PROGRAM MODELING

CONCEPTS

UNIT IV
Introduction

- Modelling processes are used for software analysis and design before software implementation.
- A software analysis and design helps
  - A description of the system requirement
  - A description of how system works
  - Shows system validation
- Graphical modelling languages use a **diagram technique** with named symbols that represent concepts and lines that connect the symbols and represent relationships and various other graphical notation to represent constraints.
Five Ways of Modelling an Embedded System

1. Sequential Program Model
2. Data Flow graph (DFG) Model
3. Control Data Flow Graph (CDFG) Model
4. Finite States Machine (FSM) Model
5. Petri Net Model
1. Programming using Sequential functions

• Idea: Use of multiple function calls sequentially
Programming model of Six sequential function calls

- function f1
  Run code
  return

- function f2
  Run code
  return

- function f3
  Run code
  return

- function f4
  Run code
  return

- function f5
  Run code
  return

- function f6
  Run code
  return
ACVM Example Sequential Program Model

• Run function `get_user_input()` for obtaining input for choice of chocolate from child.
• Run function `read_coins()` for reading the coins inserted into ACVM for cost of chocolate.
• Run function `deliver_chocolate()` for delivering the chocolate.
• Run function `display_thanks()` for displaying ‘Collect the nice chocolate. Visit again!’
Sequential Programming Model of an ACVM

Sequential function calls
while () {
get_user_input ();
read_coins ();
deliver_chocolate ();
display_thanks ();};
2. Data Flow graph (DFG)

• A program is modelled as handling the input data streams and creating output data streams.

• A data-flow means that a program flow such that specifically the data only determines all the program execution steps and program flows are determined specifically only by the data.

• Programmer predetermines the data inputs and designs the programming steps to generate the data output.
Data Flow Graph (DFG) Model for Program Analysis

• A set of data-input generates a set of data-output and another set of input will generate another output
Example of Data Flow graph (DFG)

• A program for finding an *average* of grades in various subjects will have the data inputs of grades and data output of the *average*.

• Program executes a function to generate the appropriate output.

• The data flow graph model is appropriate to model the program for the average.
Data Flow graph (DFG) Modelling

- A circle represents each process (set of program steps) in DFG.
- An arrow directed towards the circle represents the data input (or set of inputs) and an arrow originating from the circle represents a data output (or a set of outputs).
- Data input along an input edge is considered as token.
- An input edge has at least one token.
Data Flow graph (DFG) Modelling

- The circle represents the node.
- The node is said to be fired by the tokens from all input edges.
- The output is considered by outgoing tokens, which are produced by the node on firing.
Data Flow graph (DFG) Modeling

• There are no control conditions in steps at DFG

• A DFG does not have any conditions within it so that the program has one data entry point and one data output point.

• There is only one independent path for program flow when program is executed
Fig. 6.4 (a) Data flow graph (DFG) for a process for the sixth finite impulse response (FIR) sequence. (b) DFG for a set of processes of the same sequence for an FIR filter with 6 inputs and 6 coefficients.
Data Flow graph (DFG) Modelling

• Data flow graph models help in a simple code design.

• A simple code design can be defined as that in which the program mostly breaks into DFGs.

• A DFG models a fundamental program element having an independent path.
Data Flow graph (DFG) Modelling

• A DFG gives that unit of a system, which has no control conditions and thus a single path for the program flow.
3. Control Data Flow Graph (CDFG) Modelling

• A control flow means that specifically *only the program determines* all program execution steps and the flow of a program

• The programmer predetermines these steps
Control Data Flow Graph (CDFG) Modelling

• The steps may have loops or condition statements in-between.

• Data that is input generate the data output after a control data-flow as per controlling conditions.

• Output(s) and paths taken after the steps depends on the control statements for various decisions in a process.
Control Data Flow Graph (CDFG) Modelling

• In the CDFG model, there is a diagram, which graphically represents the conditions and the program flow along a condition dependent path.

• The CDFG diagram also represents the effect of events among the processes and shows which processes are activated on each specific event.
Control Data Flow Graph (CDFG) Modelling

• Here, a variable value changing above a limit or below a limit or falling within a range is also like an event that activates a certain process.
Control Data Flow Graph (CDFG) Modelling

- A circle also represents each process (called node) in a CDFG.
- A directed arrow towards the circle represents the data input (or set of inputs)
- A directed arrow from the circle represents a data output (or a set of outputs).
- A box (square or rectangle with its diagonal axes horizontal and vertical) may represent a condition.
Example 1

• Modelling of the steps (a) Data inputs and Controlling input (decision) nodes shown by test boxes in a CDFG for an FIR filter with ten inputs and 10 coefficients
CDFG Modeling

• Helps in understanding all conditions and in determining the number of paths a program may take.
• Guides us how software to be tested for each path starting from a decision node
• Helps in analysing the program in terms of complexity
4. Finite States Machine (FSM) Model

- A finite state machine is one that has a limited or finite number of possible states.
- A finite state machine can be used both as a development tool for approaching and solving problems and as a formal way of describing the solution for later developers and system maintainers.
- There are a number of ways to show state machines, from simple tables through graphically animated illustrations.
A state machine as:

- A set of input events
- A set of output events
- A set of states
- A function that maps states and input to output
- A function that maps states and inputs to states (which is called a state transition function)
- A description of the initial state
- A description of the output state
Example

- The formal description of this state machine is the following:

**States:**
- A: if input == 1 then { output 0, state = B} else { output 0, state = A}
- B: if input == 1 then { output 0, state = C} else { output 0, state = A}
- C: if input == 1 then { output 0, state = C} else { output 1, state = A}
- The initial state is A.
- The final state is C.
Example: Process State diagram

- Fork (New Job)
- Wakeup (Got I/O)
- Sleeping on I/O
- Schedule Process
- Preempt
- Sleep (Need I/O)
- Running
- Exit
Ex1: State diagram of a Simple Soda Vending Machine
Exercises

• Draw the FSM model / State diagram of ACVM.
• Draw the FSM model / State diagram of Digital Camera.
5. Petri Net Model

Idea: Abstract formal model of information flow

Major use:

Modeling of systems of events in which it is possible for some events to occur concurrently, but there are constraints on the occurrences, precedence, or frequency of these occurrences.
Example

• Consider the computer program shown below. Normally, the instructions would be processed sequentially—first, \( A = 1 \), then \( B = 2 \), and so on. However, notice that there is no logical reason that prevents the first three instructions—\( A = 1; B = 2; C = 3 \)—from being processed in any order or concurrently.

• With the continuing decline of the cost of computer hardware, and processors in particular, there is increasing interest in concurrent processing to achieve greater speed and efficiency.

• The use of **Petri nets**, graph models of concurrent processing, is one method of modelling and studying concurrent processing.
Petri Net as a Graph

:Models static properties of a system

• Graph contains 2 types of nodes
  – Circles (Places)
  – Bars (Transitions)

• Petri net has dynamic properties that result from its execution
  – Markers (Tokens)
  – Tokens are moved by the firing of transitions of the net.
we have modelled the above computer program. Here the events (transitions) are the instructions, and the places represent the conditions under which an instruction can be executed.
A NEW JOB ENTERS THE SYSTEM

A JOB IS ON THE INPUT LIST

THE PROCESSOR IS IDLE

JOB PROCESSING IS STARTED

A JOB IS BEING PROCESSED

JOB PROCESSING IS COMPLETED

A JOB IS ON THE OUTPUT LIST

A JOB LEAVES THE SYSTEM

Figure 6. Modeling of a simple computer system.