CHAPTER 7

EXPERIMENTAL RESULTS

7.1 INTRODUCTION

This algorithm has been run in practical routing tables. This forwarding table VLSI design was written in VHDL code and synthesized by Xilinx and downloaded into Spartan 2E. There are many devices (Altera, Actel, cypress) supporting the IP lookups downloading. However, the Spartan 2E is easily available kit and all kinds of software supports this Xilinx. So in this research the Xilinx series Spartan has been preferred for verification.

7.2 BEHAVIORAL DESCRIPTION OF FORWARDING ENGINE

When an IP router receives a packet on one of its input ports, it must decide to which output port the packet will be forwarded. To make this decision, it has to match the packet's destination address against a database of destination networks and hosts. That address must then be looked up in the IP Routing Table. On backbone routers there are very few routes with prefixes longer than 24-bits.

As soon as the packet's destination address is entered into forwarding engine, the first-level table examines whether all pattern bits are matched with the destination address and the entry bit is non-zero. If these two conditions are met, the next hop is determined in the 1 level detector. This simulated output (Figure 7.1) gives the next hop address of decimal equivalent of value '10' (VP in this figure) for the destination input of “010101010101010101010101”.
And the route prefixes in the routing table that are longer than 24 bits has also been taken in this research in which the entry has been set as ‘1’ to indicate the remaining 15 bits contain a pointer to a set of entries in TBL long (second table) as shown in Figure 7.2. The lower 15 bits of the entry (123 in this example) are combined with the lower 8 bits of the destination address, and used as an index into the second table. After two memory accesses, the table returned the next hop is again 10 in this example as shown in Figure 7.2.
This forwarding engine is run into Xilinx Spartan series with the target devise of xc2v406cs144 and the synthesis report is given in Figure 7.3. It shows that about 512 + 88 LUTs are consumed for all the routing entries and the synthesis reports like Device utilization, cell usage and Timing summary are given below.
7.3 DEVICE UTILIZATION SUMMARY

After synthesis, the device utilization of our target FPGA is summarized as follows which is in term of the modules the system involves, and the percentage of utilization is given in Table 7.1.

IOs: 328
Comparators: 40
24-bit comparator equal: 40

7.3.1 Cell usage

BELS: 969
GND: 1
LUT2: 48
LUT3: 98
LUT4: 460
MUXCY: 360
MUXF5: 1
VCC: 1
Flip-Flops/Latches: 19
LDCP: 19
IO Buffers: 64
IBUF: 32
OBUF: 32
Table 7.1  Device utilization summaries

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of slices</td>
<td>342 out of 256</td>
<td>33%</td>
<td></td>
</tr>
<tr>
<td>Number of slice Flip Flops</td>
<td>19 out of 512</td>
<td>3%</td>
<td></td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>606 out of 512</td>
<td>18%</td>
<td></td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>64 out of 88</td>
<td>72%</td>
<td></td>
</tr>
</tbody>
</table>

7.3.2  Timing details

Minimum input arrival time before clock: 18.490nsec
Maximum output time required after clock: 18.222nsec
Maximum combinational path delay: 23.722nsec

7.3.3  Schematic diagram of forwarding table

The schematic diagram for this forwarding engine is shown in Figure 7.4. This figure gives connection detail of logical blocks pertaining to this forwarding engine.
7.4 IP PREFIX COMPRESSION

The number of routing table entry in the traces considered ranges from 13,000 to 100,000, in order to establish consistency of the approach for smaller as well as larger routing table. All the results were obtained on a window’s platform containing Intel Pentium Processor. The results have been summarized in Table 7.2 and Table 7.3 which shows the amount of memory required for these routing tables. For instance, the Mae-east routing table with over 18,661 entries without end points took 29,871 while it is compressed to just 19,410.
Table 7.2  End points merge for different routers

<table>
<thead>
<tr>
<th>Name</th>
<th>Entries</th>
<th>Endpoints without merge</th>
<th>Endpoints with merge</th>
<th>Non-merge rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aads</td>
<td>17,487</td>
<td>27,250</td>
<td>16,807</td>
<td>62%</td>
</tr>
<tr>
<td>Mae-east</td>
<td>18,661</td>
<td>29,871</td>
<td>19,410</td>
<td>65%</td>
</tr>
<tr>
<td>Mae-west</td>
<td>29,608</td>
<td>45,826</td>
<td>27,792</td>
<td>61%</td>
</tr>
</tbody>
</table>

Table 7.3  Compression after implementing the proposed algorithm

<table>
<thead>
<tr>
<th>Name</th>
<th>Endpoints with merge</th>
<th>Compression After implementing the proposed algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aads</td>
<td>16,807</td>
<td>4023</td>
</tr>
<tr>
<td>Mae-east</td>
<td>19,410</td>
<td>4297</td>
</tr>
<tr>
<td>Mae-west</td>
<td>27,792</td>
<td>6810</td>
</tr>
</tbody>
</table>

This gives very good scalability, which will be very important when routing tables become even larger in the future. The data in this case is compacted to around 1 byte for every entry in the routing table. Also, the overall memory consumption (DRAM) using this scheme is almost half that required in conventional implementations. So it is proved that nearly 77% of entries were reduced. This algorithm needs only 250KB memory.
7.5 DECODING STRUCTURE

By exploiting the low memory access latency and high bandwidth of on-chip memory, high-speed packet forwarding can be achieved using this data structure and the original prefix is reconstructed precisely after decompression. In this process the first 24 bit binary value (hex equivalent of 222480) is loaded into the LUT without any change the symbol ‘-’ is added next to this value for the compression (ACTIVE HDL can accept the symbol) and it was found in the simulated output that the value given is decoded perfectly as shown in Figure 7.5 and the corresponding next hop is found there after.

Figure 7.5 Simulated output for decoding architecture for the compressed data bit

This expanded architecture is run into Xilinx and the detailed synthesis report that are taken from the Xilinx report are given in Figure 7.6.
This data structure is tailored into a hardware design reference model presented in this research.

7.6 FINITE STATE MACHINE

In the 1-bit trie each node stores the prefix, output interface number, pointer to parent and its children if present. For each prefix, start at the root node at the top of the 1-bit trie. Next look at the bits in the prefix from the left. If the bit is 0, then create the left node if absent. If the bit is 1, create the right node if absent. Now change the current node to left node if the bit was 0 or to right node if the bit was 1. This process continues till all the bits in the prefix are exhausted. The output interface number as stated in prefix table is assigned to the current node.

Consider a database having the following 1-bit trie entries (prefix → port): 0* → 128, 01* → 127, 011* → 128, 000* → 256 as shown in Figure 7.7. Here each node can be associated with a state in the corresponding FSM.
Figure 7.7 One bit FSM traversing

Figure 7.8 is the path-compressed FSM in which the path 001 is compressed as 01 and the traversing will continue from path 001. Since the paths are compressed the searching operation has been improved with speed as shown in timing summary and its synthesis report in Figure 7.9.

Figure 7.8 The path-compressed FSM
The timing details of this traversing after compressing the path details are given below

7.6.1 Timing summary after compression

**Speed Grade: 6**

Minimum period : 8.519ns (Maximum Frequency: 117.385MHz)

Minimum input time arrival before clock : 8.630ns

Maximum output time required after clock : 5.880ns

7.6.2 Timing detail after path reduction

The detailed timing report is given in Table 7.4.
All values displayed in nanoseconds (nsec)
Timing constraint: Default period analysis for Clock 'clk'
Delay: 8.519ns (Levels of Logic = 10)
Source: x1_2
Destination: next_state_3
Source Clock: clk rising
Destination Clock: clk rising
<table>
<thead>
<tr>
<th>S. No</th>
<th>Cell:in-&gt;out</th>
<th>Gate fanout</th>
<th>Net delay</th>
<th>Delay</th>
<th>Logical name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FD:x-&gt;q</td>
<td>26</td>
<td>0.449</td>
<td>1.709</td>
<td>x1_2 (x1_2)</td>
</tr>
<tr>
<td>2</td>
<td>LUT3_D:I1-&gt;O</td>
<td>4</td>
<td>0.347</td>
<td>0.818</td>
<td>sf67781 (sf6778)</td>
</tr>
<tr>
<td>3</td>
<td>LUT3_D:I1-&gt;LO</td>
<td>1</td>
<td>0.347</td>
<td>0.100</td>
<td>sf67821 (N44974)</td>
</tr>
<tr>
<td>4</td>
<td>LUT3:i2-&gt;o</td>
<td>3</td>
<td>0.347</td>
<td>0.750</td>
<td>sf70231 (sf7023)</td>
</tr>
<tr>
<td>5</td>
<td>LUT3:i2-&gt;o</td>
<td>1</td>
<td>0.347</td>
<td>0.000</td>
<td>n0234&lt;3&gt;1241_g (n44788)</td>
</tr>
<tr>
<td>6</td>
<td>MUXF5:i1-&gt;o</td>
<td>1</td>
<td>0.345</td>
<td>0.312</td>
<td>n0234&lt;3&gt;1241 (choice4241)</td>
</tr>
<tr>
<td>7</td>
<td>LUT4:i0-&gt;o</td>
<td>1</td>
<td>0.347</td>
<td>0.000</td>
<td>n0234&lt;3&gt;2591_f (n44656)</td>
</tr>
<tr>
<td>8</td>
<td>MUXF5:i0-&gt;o</td>
<td>1</td>
<td>0.345</td>
<td>0.312</td>
<td>n0234&lt;3&gt;2591 (choice4266)</td>
</tr>
<tr>
<td>9</td>
<td>LUT4:i1-&gt;o</td>
<td>1</td>
<td>0.347</td>
<td>0.312</td>
<td>n0234&lt;3&gt;306 (choice4271)</td>
</tr>
<tr>
<td>10</td>
<td>LUT4:i2-&gt;o</td>
<td>1</td>
<td>0.347</td>
<td>0.000</td>
<td>n0234&lt;3&gt;9101_g (n44683)</td>
</tr>
<tr>
<td>11</td>
<td>MUXF5:i1-&gt;o</td>
<td>1</td>
<td>0.345</td>
<td>0.000</td>
<td>n0234&lt;3&gt;9101 (_n0234&lt;3&gt;)</td>
</tr>
</tbody>
</table>

Total 8.519nsec (4.206nsec logic, 4.313nsec route)  
(49.4% logic, 50.6% route)
The timing details before the compression is given below.

7.6.3 Timing summary before path reduction

Speed Grade: -6

Minimum period : 15.887ns Maximum Frequency: 62.945MHz

Minimum input arrival time before clock: 16.658ns

Maximum output required time after clock: 6.788ns

7.6.4 Timing Detail before path reduction

The Xilinx reports for this algorithm is explained in Table 7.5. All values displayed in nanoseconds (nsec)

Timing constraint: Default period analysis for Clock 'clk'
Delay: 15.887ns (Levels of Logic = 9)
Source: x1_2
Destination: next_state_0_3
Source Clock: clk rising
Destination Clock: clk rising
Data Path: x1_2 to next_state_0_3

Table 7.5 Timing detail for FSM before path reduction

<table>
<thead>
<tr>
<th>S. No</th>
<th>Cell:in-&gt;out</th>
<th>Gate Fanout</th>
<th>Net delay</th>
<th>Delay</th>
<th>Logical name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FD:c-&gt;q</td>
<td>18</td>
<td>1.085</td>
<td>2.70</td>
<td>x1_2 (x1_2)</td>
</tr>
<tr>
<td>2</td>
<td>LUT3_D:I1-&gt;O</td>
<td>10</td>
<td>0.549</td>
<td>0.549</td>
<td>sf67781 (sf6778)</td>
</tr>
<tr>
<td>3</td>
<td>LUT4:i0-&gt;o</td>
<td>1</td>
<td>0.549</td>
<td>0.0</td>
<td>n0234&lt;0&gt;2031_g (n45788)</td>
</tr>
<tr>
<td>4</td>
<td>MUXF5:i1-&gt;o</td>
<td>1</td>
<td>0.305</td>
<td>1.035</td>
<td>n0234&lt;0&gt;2031</td>
</tr>
<tr>
<td>5</td>
<td>LUT4:i3-&gt;o</td>
<td>1</td>
<td>0.549</td>
<td>1.035</td>
<td>n0234&lt;0&gt;276 (choice5122)</td>
</tr>
<tr>
<td>6</td>
<td>LUT3:i1-&gt;o</td>
<td>1</td>
<td>0.549</td>
<td>0.549</td>
<td>0.549 0.000 _n0234&lt;0&gt;4061_f (n45653)</td>
</tr>
<tr>
<td>7</td>
<td>MUXF5:i0-&gt;o</td>
<td>2</td>
<td>0.315</td>
<td>0.315</td>
<td>n0234&lt;0&gt;4061 (choice5145)</td>
</tr>
<tr>
<td>8</td>
<td>LUT3:i2-&gt;o</td>
<td>1</td>
<td>0.549</td>
<td>0.000</td>
<td>n0234&lt;0&gt;9021_f (n45688)</td>
</tr>
<tr>
<td>9</td>
<td>MUXF5:i0-&gt;o</td>
<td>9</td>
<td>0.315</td>
<td>1.908</td>
<td>n0234&lt;0&gt;9021 (choice5241)</td>
</tr>
<tr>
<td>10</td>
<td>LUT4_L:I1-&gt;LO</td>
<td>1</td>
<td>0.549</td>
<td>0.000</td>
<td>n0234&lt;0&gt;2130_3 (n45366)</td>
</tr>
</tbody>
</table>
The timing diagram detail before the path has been reduced is shown in Figure 7.10.

Figure 7.10 Synthesis report for total delay before path reduction

Finally the program is combined into structural design and synthesized and the details schematic diagram is shown in Figure 7.11 and 7.12 and 7.13.
Figure 7.11 Schematic view of total delay after breakdown (stage 1)

Figure 7.12 Schematic view of total delay after breakdown (stage 2)
Figure 7.13 Schematic view of total delay after break down (stage 3)

The work has been mapped in Xilinx (Figure 7.14) and designed information has been compared with other algorithms.

Figure 7.14 Mapping report of top level design
Finally this work has been combined and run in the same Xilinx tool and synthesized. The Table content and the Design Information is shown in figure 7.15 and its IO reports are available in Figure 7.16.

Figure 7.15 Synthesis report of top level design

Figure 7.16 I/O reports in place and route
The designed forwarding table was carried out into Post-layout simulation, using Active HDL, which shows that the design can correctly work at 30 MHz. Consequently, the design can furnish approximately 85 lookups/s. Compact stride data structure to search the prefixes (Xuehong Sun and Yiqiang Q.Zhao 2005) is an efficient algorithm they have also aimed at hardware implementation. But they reported 400kb memory for storing a 20k entry and 85 lookups/sec.

The algorithm allows in a hardware pipeline configuration; this design is able to achieve one route lookup in every memory access. All these occupy just 2100 logic cells (roughly 10 thousand gates), 1100 flip-flops, and 10 kb of on-chip SRAM (8 kb for the free list and 1.6 kb for the cell buffer), for both directions of the filter.

7.7 CONCLUSION

Thus a novel algorithm has been developed and tailored to hardware technology. The distinguishing merit of this algorithm is that it has a very small memory requirement. With this merit, a routing table can be put into a single chip; thus, the memory latency to access the routing table can be reduced. With the FSM-based pipeline implementation of this algorithm, the IP address lookup speed can only be limited by the memory access technology. Experiment analysis shows that this algorithm needs only 250kb memory for storing a 47000-entry IPv4 routing table.