Introduction to Embedded Systems
EHB326E
Lectures

Prof. Dr. Müştak E. Yalçın

Istanbul Technical University

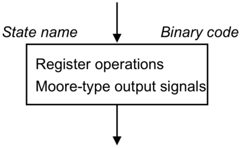
mustak.yalcin@itu.edu.tr

Describing the desired behaviour :Algorithmic State Machines (ASM)

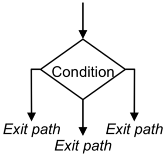
- Flowchart
  - Convenient way to graphically specify sequence of procedural steps and decision paths for algorithm
  - Enumerates sequence of operations and conditions necessary for execution
- Algorithmic State Machine (ASM)
  - Flowchart defined specifically for digital hardware algorithms
- Flowchart vs. ASM
  - Conventional flowchart
    - Sequential way of representing procedural steps and decision paths for algorithm
    - No time relations in corporated
  - ASM chart
    - Representation of sequence of events together with timing relations between states of sequential controller and events occurring while moving between steps

Algorithmic State Machines (ASM)

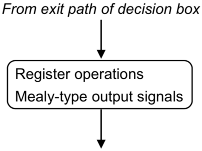
State box



Decision box



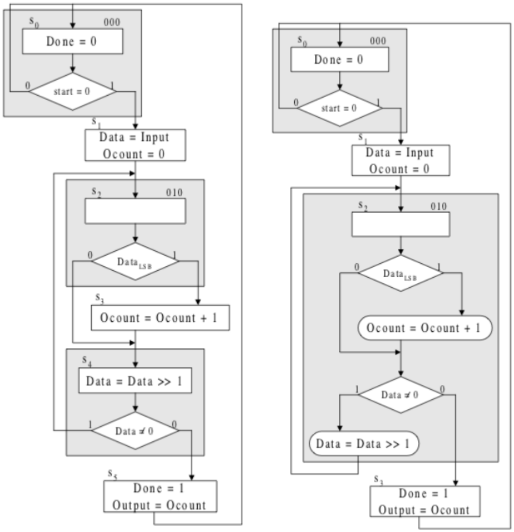
Conditional box



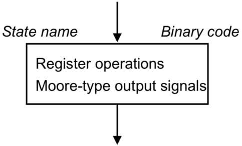
- The chart must define a unique next state for each state and set of condition.
- Every path defined by the network of conditions boxes must lead to another state.

- Block has one entrance and any number of exits paths
- Each block in ASM dedicated to state of system during one clock cycle
- Can label just the "1" and omit the "0"

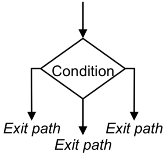
One's counter: State and Input-based charts



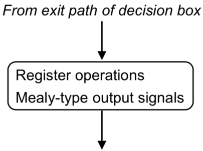
Source: D. Gajski, Principles of Digital Design



- The state represented by the state box takes a clock cycle to complete.
- The output signals in the box take the specified values during this clock cycle.
- The signal (x=1) is assigned during the next clock cycle and holds its value until otherwise set elsewhere.

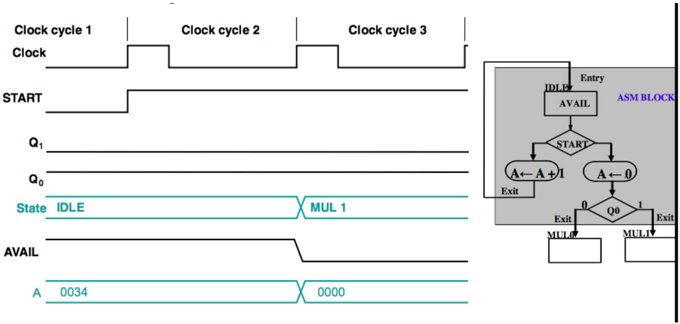


- Decision box has two or more branches.
- Decision is made based on the value of one or more input signals.
- Decision box must follow and be associated with a state box.
- Thus, the decision is made in the same clock cycle as the other actions of the state.
- Hence, the input signals must be available and valid at the start of the clock cycle.

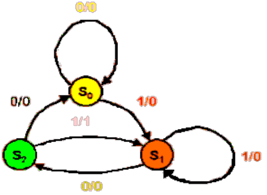


- A conditional output box must follow a decision box.
- A conditional output box is attached to a state box through one or more decision boxes.
- Therefore, the output signals in the conditional output box are asserted in the same clock cycle as those in the state box to whic it is attached.
- The output signals can change during that state as a result of changes on the inputs.
- The conditional output signals a sometimes referred as Mealy outputs since they depend on th input signals as well.

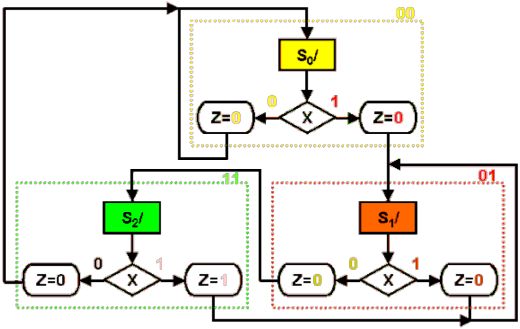
Timing


Clock cycle 1
Clock cycle 2
Clock cycle 3
Clock
START
Q_1
Q_0
State
IDLE
MUL 1
AVAIL
A
0034
0000
Entry
IDLE
AVAIL
ASM BLOCK
START
A← A +1
A ← 0
Exit
0
Q0
1
Exit
Exit
MUL0
MUL1

FSM to ASM



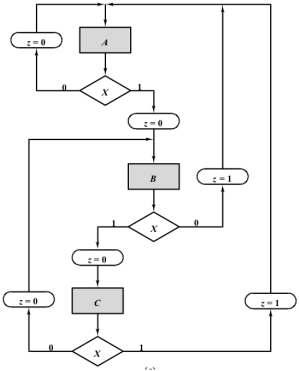
State Diagram for a Sequence Detector

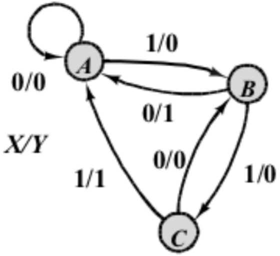


ASM Chart for The Sequence Detector (101)

FSM & ASM

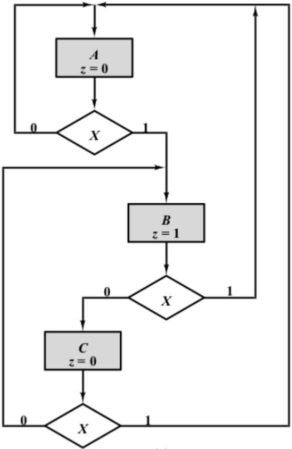
ASM Representation of a Mealy Machine (Input-based):



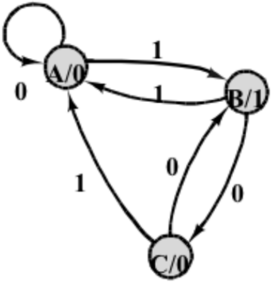


FSM & ASM

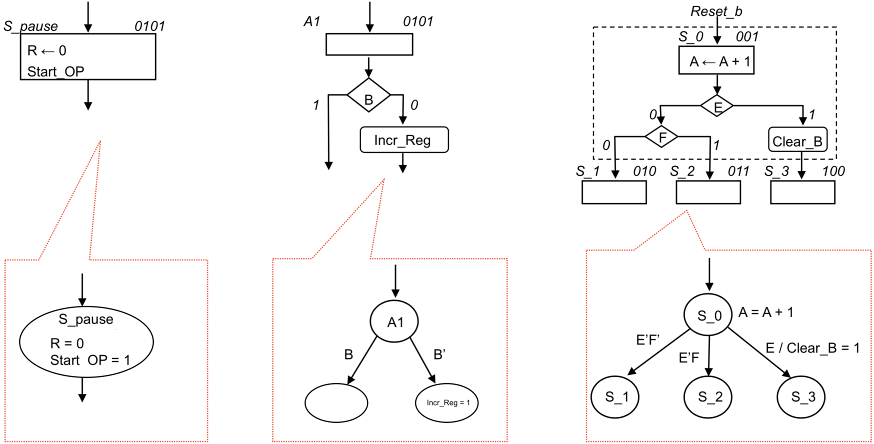
ASM Representation of a Moore Machine (State-based):



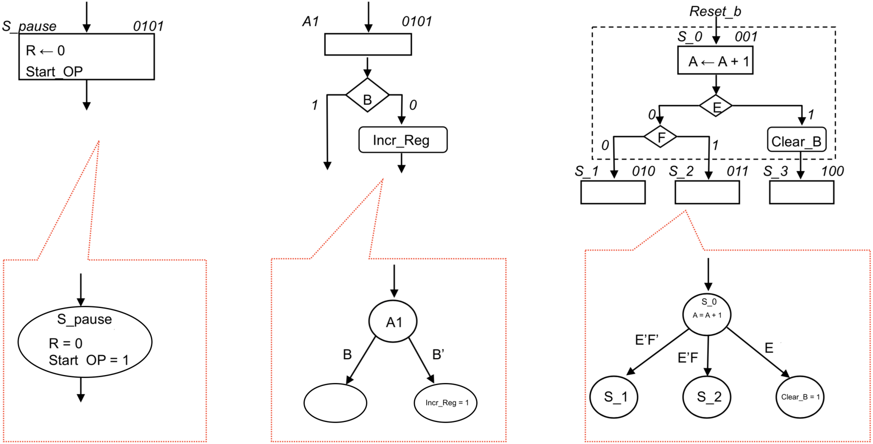
(a)



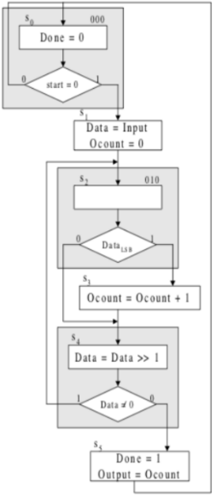
ASMs to FSMDs

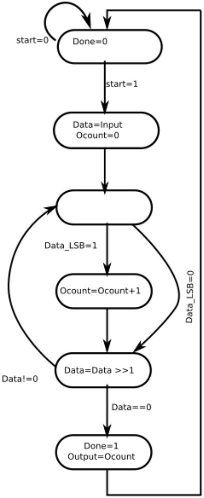


ASMs to FSMDs



Example : ASM to Application-specific processor





Source: D. Gajski, Principles of Digital Design

Design Technology

The system specification:

The designer describes the desired functionality in some language, often a natural language like English, but preferably an executable language !

Designers must spend much time and effort simply understanding and describing the desired behaviour of a system, and some studies have found that most system bugs come from mistakes made describing the desired behaviour rather than from mistakes in implementing that behaviour.

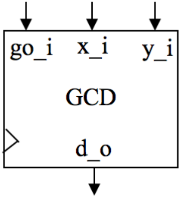
D. Gajski, F. Vahid, S. Narayan and J. Gong, "Specification an Design of embedded systems," page 10 - 13.

Program to FSMD

- The system specification

  `" compute a greatest common divisor of two inputs"`

Black-box view

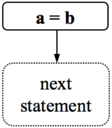


```
0:  int x, y;
1:  while (1) {
2:  while (!go_i);
3:  x = x_i;
4:  y = y_i;
5:  while (x != y) {
6:  if (x < y)
7:  y = y - x;
else
8:  x = x - y;
}
9:  d_o = x;
}
```

Templates for Program statement to FSMD

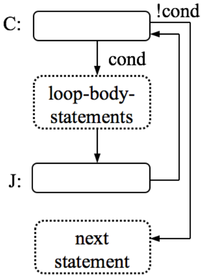
Assignment statement

```
a = b
next statement
```



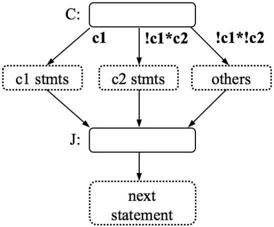
Loop statement

```
while (cond) {
loop-body-
statements
}
next statement
```



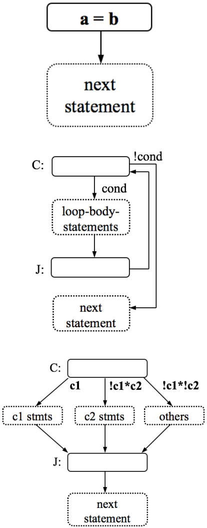
Branch statement

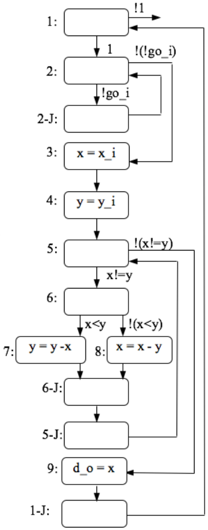
```
if (c1)
c1 stmts
else if c2
c2 stmts
else
other stmts
next statement
```



Algorithm→ Templates → FSMD

```
0:  int x, y;
1:  while (1) {
2:  while (!go_i);
3:  x = x_i;
4:  y = y_i;
5:  while (x !=
y) {
6:  if (x < y)
7:  y = y - x;
else
8:  x = x - y;
}
9:  d_o = x;
}
```

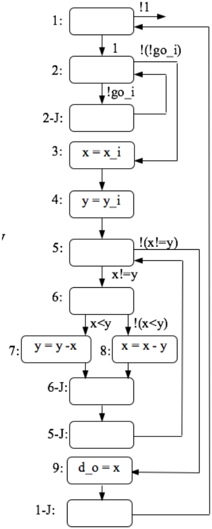


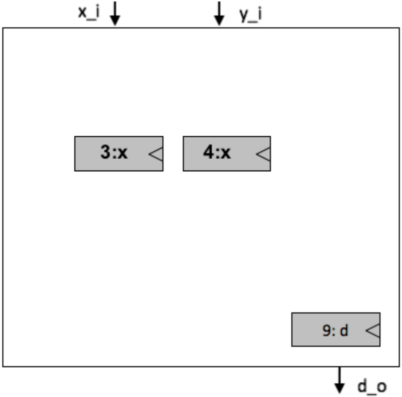


Step$_1$ Behavioral Specification !

Register Transfer Level (RTL) specification (Step$_2$)

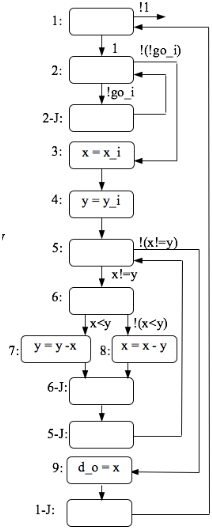
- Define I/O and Create a register for any declared variable

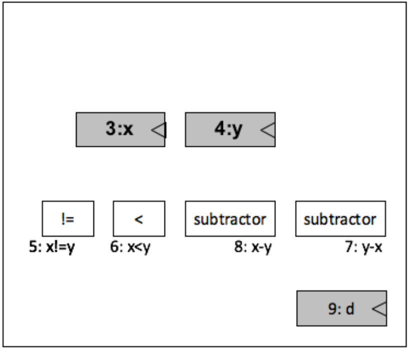




Register Transfer Level (RTL) specification (Step$_2$)

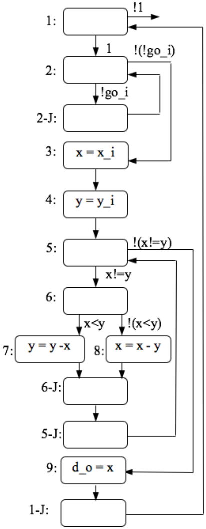
- Create a functional unit for each arithmetic operation

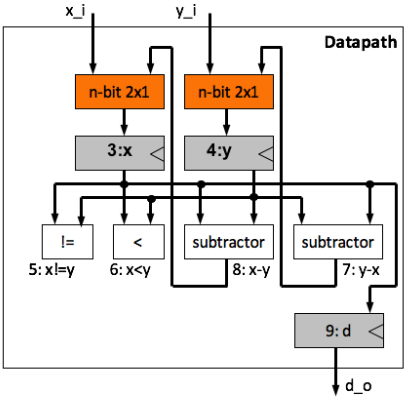




Register Transfer Level (RTL) specification (Step$_2$)

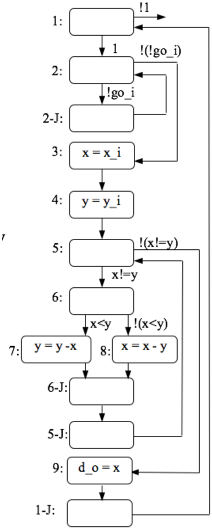
- Connect the ports, registers and functional units

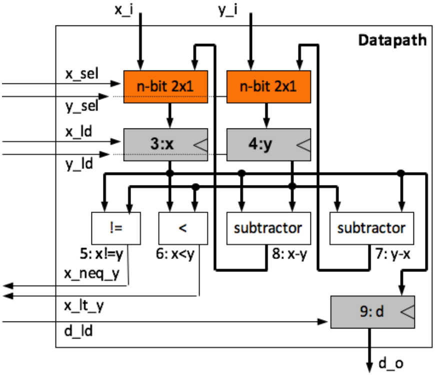




Step3

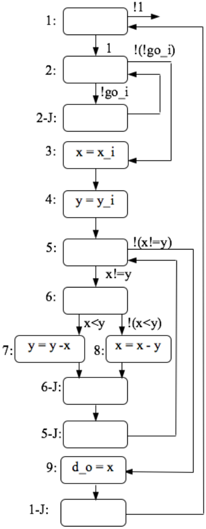
- Create unique identifier

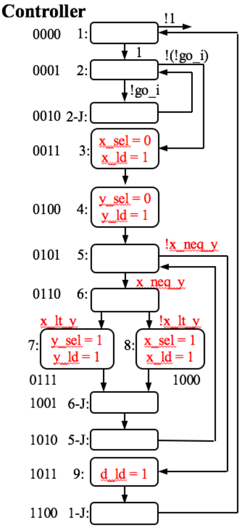




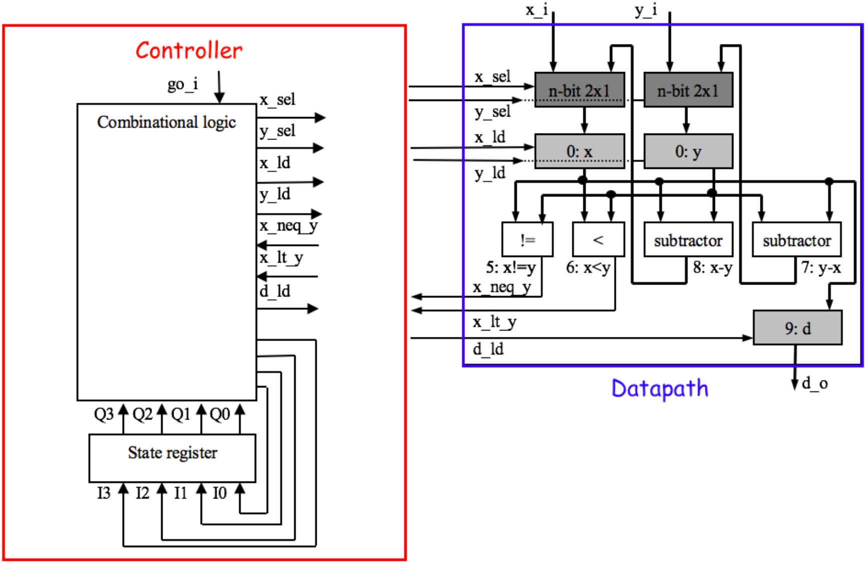
Step$_4$

Creating the controller's FSM: Replace complex actions/conditions with datapath configurations





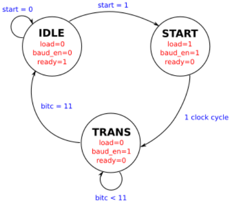
Creating the controller's FSM:



How to design combinational logic in the controller ? Figure 2.12, page: 43

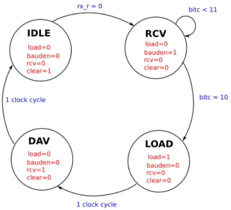
Optimizing single-purpose Processor ! (Read Chp. 2.6)

Asynchronous serial transmitter unit, ▸ Github



▸ Github:uart_tx.v

Asynchronous serial receiver unit, ▸ Github



Source: Juan González-Gómez, GitHub