Design and Implementation of A Novel FPGA-Based Pipelined-Parallel Processor Architecture for Shortest Path Search

Jassim M. Abdul-Jabbar
Computer Engineering Department, College of Engineering,
University of Mosul, Mosul, Iraq.
drjssm@gmail.com

Majid A. Alwan
Computer Engineering Department, College of Engineering,
University of Basra, Basra, Iraq.
altimimee@yahoo.com

Mohammed A. Ali Al-Ebadi
Computer Engineering Department, College of Engineering,
University of Basra, Basra, Iraq.
mohammed.alebadi@yahoo.com

Abstract

In this paper, a pipelined-parallel hardware architecture to compute the shortest paths of OSPF networks is proposed based on parallel shortest path searching algorithm. OSPF protocol uses the software version of the sequential Dijkstra algorithm for computing and constructing the routing table for each router in an autonomous system area. The proposed Pipelined-Parallel Shortest Path Searching (PPSPS) processor overcomes the time delay of executing the sequential Dijkstra algorithm by conventional processors by reducing the execution time from $O(n^2)$ to $O(n-1)$, where n is the number of nodes. The design is targeted to Xilinx Virtex 7 FPGA chip, and in order to make the parallel processor capable to handle an OSPF area with 256 nodes, a pipelining feature is adopted and successfully implemented within chip resources availability. Very large speed up factors (approximately within the range of 20 – 280) have been achieved by the proposed ultra-high performance PPSPS processor when compared with the conventional software Dijkstra algorithm.

Keywords:- Pipelined-Parallel Hardware Architecture, OSPF Networks, Dijkstra Algorithm, Pipelined-Parallel Shortest Path Searching (PPSPS) Processor, FPGA, Speed Up Factor.

التصميم والتنفيذ باعتماد رقاقة FPGA لهيكلية معالج البحث عن المسار الأقصر نوع خط الأنابيب- المتوازي

د. جاسم محمد عبد الجبار
قسم هندسة الحاسوب- كلية الهندسة
جامعة الموصل- الموصل- العراق
drjssm@gmail.com

د.ماجد عبد النبي علوان
قسم هندسة الحاسبات- كلية الهندسة
جامعة البصرة- البصرة- العراق
altimimee@yahoo.com

محمد عبد علي جودة العبادي
قسم هندسة الحاسبات- كلية الهندسة
جامعة البصرة- البصرة- العراق
mohammed.alebadi@yahoo.com

الخلاصة:

في هذه البحث، تم إقتراح معمارية مادية بخطوط الأنابيب المتوازية لحساب أقصر مسارات شبكات OSPF بإعتماد خوارزمية متوازية للبحث عن أقصر مسار. يستخدم بروتوكول OSPF الإصدار البرمجي من الخوارزمية جيكسترا المتتابعة لحساب وبناء جدول التوجيه لكل مُوَجِّه في منطقة ذاتية التحكم. معالج البحث المقترح نوع خطوط الأنابيب الموازي لأقصر مسار (PPSPS) يتغلب على تأخير وقت تنفيذ خوارزمية جيكسترا المتتابعة في المعالجات التقليدية عن طريق تقليل وقت تنفيذ من $O(n^2)$ إلى $O(n-1)$ حيث أن n تمثل عدد العقد. وتم تنفيذ التصميم على رقاقة Xilinx Virtex 7 FPGA، وسعيا لجعل المعالج الموازي قادر على التعامل مع منطقة OSPF بـ 256 عقدة، تم اعتماد ميزة النقل بواسطة خطوط الأنابيب ونفذت بنجاح بإستخدام الموارد المتوفرة بالرقاقة. ومن خلال الأداء الفائق للمعالج PPSPS المقترح بالمقارنة مع خوارزمية جيكسترا التقليدية تحققت معاملات تسريع كبيرة جدا (تقريبا ضمن المدى من 20 الى 280).

I. Introduction

Open Shortest Path First (OSPF) routing protocol is an instance of a link state protocol based on hop-by-hop communication of routing information, specifically designed for intradomain routing in an IP network [1], [2]. It executes Dijkstra algorithm for constructing the routing table in each router (node) belongs to such OSPF area. Dijkstra algorithm, as well known, has execution time as long as $O(n^2)$ in its software version when executed by conventional processors. This property increases the delay to setup routing information exponentially as the network size (number of nodes per network) is increased. Thereby, the network performance will be slow down.

Parallel computing systems are designed to solve the long time execution of Dijkstra algorithm, in software and hardware solutions. By software solutions, multi-processor and multi-thread system was designed to calculate pieces of routing table concurrently, by partitioning the network into many sub-networks and each processor or thread calculates a part of whole routing table in parallel [3] - [5]. Hardware solutions, such as reconfigurable processors and field programmable gate array (FPGA) technology are often designed for calculating the routing tables [6] - [9].

This paper follows our previous paper presented in *Ref.* [10], when parallel processors for shortest path searching was designed and implemented on Xilinx Virtex7 FPGA chip for 8, 16, 32, 64, and 128 nodes OSPF area size. In that paper, the 128-node processor occupied 65% of FPGA chip resources. It has been concluded that when we go on with the same parallel processor design strategy for a 256-node network, the logic resources required will exceed the total logics available on the FPGA chip.

In this paper, the divide and conquer concept and pipelining processing are used besides the previous proposed parallel shortest path searching (PSPS) algorithm and its units design [10]. By this combination, a pipelined-parallel shortest path searching (PPSPS) processor is achieved for a 256-node network with an ultra-high performance on the same Xilinx Virtex7 FPGA chip.

The rest of this paper is organized as follows: an explanation of the pipelined-parallel shortest path searching (PPSPS) algorithm is presented in section II. Section III explains the proposed hardware implementation of the corresponding PPSPS processor. In section IV, the pipelining data flow and the synchronization of such PPSPS processor is described. The FPGA resources utilization and the performance evaluation of PPSPS processor are discussed in section V. Finally, section VI concludes this paper.

II. Pipelined-Parallel Shortest Path Searching (PPSPS) Algorithm for A 256-Node OSPF Network

The network topology of 256-node OSPF network is represented as a 256×256 square matrix of integer numbers called adjacency matrix or cost matrix. Each element in cost matrix represents the cost (or weight) of travelling from node $\mathbf{i}$ to node $\mathbf{j}$, $\mathrm{w}(\mathrm{i},\mathrm{j})$. The link cost is supposed to be 8-bit positive value. A 256-bit flag vector is

introduced in PPSPS algorithm to be associated with the cost matrix. The bit position of the flag vector works with the row and column of the cost matrix that have their corresponding index numbers (*i.e.*, flag bit 1 with row 1 and column1). In other words, each bit of the flag vector is associated with a single node and its links to the other nodes. Initially, the flag bit position of the row of source node is set to 1, and other flag bits are reset. Flag bit status will decide if the corresponding row will be searched for minimum value by PPSPS algorithm. The parallelism and pipelining processes of PPSPS algorithm can be described as follow:

A- Parallel operation

The parallel searching of shortest paths of 256-node OSPF network can be calculated by the following steps:

1. Set all elements of each column of cost matrix that has flag bit equals 1 to FF (this cost value is chosen as the value to be saved representing infinity).

2. Find the minimum value among the elements of rows whose flag bits are 1.

3. Subtract the minimum value that was found in the step 2 from all values of rows whose flag bits are 1.

4. Zero elements of the resulted cost matrix have minimum costs from source node to their destinations.

5. For columns that have zero elements, update their flag bits to 1.

6. If all flag bits are 1's, all shortest paths are completely found and the matrix will be totally infinity next step, else, go to 1.

Unlike Dijkstra algorithm which searches one node a time, PSPS algorithm of *Ref.* [10] searches multiple nodes simultaneously for shortest paths, and can find multiple destination nodes. The parallel searching will give maximum execution time as $O(n-1)$.

B- Pipelining operation

Pipelining processing design is adopted in this paper in order to be capable to process large network sizes (such as 256 nodes) in a divide and conquer style. The pipelining process can be explained as follows:

1. The network cost matrix 256×256 is decomposed into 16 sub-cost matrices, each 256×16 (16 columns), to be processed sequentially.

2. The flag vector also is divided into sixteen 16-bit sub-flag vectors; each contains the flag bits of the corresponding columns of sub-cost matrix.

3. Each sub-cost matrix and its sub-flag vector are introduced to the PPSPS algorithm separately one after the other in pipelining fashion, and the shortest paths calculation is done according to the PSPS algorithm steps presented in (A). The steps 1-5 of the parallel operation are applied to the 16 sub-cost matrices and their sub-flag vectors, and before step 6 the processed sub-cost matrices and sub-flag vectors are accumulated for recombining and reproducing the updated 256×256 cost matrix with256 bit flag vector. In step 6, if all 256 bits of updated flag vector are 1's, the calculation of shortest paths is completed, otherwise the updated 256×256 cost matrix will enter again in new iteration

of the PPSPS algorithm. Figure 1 shows the decomposition of cost matrix and flag vector into sixteen 256×16 sub-cost matrices and sixteen 16-bit sub-flag vectors.

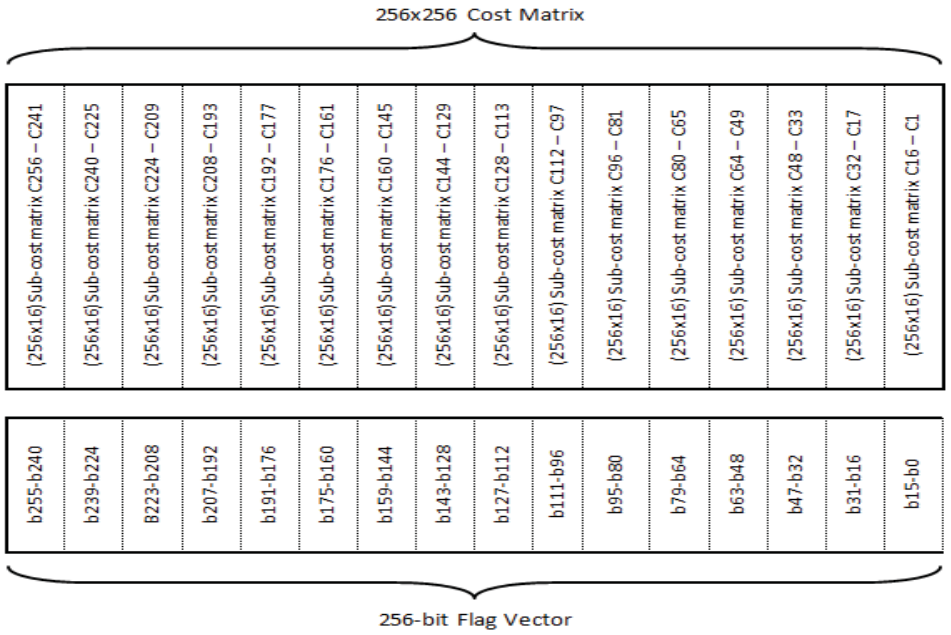


Fig. 1 Decomposition of cost matrix and flag vector; C and b stand for column and bit, respectively.

III. Hardware Architecture of The FPGA-Based PPSPS Processor

Figure 2 shows the block diagram of the proposed PPSPS processor. The hardware architecture consists mainly of two blocks operate sequentially one after the other. The first block is the ***latch block***, and its function is to latch the complete cost matrix of the 256-node OSPF network. It has been assumed that, the data bus of the processor is 64-bit width.

Since the 256- node network size has 256×256 elements of cost matrix and each element is 8-bit value representing the weight of the link that connect upstream node to downstream node, the cost matrix is 256×256×8 bits, formed as 256 rows of 256×8 bits. Thereby, the cost matrix will enter and latched internally as 8192 vectors; each has 64-bit length. The address decoder is used for this function; it receives 13-bit address ADDR1 and decodes it to produce the latch signal for the currently available 64-bit data of cost matrix. The complete cost matrix will be latched and recomposed internally after 8192 clock cycle.

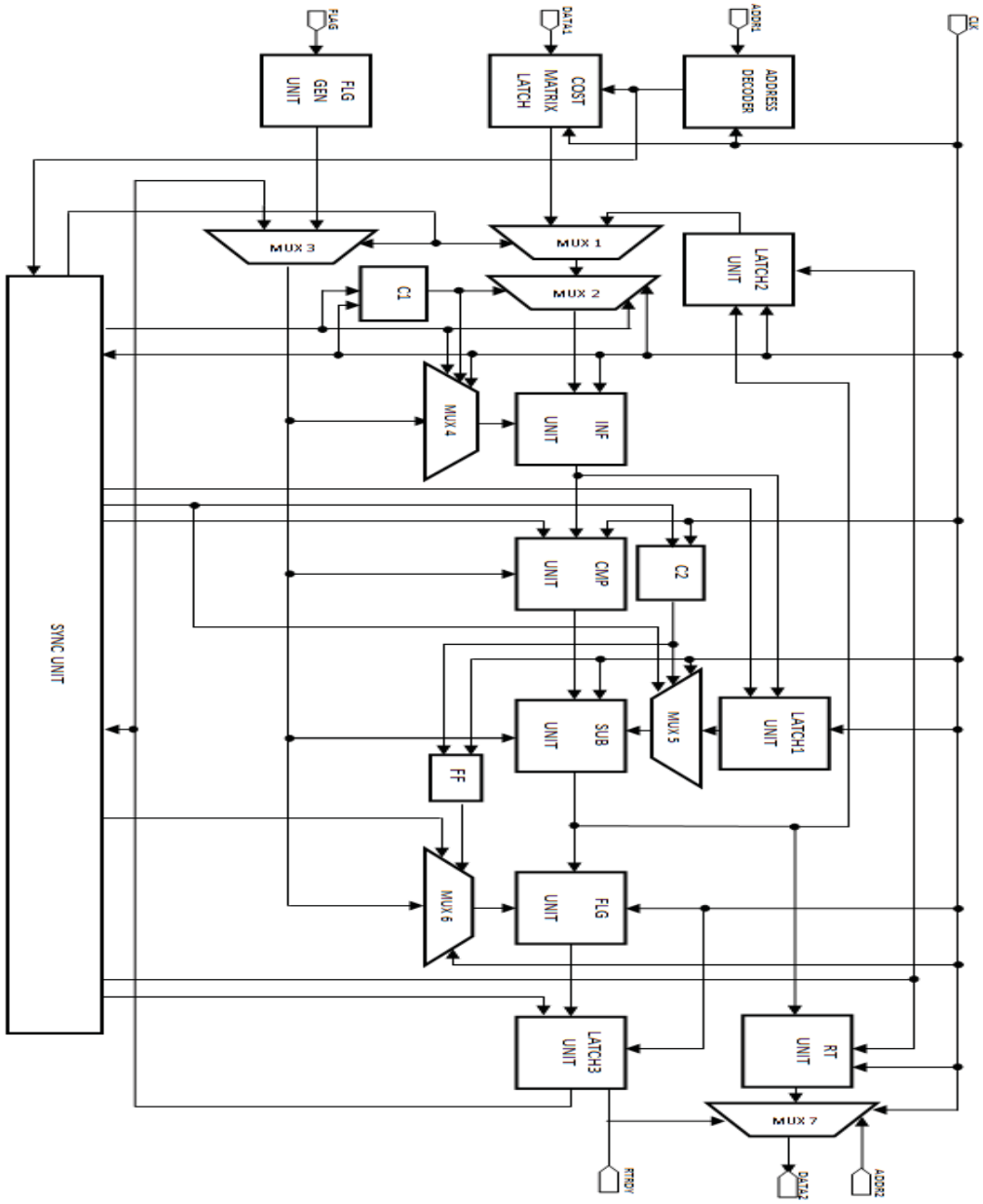


Fig. 2 Block diagram of the PPSPS processor.

Flag generation unit (FLG GEN) creates the initial 256-bit flag vector. The bit pattern of the flag vector has logic level 1 of LSB and logic level 0 of the other bits. This means that the first node of cost matrix will be the source node of shortest paths calculations to the other nodes (destinations) in the network.

The second block of PPSPS processor is the ***processing block*** which computes the shortest paths from the source node to the rest nodes. The followings are descriptions of the components contained in this block.

1- Multiplexer 1 (MUX1):A 524288-bit 2-to-1 multiplexer. Depending on the logic state of the selector, generated by control and synchronized unit. MUX1 passes either the initial cost matrix of the network which is provided by latch block (when selector = 0)or it passes the updated cost matrix resulted from processing block to be processed for next time (when selector = 1).

2- Multiplexer 2 (MUX2): A 32768-bit16-to-1 multiplexer used to divide the 256×256 cost matrix into 16 sub-cost matrices each is 256x16. The 4-bit binary counter C1 supplies its output to 4-bit selector input of MUX2. For each clock cycle, the cost matrix will pass through MUX2 as 256×16, *i.e.*, 16 columns. By MUX2 the pipelining operation is started and it will provide the other processing block components a sub-cost matrix each clock cycle.

3- Multiplexer 3 (MUX3): A 256-bit 2-to-1 multiplexer, its inputs are the initial flag vector and updated flag vector resulted from the processing block. When its selector equals to 0, it passes the initial flag vector, else, updated flag vector will be passed. The selector is generated by a synchronized unit.

4- Multiplexer 4 (MUX 4): A 16-bit 16-to-1 multiplexer, its inputs are the flag vector passed from MUX3and its selector is the 4-bit output of C1 counter. Every 16 bits of flag vector are connected to one input of MUX4. When its enable signal is active, each clock cycle MUX4passes one sub-flag vector that contains the 16 flag bits of its associated 16 columns of the sub-cost matrix passed from MUX3 for the pipelining operation.

5- Set-Infinity-Column unit (INF unit): The operation of INF unit is to set all elements of each column of the sub-cost matrix whose associated flag bit is at logic 1, to infinity (*i.e.*, FF). The inputs of INF unit are the sub-cost matrix and the sub-flag vector coming from MUX2 and MUX4, respectively, and the output of INF unit is the updated sub-cost matrix having some columns with totally infinity value elements. If the$(i,j)^{th}$element in the cost matrix has an infinite value, this indicates that there is no available connection link between thei^{th} and thej^{th} nodes. Setting some columns to infinity values by INF unit is to prevent update their element values (links costs) in the next processing units.

6- Comparator Bank unit (CMP unit): This unit consists of the following three parts working in sequence: first binary tree-comparators, sixteen 8-bit latches, and second binary tree-comparators. The first tree-comparators has 12 stages, and consists of 4095 2-input comparators. The two inputs of the first tree-comparators are the sub-cost matrix updated by INF unit and the complete 256-bit flag vector. In the first stage, multiple rows of the sub-cost matrix will be searched independently in parallel, for their local minimum costs. When the associated flag bit of the row is set, its local minimum cost will pass to the next stages, else an infinity value will be placed instead. The rest stages of the first comparators tree will search the global

minimum among the local minimums of the first stage. Finally, the output of this part is the minimum link cost of the participated links. This minimum value of the sub-cost matrix will be latched, waiting for all other minimum values of the rest sub-cost matrices. By the pipelining operation, 16 different minimum values are latched in the second part of the comparator bank unit. The third part is another tree-comparators, having 4 stages with fifteen 2-input comparators. The operation of this part is to find the final minimum cost from the 16 latched minimum values of the previous two parts.

7- Latch 1unit (LATCH1): This unit is to latch all sub-cost matrices after processing by INF unit. LATCH1 basically consists of 65536 8-bit latches, distributed as 16 groups; each with 4096 8-bit latches, and each group will latch a single sub-cost matrix. The output of this unit is the complete 256×256 cost matrix.

8- Multiplexer 5 (MUX5): MUX5 is similar toMUX2 in its construction and operation. MUX5 receives the output of LATCH1 and reforms it as 16 groups of sub-cost matrices to be passed in sequence according to the selector bits status that generated by C2 4-bit binary counter. In other word, MUX5 divides the 256×256 cost matrix into 16 sub-cost matrices each with 256×16.

9- Subtractor Matrix unit (SUB unit):The subtractor matrix unit is a two-dimensional 256×16 array of simple subtractors. The inputs of SUB unit are the 256×16 sub-cost matrixpassed from MUX5, the minimum cost produced by CMP unit and the 256-bit flag. The subtractor of a specific position in the subtractors array will subtract the minimum cost from the non-infinity cost value at the same position of the sub-cost matrix. This is done only for the rows of the sub-cost matrix which have flag bits equal to 1, other rows will remain unchanged. All subtractors will operate simultaneously, and as a result the updated sub-cost matrix will have zero values.

10- Latch 2 unit (LATCH2): This unit accumulates the sub-cost matrices processed by SUB unit to reform the updated 256×256 cost matrix. The output of this unit is supplied to MUX1 for the next iteration of shortest path calculations.

11- Multiplexer 6 (MUX6): The same as MUX4 in its operation. The 4-bit selector is the output of counter C2 but delayed by one clock cycle for synchronization.

12- Flag Update unit (FLG unit): FLG unit receives the updated sub-cost matrix produced by SUB unit and the sub-flag vector passed from MUX6. The function of this unit is that, when a zero value is found in each column of the sub-cost matrix, the corresponding flag bit will be set to 1. All 16 columns will be searched in parallel and all affected bits of sub-flag vector will be updated at the same time.

13- Latch 3 unit (LATCH3): Because of FLG unit produces only 16 bits updated sub-flag vector for one time, the 16 sub-flag vectors will be latched by LATCH3 unit to reproduce the complete new 256-bit flag vector that will be used for the next

iteration of the processing block. This unit consists of two levels of 256-bit latches, the first level is to latch the sub-flag vectors one after the other, and the second level to release all 256 bits of the complete new flag vector. Also, a signal RTRDY will produce if all 256 bits of the flag vector are 1's to indicate that all shortest paths are found and the routing table information of the network can be downloaded from the hardware. It is generated by ANDing all new 256 bits of flag vector.

14- Routing Table unit (RT unit): The RT unit will have the final information of the shortest paths of the network routs. It consists of 265×256 bits array viewed as 16 partitions; each partition is 256×16 bits sub-array. The input of this unit is the sub-cost matrix resulted from SUB unit. The zero values of sub-cost matrix elements will set the corresponding bits positions in a sub-array of RT unit. Each sub-cost matrix process a sub-array of RT unit, and the 16 sub-cost matrices covers the 256×256 bit array. At the end of the shortest path computations, when $(i,j)^{th}$bit is set, this means that the link from i^{th} node to j^{th} node is participated in the shortest path because of its minimum cost.

15- Multiplexer 7 (MUX7): Finally, the routing table can be read out from the RT unit using a MUX7. This multiplexer has 1024 inputs and one output; each is of 64-bit wide. Its selector is 10-bit address bus ADDR2 for reading the 256×256 bits of RT units as 64 bits each click cycle. MUX7 will be enabled if RTRDY signal of LATCH3 unit is logic 1.

16- Synchronization unit (SYNC unit): The synchronization and control the flow of the data from one unit to the other in PPSPS processor is done by SYNC unit. It controls the cost matrix processing steps by releasing signals for enabling/disabling each component of the processor hardware. SYNC unit consists of two53-bit rotation registers (SRR1 and SRR2) and one256-bit inputs OR-gate. The function of OR-gate is to produce the selector of MUX1 and MUX3 by ORing the 256 bits of the new flag vector generated by FLG – LATCH3 units. Initially, all these 256 bits are set to logic 0, so the initial selector state is logic 0, that will cause MUX1 and MUX3 pass the origin cost matrix and flag vector, respectively. After the first iteration of the processing block, the 256 bits of the new flag vector will contain at least one bit with logic 1 level, resulting the selector will change its state to logic 1 and accordingly MUX1 and MUX3 will pass the updated cost matrix and flag vector, respectively. The selector will remain at logic 1 till the computation of shortest paths is completed. The two synchronization rotation registers of SYNC unit are described in the next section.

IV. Pipelining Data Flow and The Synchronization of PPSPS Processor

A single clock signal drives all components of the PPSPS processor except MUX1 and MUX3 which are non-clock driven units. Table 1 shows the clock cycles and data pipelining flow starting from the pass of the first sub-cost matrix from MUX2 and ending when the 256-bit new flag vector generated by LATCH3 unit. This time period represent

the time consumed by PPSPS processor for a single iteration of the processing block operation.

Table 1 Data pipelining flow of processing block of the PPSPS processor.

Clocks / Unit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
MUX2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X																																						
INF		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X																																						
CMP			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X																				
SUB																																			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X			
FLG																																				X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
LATCH3																																					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

In Table 1, the mark X means that a new data is processed and out from that unit. MUX1 takes 16 clock cycles (from 1 to 16 clocks event) to pass the complete cost matrix as a pipelined 16 sub-cost matrices flow. Also INF unit takes 16 clock cycles (from 2 to 17 clocks event) to process the 16 sub-cost matrices sequentially, each during a single clock period. The CMP unit needs 32 clock cycles (from 3 to 34 clocks event) to find the global minimum cost of the cost matrix. In pipelining flow, 28 clocks for finding the minimum values of the 16 sub-cost matrices and 4 clocks for finding the global minimum of the cost matrix. Both SUB and FLG units consume 16 clock cycles to process the complete cost matrix and flag vector, respectively. LATCH3 unit needs 17 clock cycles to produce the new 256-bit flag vector, 16 clocks for latch the 16 sub-flag vectors in the first level latches and 1 clock for enable all second level latches to release the new flag vector.

From Table 1, it is clear that a single iteration of the processing block of the PPSPS processor is finished within 53 clock cycles. At clock cycle 54, a new iteration will be begun and the data will start flow in pipelining manner with the same scenario. In order to synchronize the pipelining data flow for each iteration, SYNC unit has two synchronization rotation registers SRR1 and SRR2 designed for this purpose. Both registers are 53-bit length, and initialized with bit patterns to enable the controlled units in its time and disable them in other times. Figure 3shows the architecture of these registers and their initial bits patterns. SRR1 and SRR2 start rotate their bits pattern to the right when their enable inputs are active and they still in rotation unless the processor is reset.

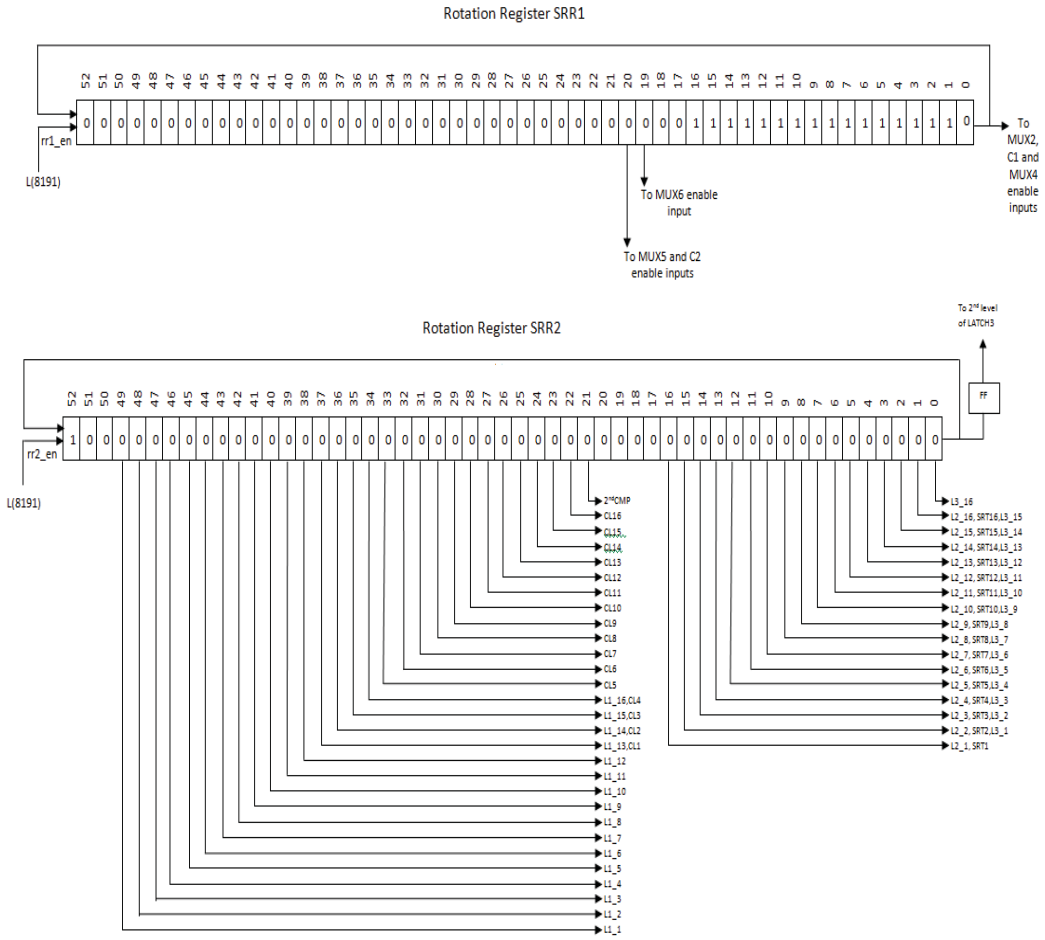


Fig. 3 SRR1 and SRR2 of SYNC unit.

The synchronization operation of the two rotation registers can be explained as follows:

- SRR1: It starts rotate its bits pattern when its rr1_en input is active. The last address decoder output L(8191), which used for latching the last 64 bits of cost matrix, is connected to rr1_en. When this line has logic level 1, next clock cycle will make RR1 start rotation. Bit SRR1(0) will enable MUX2, C1, and MUX4 when it is at logic 1 state. Because of the appropriate initial bits pattern of this register, these three units will stay enabled for 16 clocks in order to pass the complete cost matrix and flag vector from MUX2 and MUX4, respectively. After, MUX2 and MUX4 will be disabled and latch the last output, and C1 will disable counting and force its 4-bit output to 0000. These three units will stay at these states for the next 37 clock cycles before they restart the same operation for a new iteration of the processing block for new 53 clock cycles.

The SRR1(20) bit is connected to C2 and MUX5 enable inputs. When its logic state is 1, these two units will start working in next clock cycle. The 4-bit output of counter C2

is pre-set to 0000. So, if MUX5 is enabled, it will pass the first sub-cost matrix toward the SUB unit. In the same clock cycle, C2 will increment its output by 1 to prepare the selector of MUX5to select the second sub-cost matrix in the next clock cycle. This operation will continue for 16 clock cycles in order to complete passing all the 16 sub-cost matrices which stored in LATCH1 unit. SRR4(20) bit will be at logic 1 level after 34 clocks from enabling the register, and stay at this level for a 16 clocks, then return to 0 for 37 clocks. By this way, SUB unit will process the 16 sub-cost matrices in a pipelining manner during each iteration of the processing block. On the other hand, SRR4(19) bit enables MUX6 when its logic level is 1, *i.e.*, after one clock cycle from C2 and MUX5 enabled, for 16 clock cycles.

- SRR2: Latch 1 unit stores the 16 sub-costs matrices after updating by INF unit in pipelining manner. The 16 enable signals of this unit are provided by SRR2. After 3 clocks from the time of L(8191) signal is set to logic 1, the first sub-cost matrix is arrived to LATCH1 unit, and the rest matrices will come consequently during the next 15 clocks. The initial bits pattern of SRR2 is set so that SRR2(49) will be at logic 1 at the same time of the arrival of the first sub-cost matrix to LATCH1, and because of connecting this bit to the first latch enable, L1_1, next clock cycle will save this sub-cost matrix in its latches. The other sub-cost matrices will be also stored in LATCH1 unit by logic 1 state of SRR2(48) to SRR2(34) bits which represent the enable signals of the other 15 latches L1_2 to L1_16.

The second function of SRR2 register is that to enable the intermediate stage of CMP unit, which is the latch part. The logic states of bits SRR2(37) through SRR2(22) are used to enable the sixteen 8-bit latches CL1 through CL16, respectively, in sequence and one latch per a clock cycle. The result of this operation is the latching of the 16 minimum values of the sub-cost matrices processed by first comparator of CMP unit. It is required 15 clock cycles, from the time of L8191 is logic 1, to find the minimum cost of the first sub-cost matrix. The other 15 minimums of the rest sub-cost matrices will be found during the next 15 clock cycles. When all the 16 minimum values are found and latched, the second comparator will be enabled at the next clock by SRR2(21) bit to find the global minimum during 4 clock cycles, and the CMP unit finishes its work for this iteration and will repeat the same operation next iteration.

The other function of this rotation register is to provide latch signals for three units, LATCH2, RT, and LATCH3. SRR2(16) will be at logic 1after 36 clocks from SRR2 starts rotating. During the next clock cycle, SRR2(16) will latch the first updated sub-cost matrix produced by SUB unit in LATCH2 unit, and also will enable first 16 columns of the bits array of RT unit for updating its bits from logic level 0 to 1 if the corresponding costs of the updated sub-cost matrix have zero values. In the same manner, SRR2(15) to SRR2(1) will be used for the rest parts of LATCH2 unit and RT unit. On the other hand and by the same way, SRR2(15) through SRR2(0) will be used to latch the 16 sub-flag vectors resulting from FLG unit in the first level latches of LATCH3 unit. Then, the logic state of SRR2(0) bit is delayed by one clock cycle. This will activate the second

level latches of LATCH3 unit to release the complete 256-bit new flag vector. At this clock cycle, the synchronization of the pipelining data flow operation for a single processing block iteration cycle will be terminated after 53 clock cycles.

V. FPGA-Based Implementation and Results

The proposed architecture of PPSPS processor is targeted to Xilinx Virtex7 (XC7V2000T)FPGA chip [11] for implementing. Xilinx ISE design suite 13.2 [12] is used for writing VHDL codes, synthesizing the design, and also specifying timing constraints for the best and fastest performance. The FPGA area utilization and the minimum clock period resulted are presented in Table 2, and post place and route (post-PAR) simulation using Xilinx ISIM simulator [13] is successfully done.

Table 2 XC7V2000T FPGA chip utilization and clock period.

LUTs	LUTs utilization ratio	FFs	FFs utilization ratio	Clock period (ns)	Frequency (MHz)
730575	59.8%	1803542	73.7%	14.31	69.893

A) Performance evaluation of PPSPS processor

From the previous section, a single iteration of the processing block of PPSPS processor consumes 53 clock cycles for shortest paths computations. Thereby, the total clock cycles for finding all shortest paths of the whole 256-node network from a single source node to all destination nodes can be calculated as:

$$Total\ clock\ cycles = \frac{256\times256\times8}{64} + 1 + (K \times 53) + \frac{256\times256}{64} = 9217 + (K \times 53) \quad \ldots (1)$$

The variable *K* in Eq. (1) is the number of iterations performed by processing block until all shortest paths are found, *i.e.*, all bits of flag vector become 1's. Theoretically, the parallel algorithm takes no more than N-1 iterations to complete the searching process, where N is the number of network nodes. So for the 256-nodes network, if *K*=255, then the total clock cycles will be 22732 clocks.

Actually, the number of iterations of the processing block is much less than N-1 because that the parallel algorithm can find multiple shortest paths from the source node to many destination nodes. This feature reduces the number of iterations to find all shortest paths, thus it provides an early-termination of the searching process. The calculation time will depend on the network connections density and the variation of links costs. Network connections density means the number of links that connect all nodes in the network, and the variation of link costs means the differences between them. It has been found that the connections density has a minor effect of the searching time while the major effect comes from the variations of link costs. In Table 3, different random 256-node network topologies are generated with the pre-definition of their link cost values and their connection densities. The same network topologies are searched by software Dijkstra algorithm (run on standard PC with 2 GHz Intel core i7 microprocessor and 8 GB of

RAM). The execution times are calculated and compared with the corresponding execution times of the PPSPS processor. The results in all tested cases illustrate the high performance of PPSPS processor which high offers speed up factors (approximately in the range of 20 – 280). The speed up factors of the 36 experiments presented in Table 3 are plotted in Figure 4.

It is clear that from Table 3, when the network connection links have small and more similar costs, the processing block iterations will be decreased and the speed up factor of the PPSPS processor will be increased, and vice versa. Also, it is noted that, the decrease in the connection density corresponds to the increase in the processing block iterations.

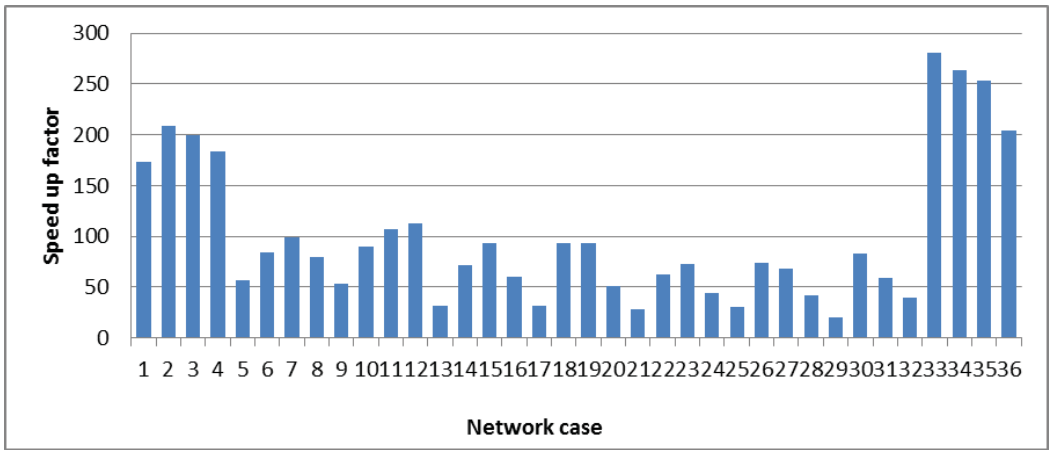


Fig. 4 The speed up factors of the 20 experiments presented in Table 3.

B) Simulation results using Xilinx ISim

FPGA-based PPSPS processor for 256 nodes has been designed in VHDL codes according to the proposed architecture, targeted to FPGA chip of the type Xilinx Vertix-7 (XC7V2000T), and tested with the two network topologies. Such topologies are different in their complexity, link cost values and connection density. As an example, Figure 5 shows parts of the post place and route simulation waveforms of a cost matrix of 256-node network. It should be noted that it takes 66 iterations of the processing block to find all shortest paths. The Post-Route simulation is done by Xilinx ISim simulator [13], with link costs of the cost matrix being represented in hexadecimal numbers, and FF_h link cost represents the infinity. The number of clocks and the number of iterations consumed by FPGA-based PPSPS processor can be described as follows:

Table 3 Various topologies of 256-node network.

Case no.	Link cost ranges	Network connection densities	Processing block iterations (K)	Total clock cycles	Execution time of s\w Dijkstra (sec.)	Execution time of PPSPS (ns)	Speed up factor
1	1-25	100%	4	9429	0.0233	134834.7	172.8042
2	1-25	75%	4	9429	0.0281	134834.7	208.4033
3	1-25	50%	5	9482	0.027	135592.6	199.1259
4	1-25	25%	8	9641	0.0253	137866.3	183.5111
5	10-25	100%	13	9906	0.0081	141655.8	57.18086
6	10-25	75%	17	10118	0.0121	144687.4	83.62857
7	10-25	50%	19	10224	0.0145	146203.2	99.17704
8	10-25	25%	25	10542	0.012	150750.6	79.60167
9	20-50	100%	25	10542	0.008	150750.6	53.06778
10	20-50	75%	30	10807	0.0138	154540.1	89.29721
11	20-50	50%	34	11019	0.0169	157571.7	107.2528
12	20-50	25%	49	11814	0.019	168940.2	112.4658
13	50-100	100%	51	11920	0.0053	170456	31.09307
14	50-100	75%	58	12291	0.0125	175761.3	71.11918
15	50-100	50%	65	12662	0.0169	181066.6	93.33582
16	50-100	25%	85	13722	0.0119	196224.6	60.64479
17	75-125	100%	51	11920	0.0053	170456	31.09307
18	75-125	75%	61	12450	0.0166	178035	93.24009
19	75-125	50%	68	12821	0.0171	183340.3	93.26918
20	75-125	25%	94	14199	0.0103	203045.7	50.7275
21	125-200	100%	74	13139	0.0052	187887.7	27.67611
22	125-200	75%	85	13722	0.0122	196224.6	62.17365
23	125-200	50%	89	13934	0.0146	199256.2	73.2725
24	125-200	25%	111	15100	0.0095	215930	43.99574
25	175-225	100%	49	11814	0.0052	168940.2	30.78012
26	175-225	75%	69	12874	0.0137	184098.2	74.41681
27	175-225	50%	79	13404	0.0131	191677.2	68.34407
28	175-225	25%	112	15153	0.0091	216687.9	41.99588
29	200-254	100%	54	12079	0.0035	172729.7	20.26287
30	200-254	75%	71	12980	0.0155	185614	83.50663
31	200-254	50%	79	13404	0.0113	191677.2	58.95328
32	200-254	25%	123	15736	0.0089	225024.8	39.5512
33	1-254	100%	14	9959	0.0399	142413.7	280.1697
34	1- 254	75%	16	10065	0.0379	143929.5	263.3234
35	1-254	50%	22	10383	0.0376	148476.9	253.238
36	1-254	25%	38	11231	0.0328	160603.3	204.2299

- PPSPS processor starts receive the cost matrix within 7.15 ns. At the time = 464.75 ns, the first row of cost matrix is latched completely after 32 clock cycles. Note that 464.75-7.15=457.6 ns and 457.6/14.3=32 clock cycles, 32x64bit=2048bit which is the total bits of the first row (256x8bits)- See Figure 5(a).

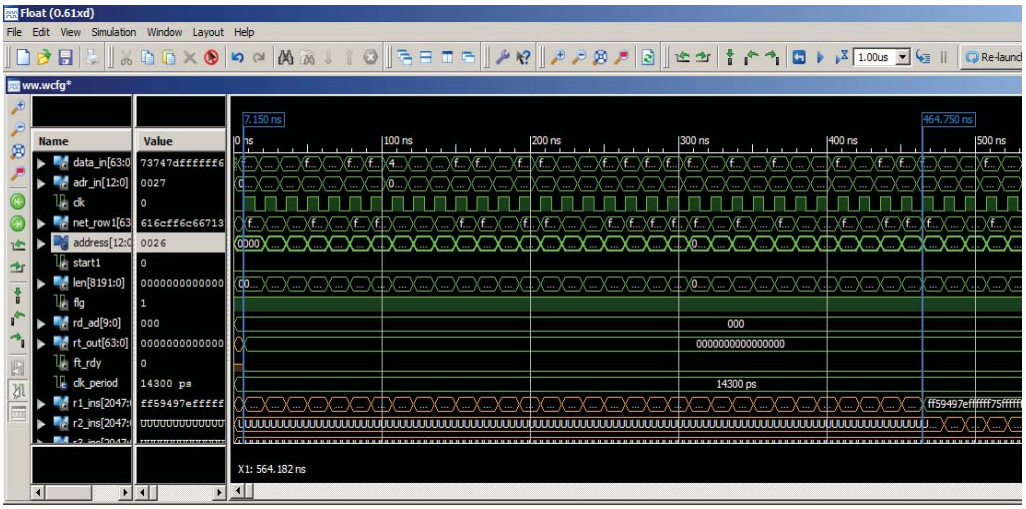
- So, every 32 clock cycles, a new row will be latched completely in PPSPS processor, and 256x32 = 8192 clock cycles are required for latching the complete cost matrix. At the time = 117152.750 ns, the last row of the cost matrix is latched. Again, (117152.750-7.15)/14.3=8192 clock cycles- See Figure 5(b).

- Now, after one clock, the pipeline operation starts at the time = 117167.05 ns and the first sub-cost matrix will inter the first iteration of the processing block- See Figure 5(c).

- The last clock of the 16 clocks of the pipeline operation starts at the time = 117381.550 ns. At this clock cycle the last sub-cost matrix will enter the processing block- See Figure 5(d).

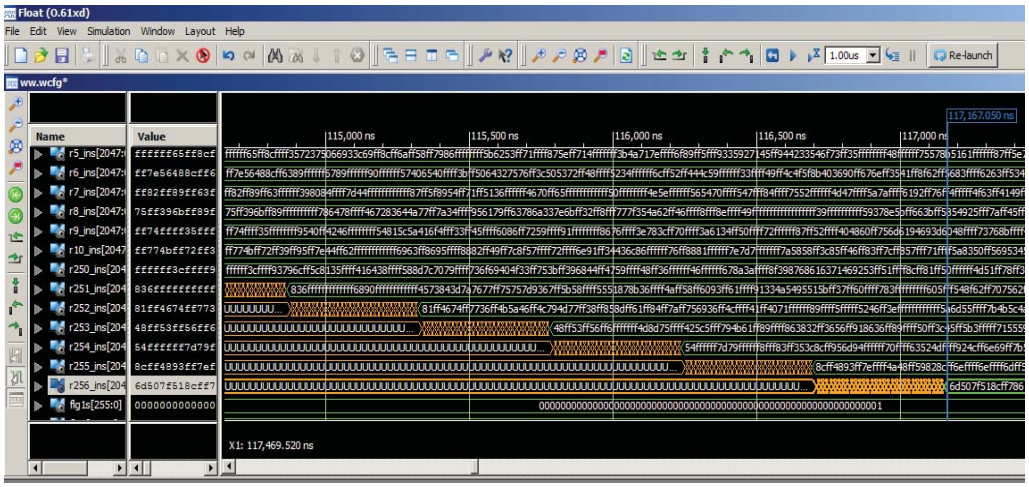
- At the time = 167174.150 ns, PPSPS processor completes the computations and found all the shortest paths after 66 iterations of the processing block. Note that (167174.150-7.15)/14.3 = 167167; 167167- 8192 = 3498, and 3498/53 = 66 iterations- See Figure 5(e).

- After one clock, the routing table information can be downloaded from PPSPS RT unit for 1024 clock cycle.

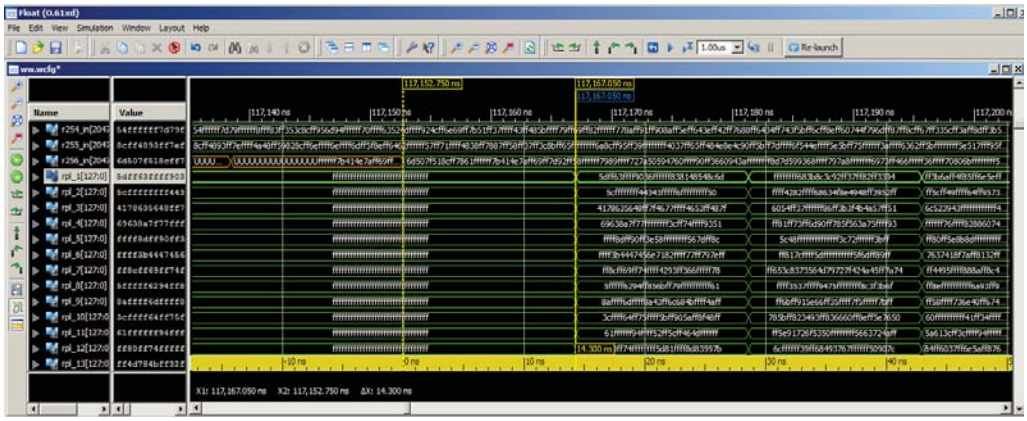


(a)

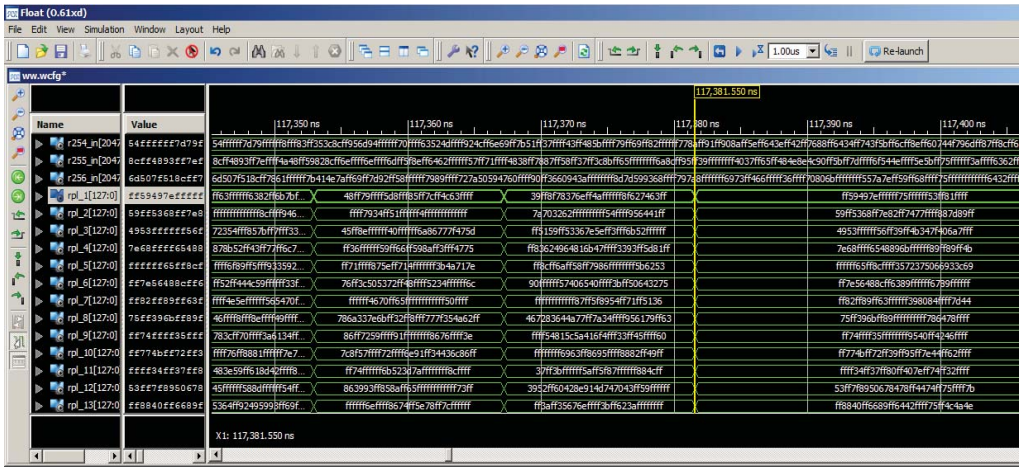
Fig. 5 Xilinx ISim simulation results of PPSPS processor.



(b)

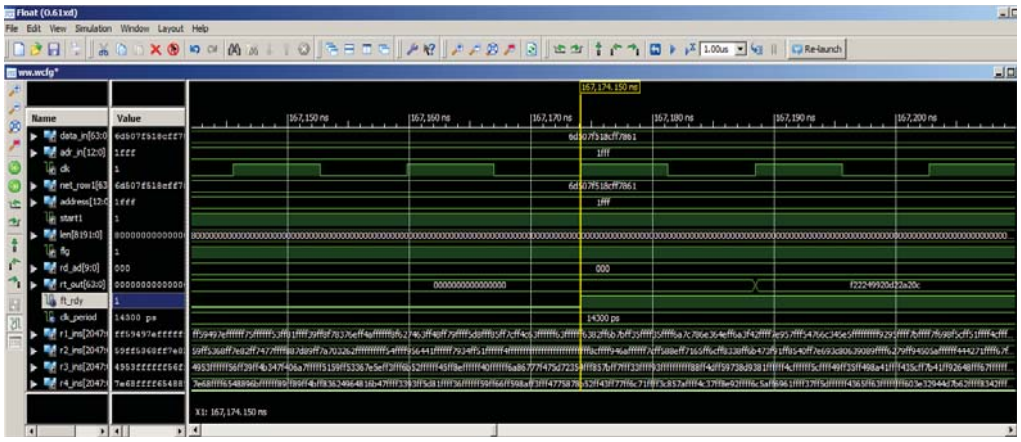


(c)



(d)

Fig. 5 (continued)



(e)

Fig. 5 (continued)

VI. Conclusions

In this paper, a pipelined-parallel hardware architecture for shortest path searching (PPSPS) processor has been designed and implemented successfully for 256-node OSPF networks. The synchronization of such proposed PPSPS processor is guaranteed with fewer logics (73.7% FPGA chip flip-flops and 59.8% chip LUTs utilizations on Xilinx Virtex7-XC7V2000T FPGA chip). These reduced implementation complexities have been achieved because the processing block units are designed only to process 256×16 sub-cost matrices. The cost matrix loading process to FPGA chip takes 8192 clock cycle and the routing table information extraction takes 1024 clock cycle, while the processing block consumed ($K \times 53$) clock cycles, where K is the number of iterations of processing block for finding all shortest paths from a single node to the rest nodes. It has been noticed that, two factors can affect the K value; the link cost range and the network connection density. When the network links costs are more similar with high connection density, the K value will be small and the total computing time will be then reduced. The resulting high speed up factors of our experiments with various network topologies that differ in link cost ranges and network connection densities have proved that the performance of the proposed PPSPS processor can outperform the software Dijkstra algorithm running on a conventional microprocessor system.

References

[1] D. Medhi and K. Ramasamy, Network Routing Algorithms, Protocols, and Architectures, Elsevier Inc, 2007.

[2] B. A. Forouzan, Data Communications and Networking, 4th edit, McGraw-Hill, 2007.

[3] X. Xiao and L. Ni, "Reducing Routing Table Computation Cost in OSPF", Internet Workshop, 1999. IWS 99, pp.119-125, 1999.

[4] B. Xiao, Jiannong Cao, QingfengZhuge, Zili Shao, Edwin H.-M. Sha, "dynamic Update of Shortest Path Tree in OSPF", in Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04), pp. 18-23, May 2004.

[5] A. K. Phipps, "Parallel Algorithms for Geometric Shortest Path Problems", Master thesis, School of Informatics, University of Edinburgh, 2004.

[6] H. Ishikawa, S. Shimizu, Y. Arakawa, N. Yamanaka, K. Shiba, "New Parallel Shortest Path Searching Algorithm based on Dynamically Reconfigurable Processor DAPDNA-2", IEEE International Conference on Communications, 2007 ICC '07, pp.1997-2002, 24-28 June 2007.

[7] S. Shimizu, T. Kihara, Y. Arakawa, N. Yamanaka, K. Shiba, "A Prototype of a Dynamically Reconfigurable Processor Based Off-loading Engine for Accelerating the Shortest Path Calculation with GNU Zebra", International Conference on High Performance Switching and Routing, 2008. HSPR 2008, pp.131-136, 15-17 May 2008.

[8] K. Sridharan, T. K. Priya, P. Rajesh Kumar, "Hardware Architecture for Finding Shortest Paths", TENCON 2009 - 2009 IEEE Region 10 Conference, pp.1-5, 23-26 Jan. 2009.

[9] I. Fernandez, J. Castillo, C. Pedraza, C. Sanchez, J. Ignacio Martinez, "Parallel Implementation of The Shortest Path Algorithm on FPGA", 2008 4th Southern Conference on Programmable Logic, pp.245-248, 26-28 March 2008.

[10] J. M. Abdul-Jabbar, M. Alwan, M. Al-Ebadi, "A New Hardware Architecture for Parallel ShortestPath Searching Processor Based-on FPGA Technology", International Journal of Electronics and Computer Science Engineering (IJECSE), vol. 1, no. 4, pp.2572-2582, Oct. 2012.

[11] Xilinx, "Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics", Advance Product Specification, DS183 (v1.2) November 7, 2011, available on www.xilinx.com

[12] Xilinx, "ISE In-Depth Tutorial", UG695 (v13.1) March 1, 2011, available on www.xilinx.com

[13] Xilinx, "ISE Simulator (ISim) In-Depth Tutorial", UG682 (v 13.2) July 6, 2011, available on www.xilinx.com