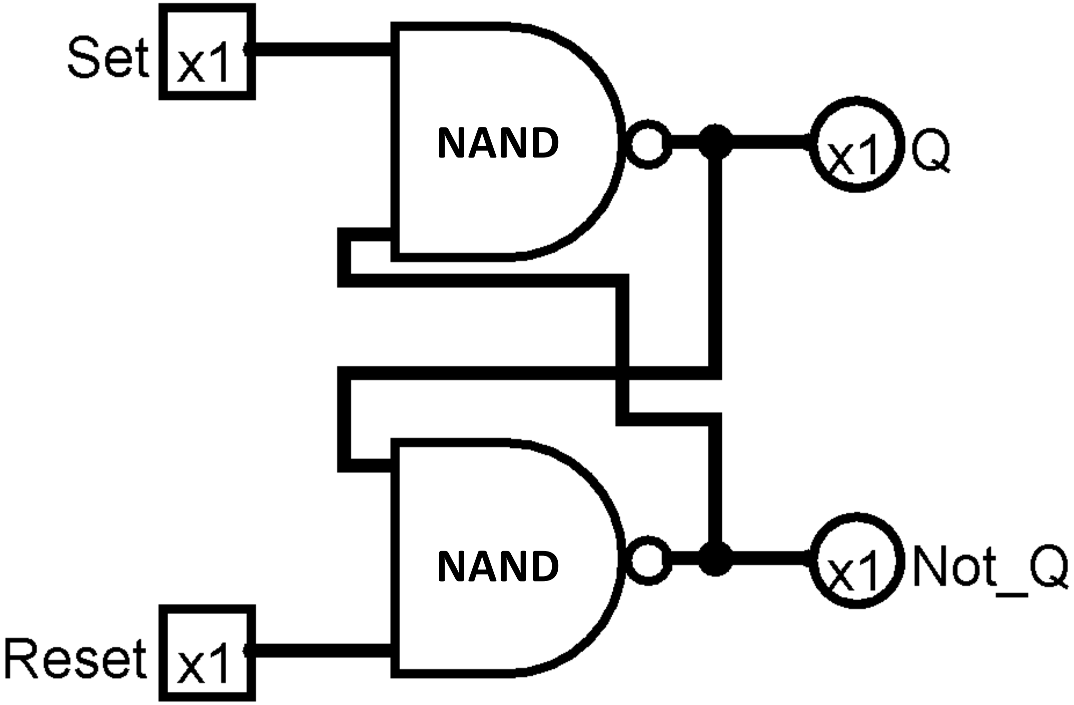
Digital systems and basics of electronics - SYC

10 Sequential Circuits - introduction

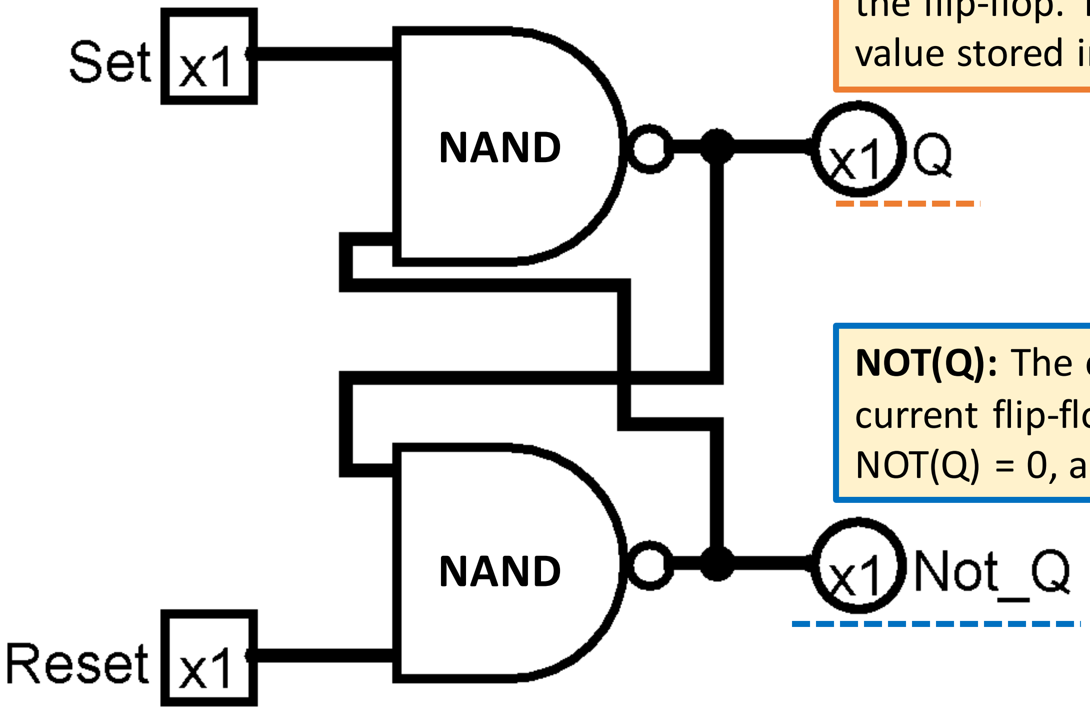
Flip-flops and “memory effect”

- **Flip-Flops and Digital Memory:** Flip-flops are fundamental building blocks in digital electronics, used **to store a single bit of data**. They play a key role in creating memory circuits.
- **Memory Effect:** Flip-flops exhibit a "memory effect," meaning they retain their state (0 or 1) until explicitly changed by an input signal, making them ideal for sequential logic and state retention.
- **Types and Applications:** Common types include **SR, D, JK, and T flip-flops**, each designed for specific use cases like counters, data storage, and synchronization in digital systems.

RS Flip-Flop NAND



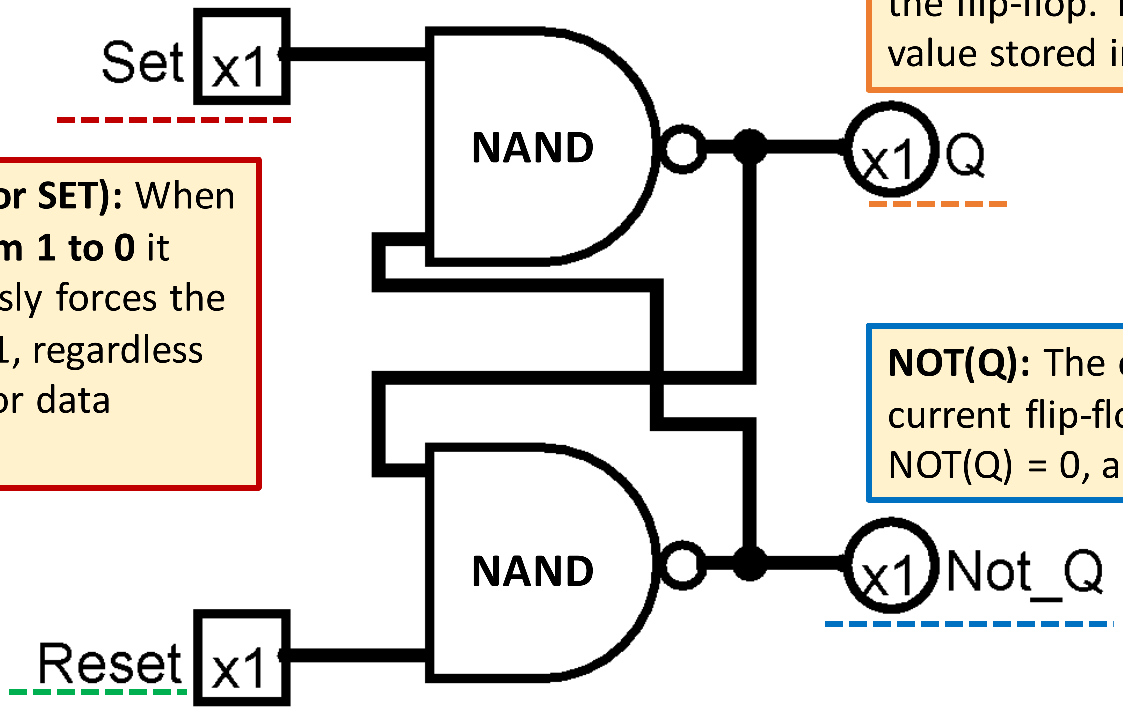
RS Flip-Flop NAND



Q: The current output state of the flip-flop. This reflects the value stored in the flip-flop.

NOT(Q): The complement of the current flip-flop state. If Q = 1, NOT(Q) = 0, and vice versa.

RS Flip-Flop NAND



Preset (PRE or SET): When Set is set **from 1 to 0** it asynchronously forces the Q output to 1, regardless of the clock or data inputs.

Clear (CLR or RESET): When Reset is set **from 1 to 0**, it asynchronously forces the Q output to 0, regardless of the clock or data inputs.

Q: The current output state of the flip-flop. This reflects the value stored in the flip-flop.

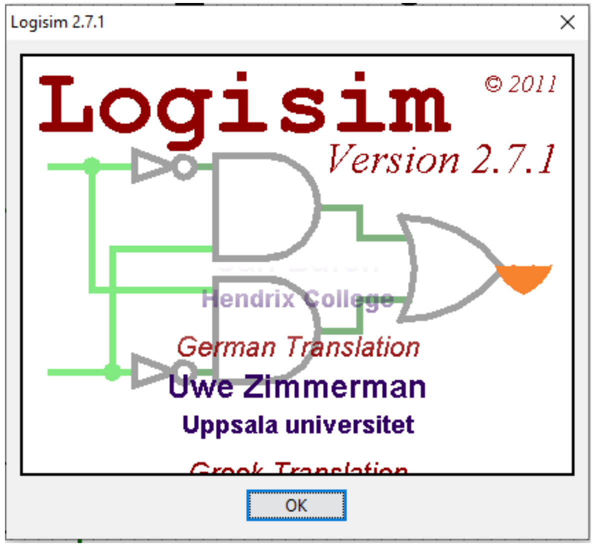
NOT(Q): The complement of the current flip-flop state. If Q = 1, NOT(Q) = 0, and vice versa.

S_t	R_t	Q_{t+1}
0	0	*forbidden*
0	1	1
1	0	0
1	1	Q_t (*previously*)

Start logisim

Icon

Version



Logisim 2.7.1

Logisim

© 2011

Version 2.7.1

Hendrix College

German Translation

Uwe Zimmerman

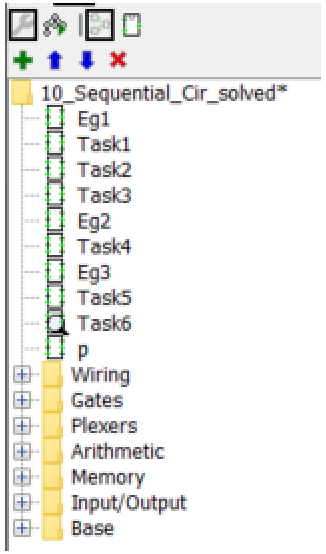
Uppsala universitet

Greek Translation

Open :

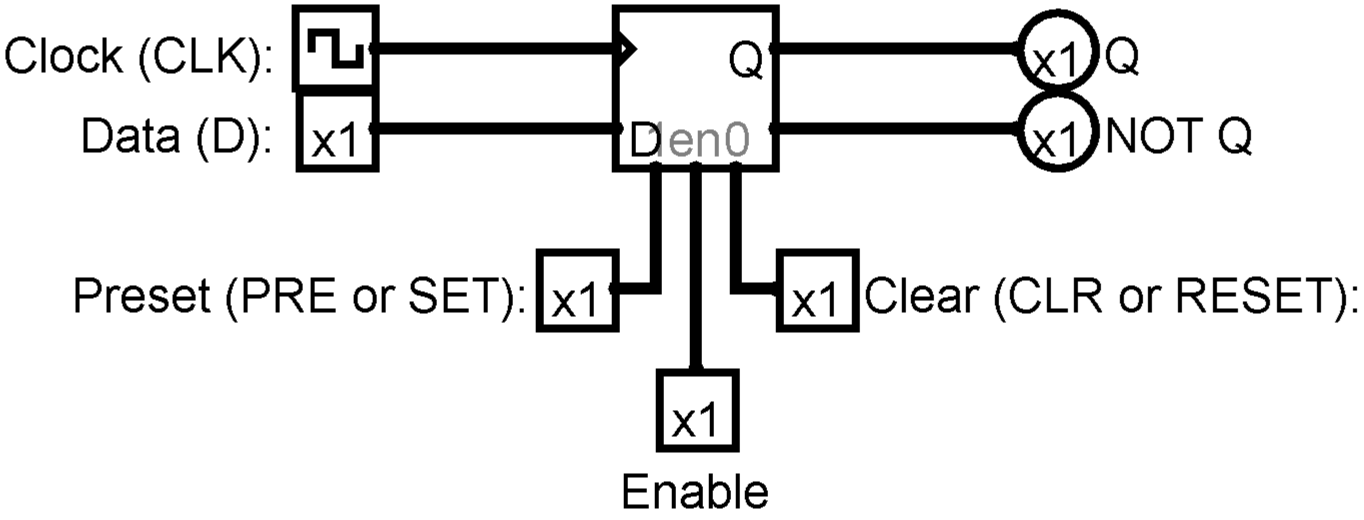
mmajew/SYC/10/**10_template_Seq.circ**

➔ **eg1**



- 10_Sequential_Cir_solved*
  - Eg1
  - Task1
  - Task2
  - Task3
  - Eg2
  - Task4
  - Eg3
  - Task5
  - Task6
  - p
- Wiring
- Gates
- Plexers
- Arithmetic
- Memory
- Input/Output
- Base

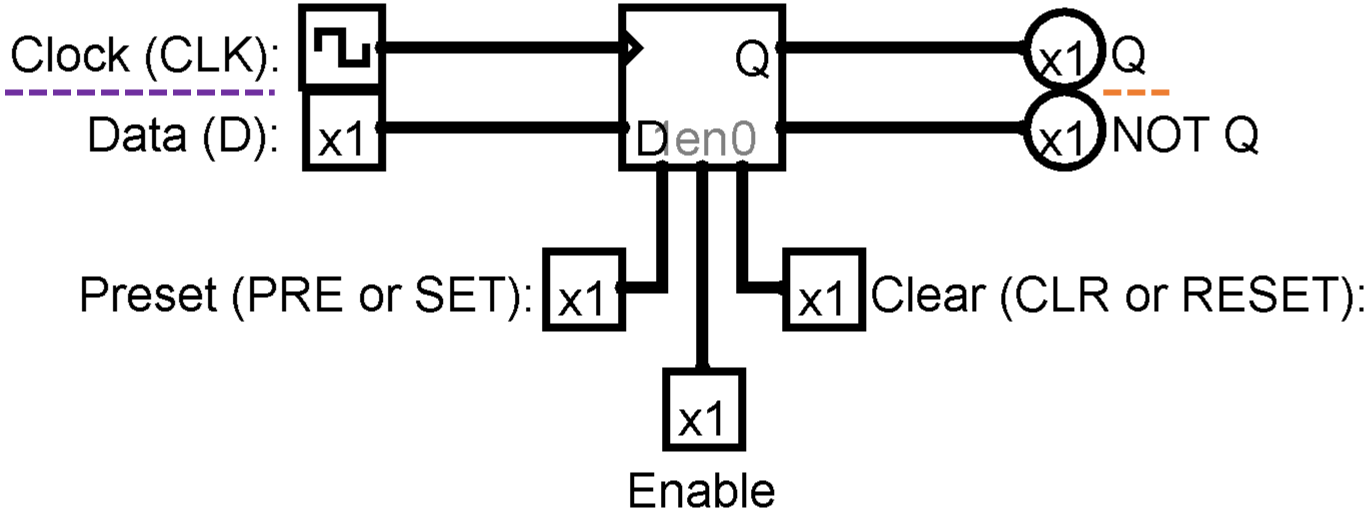
D Flip-flop


Clock (CLK):
Data (D):
x1
D
1
en
0
Q
x1
Q
x1
NOT Q
Preset (PRE or SET):
x1
x1
Clear (CLR or RESET):
x1
Enable

D Flip-flop

Clock (CLK): The **flip-flop updates** its **state on the rising (from 0 to 1)** edge of the **clock signal**, depending on the design. The clock is the primary control signal for synchronous operation.

Q: The current output state of the flip-flop. This reflects the value stored in the flip-flop.

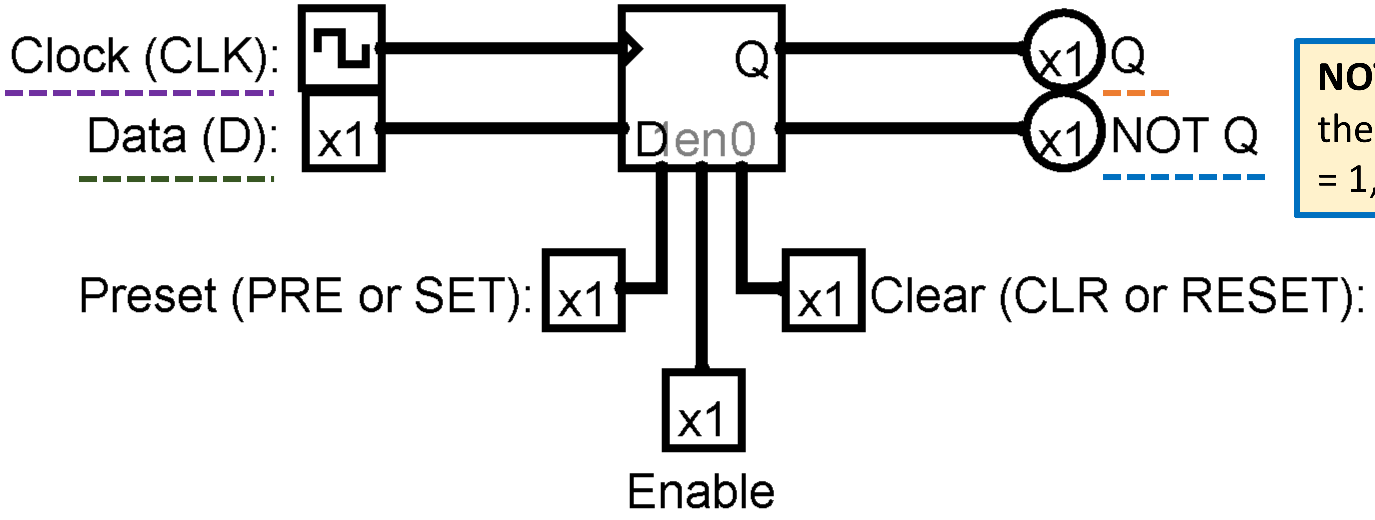


D Flip-flop

Clock (CLK): The **flip-flop updates** its **state on the rising (from 0 to 1)** edge of the **clock signal**, depending on the design. The clock is the primary control signal for synchronous operation.

Q: The current output state of the flip-flop. This reflects the value stored in the flip-flop.

Data (D): This is the input data pin. The state of this pin (0 or 1) is sampled and stored in the flip-flop on the triggering edge of the clock.



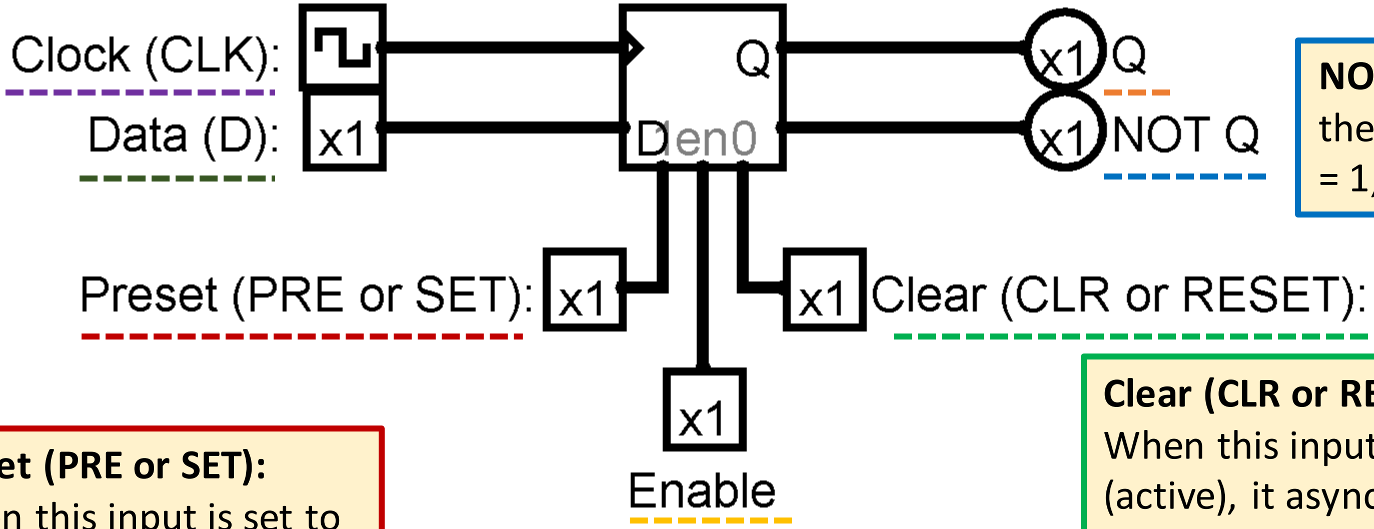
NOT(Q): The complement of the current flip-flop state. If Q = 1, NOT(Q) = 0, and vice versa.

D Flip-flop

Clock (CLK): The **flip-flop updates** its **state on the rising (from 0 to 1)** edge of the **clock signal**, depending on the design. The clock is the primary control signal for synchronous operation.

Q: The current output state of the flip-flop. This reflects the value stored in the flip-flop.

Data (D): This is the input data pin. The state of this pin (0 or 1) is sampled and stored in the flip-flop on the triggering edge of the clock.



NOT(Q): The complement of the current flip-flop state. If Q = 1, NOT(Q) = 0, and vice versa.

Preset (PRE or SET): When this input is set to 1 (active), it asynchronously forces the Q output to 1, regardless of the clock or data inputs.

Enable (EN): When this input is 0, the clock is ignored, and the flip-flop does not respond to clock edges. When EN = 1, the flip-flop operates normally, responding to the clock.

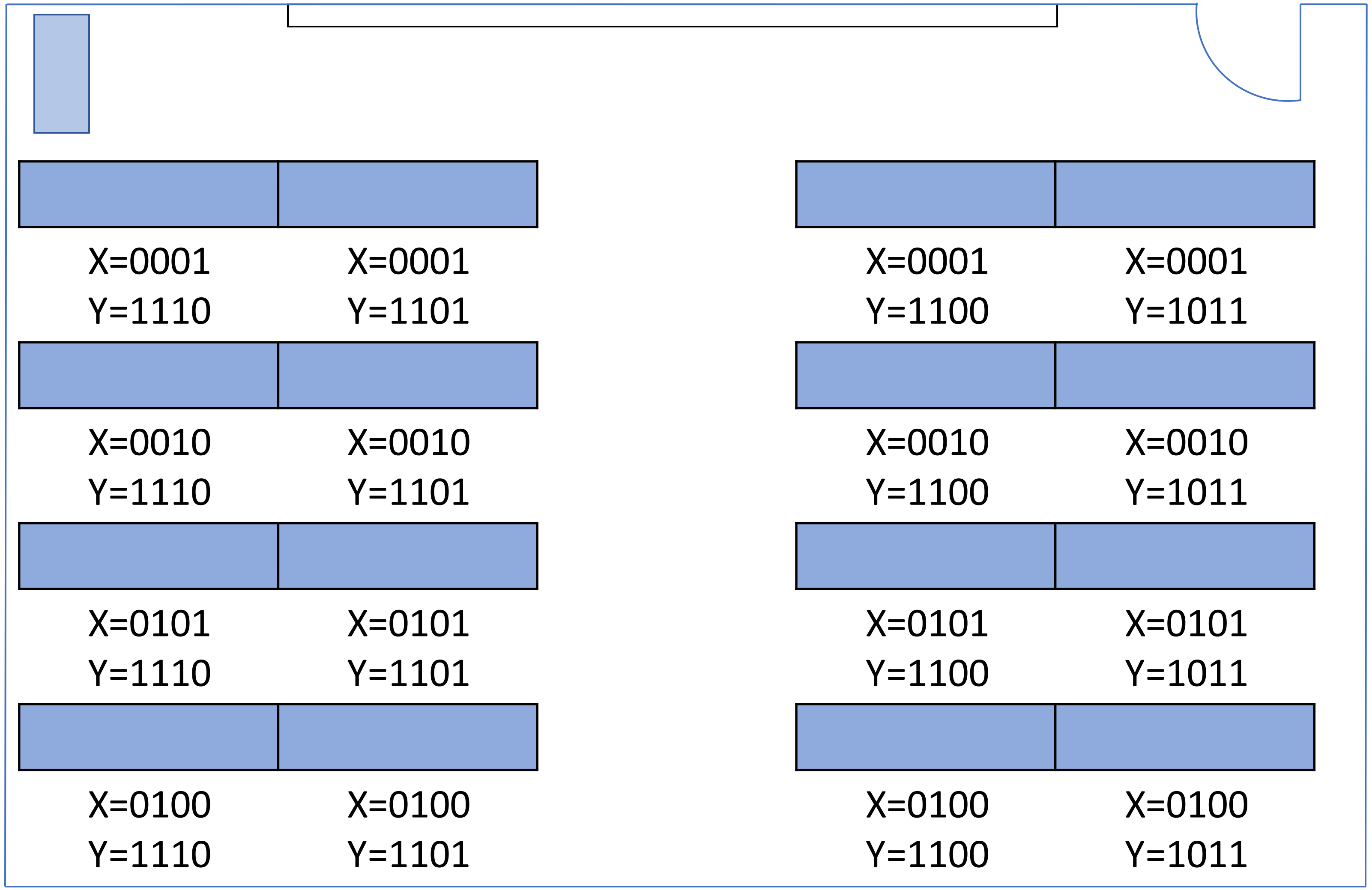
Clear (CLR or RESET): When this input is set to 1 (active), it asynchronously forces the Q output to 0, regardless of the clock or data inputs.

Tasks description

1. **4-Bit Shift Register:** Using D-type flip-flops, design a 4-bit shift register with the following sequence: 0000 → 1000 → 1100 → 1110 → 1111
   1 point
2. **Ring Counter:** Build a ring counter using the shift register from Task 1. The bits should circulate through the connected stages in a circular fashion
   0.5 point
3. **Johnson Counter:** Design a Johnson counter (also known as a twisted ring counter) using the shift register from Task 1. The counter should follow the sequence: 0000 → 0001 → 0011 → 0111 → 1111 → 1110 → 1100 → 1000 → 0000
   0.5 point
4. **Pseudo-Random Binary Sequence (PRBS) Generator:** Create a PRBS generator by using the shift register from Task 1 along with an XOR logic gate. Ensure the sequence generated is a maximal-length pseudo-random binary sequence.
   0.5 point
5. **Mod-9 Counter:** Using D-type flip-flops, design a counter that cycles through a modulus of 9, with states ranging from `0000` to `1000`.
   0.5 point
6. **Custom Counter (X to Y):** Using D-type flip-flops, build a counter that starts at `X` and ends at `Y`, cycling through the sequence. The values of X and Y are assigned to each student according to the scheme on the next slide.
   2 point

Task 6. Custom Counter (X to Y)

The values of X and Y in task 6 are assigned according to your place in the classroom.



X=0001
Y=1110
	X=0001
	Y=1101
		X=0001
		Y=1100
			X=0001
			Y=1011

X=0010
Y=1110
	X=0010
	Y=1101
		X=0010
		Y=1100
			X=0010
			Y=1011

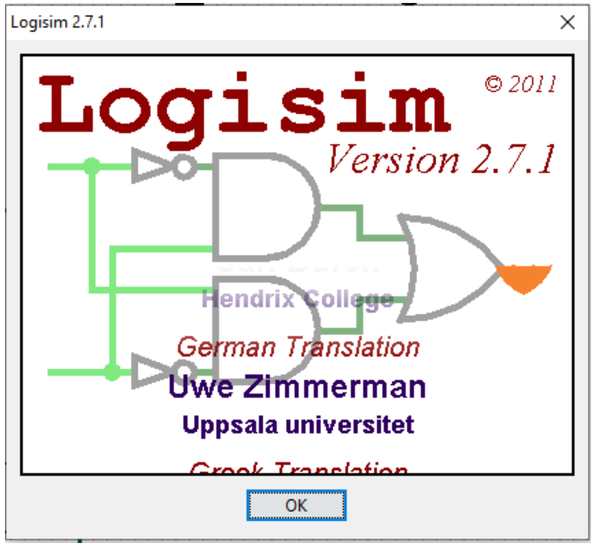
X=0101
Y=1110
	X=0101
	Y=1101
		X=0101
		Y=1100
			X=0101
			Y=1011

X=0100
Y=1110
	X=0100
	Y=1101
		X=0100
		Y=1100
			X=0100
			Y=1011

Start logisim

Icon

Version



Logisim 2.7.1

Logisim

© 2011

Version 2.7.1

Hendrix College

German Translation

Uwe Zimmerman

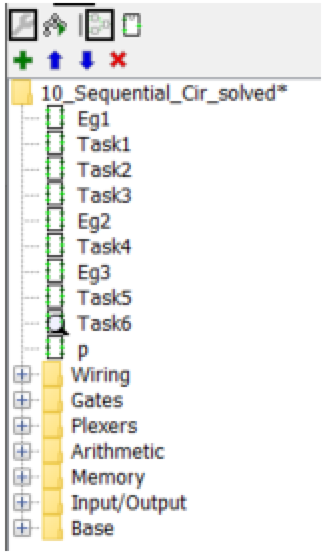
Uppsala universitet

Greek Translation

Open :

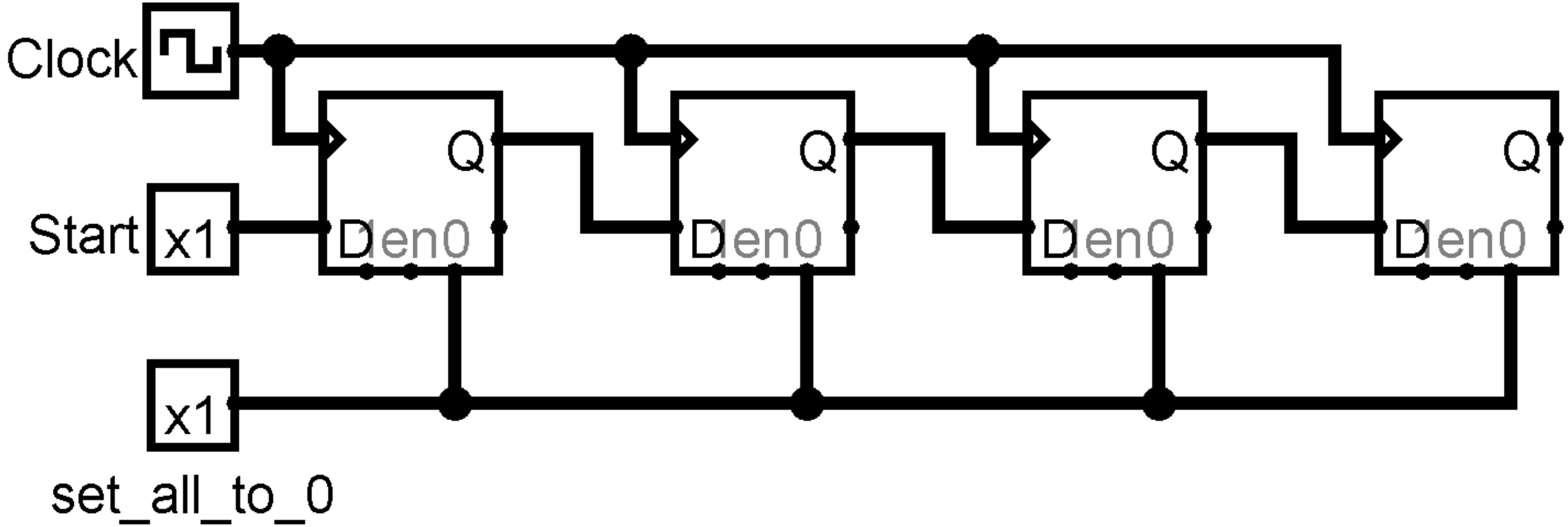
mmajew/SYC/10/**10_template_Seq.circ**

- → **task_1**
- → **task_2**
- → **task_3**
- → **task_4**
- → **task_5**
- → **task_6**

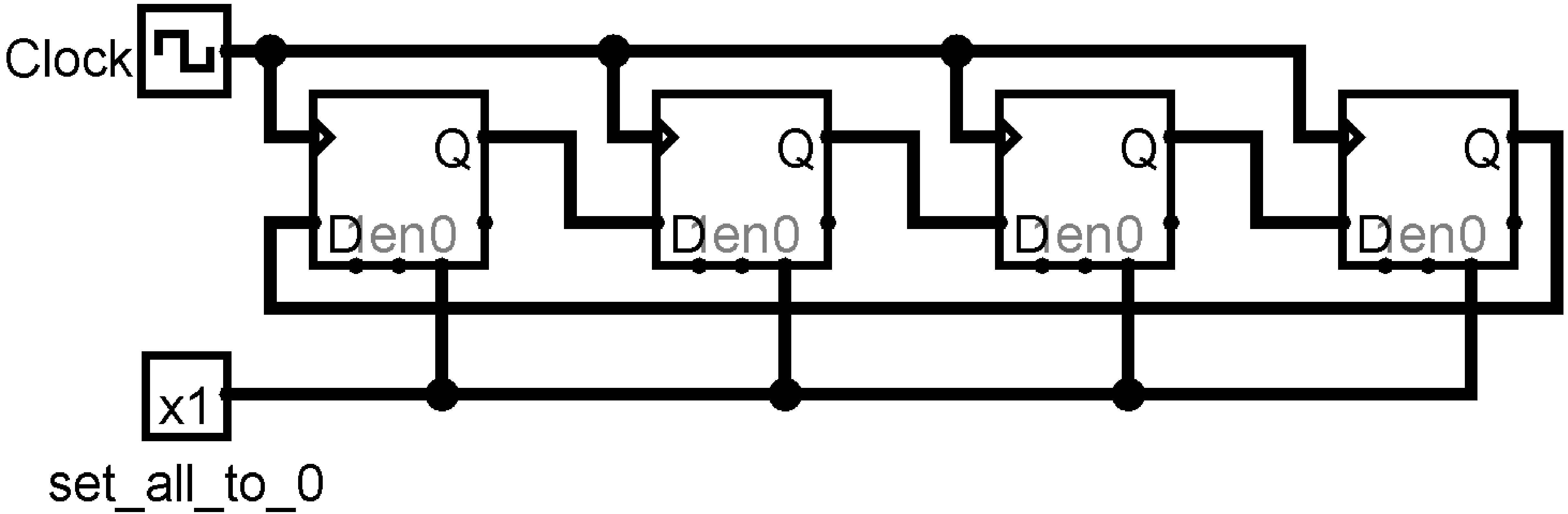


10_Sequential_Cir_solved*
Eg1
Task1
Task2
Task3
Eg2
Task4
Eg3
Task5
Task6
p
Wiring
Gates
Plexers
Arithmetic
Memory
Input/Output
Base

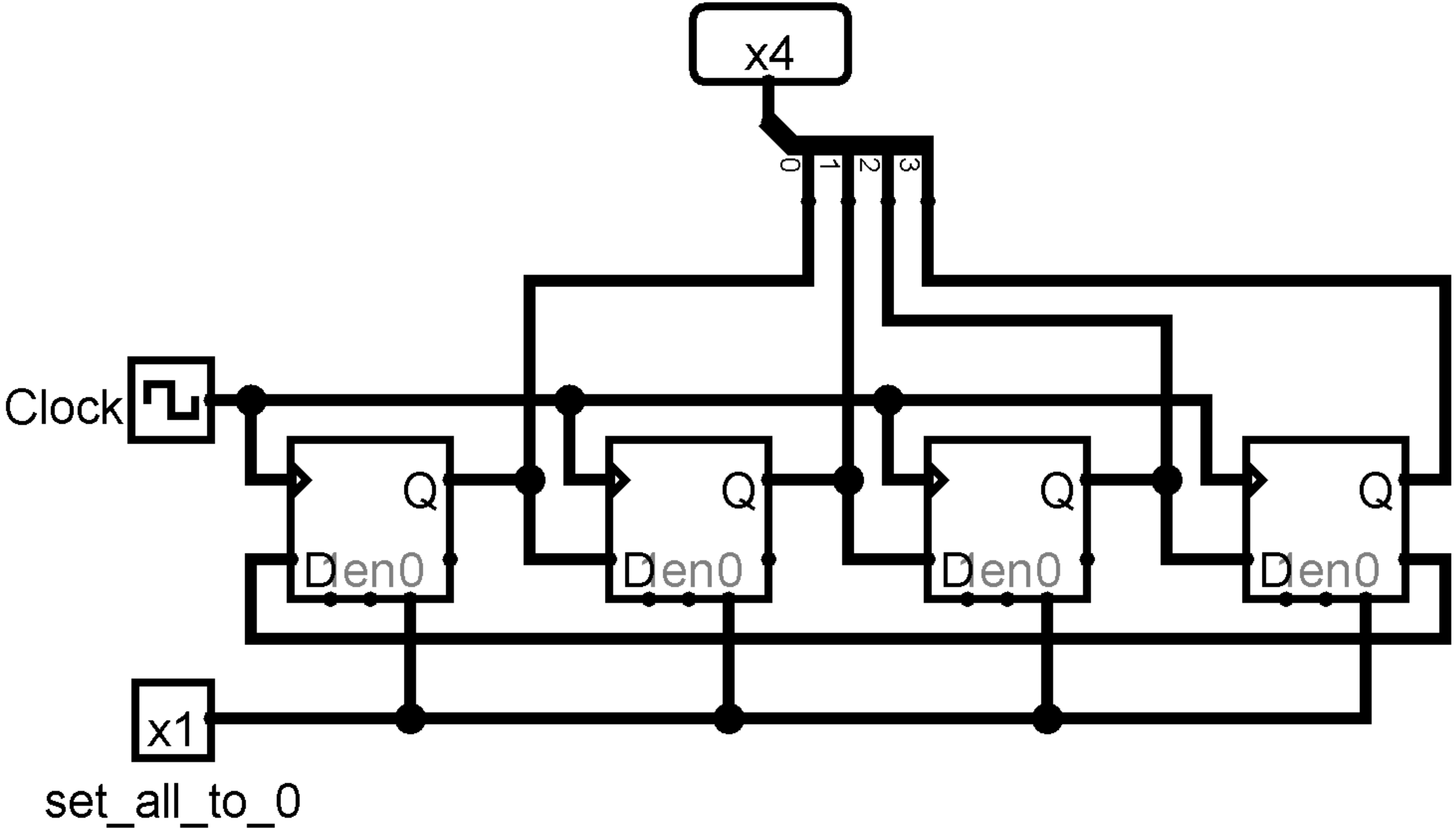
Task 1 Using D-type flip-flops build 4-bit shift register: 0000,1000,1100,1110,1111.



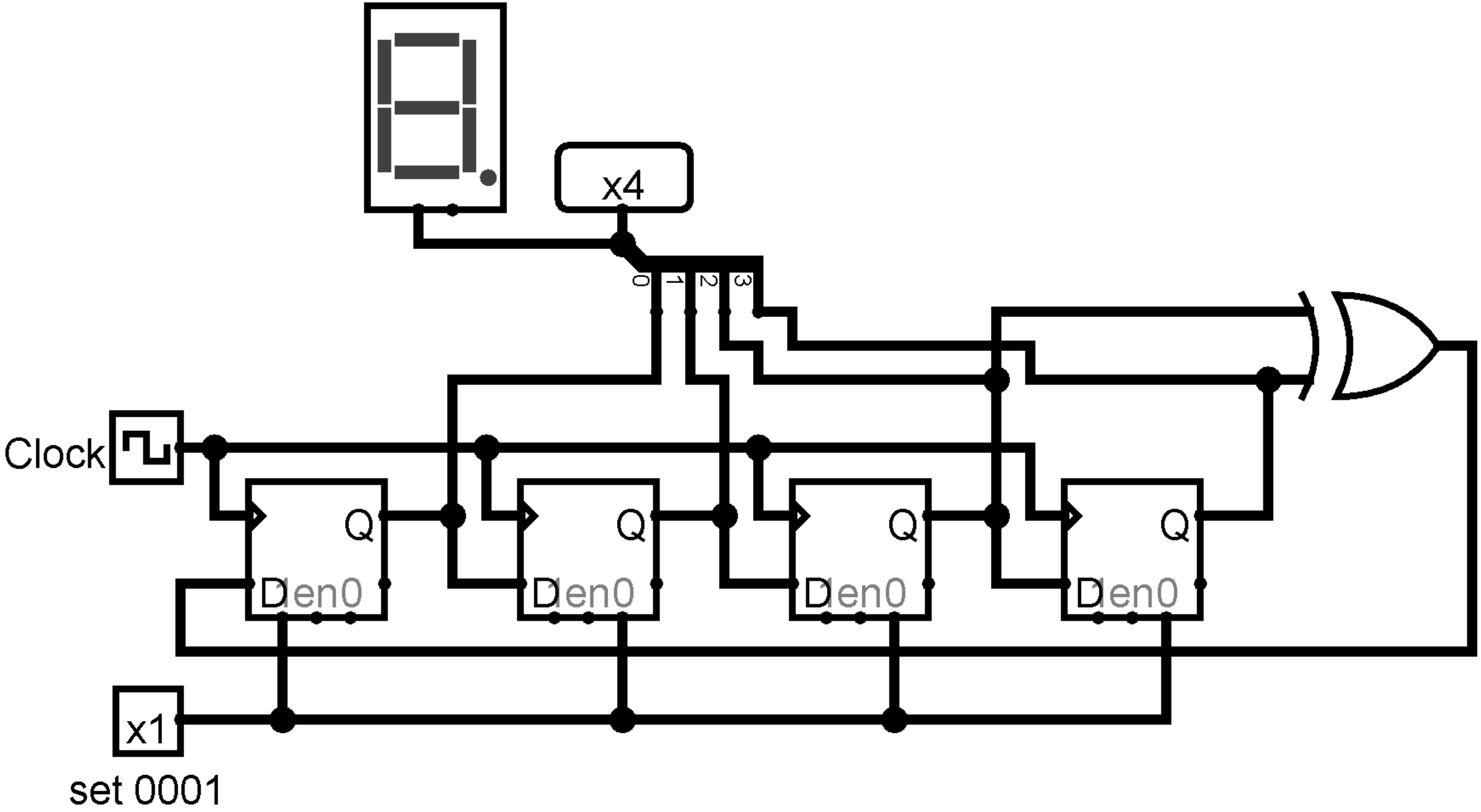
Task 2 Using shift register from point 1. build ring counter:
the bits circulate through the connected stages in a circular fashion.



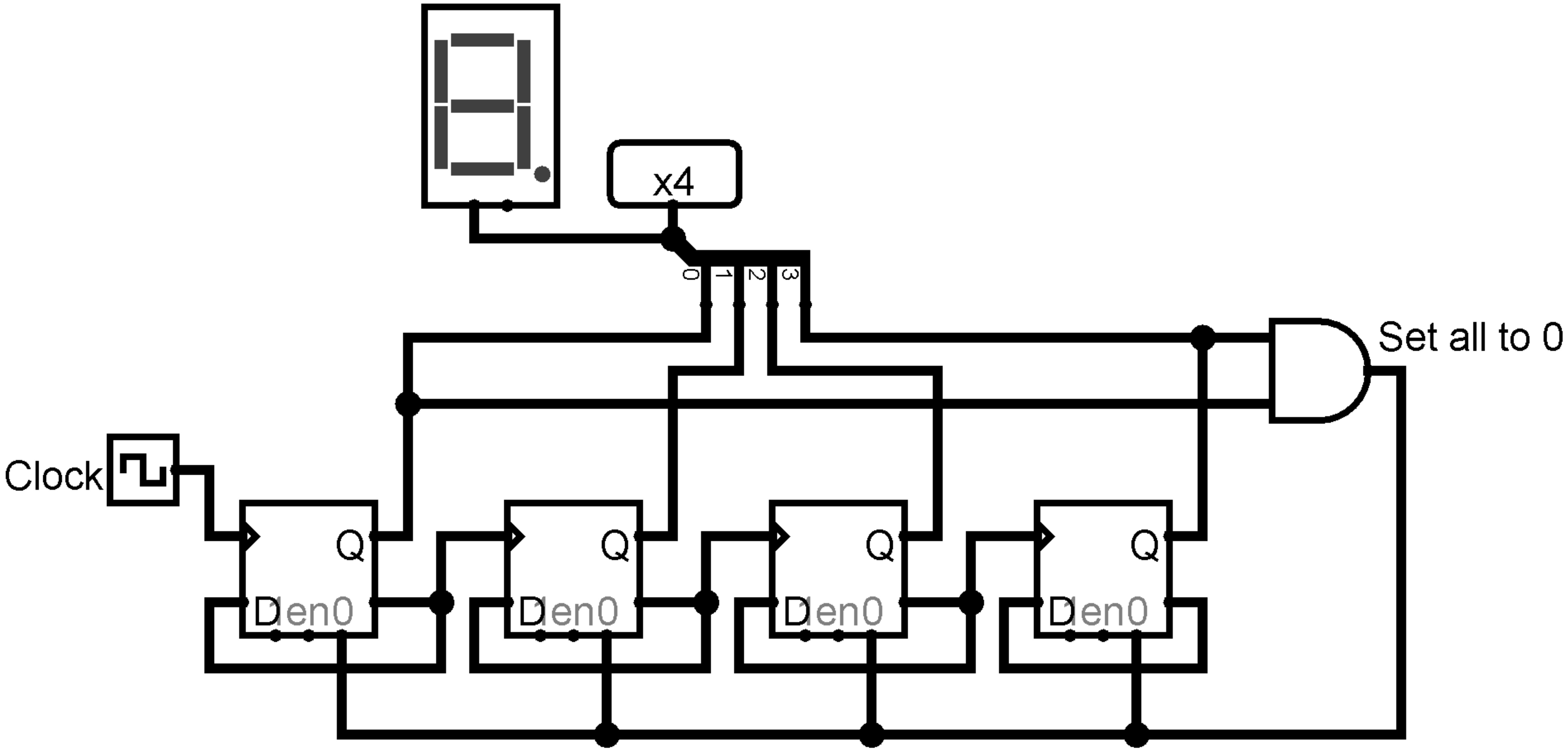
Task 3 Using shift register from point 1. build Johnson counter:
0000,0001,0011,0111,1111,1110,1100,1000,0000



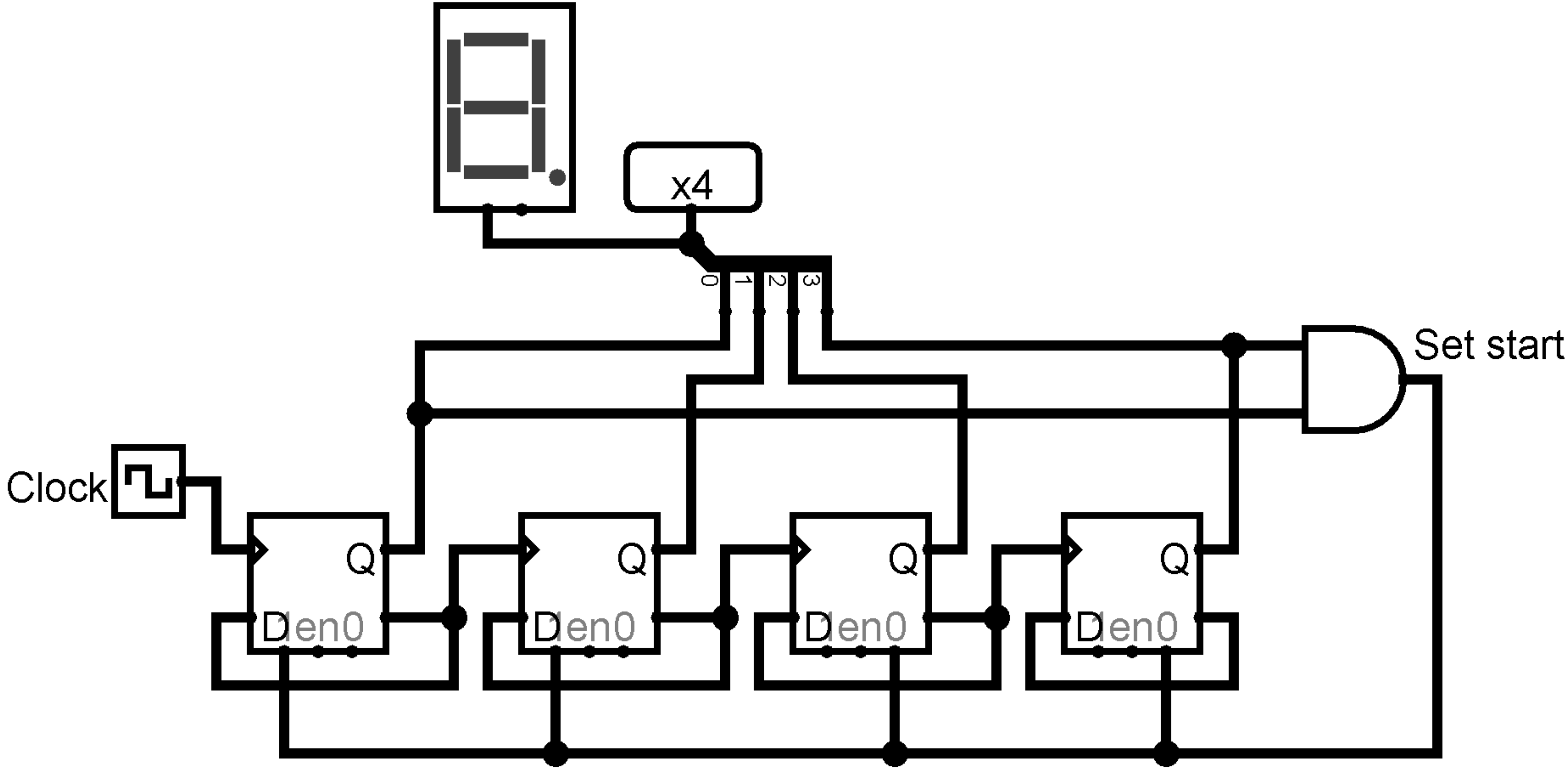
Task 4 Using shift register from point 1. and XOR logic gate make PRBS Pseudo Random Binary Signal generator



Task 5 Using D-type flip-flops build mod9 counter (from 0000 to 1000)



Task 6 Using D-type flip-flops build counter from 0011 to 1000


x4
0
1
2
3
Set start
Clock
Q
D
en
0
Q
D
en
0
Q
D
en
0
Q
D
en
0