Chapter 8 Synchronous Sequential Circuits

- Finite State Machine (FSM) (Afr: toestand masjien)
- Combination of combinatorial and sequential logic. Mealy and Moore types.
- Digital control circuits.
- See figure 8.1.

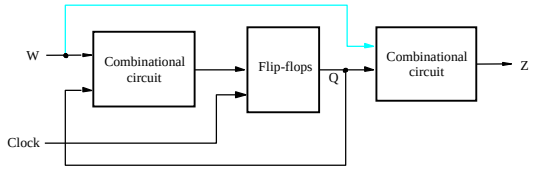


Figure 8.1 The general form of a sequential circuit

8.1 Basic Design Steps

- Example:
  - A circuit has 1 input w and one output z.
  - Use positive edge clock.
  - z = 1 if during two immediate preceding clock cycles the input w was equal to 1. Otherwise z = 0.
  - See figure 8.2.

Clockcycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	0	1	0	0	1	1	0

Figure 8.2 Sequences of input and output signals

8.1.1 State diagram (Afr: Toestanddiagram)

- A designer maps the descriptive specification onto a state diagram.
- Circles represent states (Afr: toestande) and arcs (Afr: boeë) represent transitions (Afr: oorgange of transisies) from one state to another.
- Starting state after reset.
- See figure 8.3.

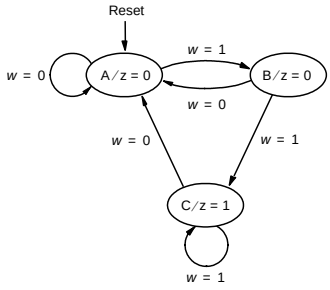


Figure 8.3 State diagram of a simple sequential circuit

8.1.2 State Tables (Afr: Toestandtabel)

- A State Table tabulates Next States as a function all possible Present States and the inputs to the circuit.
- The State Machine moves from the Present State to Next State after each clock transition.
- See figure 8.4.

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	A	B	0
B	A	C	0
C	A	C	1

Figure 8.4 State table

8.1.3 State Assignment

- In an implementation each State is represented by a particular valuation (combination of values) of *State Variables*.
- Each *state variable* is implemented in the form of a *flip-flop*.
- See figure 8.5.

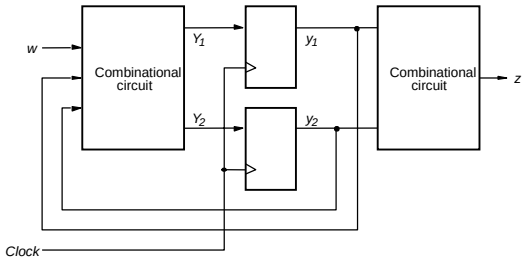


Figure 8.5 A general sequential circuit

State-assignment table

- Similar to the state table, but with state names replaced by state variables.
- y_1 and y_2 are called present-state variables.
- Y_1 and Y_2 are called next-state variables.
- Note don't care state in figure 8.6.

	Present state	Next state		Output
		$w = 0$	$w = 1$	
	y_2y_1	Y_2Y_1	Y_2Y_1	z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Figure 8.6 A State-assigned table

8.1.4 Choice of flip-flops and Derivation of Next-State and Output Expressions

- Using D-type flip-flops is easy and straight forward, because the input is transferred unchanged to the output.
- See figures 8.7 and 8.8 for implementation.

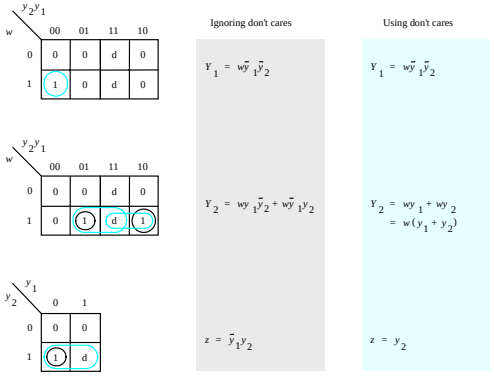


 Figure 8.7 Derivation of logic expressions

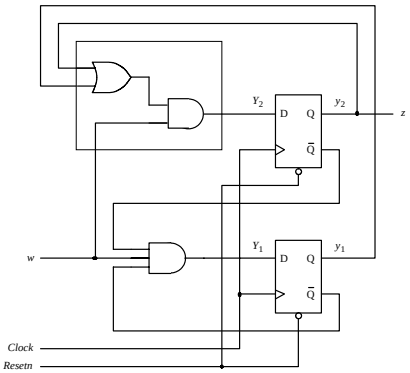


 Figure 8.8 Sequential circuit

8.1.5 Timing diagram

- Figure 8.9 shows the timing diagram, with delays, for the above example.
- (Students make sure you understand this detail.)

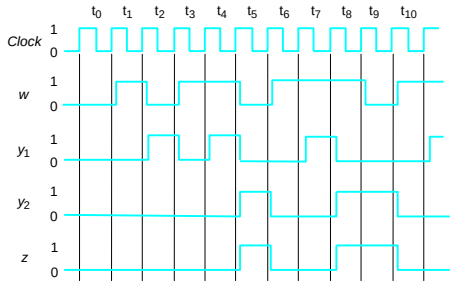


 Figure 8.9 Timing diagram

8.1.6 Summary of design steps

1. Obtain specification.
2. Derive state diagram (engineering skill!).
3. Create state table.
4. Minimize states (later).
5. State assignment.
6. Choose flip-flops.
7. Implement circuit.

Example 8.1

- A slightly more complex example of a *control circuit*.
- A typical instruction in a computer is to swap the values of two registers via a three-state bus.
- See figure 7.56 and 7.57.

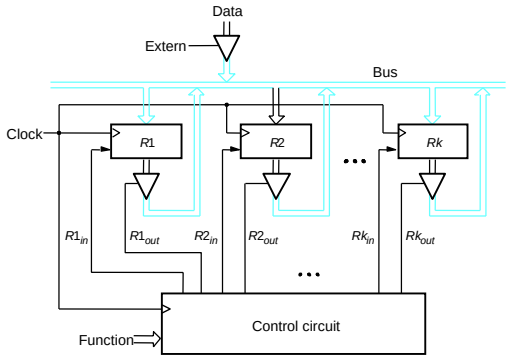


 Figure 7.56 A digital system with *k* registers

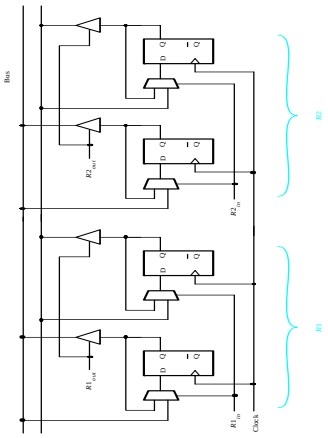


Figure 7.57 Details for connecting a register to a bus

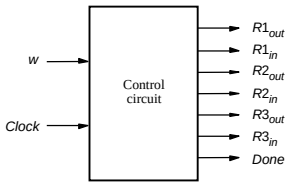


 Figure 8.10 Signals needed in Example 8.1

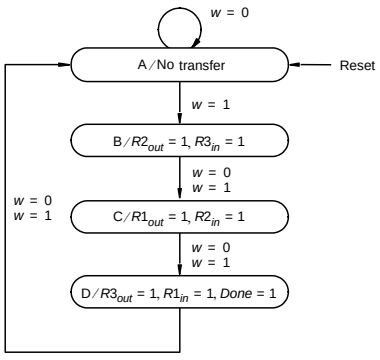


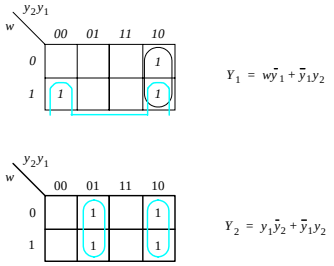
 Figure 8.11 State diagram

Present state	Next state		Outputs						
	$w = 0$	$w = 1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	A	B	0	0	0	0	0	0	0
B	C	C	0	0	1	0	0	1	0
C	D	D	1	0	0	1	0	0	0
D	A	A	0	1	0	0	1	0	1

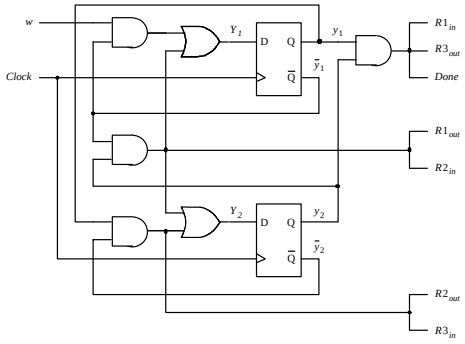
 Figure 8.12 State table

	Present state	Nextstate		Outputs						
		$w=0$	$w=1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	01	0	0	0	0	0	0	0
B	01	10	10	0	0	1	0	0	1	0
C	10	11	11	1	0	0	1	0	0	0
D	11	00	00	0	1	0	0	1	0	1

July 2000 Figure 8.13 State-assigned table 299



July 2000 Figure 8.14 Derivation of next-state expressions 300



July 2000 Figure 8.15 Sequential circuit 301

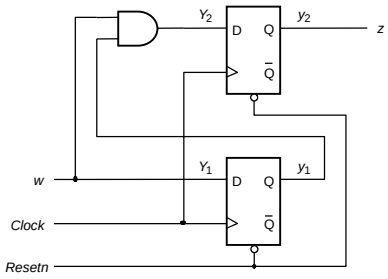
8.2 State assignment problem

- Alternate assignment of values to states can lead to simpler designs, but there is no formal procedure.
- CAD tools use proprietary heuristic techniques.
- See first example in figures 8.16 and 8.17.

July 2000 Univ. of Stellenbosch - Digital Systems 144 302

	Present state	Next state		Output
		$w = 0$	$w = 1$	
	y_2y_1	Y_2Y_1	Y_2Y_1	z
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	dd	dd	d

July 2000 Figure 8.16 Improved state assignment 303



July 2000 Figure 8.17 Final circuit for the improved state assignment 304

8.2.1 One-hot encoding

- One-hot encoding can be used to represent states in stead of binary encoding as used up to now.
- Disadvantage: requires more flip-flops.
- Advantage: simpler output circuit.

	Present state	Nextstate		Output
		$w = 0$	$w = 1$	
	$y_3y_2y_1$	$Y_3Y_2Y_1$	$Y_3Y_2Y_1$	z
A	001	001	010	0
B	010	001	100	0
C	100	001	100	1

Figure 8.20 One-hot state assignment

	Present state	Nextstate		Outputs						
		$w = 0$	$w = 1$							
	$y_4y_3y_2y_1$	$Y_4Y_3Y_2Y_1$	$Y_4Y_3Y_2Y_1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	0 001	0001	0010	0	0	0	0	0	0	0
B	0 010	0100	0100	0	0	1	0	0	1	0
C	0 100	1000	1000	1	0	0	1	0	0	0
D	1 000	0001	0001	0	1	0	0	1	0	1

Figure 8.21 One-hot state assignment

8.3 Mealy State Model

- Moore model – output only a function of the present state.
- Mealy model – output a function of the present state *and* the input(s).
- Example: modification of previous example – see figures 8.22 to 8.26.
- NB. During present clock cycle the output is determined by the state from which the arc emanates and the input value on the arc.

Clock cycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	1	0	0	1	1	0	0

Figure 8.22 Sequences of input and output signals

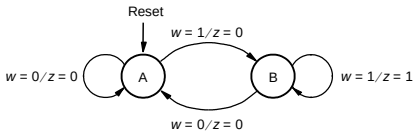


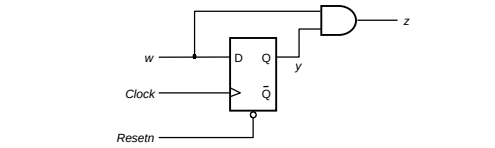
Figure 8.23 State diagram

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	B	0	0
B	A	B	0	1

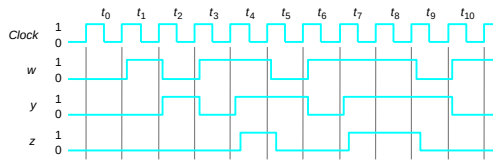
Figure 8.24 State table

	Present state	Next state		Output	
		$w = 0$	$w = 1$	$w = 0$	$w = 1$
	y	Y	Y	z	z
A	0	0	1	0	0
B	1	0	1	0	1

Figure 8.25 State-assigned table



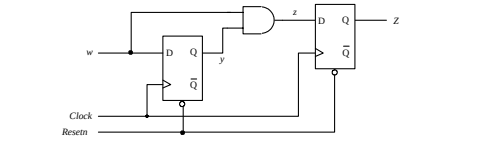
(a) Circuit



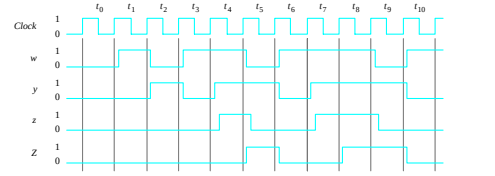
(b) Timing diagram

Figure 8.26 FSM implementation

- If the output pulse is to be delayed by one clock cycle, a D-type flip-flop can be added at the output (figure 8.27).
- This circuit is now equivalent to the one in figure 8.17 and to the original specification.



(a) Circuit



(b) Timing diagram

Figure 8.27 FSM implementation

- Figure 8.28 shows the second example in Mealy form.

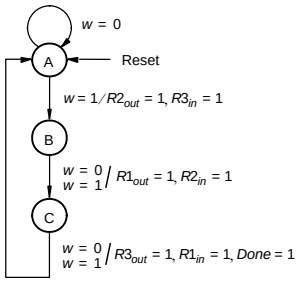


Figure 8.28 State diagram for Example 8.4

8.7 Design of a Counter Using the Sequential circuit Approach

- 8.7.1 State diagram and state table for Modulo-8 counter (figure 8.60, 61)
- 8.7.2 State assignment (figure 8.62)
- 8.7.3 Implementation using D-type flip-flops (figure 8.63, 64)

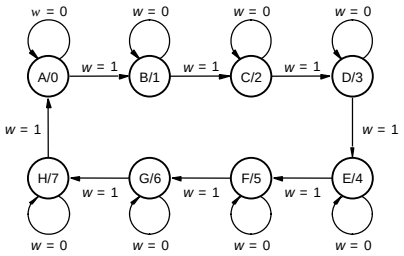


Figure 8.60 State diagram for a counter

Present state	Next state $w = 0$	Next state $w = 1$	Output
A	A	B	0
B	B	C	1
C	C	D	2
D	D	E	3
E	E	F	4
F	F	G	5
G	G	H	6
H	H	A	7

Figure 8.61 State table for the counter

	Present state $y_2y_1y_0$	Next state $w = 0$ $Y_2Y_1Y_0$	Next state $w = 1$ $Y_2Y_1Y_0$	Count $z_2z_1z_0$
A	000	000	001	000
B	001	001	010	001
C	010	010	011	010
D	011	011	100	011
E	100	100	101	100
F	101	101	110	101
G	110	110	111	110
H	111	111	000	111

Figure 8.62 State-assigned table for the counter

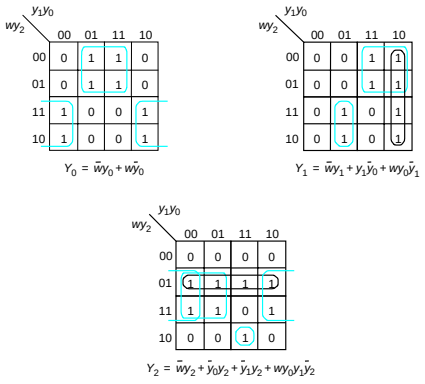


Figure 8.63 Karnaugh maps for D flip-flops for the counter

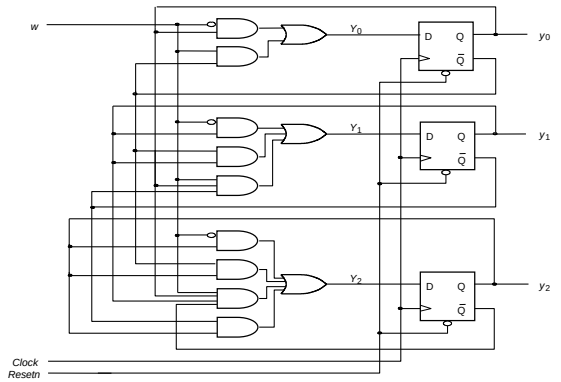


Figure 8.64 Circuit diagram for the counter

8.7.4 Implementation using JK-flip-flops

- JK properties
  - If present state = 0 and we want the next state = 0, then J = 0 and K = d.
  - If present state = 0 and we want the next state = 1, then J = 1 and K = d.
  - If present state = 1 and we want the next state = 1, then J = d and K = 0.
  - If present state = 1 and we want the next state = 0, then J = d and K = 1.

- Implementation of counter with JK's in figures 8.65 – 8.67.
- *Excitation table* (figure 8.65) same as *state assignment table.*

	Present state $y_2y_1y_0$	Flip-flop inputs								Count $z_2z_1z_0$
		$w = 0$				$w = 1$				
		$Y_2Y_1Y_0$	J_2K_2	J_1K_1	J_0K_0	$Y_2Y_1Y_0$	J_2K_2	J_1K_1	J_0K_0	
A	000	000	0d	0d	0d	001	0d	0d	1d	000
B	001	001	0d	0d	d0	010	0d	1d	d1	001
C	010	010	0d	d0	0d	011	0d	d0	1d	010
D	011	011	0d	d0	d0	100	1d	d1	d1	011
E	100	100	d0	0d	0d	101	d0	0d	1d	100
F	101	101	d0	0d	d0	110	d0	1d	d1	101
G	110	110	d0	d0	0d	111	d0	d0	1d	110
H	111	111	d0	d0	d0	000	d1	d1	d1	111

 Figure 8.65 Excitation table for the counter with JK flip-flops

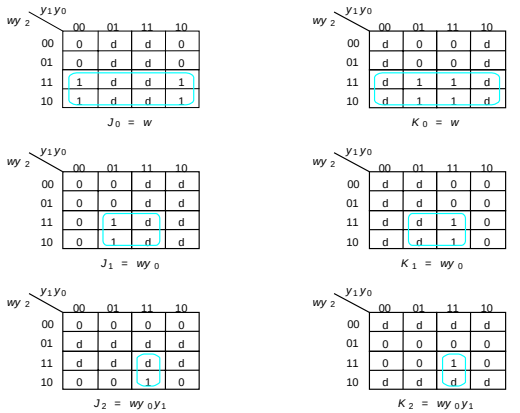


Figure 8.66 Karnaugh maps for JK flip-flops in the counter

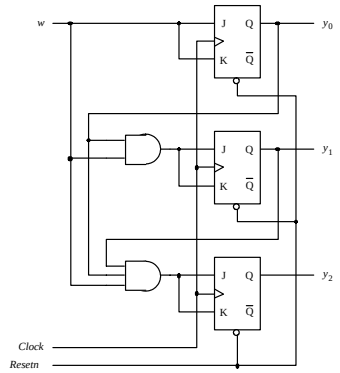
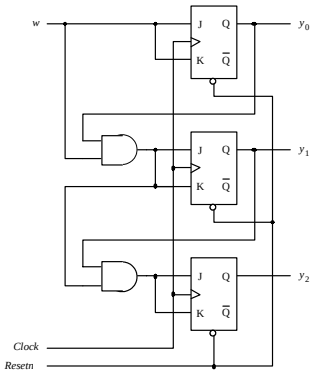


 Figure 8.67 Circuit diagram using JK flip-flops



July 2000 Figure 8.68 Factored-form implementation of the counter 329

Other counters

- Any counter can be designed using state machine principles.
- The most important application of state machines is for the implementation of control circuits.
- Designing state machines in VHDL is a very powerful tool.

The end!