CHAPTER 4: DESIGN

This chapter describes the overall system architecture, class design, algorithm designs, and other design issues.

4.1 System Architecture Design

The ATM simulator is created base on the Banyan 4x4 and Banyan 8x8 techniques. Figure 4.1 and Figure 4.2 illustrate the design architecture for both techniques.

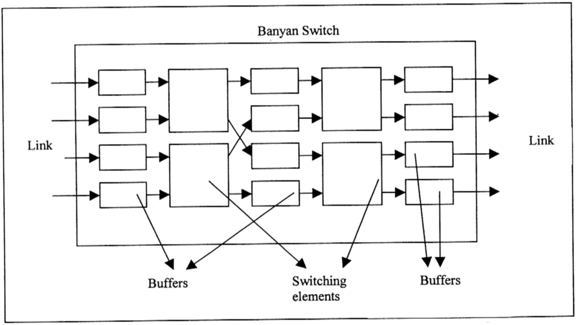

Figure 4.1: Banyan 4x4 Switch Architecture

Switching path from one inlet to one outlet for a switching element need to pay more attention. With referring to figure 4.1, the following describes Banyan 4x4 switching path.

Stage 1 switching:

- Inlet 1 and inlet 2 at first switching element (i.e. the upper left switching element) are connected to inlet 1 and inlet 3 at stage 2 switching elements.

- Inlet 3 and inlet 4 at second switching element (i.e. the lower left switching element) are connected to inlet 2 and inlet 4 at stage 2 switching elements.

Stage 2 switching:

- Inlet 1 and inlet 2 at first switching element (i.e. the upper right switching element) are connected to outlet 1 and outlet 2.
- Inlet 3 and inlet 4 at second switching element (i.e. the lower right switching element) are connected to outlet 3 and outlet 4.

With reference to figure 4.2, the followings describe Banyan 8x8 switching path.

Stage 1 switching:

- Inlet 1 and inlet 2 at first switching element are connected to inlet 1 and inlet 4 at stage 2 switching elements.
- Inlet 3 and inlet 4 at second switching element are connected to inlet 3 and inlet 7 at stage 2 switching elements.
- Inlet 5 and inlet 6 at third switching element are connected to inlet 2 and inlet 6 at stage 2 switching elements.
- Inlet 7 and inlet 8 at forth switching element are connected to inlet 5 and inlet 8 at stage 2 switching elements.

Stage 2 switching:

- Inlet 1 and inlet 2 at first switching element are connected to inlet 1 and inlet 3 at stage 3 switching elements.
- Inlet 3 and inlet 4 at second switching element are connected to inlet 2 and inlet 4 at stage 3 switching elements.
- Inlet 5 and inlet 6 at third switching element are connected to inlet 5 and inlet 7 at stage 3 switching elements.

- Inlet 7 and inlet 8 at forth switching element are connected to inlet 6 and inlet 8 at stage 3 switching elements.

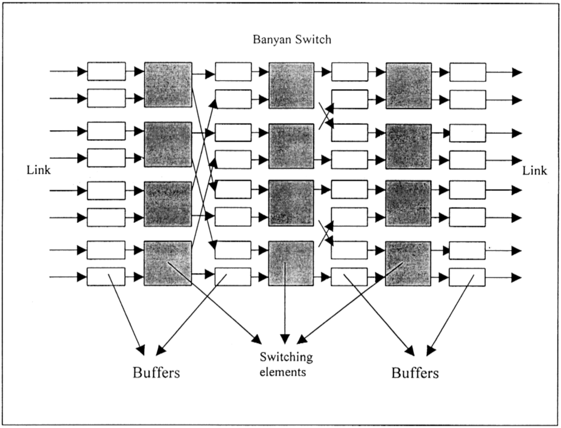

Figure 4.2: Banyan 8x8 Switch Architecture

Stage 3 switching:

- Inlet 1 and inlet 2 at first switching element are connected to outlet 1 and outlet 2.
- Inlet 3 and inlet 4 at second switching element are connected to outlet 3 and outlet 4.
- Inlet 5 and inlet 6 at third switching element are connected to outlet 5 and outlet 6.
- Inlet 7 and inlet 8 at forth switching element are connected to outlet 7 and outlet 8.

4.2 Object-Oriented Design

Figure 4.3 and figure 4.4 are the class diagrams for ATM Switching simulator. *Switch* inherits *Thread;* therefore it is a thread, which inherits *Thread. Banyan Switch* is a specific switching architecture inherits *Switch.* It consists of three important attributes, i.e. *Buffers, Injected Header Cell, Routing Table.*

Buffer consists of a *Queue* of *Queue Node* and the contain of *Queue Node* is *ATM Cell. Switch* .In other words, the buffer consists of a *Queue* of *ATM Cell. Injected Header Cell* is used to store the information for each head-of-line cell in the corresponding input buffers. There is no containment or referential to other object for *Injected Header Cell.* Lastly, *Routing Table* consist of nine *Queue* of *Queue Node*. The contains inside these *Queue Node are* Input Port, Input VPI, Input VCI, Output Port, Output VPI, Output VCI, PCR, MCR, and TCT.

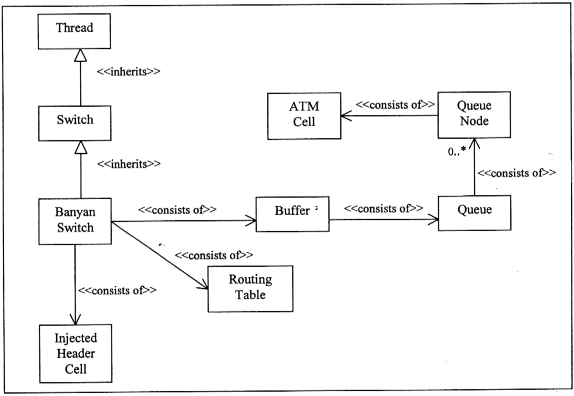

Figure 4.3: Class Diagram I

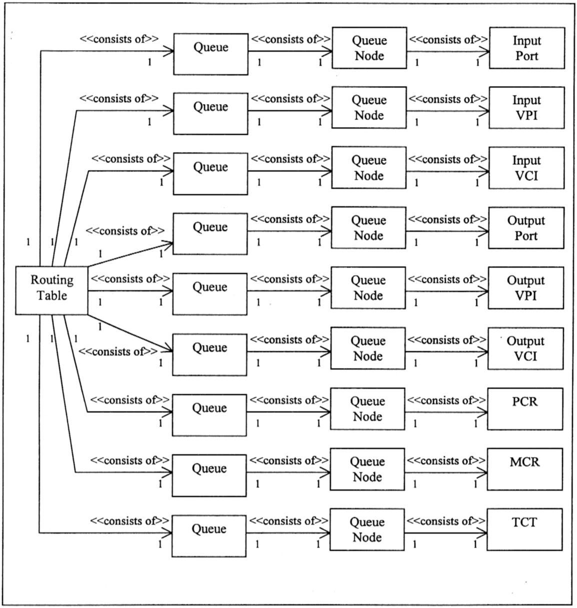

Figure 4.4: Class Diagram II

4.3 Class Design

This section gives a description on classes design in this project. Among the classes highlighted are *Node, Queue Node, ATM Cell, Routing Table, ATM Switch, Banyan Switch, Buffer,* and *Injected Header Cell.* At the end of this section, a table that summarises the major attributes and functions within the corresponding classes is presented.

Class Node and Queue Node

Both class *Queue* and *Queue Node* are used together to form the enhanced linked list structure. As the name implied, *Queue Node* is the item consisted within *Queue.* It is just a simple class with two attributes: *Object* for storing information and *Next Queue Node* for referring to next node. The use of *Object* in *Queue Node* brings the advantage that the *Queue* becomes a powerful linked list which can store different types of items.

A *Queue* can has dynamic length. This allows the length of the *Queue* can grow without limits by adding new *Node* to the *Queue. Queue Node* can be removed from the front, back any position of the *Queue.* Also, Any *Queue Node* within the *Queue* can be retrieved without removing it from *Queue.* Another extra function is the ability to modify the information of a *Queue Node* at any position in the *Queue.*

Class *Queue* is used in class Buffer to store the contained ATM cells, therefore Each *Object* in *Queue Node* stores the information for one ATM cell. Class *Queue* also used in Class *Routing Table* where records stored in *Routing Table* can range from zero up to infinite value.

Class ATM Cell

ATM cell is the main resource class for this simulator. It is containment in class *Buffer* and manipulated by other classes. The main contribution of *ATM Cell* is to provides necessary information for ATM switching. Major attributes are virtual path identifier and virtual channel identifier (i.e. pair value of *VPI/VCI*) which provide switching information and *MCR* which is used for controlling switching rate.

Class Routing Table

Class *Routing table* serve two main purposes. Firstly, it contains the information for internal switching, i.e. the output port for a particular ATM application. Secondly, it provides the information to the *ATM switch* for switching rate control purpose.

A *Routing Table* contains an amount of nine linked list (which is class *Queue*) to store the corresponding information for an ATM application. These information are: input port number, input VPI value, input VCI value, output port number, output VPI value after translation, output VCI value after translation, PCR, MCR, and TCT (total cells transferred in current second).

Input port number, input VPI value, and input VCI value specify an ATM application with VPI/VCI is come from that input port and is addressed to the corresponding output port. In this simulation, the VPI/VCI value need not be changed (why the value need not be changed has been explained in section 4.4.2). However, the output VPI and output VCI are included for future extensible.

PCR and *MCR* have been converted from unit *bit per second* to *cell per second* for the ease of calculation. *TCT* provides the information for how many cells for particular ATM application has been transmitted within current second. After every one second time, the value of *TCT* should be reset to zero.

Input Port	Input VPI	Input VCI	Output Port	Output VPI	Output VCI	PCR	MCR	TCT
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Number of records

Figure 4.5: Routing Table

Class ATM Switch

Class *ATM Switch* is a derived class from *Thread* class as an effort to make the switch as an independent thread in simulation environment. This class contains the general information for an ATM switch to ensure the extensibility of ATM switch to different models in future. For instance, class *Banyan Switch* is one of the ATM switching model inherited from class *ATM Switch.* General attributes in this class are *Switch Size, Switching Rate, Routing Table, Cell Credit, Switching Cell Credit, Clock,* and *Indicator.* The usage of *Switch Size, Switching Rate,* and *Routing Table* are implied by their names. *Switching Cell Credit* and *Cell Credit* are used to indicate the number of cells allowed to be switched within one switching cycle. These attributes will be further elaborated in section 4.4.1. *Clock* simulates physical clock and *Indicator* is used to indicate or signal other event/process to continue or stop. The function *Calculate the time for one second* is necessary for controlling the switching rate, which is measured in cells per second.

Class Banyan Switch

Class *Banyan Switch* is a derived class from *ATM Switch.* This class responsible for switching within an ATM switch. In here, Banyan 4x4 and Banyan 8x8 architecture are implemented. This class allows the adding of new Banyan NxN architecture by just adding the source code into the class. Several important attributes are

- *Total Stages* – total stages within Banyan, For Banyan NxN, the number of stages is $\log_2 N$
- *Buffers* – Total number of buffers within a *Banyan Switch* should be *Switch Size* x (*Total Stages* + *1*). For instances, Banyan 4x4 with *Switch Size* = *4* and *Total Stages* = *2* consists of 12 buffers (4 input buffers, 4 middle buffers between stage 1 and stage 2, and 4 output buffers). Banyan 8x8 with *Switch Size* = *8* and *Total Stages* = *3* consists of 32 buffers (8 input buffers, 8 middle buffers between stage 1 and stage 2, 8 middle buffers between stage 2 and stage 3, and 8 output buffers).

- *Injected Header Cell* – stores switching information for ATM cells.

Several important functions design here are:

- *Switch Cell* – perform switching upon the condition of PCR, MCR, and PCR.
- *Routing within Switching Element* – perform switching for a switching element.
- *Banyan 4x4 Switching* – perform Banyan 4x4 switching.
- *Banyan 8x8 Switching* – perform Banyan 8x8 switching.

Class Buffer

Class *Buffer* is used for both input buffers and output buffers. Every inlet of switching elements consists of an input buffer and one output buffer for the outlet of the last stage's switching elements. Class *Buffer* primarily contains *Queue* of *ATM Cell.* The attributes maximum buffer size while current buffer size are used to store the information for buffer length and number of cells contained in buffer respectively.

Class Injected Header Cell

Class *Injected header cell* stores the information for the first cell in the buffer. These information are useful to instruct the switching element about VPI/VCI and destination of the cell. Attribute *Activity* within this class indicates that the buffer contains cell and *Destination* is the outgoing port for that cell with the associated *VPI/VCI.* Every time before the switching is performed, the information of the ATM cell will be copied to this class.

Table 4.1 Class Design

Class	Major Attributes	Major Functions
Queue Node	Object	Set Object
	Next queue node	Get next node
		Set next node
Queue	First node	Get node information based on position in Queue

	Last node	Get number of nodes
	Number of nodes	Insert node at front
		Insert node at back
		Delete node based on position in Queue
		Update node's information based on position in Queue
ATM cell	VPI	Get VPI, Set VPI
	VCI	Get VCI, Set VCI
	MCR	Get MCR
Routing table	List of input port	Get output port for particular VPI/VCI
	List of input VPI	Increase TCT value
	List of input VCI	Input all the necessary information into routing table
	List of output port	Reset TCT
	List of output VPI	Get number of records
	List of output VCI	
	List of PCR	
	List of MCR	
	List of TCT	
	Number of records	
ATM Switch	Switch size	Calculate the time for one second
	Switching rate	
	Routing table	
	Cell credit	
	Switching cell credit	
	Clock	
	Indicator	
Banyan switch	Total stages	Switch cell
	Buffers	Switching within switching element
	Injected Header Cell	Banyan4x4 switching
		Banyan8x8 switching

Buffer	Maximum buffer size	Get the first ATM cell in this buffer
	Current buffer size	Insert ATM cell into this buffer
	Queue for storing ATM cell	
Injected Header Cell	Activity	Insert first ATM cell's information into injected header cell
	Destination	
	VPI	
	VCI	

4.4 Algorithm Design

Two important algorithms are discussed here. First algorithm describes how switching is performed within a single switching element while second algorithm describes fair switching for two cells addressed to same output.

4.4.1 Switching Within Switching element

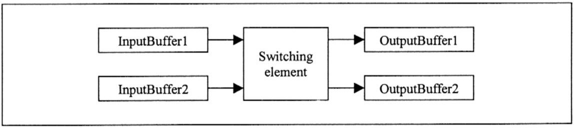

Figure 4.6: A Basic Switching Element

Figure 4.6 illustrates a switching element connected to two input buffers and two output buffers. The algorithm for performing the switching of cell is:

```
If both inputbuffer1 and inputbuffer2 contain cell {
    Get the destination of both cells
    Calculate which output buffer for the two cells
    If both cells are addressed to same output buffer {
        Determine which cell should be switched by Fair-Switching algorithm
        If the addressed output buffer is not full switch that cell
```

```
        }
        Else if both cells are addressed to different output buffer {
            If the inputbuffer1's cell is addressed to outputbuffer1 {
                If outputbuffer1 is not full switch the inputbuffer1's cell
                If outputbuffer2 is not full switch the inputbuffer2's cell
            }
            Else if inputbuffer1's cell is designated to outputbuffer2 {
                If outputbuffer1 is not full switch the inputbuffer2's cell
                If outputbuffer2 is not full switch the inputbuffer1's cell
            }
        }
    Else if only inputbuffer1 contains cell {
        Get the destination of the cell
        Calculate the output buffer
        If the cell is designated to outputbuffer1 and outputbuffer1 is not full
            Switch that cell
        If the cell is addressed to outputbuffer2 and outputbuffer2 is not full
            Switch that cell
    }
    Else if only inputbuffer2 contains cell {
        Get the destination of the cell
        Calculate the output buffer
        If the cell is addressed to outputbuffer1 and outputbuffer1 is not full
            Switch that cell
        If the cell is addressed to outputbuffer2 and outputbuffer2 is not full
            Switch that cell
    }
```

4.4.2 Fair-Switching

Refer to figure 4.6 again; Fair-switching algorithm will take the PCR, MCR and TCT value for both cells in inputbuffer1 and inputbuffer2 for calculation. Assuming that upperPCR, upperMCR, upperTCT are the PCR, MCR, and TCT values for inputbuffer1's PCR, MCR, and TCT respectively. lowerPCR, lowerMCR, and lowerTCT are the PCR, MCR, and TCT values for inputbuffer2's PCR, MCR, and TCT respectively.

```
If both TCT are less than MCR {
    Calculate ratio of TCT/MCR
    Select cell with smaller TCT/MCR value
}
Else if only upperTCT is less than upperMCR
```

Select inputbuffer1's cell
Else if only lowerTCT is less than lowerMCR
Select inputbuffer2's cell
Else if both the TCT are higher or equal to PCR
Neither one is selected
Else if only upperTCT is greater or equal than upperPCR
Select inputbuffer2's cell
Else if only lowerTCT is greater or equal than lowerPCR
Select inputbuffer1's cell
Else if both are lower than PCR {
Calculate ratio of TCT/PCR
Select cell with smaller TCT/PCR value
}

The algorithm guarantees a more fairness switching compare to other method like round robin. Another important feature is that, although both input buffers are containing cell, there is no guarantee that either one must be selected for transmission. If the TCT value for an ATM application is equal to PCR, the ATM cells belong to this application would not be switched within the current second until next second. This algorithm guarantees that the switching rate for an ATM cell must in between MCR and PCR value.

4.5 Other Designs

Besides system architecture, class design, and algorithm design, there are others design which are described in this section.

4.5.1 ATM Switch Switching Rate Design

The design of ATM switching rate is converted from unit bit per second to cell per second for ease of calculation. The smallest time unit is *tick* and timing information is stored in class *GlobalClock*. One of the attributes is the value of microsecond per tick. Hence, for one tick time, number of cell to switch is:

$$N = \frac{\textit{Microsecond per tick} \text{ X } \textit{Cell per second}}{1000000}$$

If N is less than 1, no cell will be switched within current tick and this value is accumulated for every tick. When N is greater or equal to 1, then a number of round N cell(s) will be switched. The following describes the algorithm, the term used for N is *switchingCellCredit* and number of cell which allowed to be switched is *cellCredit.* There is difference between *switchinCellCredit* and *cellCredit.* *SwitchinCellCredit* is a type of floating point but *cellCredit* is measured in type of integer.

```
while( ATM switch is runngin ) {
    switchingCellCredit = switchingCellCredit + switchingRate
    cellCredit =  Integer value of switchingCellCredit
    switchingCellCredit = switchingCellCredit – cellCredit
    while( cellCredit > 0 ) {
        Perform switching
        Decrease cellCredit by 1
    }
}
```

4.5.2 VPI/VCI Value Assumption

The value of VPI/VCI must be unique among all the input port. This assumption make the switching become easier since no conversion of VPI/VCI value need to be done while maintaining the switching function of ATM switch.

4.6 Summary

This chapter covers the major design issues for the ATM switching simulator. This includes an overview of the system architecture which focuses on Banyan 4x4 and Banyan 8x8 switching architecture. The routing path for a cell from input port to output has to be designed properly to prevent misroute. The class design gives an illustration of the defined attributes and functions for each class. In the following section, a description of the algorithm design is presented. Two main algorithms;

switching within switching element and fair switching are covered here. The end of this chapter discusses the other design issues which have yet been covered.