Detektor posloupnosti pro zelenáče
Sequence detector for dummies

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické v Praze

Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague

2021 Petr Socha, petr.socha@fit.cvut.cz

Zadání
Assignment

- Navrhněte konečný automat, který bude jedničkou na výstupu (po dobu jednoho hodinového taktu) signalizovat detekci vstupní posloupnosti bitů “0110” nebo “1000” (první vstupní bit je vlevo). Posloupnosti se nesmějí překrývat.

  Design a finite state machine, which outputs ‘1’ (for one clock cycle) if it detects input sequences “0110” or “1000” (the first input bit is leftmost). The sequences must not overlap.

Vstupy a výstupy
Inputs and outputs

- Vstup Input
  - **IN**
    - jeden bit posloupnosti (za hodinový takt)
      a single bit of sequence (per clock cycle)
- Výstup Output
  - **DETECT**
    - ‘1’ (po dobu jednoho hodinového taktu) právě tehdy, když byla detekována na vstupu **IN** zadaná posloupnost; jinak ‘0’
      ‘1’ (during a single clock cycle) iff the defined sequence was detected on the input **IN**; else ‘0’

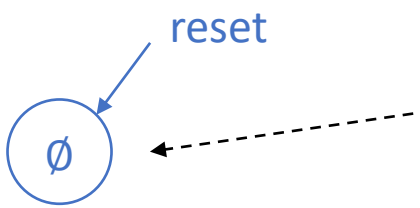
Příklad vstupu a výstupu
Input and output example

- 0110, 1000

- IN: 0111010**0110**00**1000**101
- DETECT: 000000000**1**00000**1**000 (Mealy)
- DETECT: 000000000**1**00000**1**00 (Moore)

Návrh krok po kroku
Step-by-step design

0110
1000

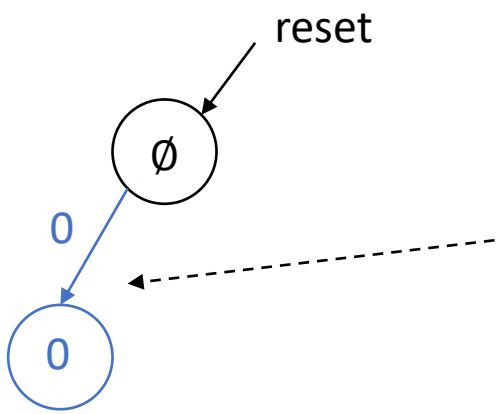


Po resetu nemáme načteno nic
After reset, nothing is loaded

Chybí přechody pro ‘0’ a ‘1’
Transitions for ‘0’ and ‘1’ are missing

Návrh krok po kroku
Step-by-step design

0110
1000

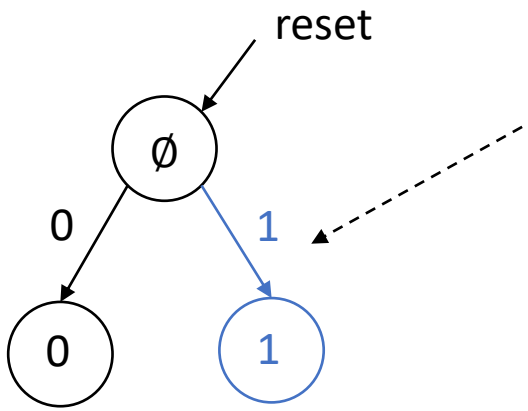


‘0’ si zapamatujeme: je prefixem hledané posloupnosti

We will remember ‘0’: it is a prefix of the searched sequence

Návrh krok po kroku
Step-by-step design

0110
1000



Stejně tak si zapamatujeme načtení ‘1’
Similarly, we will remember ‘1’

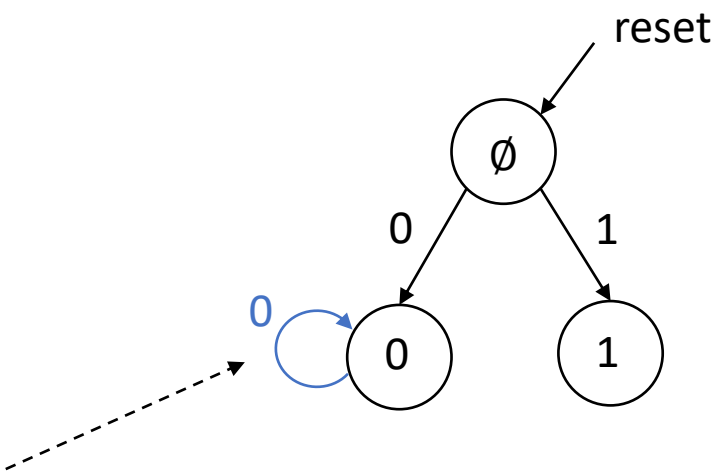
Pokračujeme opět chybějícími přechody...
Again, we continue with missing transitions...

Všimněte si, že kdyby posloupnosti byly třeba “0110” a “0100”, tak by nás tahle jednička nezajímala: cyklili bychom se v počátečním stavu. ‘1’ by nebyla prefixem ani jedné posloupnosti, a tak by nebyl důvod si ji pamatovat.

Note that if the sequences were, say, “0110” and “0100”, we would not care about this particular ‘1’: we would stay in the after-reset state. ‘1’ would not be a prefix on either sequence, so there would be no reason to remember it.

Návrh krok po kroku
Step-by-step design

0110
1000

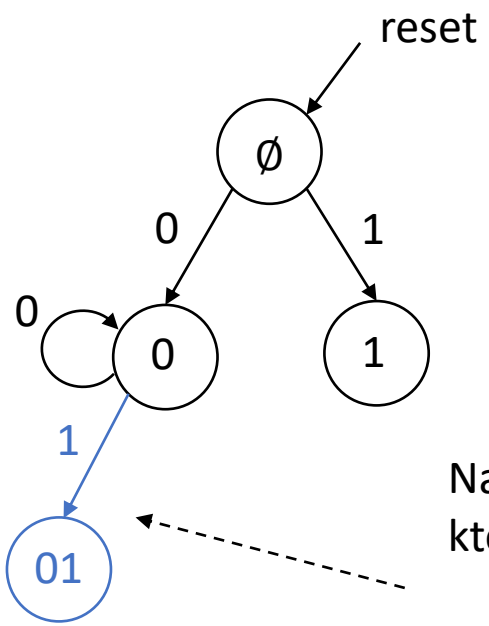


Načtení další ‘0’ znamená, že jsme přečetli “00”. Nám stačí si pamatovat jen prefix “…0”: nula, a před ní něco, co nás nezajímá (další nuly).

Reading another ‘0’ means we have read “00”. But we only need to remember prefix “…0”: a zero, and something uninteresting before it (another zeroes).

Návrh krok po kroku
Step-by-step design

01110
1000



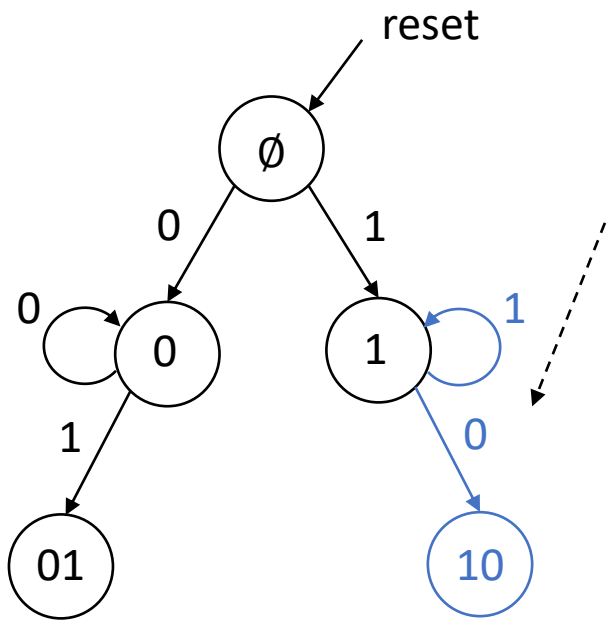
Načtení ‘1’ ale znamená načtení prefixu “...01”, který si potřebujeme zapamatovat celý.

But reading ‘1’ means we have read “...01”, which we need to remember whole.

Do stavu 01 se lze dostat třeba posloupnostmi “01”, “001”, “0001”, …
To get in state 01, sequences such as “01”, “001”, “0001”,… can be used

Návrh krok po kroku
Step-by-step design

0110
1000

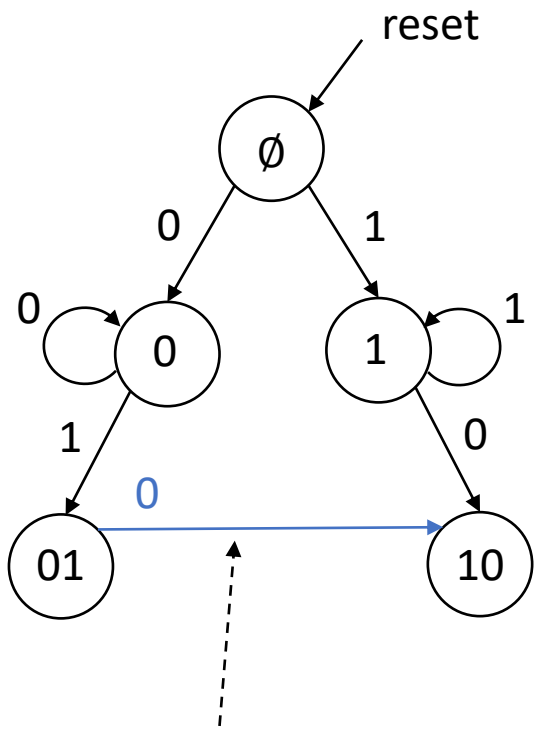


Podobně si nepotřebujeme pamatovat “11”, budeme si pamatovat jen “…1”.
Zajímá nás ale “…10”.

Similarly, we do not need to remember “11”, we handle it same as just “…1”.
But we care about “…10”.

Návrh krok po kroku
Step-by-step design

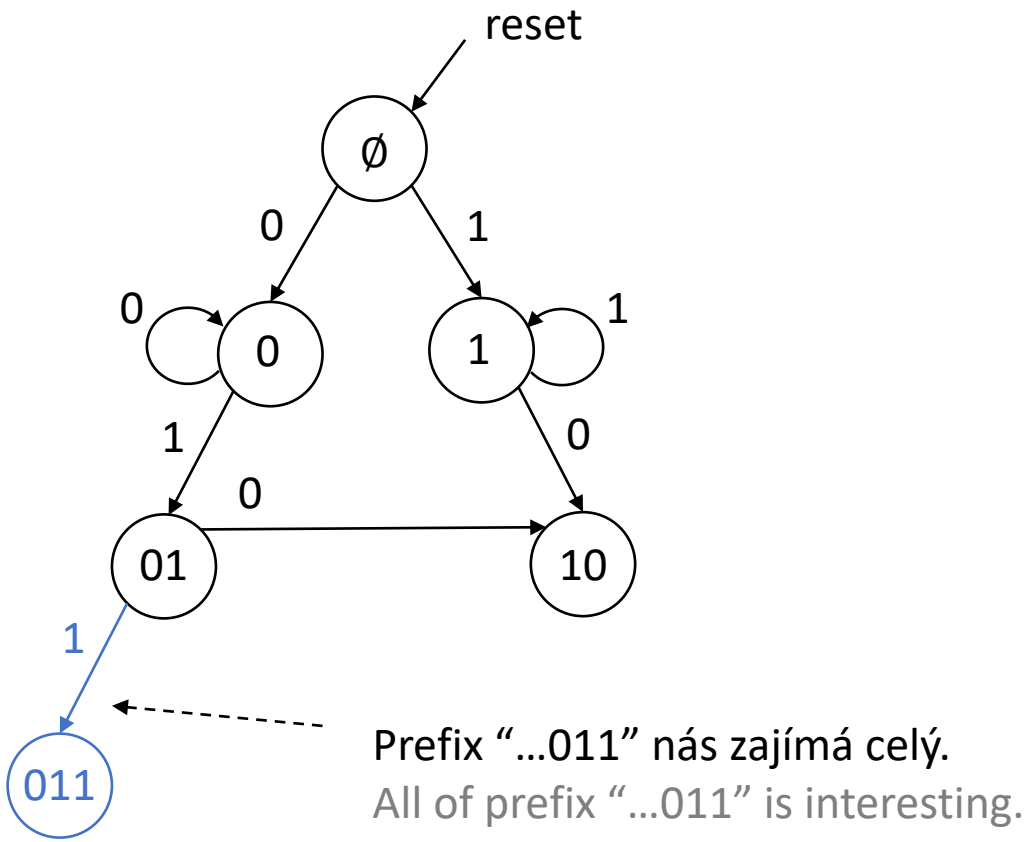
0110
1000



Z prefixu “…010” si potřebujeme pamatovat jen “…10”.
From prefix “…010”, we only need to remember “…10”.

Návrh krok po kroku
Step-by-step design

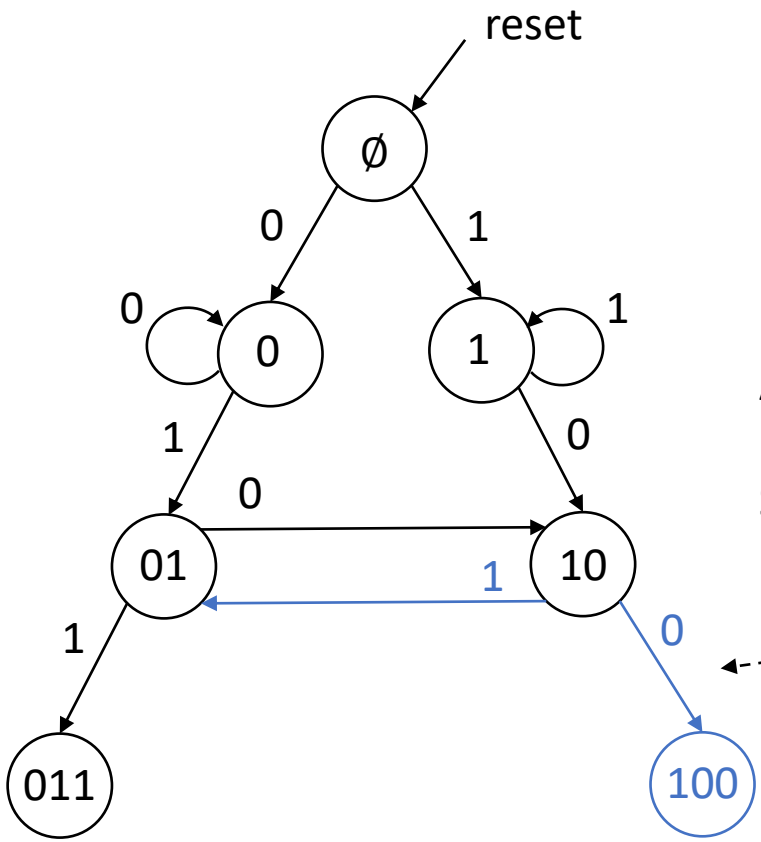
0110
1000



Prefix “…011” nás zajímá celý.
All of prefix “…011” is interesting.

Návrh krok po kroku
Step-by-step design

0110
1000

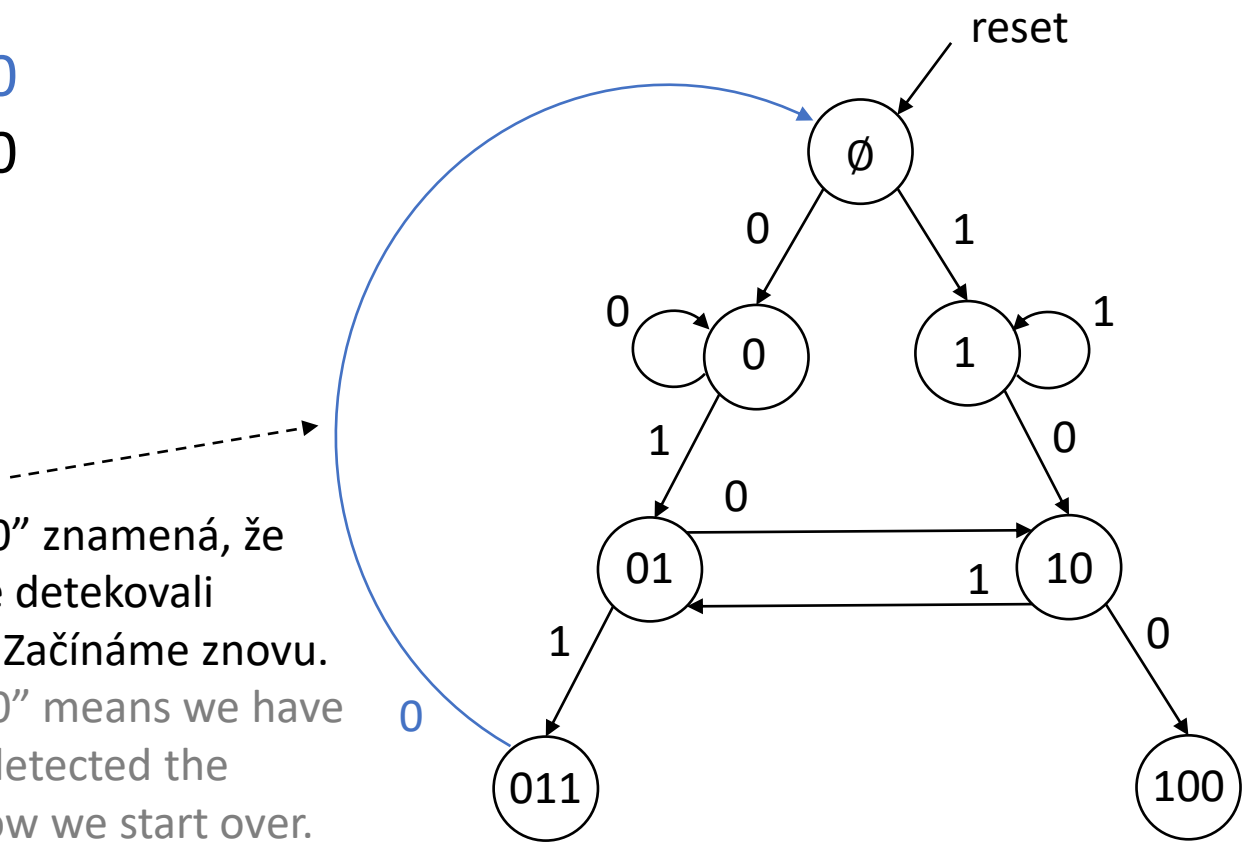


Podobně z “…101” nás zajímá jen “…01”.
Ale “...100” nás zajímá celé.

Similarly, from “…101” we only care about “…01”.
But we care about all of “…100”.

Návrh krok po kroku
Step-by-step design

0110
1000

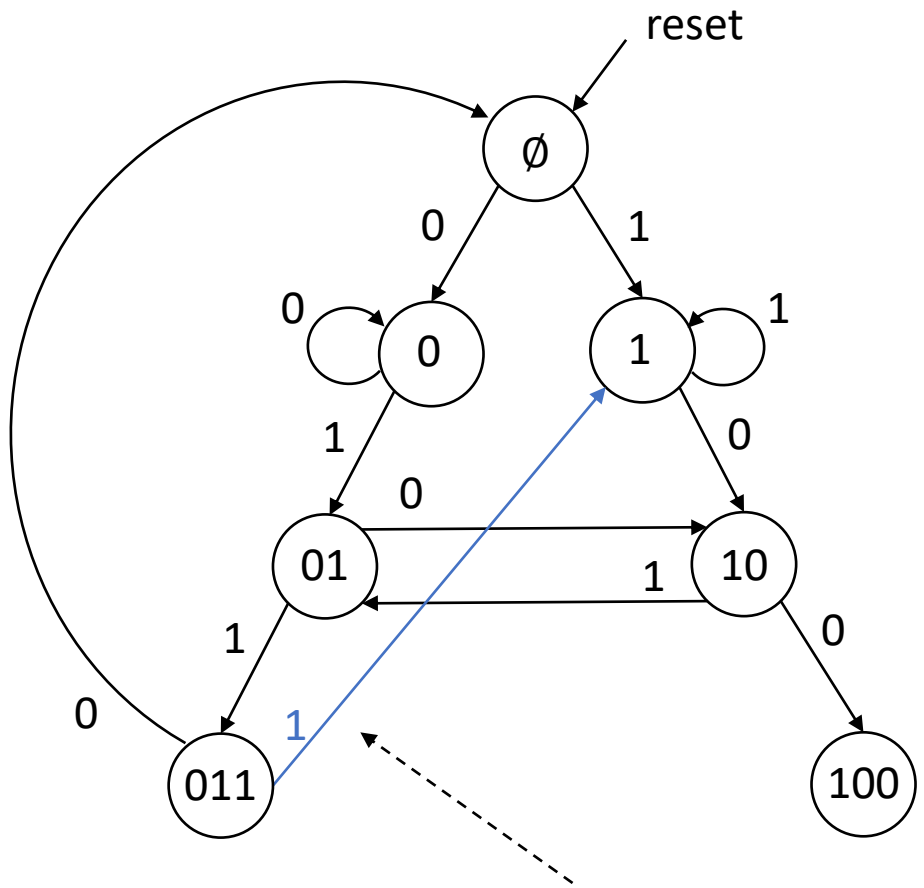


Prefix “…0110” znamená, že jsme úspěšně detekovali posloupnost. Začínáme znovu.
Prefix “…0110” means we have successfully detected the sequence. Now we start over.

Kdyby se posloupnosti směly překrývat, přešli bychom do stavu 10. Tedy bychom si dále pamatovali “…10”.
If the sequences could overlap, we would go to the state 10. So, we would still remember “…10”.

Návrh krok po kroku
Step-by-step design

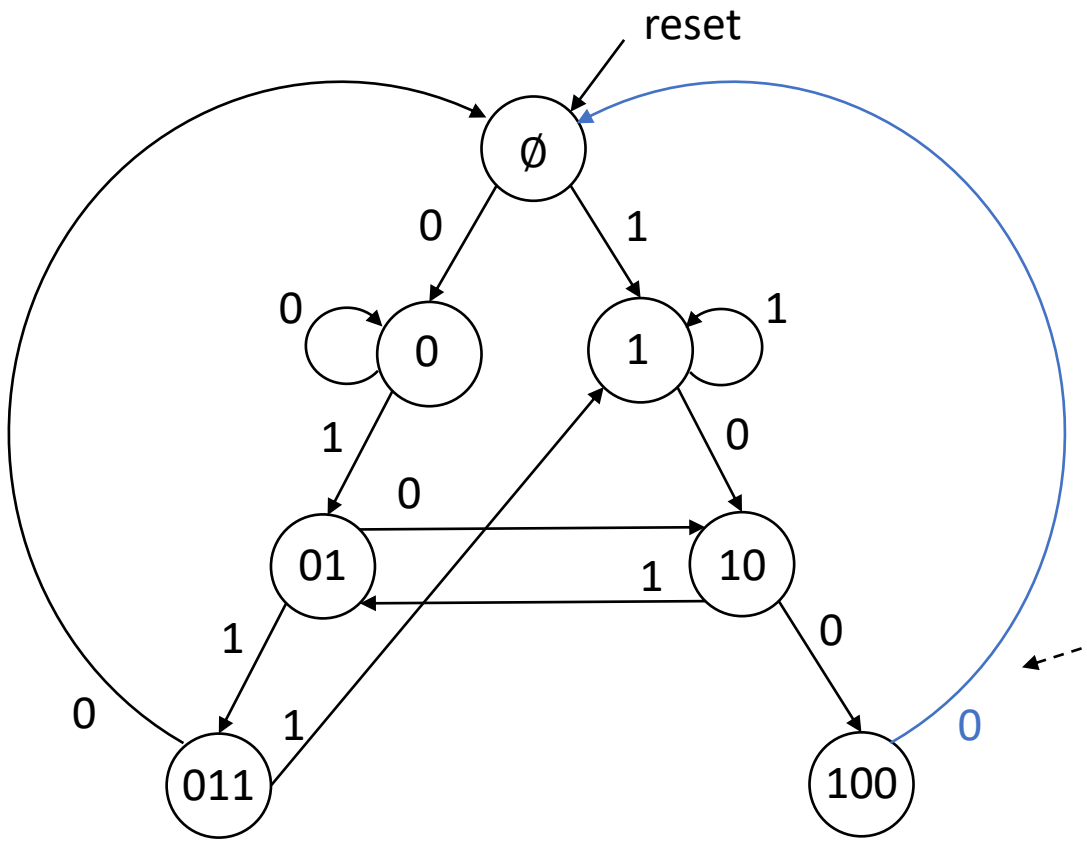
0110
1000



Z prefixu “…0111” nás zajímá pouze “…1”.
In prefix “…0111”, we only care about “…1”.

Návrh krok po kroku
Step-by-step design

0110
1000

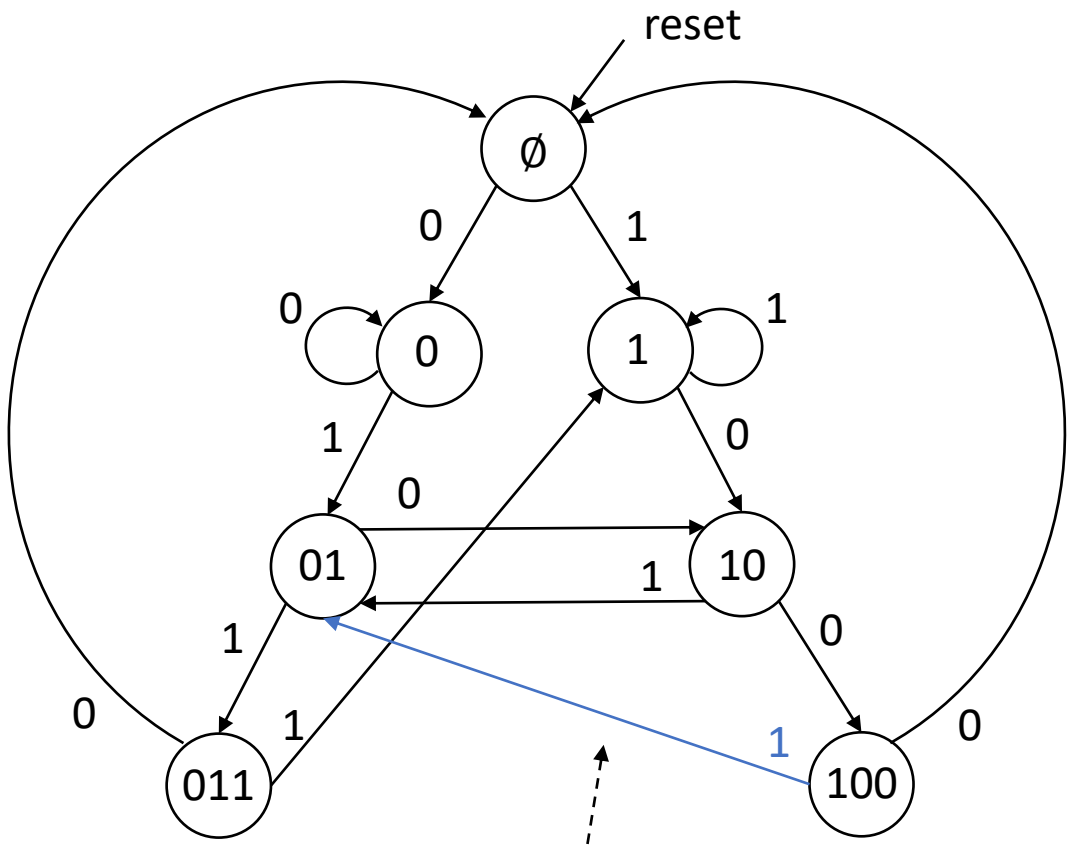


Prefix "…1000" znamená, že jsme úspěšně detekovali posloupnost. Pamatovat si dál nemusíme nic.
Prefix "…1000" means we have successfully detected the sequence. There is no prefix to remember.

Kdyby se posloupnosti směly překrývat, přešli bychom do stavu 0.
If the sequences could overlap, we would go to the state 0.

Návrh krok po kroku
Step-by-step design

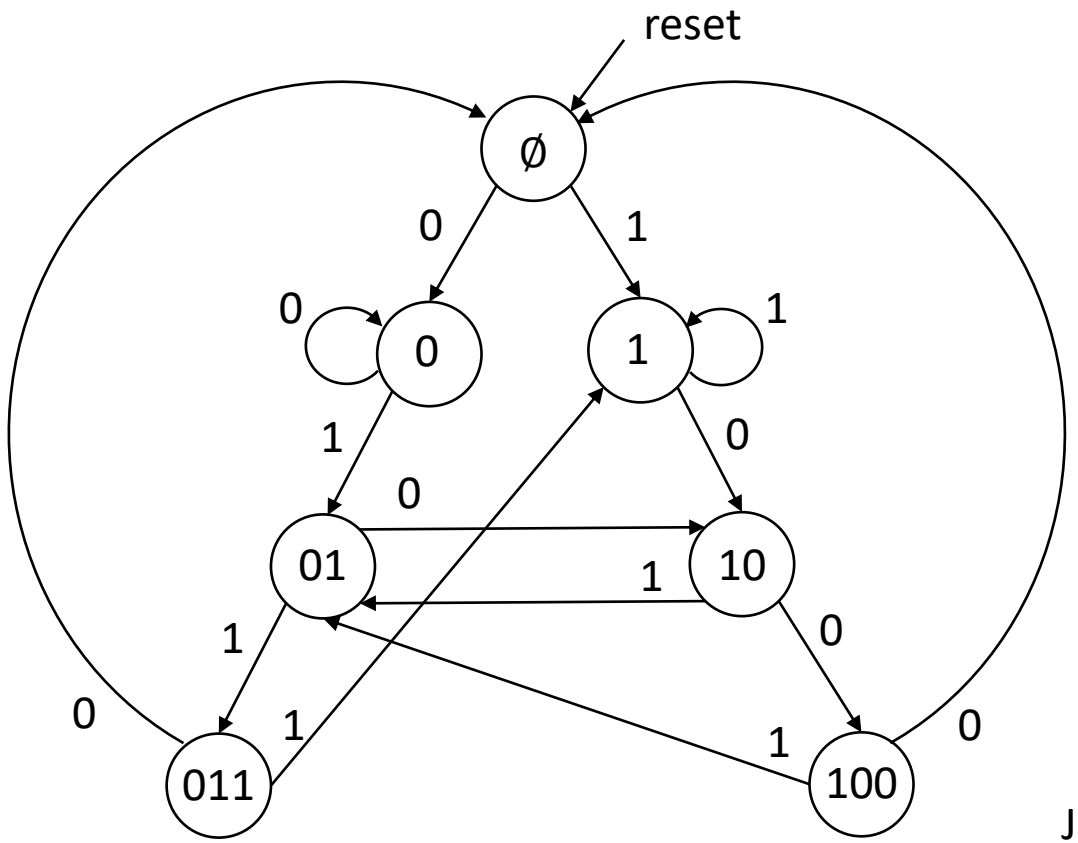
01110
1000



Z prefixu “…1001” nás zajímá pouze “…01”.
In prefix “…1001”, we only care about “…01”.

Návrh krok po kroku
Step-by-step design

0110
1000

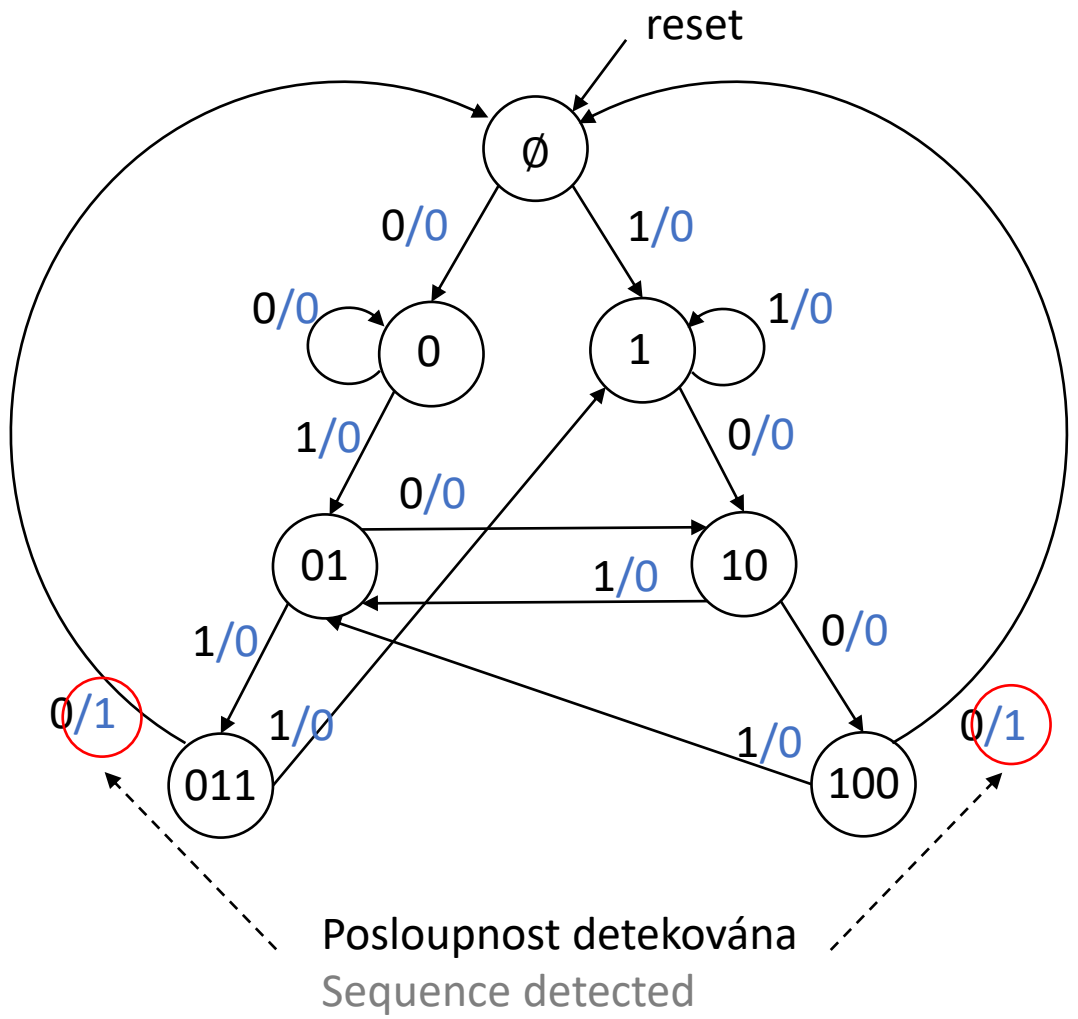


Ještě nám chybí vyřešit výstup.
We still need to take care of the output.

Návrh krok po kroku
Step-by-step design

Pro Mealyho automat stačí doplnit výstup na hrany.
For Mealy machine, it's enough to complete output on the edges.

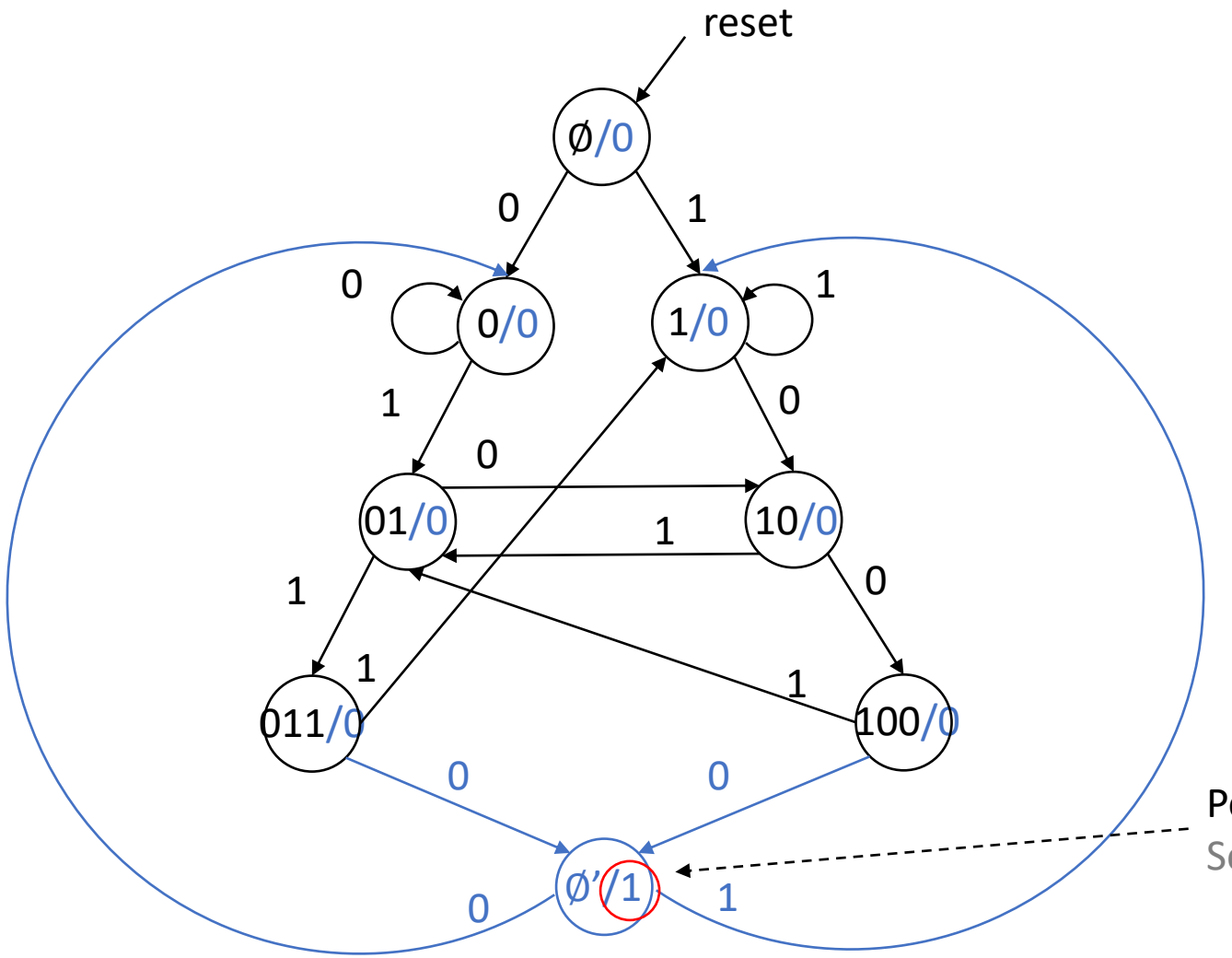
0110
1000



Návrh krok po kroku
Step-by-step design

Pro Moorův automat je nutné vytvořit další stav.
For Moore machine, another state is necessary.

0110
1000



Posloupnost detekována
Sequence detected

Algoritmus pro návrh
Design algorithm

- Vstup: hledané posloupnosti
- Výstup: graf přechodů
- Každý stav reprezentuje nějaký aktuálně načtený prefix
- Po resetu začínáme ve stavu, kdy není zatím načteno nic: Ø
- Dokud nejsou definované všechny přechody (hrany), opakuj:
  - Vyber nějaký nedefinovaný přechod
  - Zřetěz vybraný vstup přechodu (0,1) s prefixem stavu, ze kterého přechod vychází
  - Oříznì výsledek zřetězení na nejdelší část, která se překrývá s alespoň jednou hledanou posloupností
  - Přechod veď do stavu, který reprezentuje výsledek ořezu
    - **Pokud se ořez rovná hledané posloupnosti**, a ty se nesmějí překrývat, pokračuj do stavu Ø. Pokud se posloupnosti smějí překrývat, ořízni prefix tak, aby se překrýval pouze ostatními (zatím nedetekovanými) posloupnostmi.

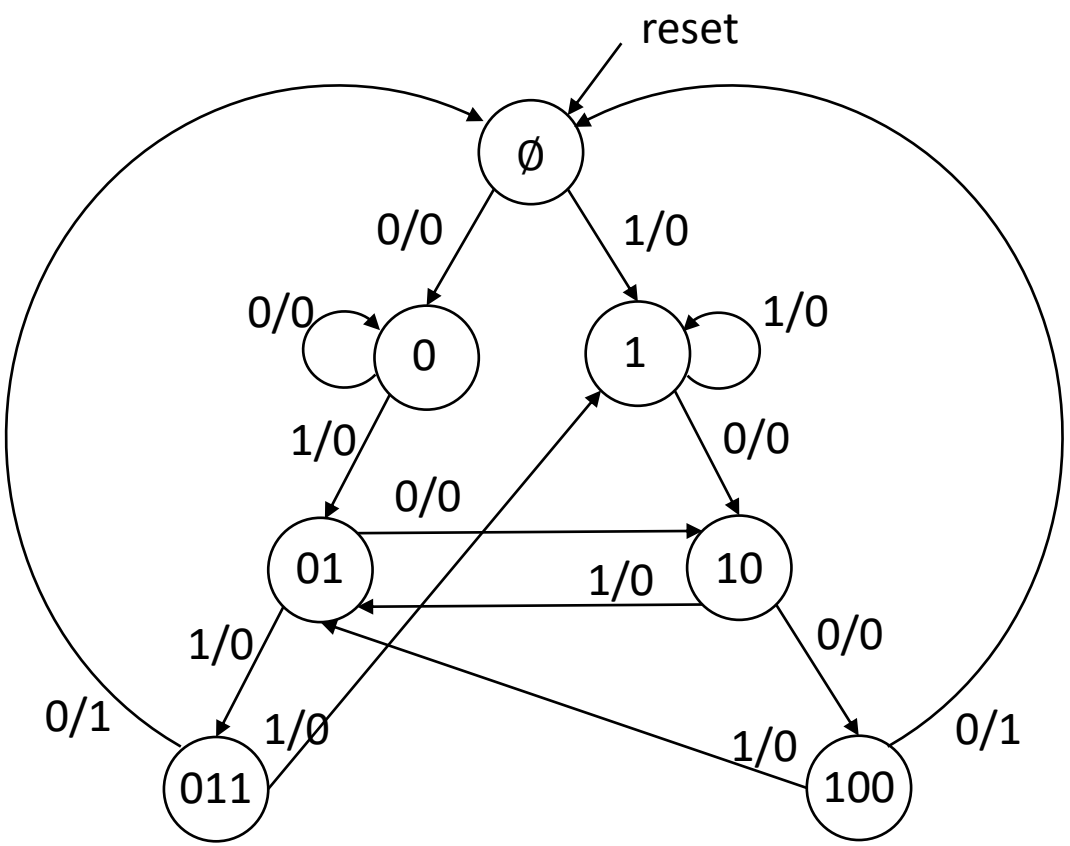
- Input: searched sequences
- Output: state-transition graph
- Every state represents some currently read prefix
- After reset, we begin in the state where nothing is read: Ø
- Until all transitions (edges) are defined, repeat:
  - Select an undefined transition
  - Concatenate selected input (0,1) with prefix of the state, in which the transition origins
  - Crop the concatenation result to the longest part that overlaps with at least one searched sequence
  - End the transition in a state, which represents the result of the crop
    - **If the crop is equal to the searched sequence**, and they must not overlap, go to state Ø. If the sequences may overlap, crop the prefix so that it overlays only with other (yet undetected) sequences.

Algoritmus pro implementaci
Implementation algorithm

1. Graf automatu
2. Tabulky přechodů a výstupů
3. Zakódování vnitřních stavů, vstupů a výstupů
4. Minimalizace výrazů pro přechodovou a výstupní kombinační logiku
5. Implementace pomocí dostupných stavebních bloků (např. hradla, klopné obvody)

1. State-transition graph
2. Tables of transitions and outputs
3. Encoding of inner states, inputs and outputs
4. Minimization of expressions for transition and output combinational logic
5. Implementation using available building blocks (e.g., logic gates, flip flops)

Implementace Mealy
Mealy implementation



Tabulka přechodů
Transitions table

Graf automatu
State-transition graph

Tabulka výstupů
Output table

State	Next state (IN=0)	Next state (IN=1)
“Ø”	“0”	“1”
“0”	“0”	“01”
“1”	“10”	“1”
“01”	“10”	“011”
“10”	“100”	“01”
“011”	“Ø”	“1”
“100”	“Ø”	“01”

State	DETECT (IN=0)	DETECT (IN=1)
“Ø”	0	0
“0”	0	0
“1”	0	0
“01”	0	0
“10”	0	0
“011”	1	0
“100”	1	0

Implementace Mealy
Mealy implementation

Přeuspořádaná tabulka přechodů a výstupů
Reorganized table of transitions and outputs

Tabulka přechodů
Transitions table

State	Next state (IN=0)	Next state (IN=1)
“Ø”	“0”	“1”
“0”	“0”	“01”
“1”	“10”	“1”
“01”	“10”	“011”
“10”	“100”	“01”
“011”	“Ø”	“1”
“100”	“Ø”	“01”

Tabulka výstupů
Output table

State	DETECT (IN=0)	DETECT (IN=1)
“Ø”	0	0
“0”	0	0
“1”	0	0
“01”	0	0
“10”	0	0
“011”	1	0
“100”	1	0

State, IN	Next state	DETECT
“Ø”, 0	“0”	0
“Ø”, 1	“1”	0
“0”, 0	“0”	0
“0”, 1	“01”	0
“1”, 0	“10”	0
“1”, 1	“1”	0
“01”, 0	“10”	0
“01”, 1	“011”	0
“10”, 0	“100”	0
“10”, 1	“01”	0
“011”, 0	“Ø”	1
“011”, 1	“1”	0
“100”, 0	“Ø”	1
“100”, 1	“01”	0

Implementace Mealy
Mealy implementation

State, IN	Next state	DETECT
“Ø”, 0	“0”	0
“Ø”, 1	“1”	0
“0”, 0	“0”	0
“0”, 1	“01”	0
“1”, 0	“10”	0
“1”, 1	“1”	0
“01”, 0	“10”	0
“01”, 1	“011”	0
“10”, 0	“100”	0
“10”, 1	“01”	0
“011”, 0	“Ø”	1
“011”, 1	“1”	0
“100”, 0	“Ø”	1
“100”, 1	“01”	0

State	Encoding
“Ø”	000
“0”	001
“1”	010
“01”	011
“10”	100
“011”	101
“100”	110

Zakódovaná tabulka přechodů a výstupů

Encoded table of transitions and outputs

State, IN	Next state	DETECT
000, 0	001	0
000, 1	010	0
001, 0	001	0
001, 1	011	0
010, 0	100	0
010, 1	010	0
011, 0	100	0
011, 1	101	0
100, 0	110	0
100, 1	011	0
101, 0	000	1
101, 1	010	0
110, 0	000	1
110, 1	011	0

Implementace Mealy
Mealy implementation

Přechodová funkce
Transition function

Výstupní funkce
Output function

State, IN	Next state	DETECT
000, 0	001	0
000, 1	010	0
001, 0	001	0
001, 1	011	0
010, 0	100	0
010, 1	010	0
011, 0	100	0
011, 1	101	0
100, 0	110	0
100, 1	011	0
101, 0	000	1
101, 1	010	0
110, 0	000	1
110, 1	011	0

q_2	q_1	q_0	IN	d_2	d_1	d_0	DETECT
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0
0	0	1	1	0	1	1	0
0	1	0	0	1	0	0	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	0	0
0	1	1	1	1	0	1	0
1	0	0	0	1	1	0	0
1	0	0	1	0	1	1	0
1	0	1	0	0	0	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	1
1	1	0	1	0	1	1	0

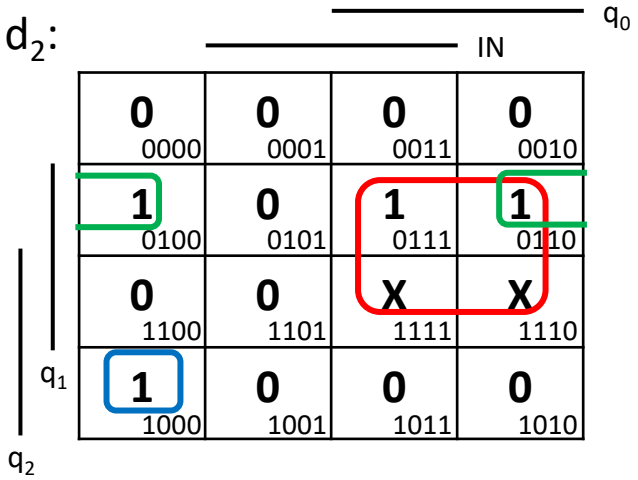
Pravdivostní tabulky
Truth tables (4 in 1)

Implementace Mealy

Mealy implementation

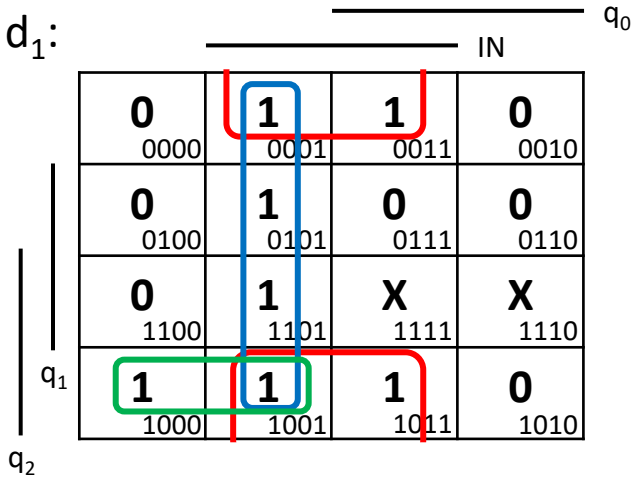
q_2	q_1	q_0	IN	d_2	d_1	d_0	DETECT
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	0	1	0
0	0	1	1	0	1	1	0
0	1	0	0	1	0	0	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	0	0
0	1	1	1	1	0	1	0
1	0	0	0	1	1	0	0
1	0	0	1	0	1	1	0
1	0	1	0	0	0	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	1
1	1	0	1	0	1	1	0

d_2:



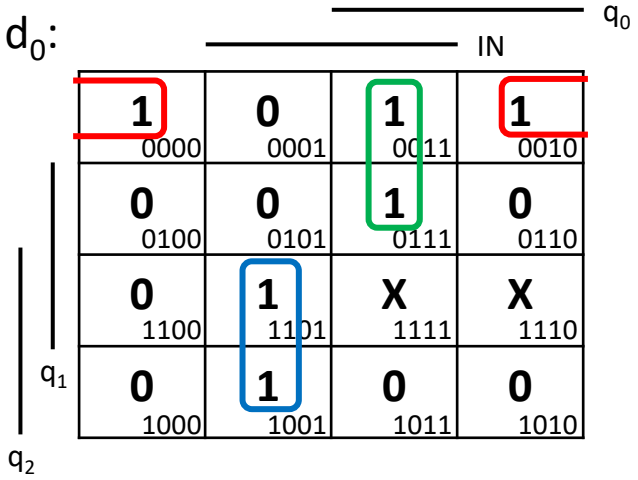
$d_2 = q_0 \cdot q_1 + \overline{q_2} \cdot q_1 \cdot \overline{IN} + q_2 \cdot \overline{q_1} \cdot \overline{q_0} \cdot \overline{IN}$

d_1:



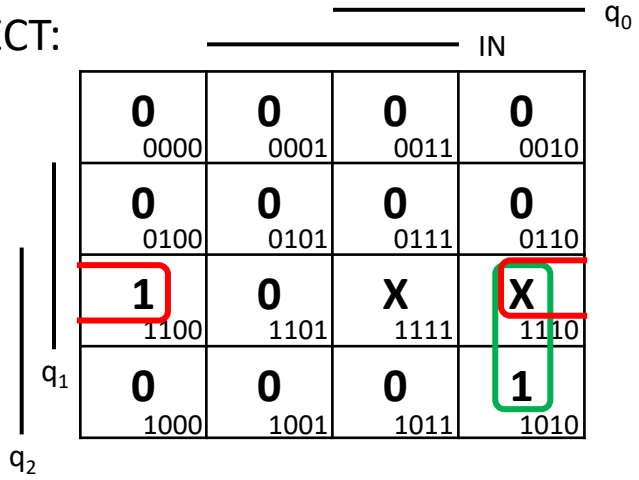
$d_1 = \overline{q_1} \cdot IN + \overline{q_0} \cdot IN + q_2 \cdot \overline{q_1} \cdot \overline{q_0}$

d_0:



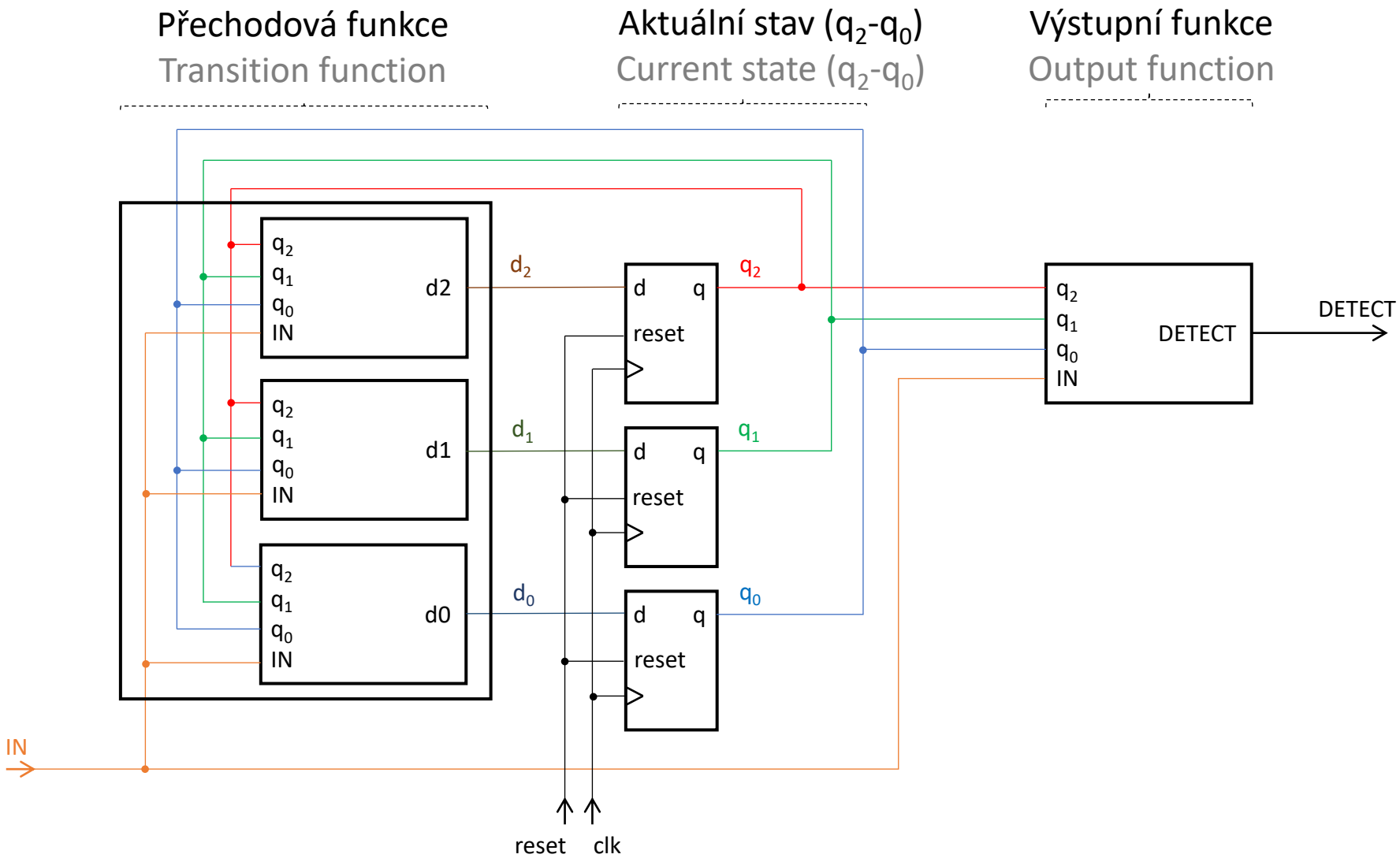
$d_0 = \overline{q_2} \cdot \overline{q_1} \cdot \overline{IN} + \overline{q_2} \cdot q_0 \cdot IN + q_2 \cdot \overline{q_0} \cdot IN$

DETECT:



$DETECT = q_2 \cdot q_1 \cdot \overline{IN} + q_2 \cdot q_0 \cdot \overline{IN}$

Implementace Mealy
Mealy implementation



A to je vše!
That’s all folks!