ЦИФРОВА СХЕМОТЕХНІКА

ЗАНЯТТЯ 9. МАШИНИ СТАНІВ. СКІНЧЕННІ АВТОМАТИ МУРА І МІЛЛІ.

МАШИНИ СТАНІВ

Скінченний автомат (Finite State Machine, FSM) — це математична модель обчислювальної системи, яка може перебувати в одному з обмеженої кількості дискретних станів та змінювати свій стан у відповідь на вхідні сигнали.

1. Роль у цифровій схемотехніці
Скінченні автомати є основою для проєктування цифрових схем, оскільки вони дозволяють:
- реалізовувати керуючі алгоритми в цифрових пристроях;
- описувати поведінку схем у часі;
- оптимізувати логічні схеми для мінімізації апаратних ресурсів.

2. Скінченні автомати застосовуються у:
- **процесорах та мікроконтролерах** – для керування виконанням команд.
- **цифрових пристроях управління** – наприклад, в контролерах пам'яті, комунікаційних протоколах (SPI, I^2C, UART).
- **автоматах обробки сигналів** – у схемах кодування, декодування, стиску даних.
- **програмованій логіці (FPGA, CPLD)** – для реалізації складних керуючих алгоритмів.
- **системах автоматизації та робототехніці** – для управління послідовністю дій.

МАШИНИ СТАНІВ

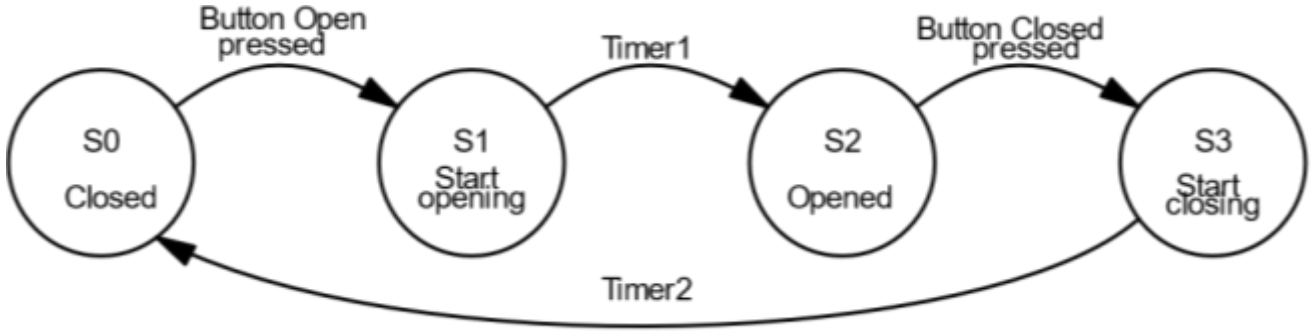
3. Основні переваги використання скінченних автоматів

- **Формальність опису** – чітка структура та математична модель.

- **Простота реалізації** – можуть бути реалізовані як апаратно (на основі тригерів і логічних елементів), так і програмно (алгоритми в мікроконтролерах).

- **Можливість оптимізації** – мінімізація кількості станів та апаратних ресурсів.

МАШИНИ СТАНІВ



Стани (Алфавіт станів, States)

Стан – це конкретний момент роботи автомата, що визначає його поведінку.

- Автомат може перебувати лише в одному стані за раз.
- Кожен стан може мати певні характеристики або вихідні сигнали.
- Визначення мінімальної кількості станів допомагає оптимізувати роботу автомата.

Приклад станів МС керування дверима ліфта:
S0 – Двері закриті
S1 – Двері відкриваються
S2 – Двері відкриті
S3 – Двері закриваються

Переходи (Transitions)
Перехід – це зміна стану автомата під впливом вхідних сигналів.

- Перехід визначається умовами (вхідними сигналами).
- Кожен перехід має визначений напрямок (від початкового стану до наступного).
- Визначається логічними рівняннями або таблицею переходів.

Приклад переходів для ліфта:
- якщо двері закриті (S0) і надійшла команда "відкрити" → переходимо в стан **S1**

- якщо двері відкриті (S2) і таймер спрацював → переходимо в стан **S3**

МАШИНИ СТАНІВ

Вхідні сигнали (Алфавіт вхідних значень, Input signals, Inputs X) – сигнали, що надходять до автомата і визначають його перехід між станами.

- Можуть надходити від користувача, сенсорів або зовнішніх систем.
- Використовуються у схемах на тригерах для зміни станів.

Приклади вхідних сигналів:
- натискання кнопки виклику ліфта
- датчик наявності пасажира
- часовий імпульс (таймер)

Вихідні сигнали (Алфавіт вихідних значень, Output signals)
Це сигнали, що формуються автоматом і можуть впливати на зовнішні пристрої.

- В автоматах **Мура** вихідні сигнали залежать тільки від стану.
- В автоматах **Міллі** вихідні сигнали залежать від стану та вхідних сигналів.

Приклади вихідних сигналів:
- включення двигуна для відкриття дверей
- зміна кольору світлодіода
- відправка сигналу на дисплей

ЦИФРОВІ АВТОМАТИ

Математичною моделлю ЦА (а в загальному випадку будь-якого дискретного пристрою) є так званий абстрактний автомат, сукупністю 6 елементів $A=(S, X, Y, \delta, \lambda, s_1)$

$S= \{s_1, s_2,\dots s_k\}$ - алфавіт станів - множина станів, в яких може знаходитися цифровий автомат

$X= \{x_1, x_2, \dots, x_n\}$ - алфавіт вхідних значень - множина значень, які можуть поступати на вхід ЦА

$Y= \{y_1, y_2,\dots y_m\}$ - алфавіт вихідних значень - множина значень, які можуть бути встановлені на виході ЦА.

$\delta: S\times X\rightarrow S$ - функція переходів $s(t+1)=((x(t), s(t))$. Функція переходів визначає, в який стан $s(t+1)$ перейде автомат під впливом вхідного сигналу $x(t)$, якщо у даний момент часу автомат знаходиться в $s(t)$.

$\lambda: S\times X\rightarrow S$ - функція виходів $y(t)= ((s(t), x(t))$. Функція виходів визначає яке вихідне значення $y(t)$ буде встановлено на виході автомата залежно від вхідного значення $x(t)$ і поточного стану $s(t)$.

$s_1 \in S$ - початковий стан автомата у якому він знаходиться у момент часу $t=0$, тобто стан в який встановлюється ЦА після подачі живлення або після скидання.

На практиці найбільшого поширення набули два класи автоматів – автомати **Міллі (Mealy)** і **Мура (Moore)**

ЦИФРОВІ АВТОМАТИ

Закон функціонування автомата **Міллі** задається рівняннями:

$$Q_{n+1} = f(Q_n(t), x(t)); y(t)) = f(Q_n(t), x(t)), \text{ де } t = 0,1,2,\ldots$$

Закон функціонування автомата **Мура** задається рівняннями:

$$Q_{n+1} = f(Q_n(t), x(t)); y(t)) = f(Q_n(t)), \text{ де } t = 0,1,2,\ldots$$

Q_{n+1}– подальший стан автомата, Q_n – початковий стан автомата, x(t) – вхідний сигнал, y(t) – вихідний сигнал.

Для того, щоб задати автомат, необхідно описати усі компоненти $A=(S, X, Y, \delta, \lambda, s_1)$

Множини S, X, Y описуються і задаються простим перерахуванням своїх елементів. Серед різноманіття різних способів завдань функцій δ і λ (отже і усього автомата в цілому) найбільшого поширення набули табличний і графічний.

При табличному способі опису цифрових автоматів застосовується два види таблиць - таблиця переходів і таблиця виходів.

АВТОМАТ МІЛЛІ

Рядки цих таблиць відповідають вхідним сигналам, а стовпці - станам, причому крайній лівий стовпець станів позначений початковим станом s_1

вхід \ стан	s_1	…	s_m
X_1	$\delta(s_1, x_1)$	…	$\delta(s_m, x_1)$
…	…	…	…
X_F	$\delta(s_1, x_F)$	…	$\delta(s_m, x_F)$

Таблиця переходів автомата Мілі

вхід \ стан	s_1	…	s_m
X_1	$\lambda(s_1, x_1)$	…	$\lambda(s_m, x_1)$
…	…	…	…
X_F	$\lambda(s_1, x_F)$	…	$\lambda(s_m, x_F)$

Таблиця виходів автомата Міллі

На перетині стовпця s_m, і рядки x_F в **таблиці переходів** ставиться стан $s_{mf} = \delta(_{sm}, x_F)$, в який автомат переходить із стану s_m під дією сигналу x_F.
В **таблиці виходів** - відповідний цьому переходу вихідний сигнал $y_{mf} = \lambda(s_m, x_F)$.

АВТОМАТ МІЛЛІ

Таблиці переходів і виходів автомата **Міллі** можуть бути подані у вигляді однієї об'єднаної таблиці, у клітинках якої вказані значення як станів,так і виходів

вхід \ стан	s_1	…	s_m
X_1	$\delta(s_1, x_1)/\lambda(s_1, x_1)$	…	$\delta(s_m, x_1)/\lambda(s_m, x_1)$
…	…	…	…
X_F	$\delta(s_1, x_F)/\lambda(s_1, x_F)$	…	$\delta(s_m, x_F)/\lambda(s_m, x_F)$

Об'єднана таблиця переходів автомата Мілі

АВТОМАТ МУРА

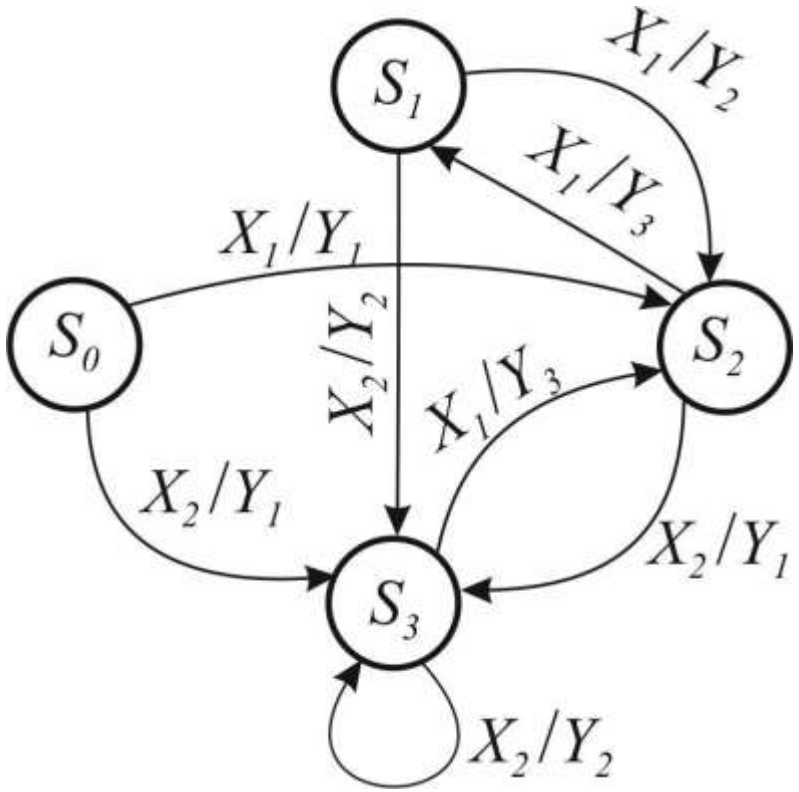
Оскільки в автоматі **Мура** вихідний сигнал залежить, тільки від стану, автомат **Мура** задається однією таблицею переходів, в якій кожному її стовпцю приписаний, окрім стану s_m, ще і вихідний сигнал $y_m=\lambda(s_m)$, відповідний цьому стану. Значення виходів автомата встановлюється окремим рядком над заголовками кожного стовпця.

вихід	$\lambda(s_1)$	…	$\lambda(s_m)$
стан / вхід	s_1	…	s_m
X_1	$\delta(s_1, x_1)$	…	$\delta(s_m, x_1)$
…	…	…	…
X_F	$\delta(s_1, x_F)$	…	$\delta(s_m, x_F)$

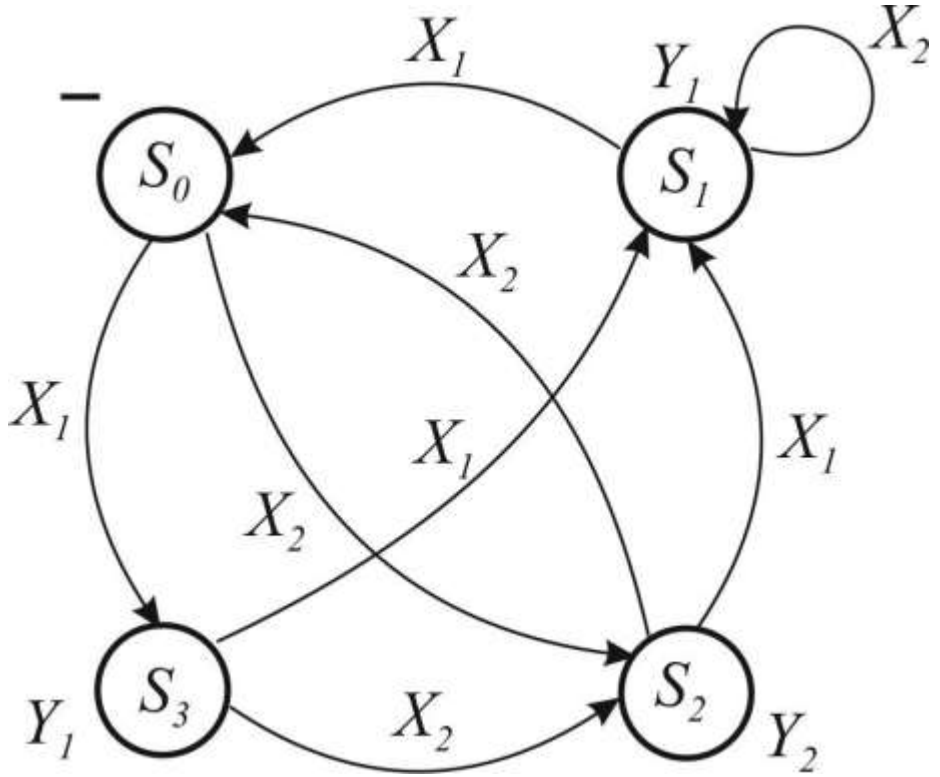
Таблиця переходів автомата Мура

МАШИНИ СТАНІВ У ВИГЛЯДІ ГРАФІВ

Більш наочним є спосіб опису автоматів за допомогою графів. Граф автомата - орієнтований зв'язний граф, вершини якого відповідають станам, а дуги переходам між ними.



Граф автомата Міллі
(це приклад, не шукайте логіки😀)



Граф автомата Мура
(і це теж приклад, не шукайте логіки😀)

ПРИКЛАД 1

Заданий автомат А в табличному виді.

1. Визначити його вхідні і вихідні алфавіти.
2. Визначити тип автомата і представити його у вигляді графа.
3. Визначити вихідну послідовність букв, якщо на вхід поступає вхідна послідовність виду **x1 x2 x3 x3 x1 x2**

	s_0	s_1	s_2	s_3
x_1	s_3	s_0	s_2	s_0
x_2	s_1	s_2	s_0	s_3
x_3	s_0	s_1	s_3	s_1

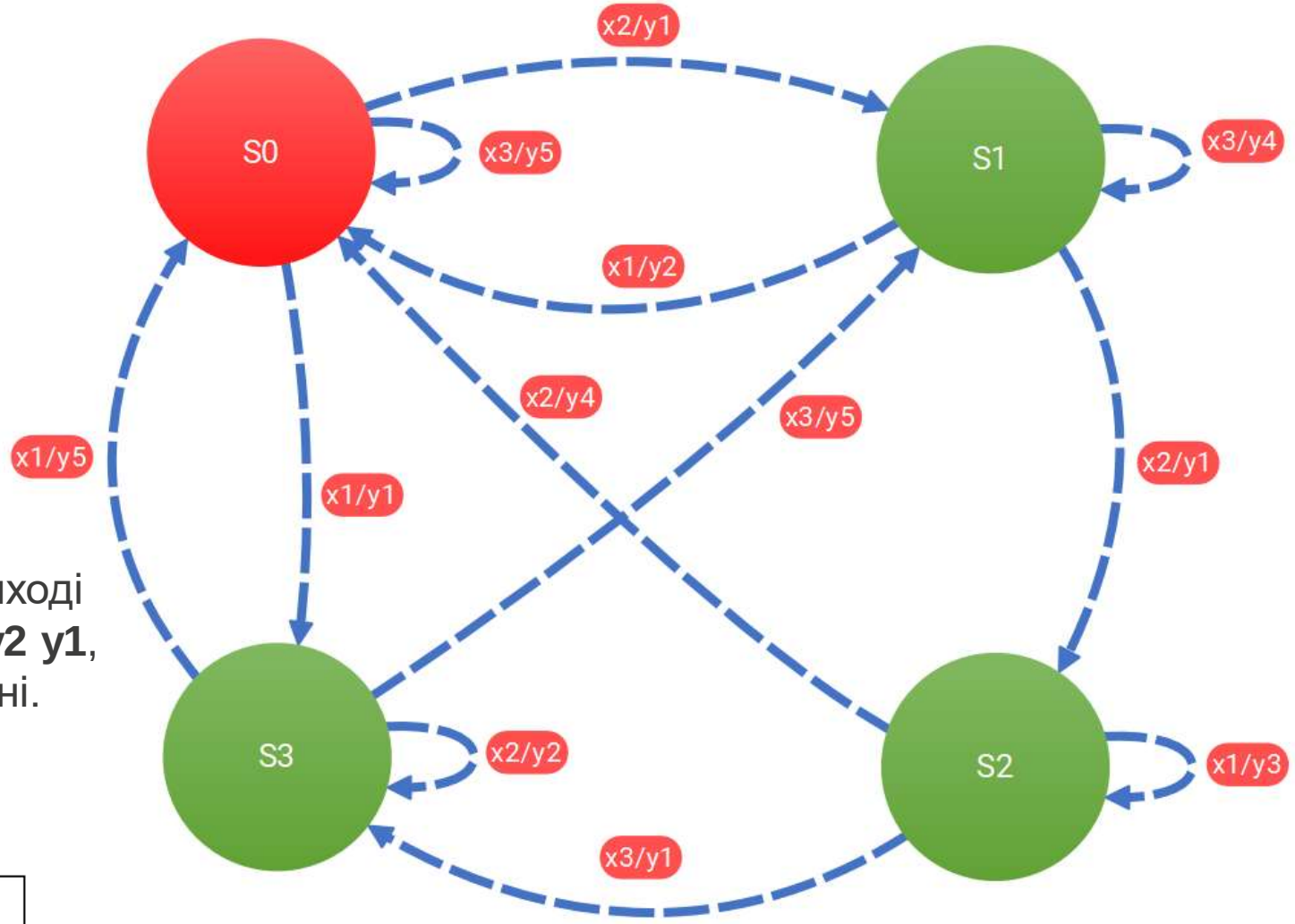
Таблиця переходів ЦА

	s_0	s_1	s_2	s_3
x_1	y_1	y_2	y_3	y_5
x_2	y_1	y_1	y_4	y_2
x_3	y_5	y_4	y_1	y_5

Таблиця виходів ЦА

ПРИКЛАД 1. РОЗВ'ЯЗОК

Вхідний алфавіт: X={x1, x2, x3}
Вихідний алфавіт: Y={y1, y2, y3, y4, y5}
Таблицями заданий автомат Міллі і граф автомата зорбажений на рисунку



При подачі на вхід автомата вхідної послідовності виду **x1 x2 x3 x3 x1 x2** на виході буде отримана послідовність **y1 y2 y5 y4 y2 y1**, на графі автомата відповідні ребра виділені.

Таблиця переходів

	s_0	s_1	s_2	s_3
x_1	s_3	s_0	s_2	s_0
x_2	s_1	s_2	s_0	s_3
x_3	s_0	s_1	s_3	s_1

Таблиця виходів

	s_0	s_1	s_2	s_3
x_1	y_1	y_2	y_3	y_5
x_2	y_1	y_1	y_4	y_2
x_3	y_5	y_4	y_1	y_5

ПРИКЛАД 2

Заданий автомат А2 в табличному виді.

1. Визначити його вхідний і вихідний алфавіти.
2. Визначити тип автомата і представити його у вигляді графа.

	-	y_1	y_3	y_3	y_2
	s_0	s_1	s_2	s_3	s_4
x_1	s_0	s_4	s_2	s_1	s_3
x_2	s_4	s_3	s_0	s_2	s_4
x_3	s_2	s_0	s_4	s_4	s_1
x_4	s_3	s_1	s_1	s_3	s_2

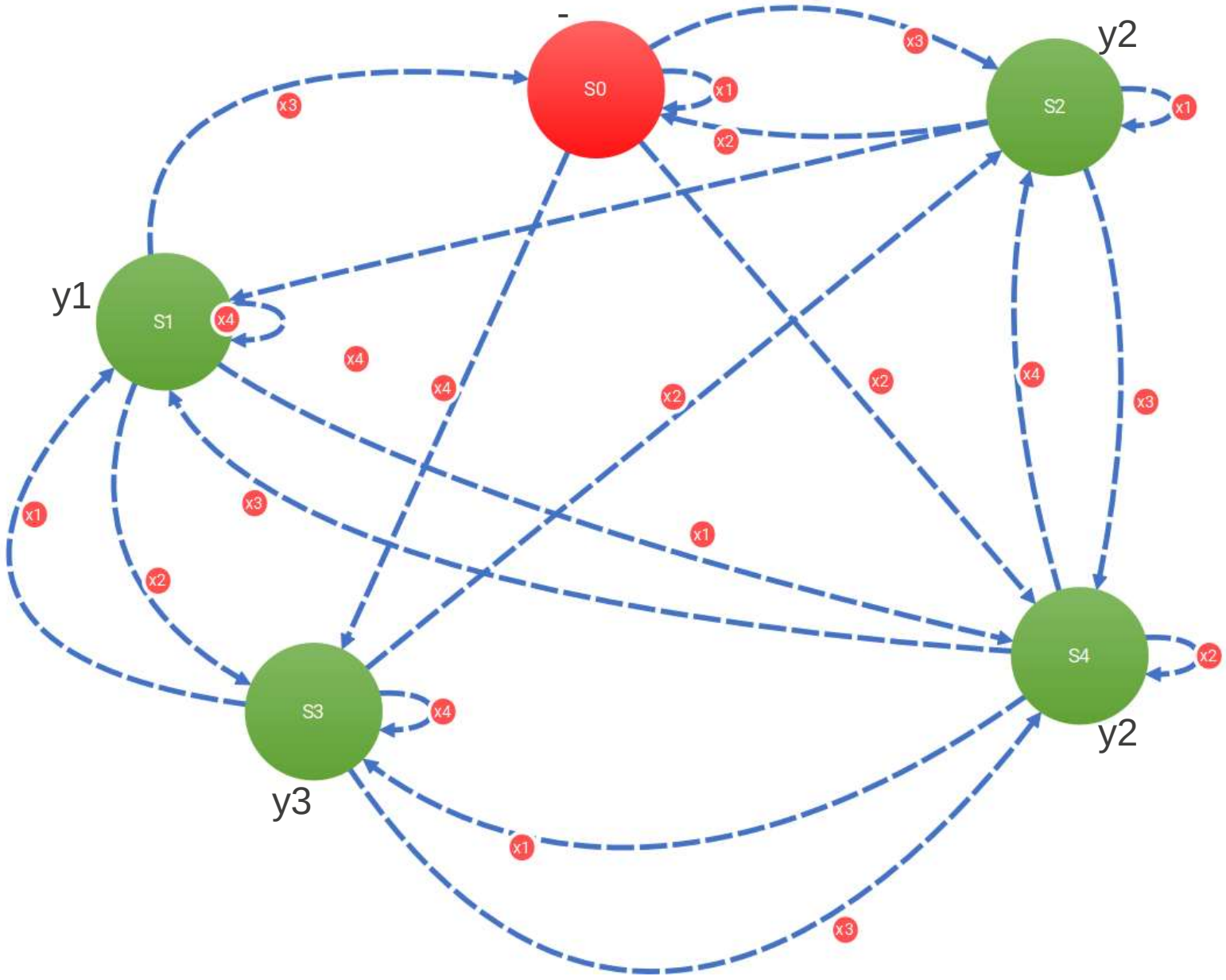
ПРИКЛАД 2. РОЗВ'ЯЗОК

Вхідний алфавіт: X={x1, x2, x3, x4}

Вихідний алфавіт: Y={y1, y2, y3}

Таблицею заданий автомат Мура

	-	y_1	y_3	y_3	y_2
	s_0	s_1	s_2	s_3	s_4
x_1	s_0	s_4	s_2	s_1	s_3
x_2	s_4	s_3	s_0	s_2	s_4
x_3	s_2	s_0	s_4	s_4	s_1
x_4	s_3	s_1	s_1	s_3	s_2



ЕКВІВАЛЕНТНІСТЬ АВТОМАТІВ МІЛІ І МУРА

Автомати Міллі та Мура є еквівалентними в тому сенсі, що будь-який автомат одного типу можна **перетворити** в еквівалентний автомат іншого типу, який виконуватиме ту саму функцію.

Однак, при переході від одного типу до іншого **змінюється кількість станів і структура виходів**.

Перехід від Міллі до Мура:

1. Якщо заданий граф, то виписумо таблицю переходів і виходів, після чого створюємо поєднану таблицю переходів

	q_0	q_1	q_2	q_3
x_1	q_1 / y_1	q_0 / y_2	q_1 / y_1	q_0 / y_1
x_2	q_2 / y_2	q_2 / y_2	q_3 / y_1	q_1 / y_2

2. Позначаємо одинакові переходи/виходи, починаючи з q_0 /y1, q_0 /y2, q_1 /y1…. Якщо такі комбінації знаходяться, то внизу записуємо s_1, s_2, s_3 …

	q_0	q_1	q_2	q_3
x_1	q_1 / y_1 s_3	q_0 / y_2 s_2	q_1 / y_1 s_3	q_0 / y_1 s_1
x_2	q_2 / y_2 s_5	q_2 / y_2 s_5	q_3 / y_1 s_6	q_1 / y_2 s_4

ЕКВІВАЛЕНТНІСТЬ АВТОМАТІВ МІЛІ І МУРА

3. Запис еквівалентних станів. Кожному початковому стану приписується множина відповідних ним позначок (розглядається побудована таблиця з позначками). Шукається в таблиці черговий початковий стан і у відповідну множину записується приписана йому позначка. Для початкового стану q_0 в множині позначок додається також s_0- для ідентифікації початкового стану автомата Мура.

Для отриманої таблиці вийшли наступні значення:

$q_0 = \{s_0, s_1, s_2\}$

$q_1 = \{s_3, s_4\}$

$q_2 = \{s_5\}$

$q_3 = \{s_6\}$

4. Будуємо таблицю переходів автомата Мура

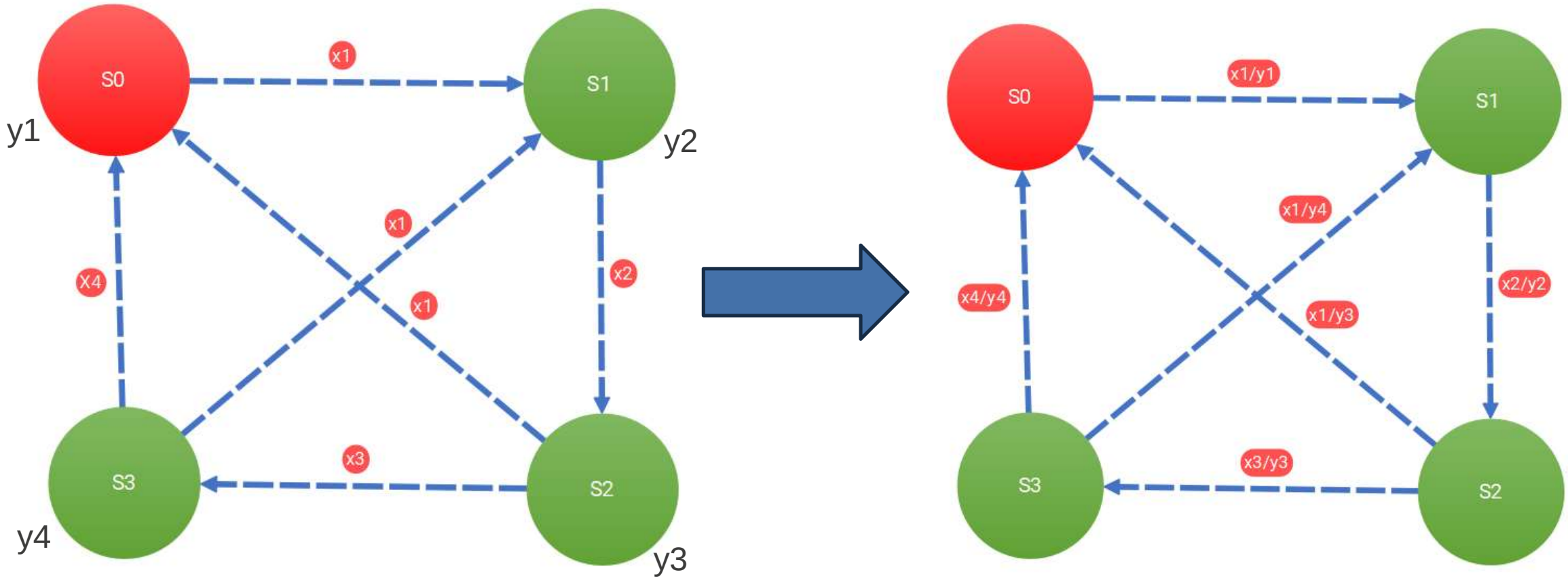
	-	y_1	y_2	y_1	y_2	y_2	y_1
	s_0	s_1	s_2	s_3	s_4	s_5	s_6
x_1	s_3	s_3	s_3	s_2	s_2	s_3	s_1
x_2	s_5	s_5	s_5	s_5	s_5	s_6	s_4

	q_0	q_1	q_2	q_3
x_1	q_1 / y_1 s_3	q_0 / y_2 s_2	q_1 / y_1 s_3	q_0 / y_1 s_1
x_2	q_2 / y_2 s_5	q_2 / y_2 s_5	q_3 / y_1 s_6	q_1 / y_2 s_4

ЕКВІВАЛЕНТНІСТЬ АВТОМАТІВ МІЛІ І МУРА

Перехід від Мура до Міллі

Перехід від автомата Мура до еквівалентного автомата Мілі полягає в тому, що вихідні реакції переносяться з вершин графа на ребра, що входять в ці вершини.



ТЕПЕР ПЕРЕХОДИМО ДО ПРАКТИКИ

Умова завдання. Автомат режимів роботи принтера. Який працює в наступних режимах:

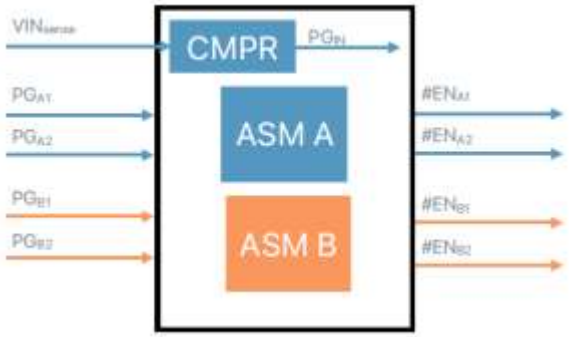
✓ **Очікування (Standby):** Принтер знаходиться в режимі очікування нових завдань. Горить зелений світлодіод

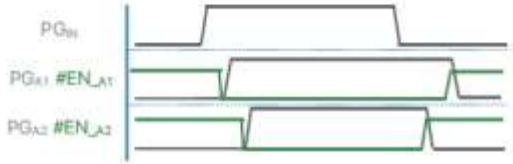
✓ **Друк (Printing):** Активний режим друку, коли принтер виконує поставлене завдання. Горить синій світлодіод

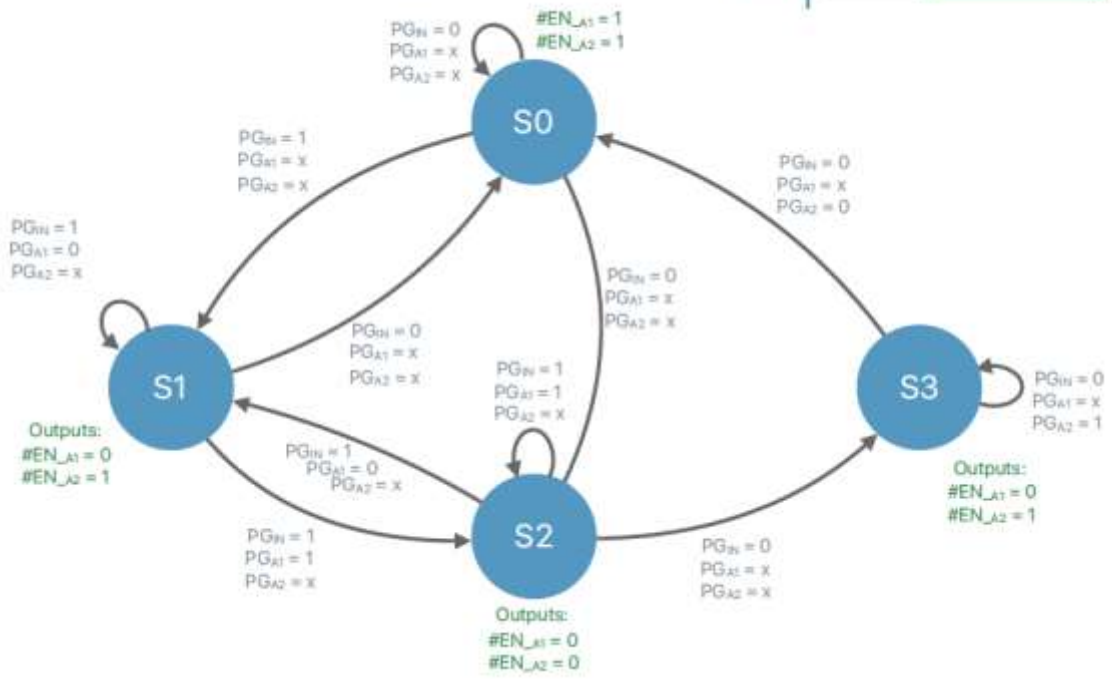
✓ **Очищення (Cleaning):** Деякі принтери автоматично очищують друкуючу головку після певного часу простою або після завершення друку. Горить жовний світлодіод

✓ Помилка **(Error):** Якщо виникла **помилка** (немає паперу, чорнила, paper stuck), принтер зупиняється до її усунення. Горить червоний світлодіод

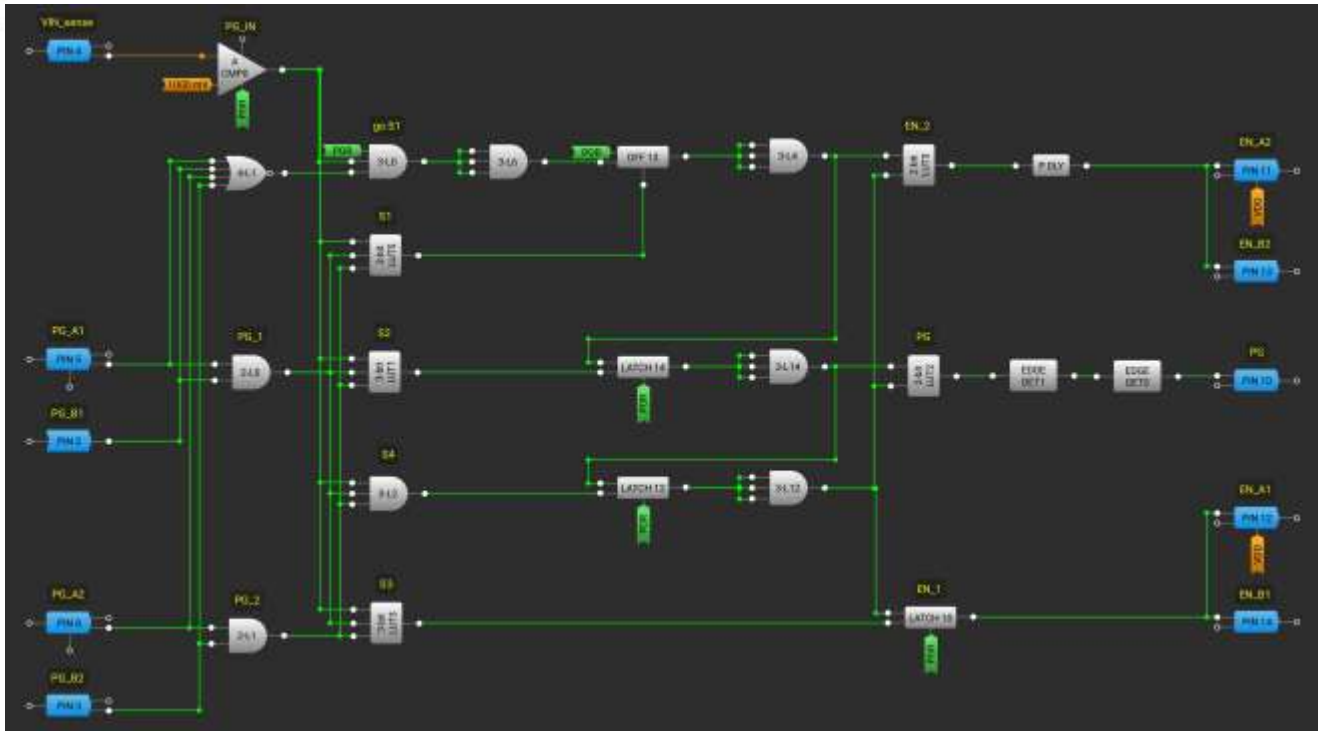
НЕ ASMКОЮ ЄДИНОЮ. ФЛЕШБЕКИ З МИНУЛОГО...



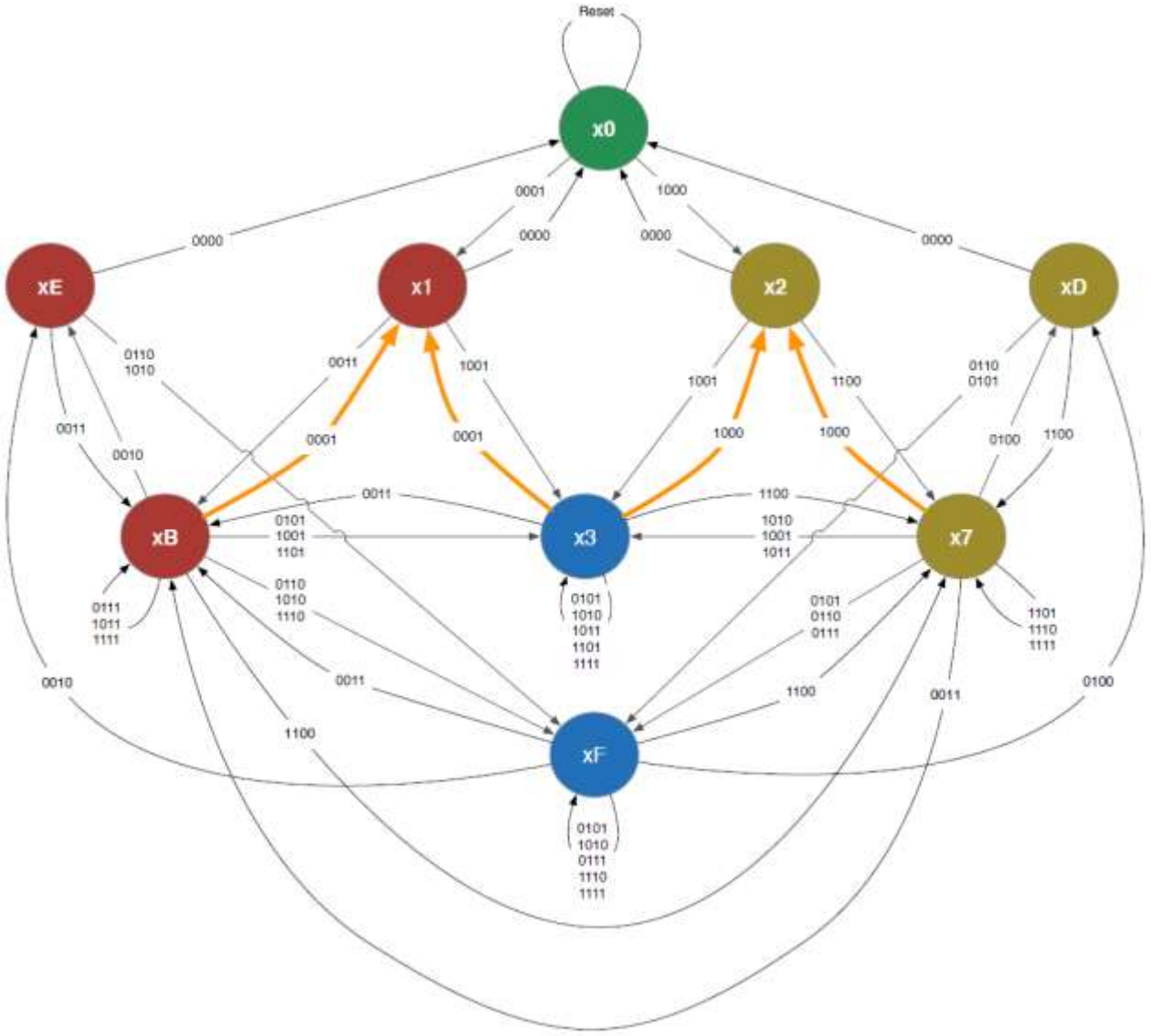




Цифрові автомати можна реалізовувати, використовуючи LUT, DFF, LATCH



НЕ ASMКОЮ ЄДИНОЮ. ФЛЕШБЕКИ З МИНУЛОГО...



Коротка інструкція по ASM в SLG46537

ASM USE CASE. BASED ON SLG46537

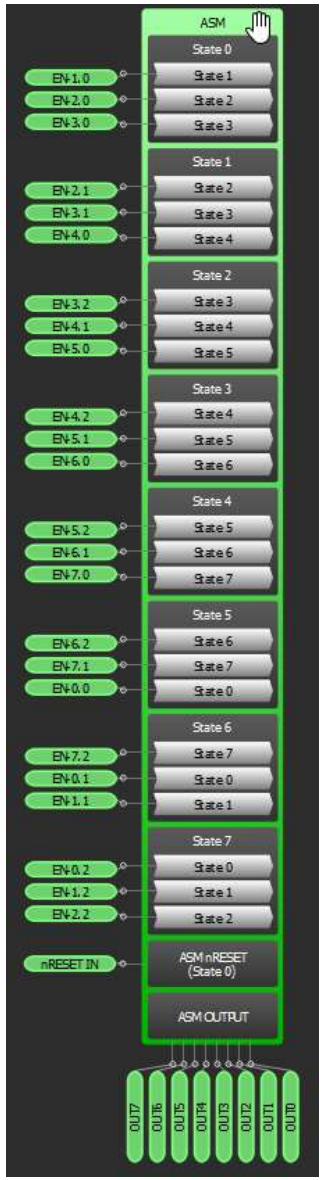
ASM MACROCELL OVERVIEW

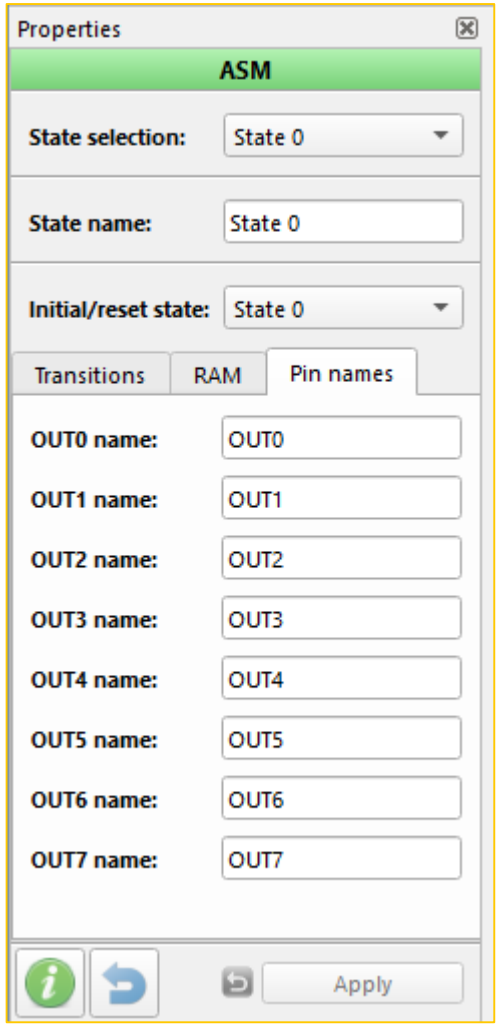
- No clock source is needed, it reacts only to input signals
- No clock means no power (less than 1uA in standby mode). The macrocell consumes power while in state transition
- The input signals do not have to be synchronized to each other, the macrocell will react to the earliest valid signal for state transition

ASM USE CASE. BASED ON SLG46537

ASM MACROCELL OVERVIEW

- ASM has a total of 25 inputs:
  - 24 are user selectable for state transitions
  - 1 is for driving a state transition to an Initial/Reset State
  - The Initial/Reset State has the highest priority
- ASM has a total of 8 outputs
- Active LOW reset to a selectable “Initial/Reset State”
- Transition control
- 8 programmable outputs per state
- Change Pin and State names

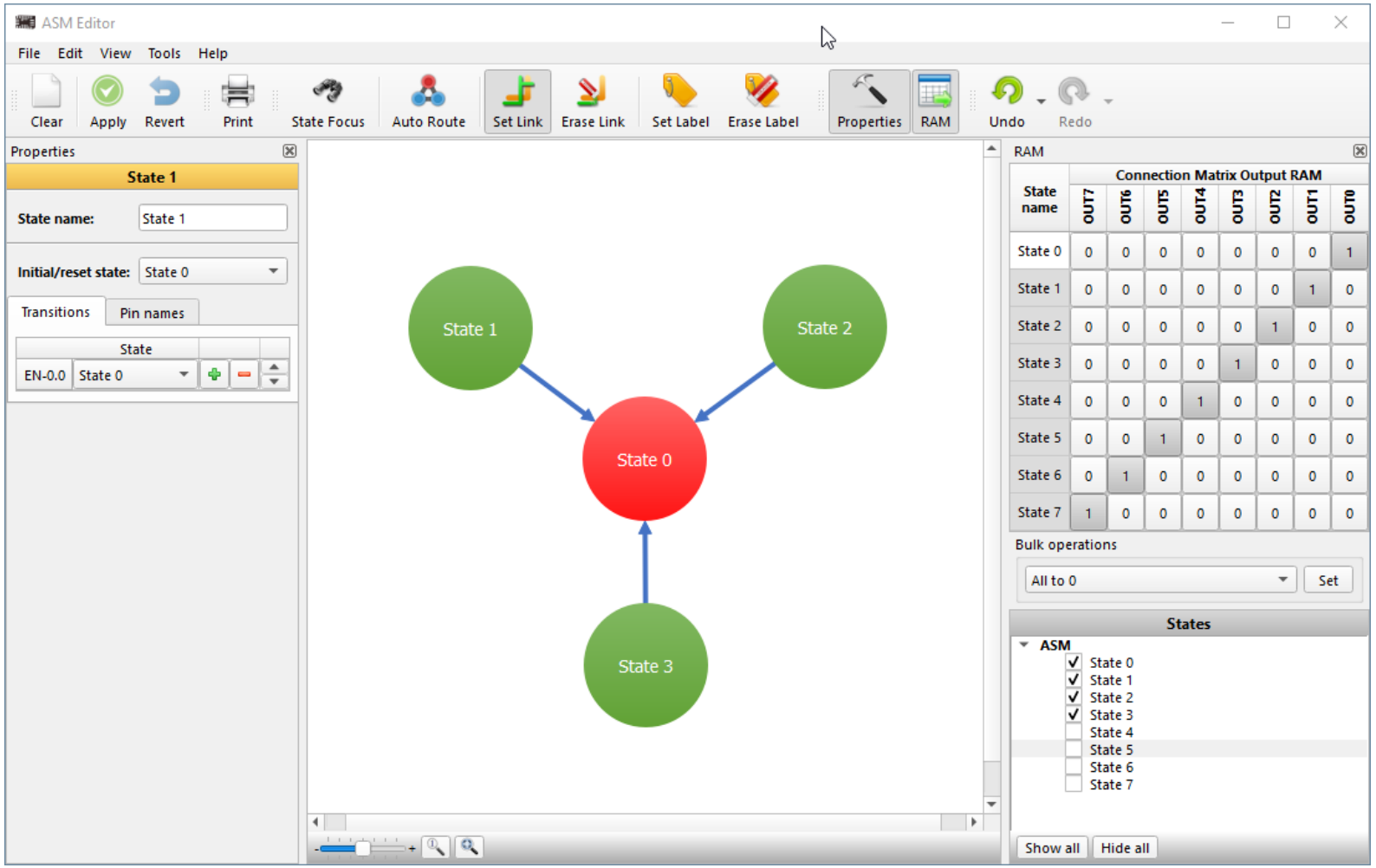




ASM USE CASE. BASED ON SLG46537

ASM EDITOR CONFIGURATION

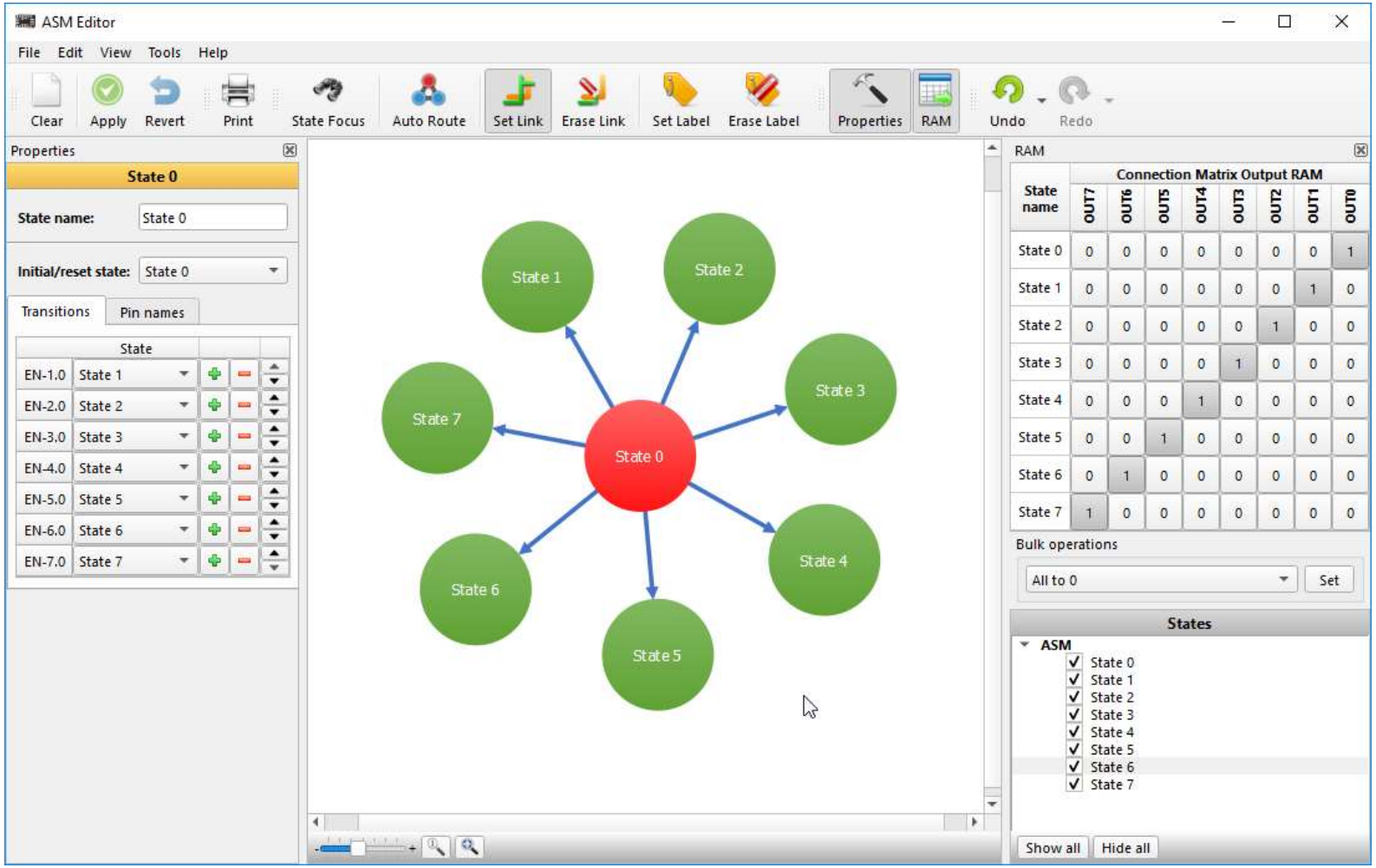
- Maximum 3 State Transition into Given State



ASM USE CASE. BASED ON SLG46537

ASM EDITOR CONFIGURATION

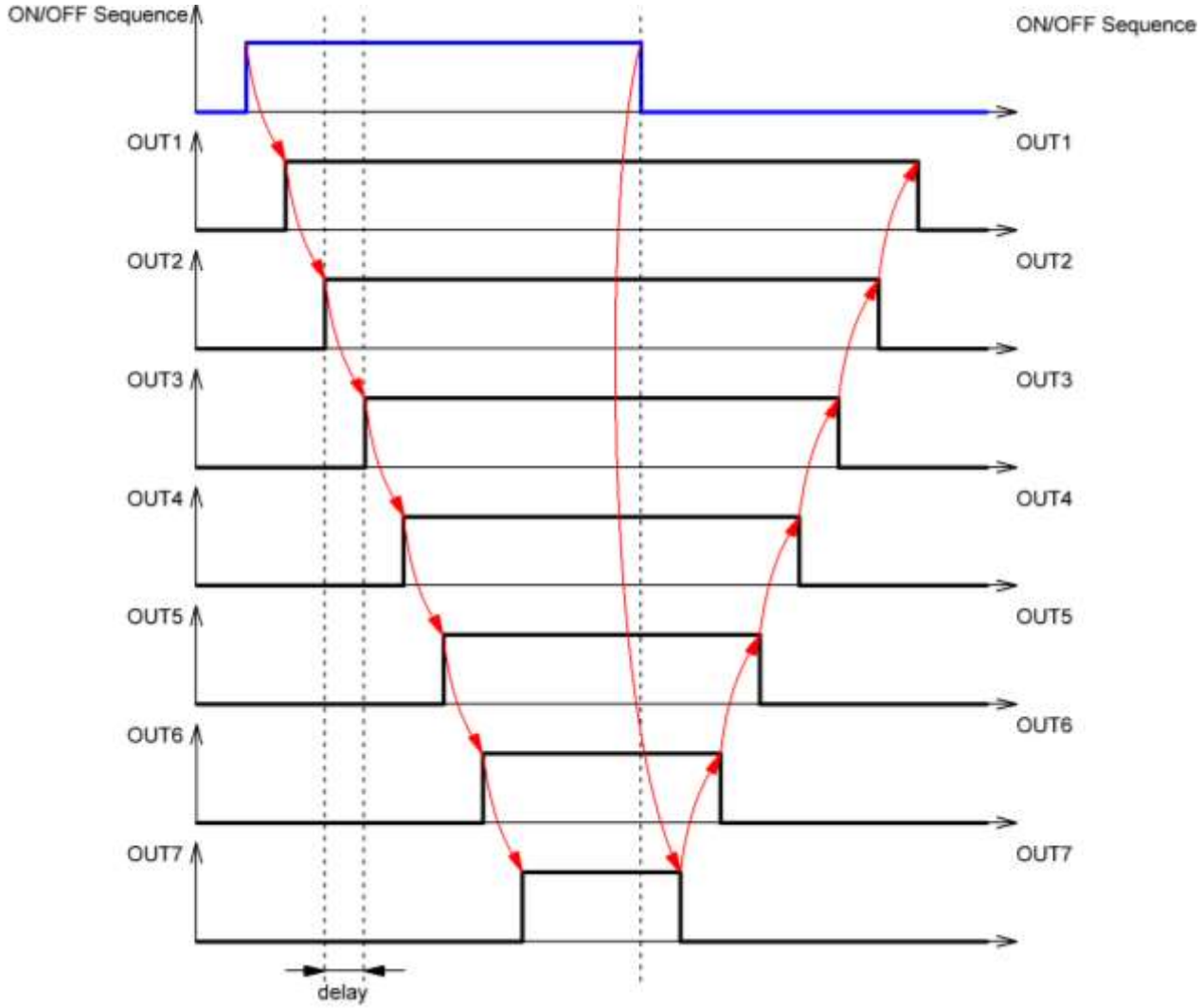
- Maximum 7 State Transition out for a Given State



ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

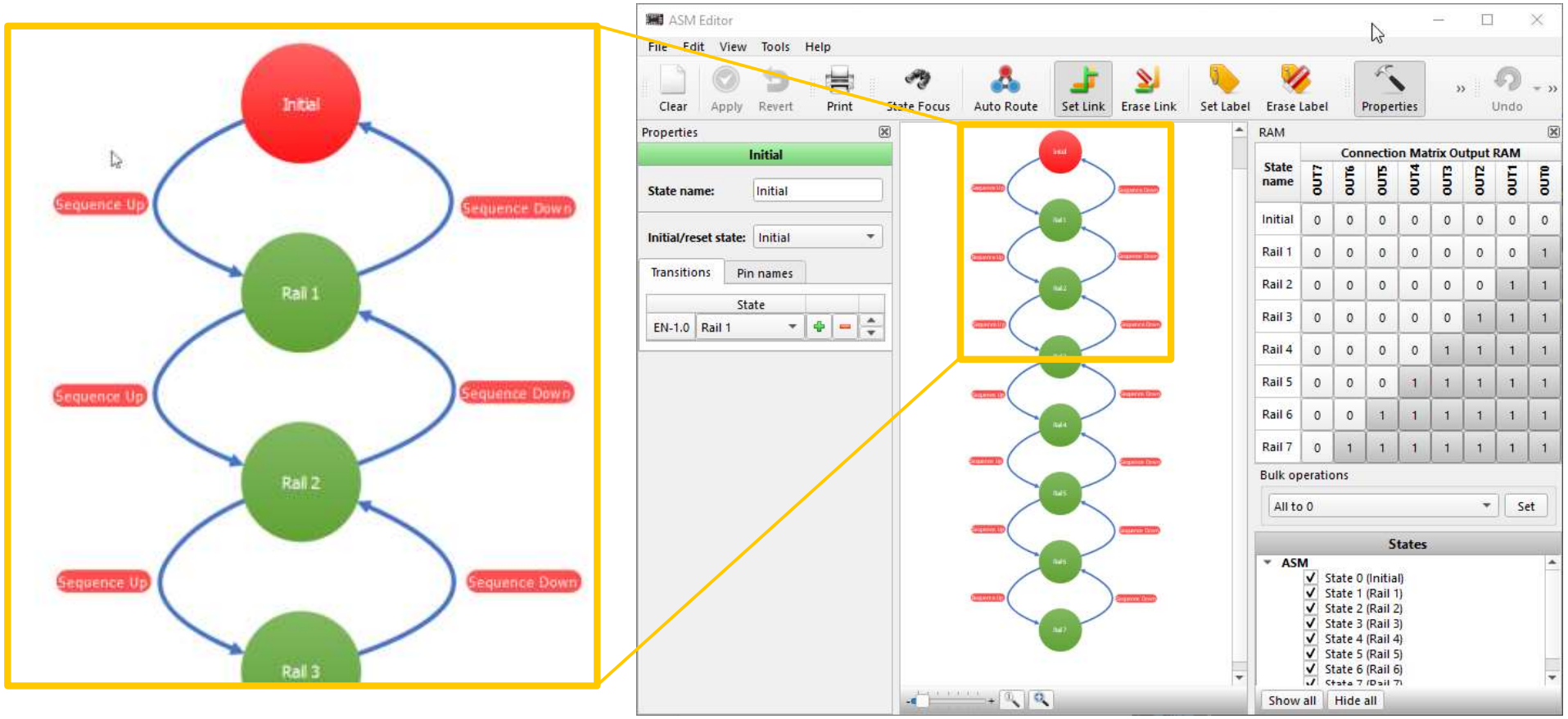
- **Power Sequencer** is used to create the order of switching either ON or OFF with some pre-definite delay
- **Power Sequencer** can be easily implemented using ASM in GreenPak, where every each state switch the next rail



ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

- **Power Sequencer.** States of ASM change in sequence from Rail1 to Rail7 and vice versa



ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

HOW TO PREVENT METASTABILITY IN ASM, WHEN THE CONDITION OF TRANSITION IS THE SAME IN A FEW STATES

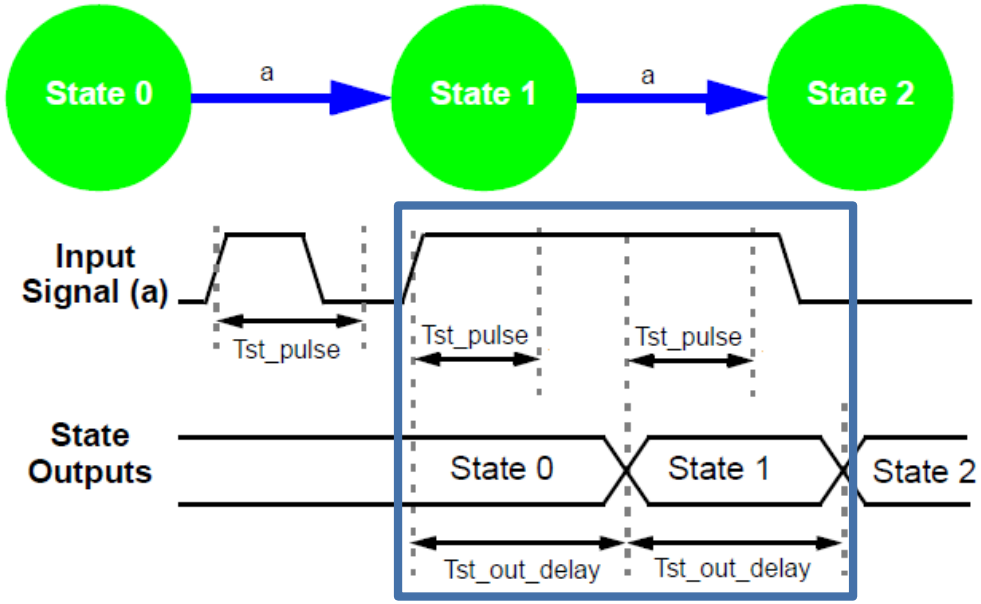


Table 1. ASM Specifications

Symbol	Parameter	Condition	Min	Typ	Max	Unit
tst_out_delay	Asynchronous State Machine Output Delay Time	V_{DD} = 1.8 V ± 5 %	104	--	213	ns
		V_{DD} = 3.3 V ± 10 %	44	--	89	
		V_{DD} = 5.0 V ± 10 %	32	--	58	
tst_out	Asynchronous State Machine Output Transition Time	V_{DD} = 1.8 V ± 5 %	--	--	165	ns
		V_{DD} = 3.3 V ± 10 %	--	--	70	
		V_{DD} = 5.0 V ± 10 %	--	--	45	
tst_pulse	Asynchronous State Machine Input Pulse Acceptance Time	V_{DD} = 1.8 V ± 5 %	14	--	--	ns
		V_{DD} = 3.3 V ± 10 %	6	--	--	
		V_{DD} = 5.0 V ± 10 %	5	--	--	
tst_comp	Asynchronous State Machine Input Compete Time	V_{DD} = 1.8 V ± 5 %	--	--	20	ns
		V_{DD} = 3.3 V ± 10 %	--	--	8	
		V_{DD} = 5.0 V ± 10 %	--	--	5	

Provide the impulse longer than but shorter than

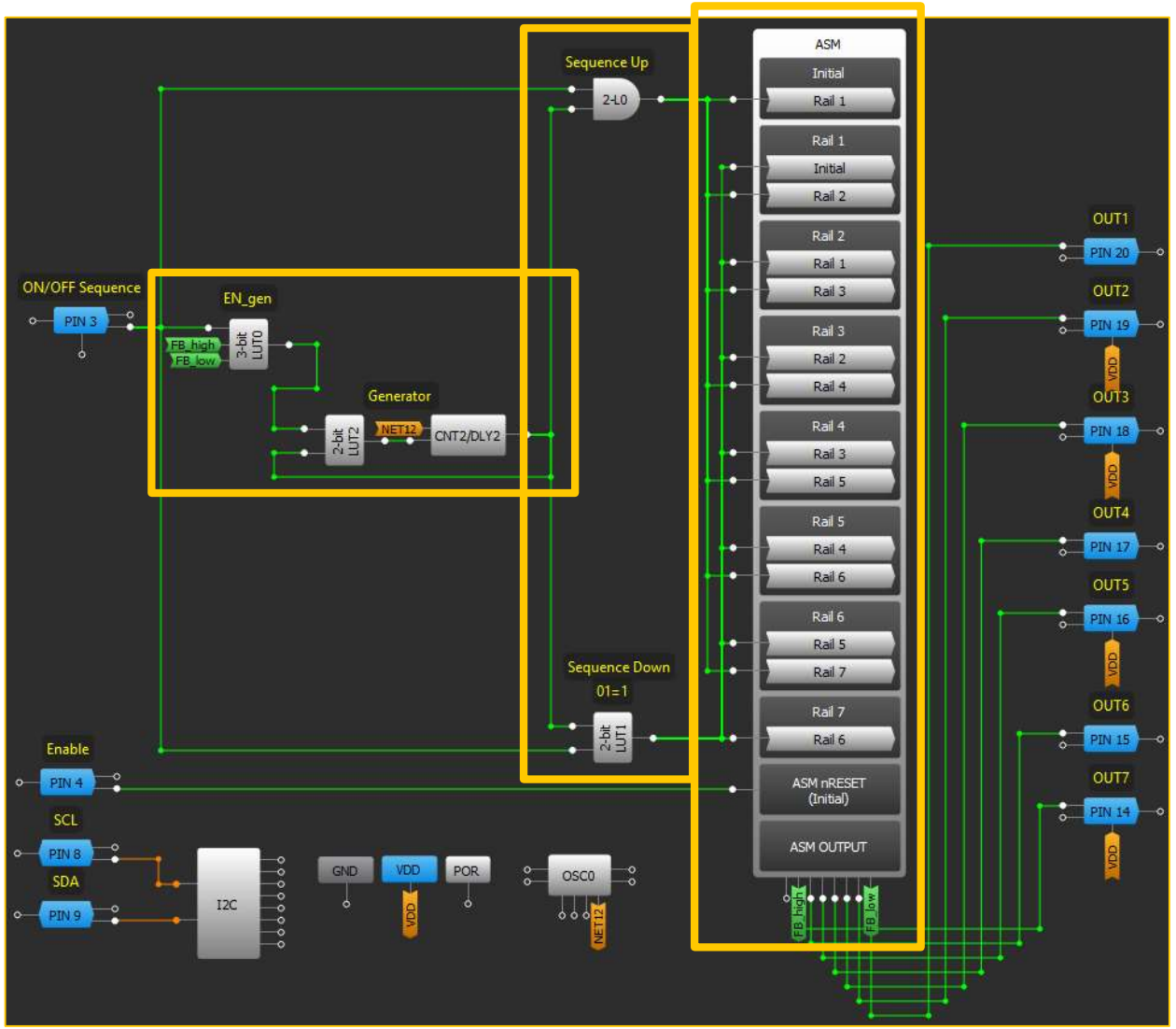
$$Tst\_pulse\_min < T < Tst\_out\_min + Tst\_pulse\_min$$

6ns<T<44ns+6ns Example for 3.3V Vdd

ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

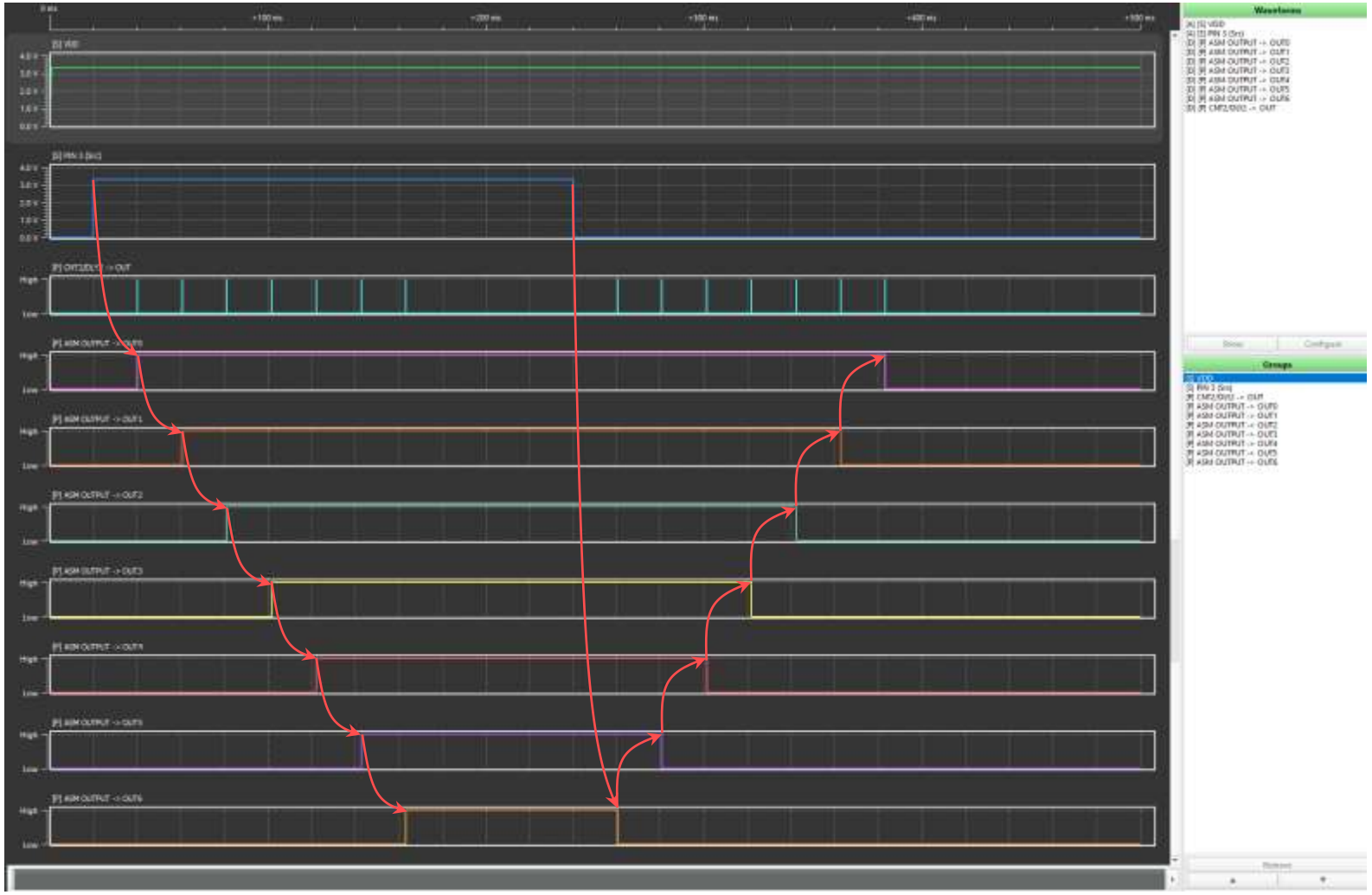
- **Power Sequencer.** This application can control up to 7 lines with a constant delay of switching
- Components of the design:
- Frequency Generator with enable logic
- Sequencer direction logic
- ASM



ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

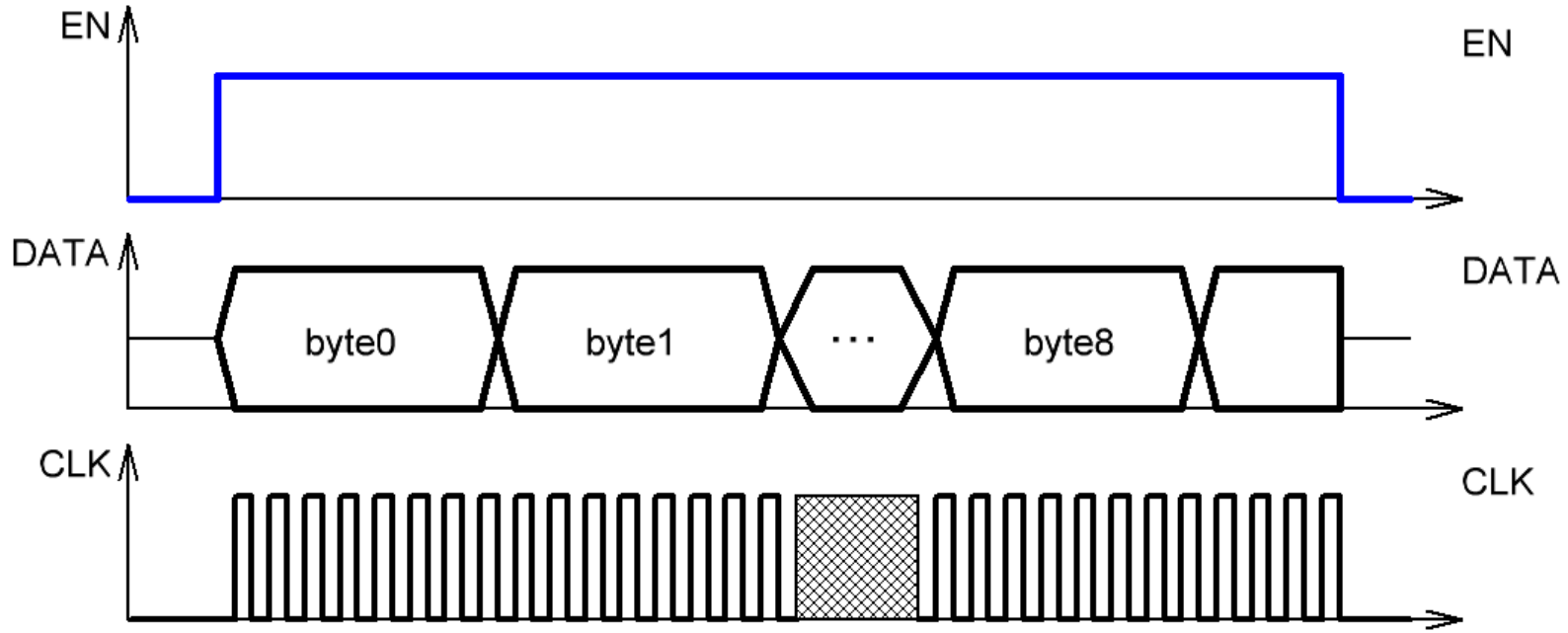
- **Power Sequencer**
  Simulation Results



ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

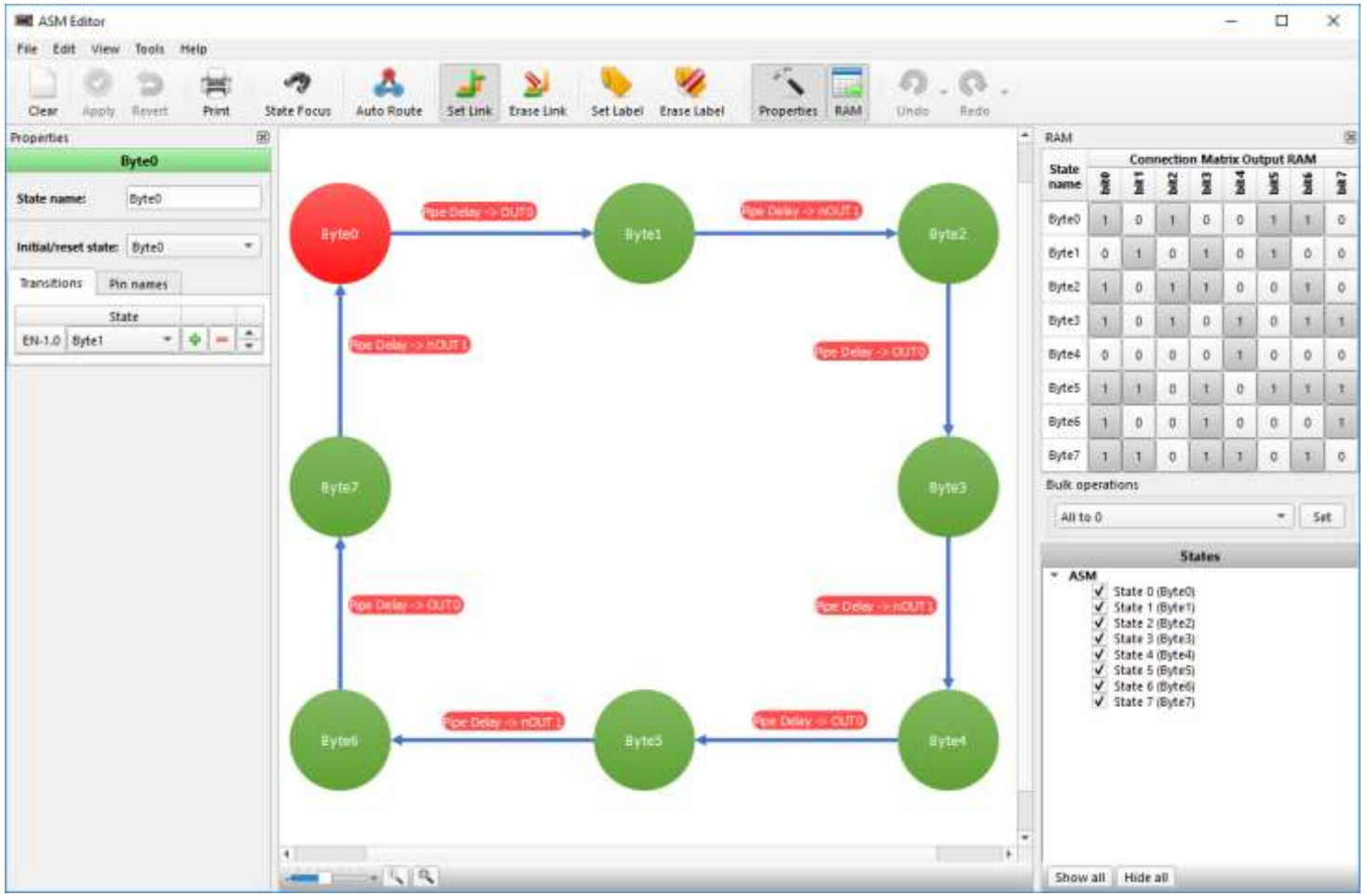
- **64-bit Transmitter** is used to send some predefined pattern of impulses
- In GreenPAK it can be realized using Pattern Generator block or Shift register based on Pipe Delay (DFFs). These methods are limited by chip resources.
- ASM can be used to create 64-bits pattern



ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

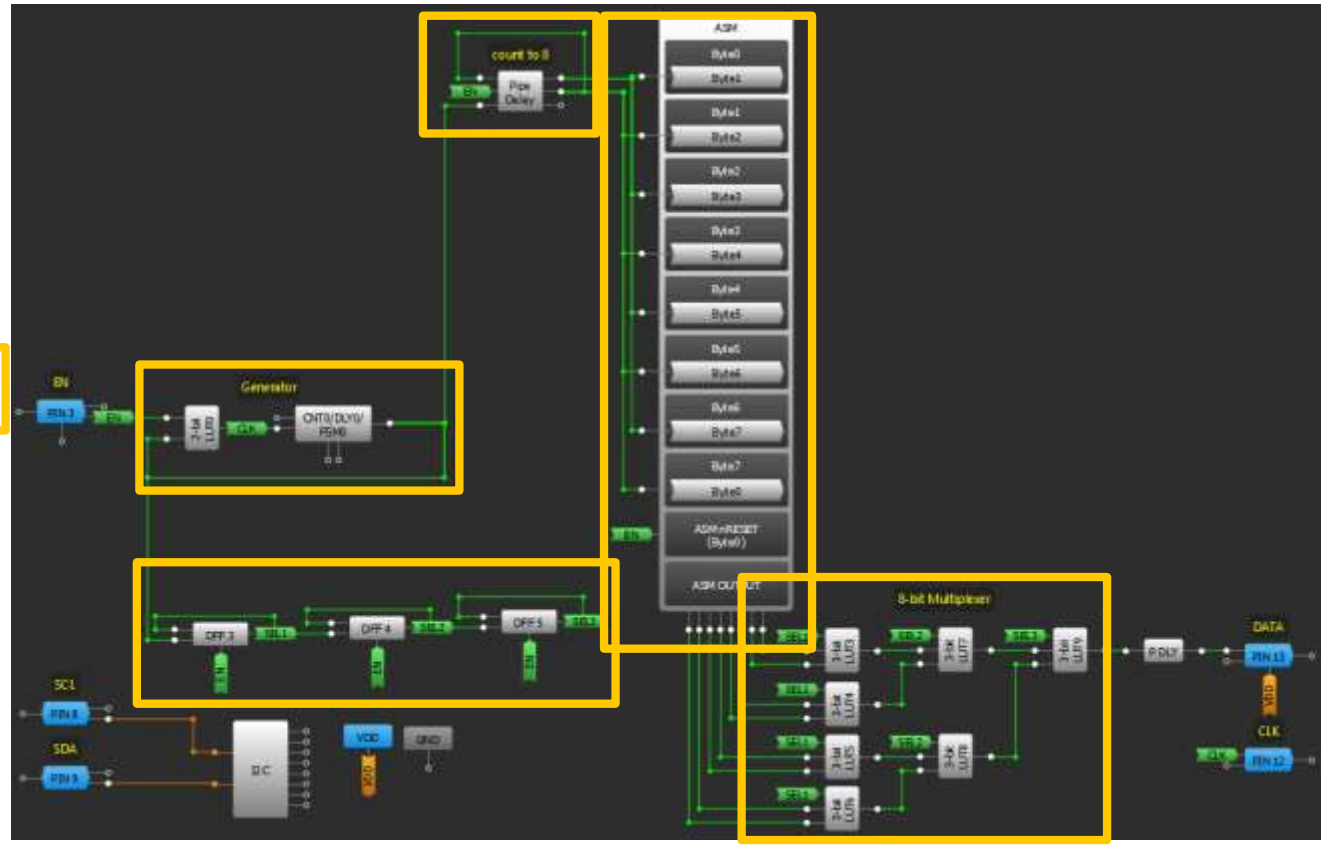
- **64-bit Transmitter** state machine



ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

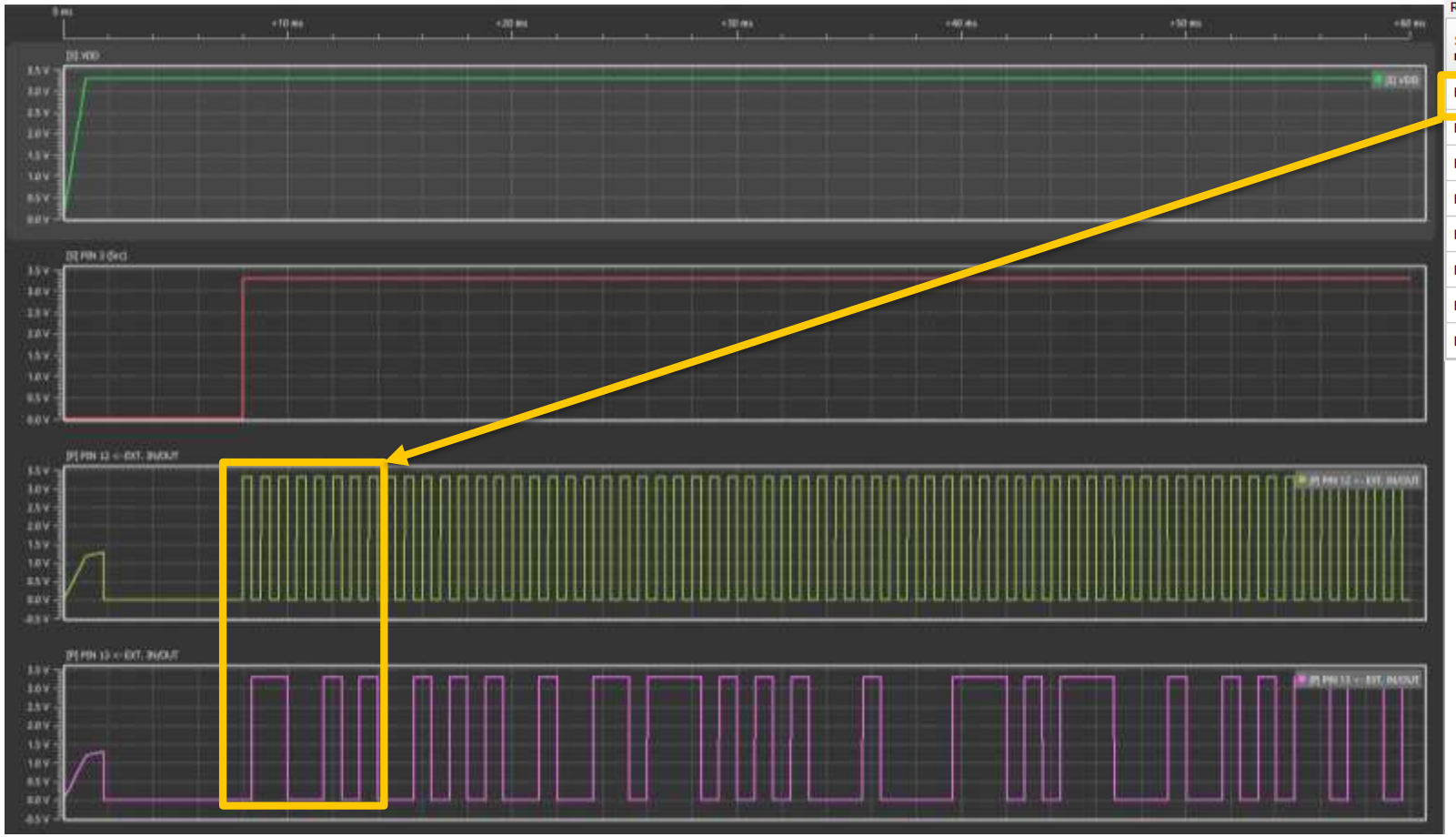
- **64-bit Transmitter.** This application can sent up to 64 bits of code with programmable frequency
- Components of the design
  - Frequency Generator with enable logic
  - 8-bit counter
  - ASM
  - Ripple counter
  - 8-bit Multiplexer



ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

- **64-bit Transmitter** Simulation Result



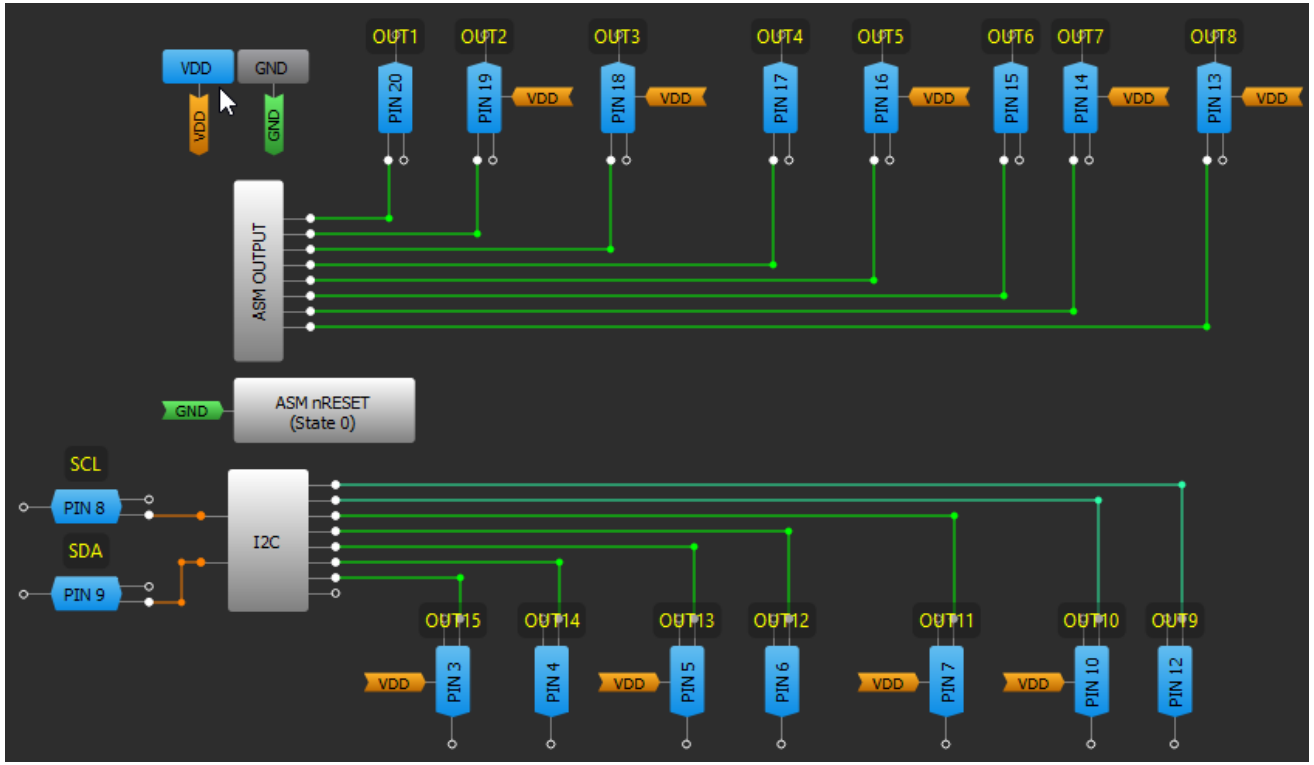
RAM

State name	Connection Matrix Output RAM							
	bit0	bit1	bit2	bit3	bit4	bit5	bit6	bit7
Byte0	1	0	1	0	0	1	1	0
Byte1	0	1	0	1	0	1	0	0
Byte2	1	0	1	1	0	0	1	0
Byte3	1	0	1	0	1	0	1	1
Byte4	0	0	0	0	1	0	0	0
Byte5	1	1	0	1	0	1	1	1
Byte6	1	0	0	1	0	0	0	1
Byte7	1	1	0	1	1	0	1	0

ASM USE CASE. BASED ON SLG46537

EXAMPLE DESIGN – ASM

- **Extended I2C Expander.** This application can control up to 16 outputs via I2C using ASM in reset state



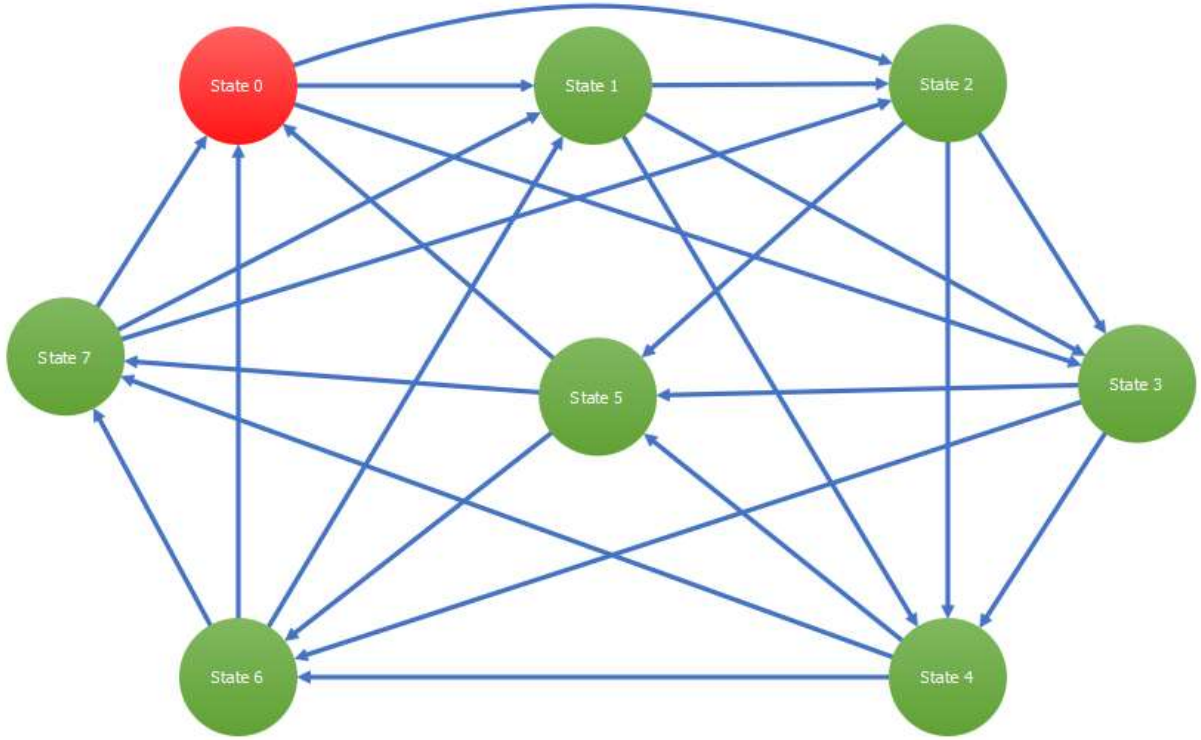
- Components of the design:
  - ASM
  - I2C

- After Connection Matrix Output RAM was updated via I2C, ASM outputs to Connection Matrix can be changed only after ASM changes its state or after reset event.
- To change ASM outputs to Connection Matrix instantly after I2C write command, ASM must be in reset all the time.

ASM USE CASE. BASED ON SLG46537

CONCLUSION

- No clock source is needed
- Zero current consumption
- Asynchronous state transition
- Possibility to create complicated State Machines



Renesas.com