# A new technique for accelerating routing information process in communication networks

# Une nouvelle technique pour accélérer le processus de routage d'information dans les réseaux de communication

Ramzi Benaicha[*1] & Mahmoud Taibi[2]

[1] *L.E.R.I.C.A Laboratory, Badji Mokhtar Annaba University, Po Box 12, 23000, Annaba, Algeria.*
[2] *Automatic and Signals Laboratory (LASA), Badji Mokhtar Annaba University, Po Box 12, 23000, Annaba, Algeria.*

*Soumis le :05/05/2015* *Révisé le :01/12/2015* *Accepté le :08/12/2015*

**ملخص**

إن حساب أقصر مسار بين زوج من أجهزة التوجيه هو مشكل هام في شبكات الاتصالات والكمبيوتر. كما أن حساب المسار في الوقت الحقيقي هو مفيد في جميع الحالات. وتشمل هذه العملية التوجيه للوصول إلى وجهتها، والتقليل من آثار الاصطدام مع العقبات. الأعمال السابقة في هذا الميدان والمتمثلة في البحث على أقصر طريق تقتصر على خوارزميات متسلسلة أو متوازية على هياكل للأغراض العامة. الباحثون يهتمون بشكل متزايد على الحلول المادية(FPGA). في هذا العمل نقترح نهجا لتنفيذ خوارزمية التوجيه التي هي فعالة مثل ديكسترا باستخدام بطاقة FPGA من نوع VIRTEX لتسريع عملية التوجيه ويستند هذا العمل على سرعة الأجهزة(FPGA). إن نتائج تنفيذ الخوارزمية في البطاقة FPGA من نوع VIRTEX 7 أتت واعدة.

**الكلمات المفتاحية :** *WLAN ، OSPF ، خوارزمية ديكسترا ، اللغة VHDL، بطاقةFPGA* .

**Abstract**

The calculation of the shortest path between a pair of routers is an important problem in telecommunication and computer networks. The calculation of the path in real time is useful in a number of situations. These include a routing process that attempts to reach its destination and minimizing the effects of collision with obstacles. Previous works on the shortest path are limited to sequential and parallel algorithms on general-purpose architectures. Researchers are increasingly interested in hardware's solutions. In this work , we propose an approach for implementing a routing algorithm which is effective than Dijkstra using a FPGA development board Xilinx Virtex-type order accelerate the process of routing based on the speed of hardware (FPGA). The results of the implementation in an FPGA card Virtex7 are promising.

**Key words:** *WLAN, OSPF, Dijkstra algorithm, Language VHDL, FPGA.*

**Résumé**

Le calcul du plus court chemin entre une paire de routeurs est un problème important dans les réseaux de télécommunication et de l'informatique. Le calcul de la trajectoire en temps réel est utile dans un certain nombre de situations. Il s'agit notamment d'un processus de routage qui tente d'atteindre sa destination, et de minimiser les effets de collision avec des obstacles. Les travaux antérieurs sur le plus court chemin sont limités à des algorithmes séquentiels et parallèles sur les architectures à usage général. Les chercheurs sont de plus en plus intéressés par les solutions de matériel. Dans ce travail, nous proposons une approche pour mettre en œuvre un algorithme de routage qui est efficace que celui de Dijkstra, utilisant une carte de développement FPGA de type Virtex, pour accélérer le processus de routage, en se basant sur la vitesse du matériel (FPGA). Les résultats de l'implémentation de l'algorithme dans une carte FPGA Virtex 7 sont prometteurs.

**Mot clés :** *WLAN, OSPF, Algorithme de Dijkstra, Langage VHDL, FPGA.*

[*] *Corresponding author: ramzibenaicha23@gmail.com*

## 1. INTRODUCTION

The most effective protocol in computer networks is MPLS (Multi -Protocol Label Switching) [1]. This protocol uses to calculate the path meets the specified constraints extensions of OSPF protocol [2]. The latter is based on the Dijkstra algorithm to determine the optimal path to be assigned to information flow [3]. But the information is not transmitted as data only, many other factors come into play now among which may be mentioned that the quality of service.

These routing algorithms are executed by ordinary processors [4], and the increase in complexity of calculation is exponentially transformed into an increase in quality of service, and for the network to achieve acceptable performance it will need an increase in operative resource [5]. The first reflex face this problem of flexibility and speed is the design of new routing protocols more complex than the previous. But protocol designers soon realized that the problem was none other than the processor that performs the calculations. This is why we turned to the FPGA technology allows to combine the flexibility of software, hardware speed [6].

Researchers at the Central School of Paris are the first who have implemented Dijkstra's algorithm in an ordinary microprocessor and other works proposed to be used in calculations in reconfigurable communication systems [7], it is an approach that promotes the hardware implementation compared to software implementation [8]. A comparison was made between the performance of the implementation of Dijkstra algorithm on FPGA and implementation on regular processor [9]. For these reasons, we propose to use FPGA cards in parallel with an ordinary processor.

This paper's main purpose is to highlight the problems encountered in the routing information in computer networks. To solve them we offer a solution that can improve or accelerate this process. We propose to use cards FPGA in parallel with an ordinary processor. This allows them to accelerate certain tasks materially while keeping a machine (CPU). Quick for any task it is asked for.

## 2. ROUTING IN COMPUTER NETWOTKS

Routing is a function of the layer 3 (network layer) of the OSI model (Open System Internet). It is a hierarchical organization system (Fig. 1) that lets you group individual addresses. These are treated as a whole until the destination address is required for the final delivery of data [10].

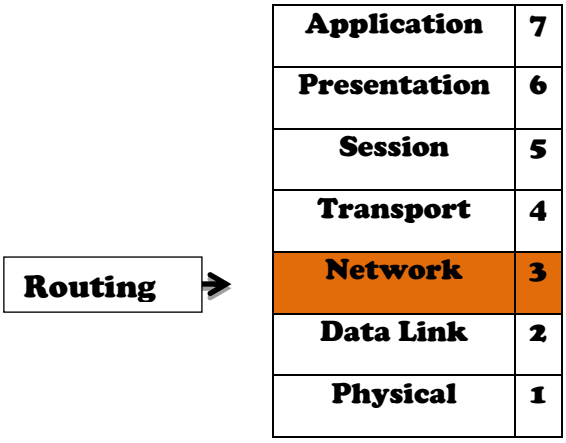


Figure 1: Network Layer.

A router is a network layer unit that uses one or more metrics to determine the optimal path through which to route network traffic. The routing metrics are the values that define the best path.

Routing seeks the most effective one unit to another path (Fig. 2).

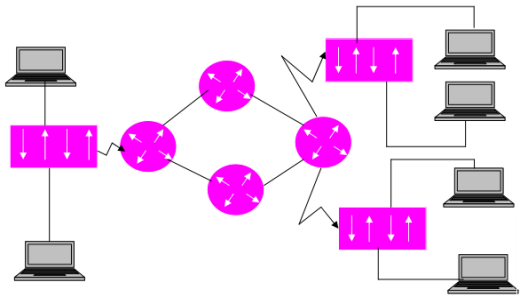

Figure 2: The Routing.

It has got the two following main functions:

• The router handles the routing tables and ensures that other routers are aware of the changes in the network topology. It uses routing protocols to exchange network information.

• The router determines the destination of packets by using the routing table when they arrive at one of its interfaces. He transferred to the correct interface, adds the information frame of this interface, and then forwards the frame to the outside.

## 3. DETERMINING THE PATH

Determining the path allows the router to choose the port from which to send a packet that arrives at the destination. This process is called routing of a packet. Each met on the way packet router is called a jump. The number of hops is the distance traveled.

Routing tables contain the necessary information for the transmission of data packets over the network connected. The router checks its routing table to allow a flow of a packet to another router in each jump function of the load. Bandwidth, delay, cost and reliability of connection network routers will make decisions that is to say, and the router chooses the output port from which the packet is sent. The processes involved in the selection of the path for each packet are illustrated in figure 3.

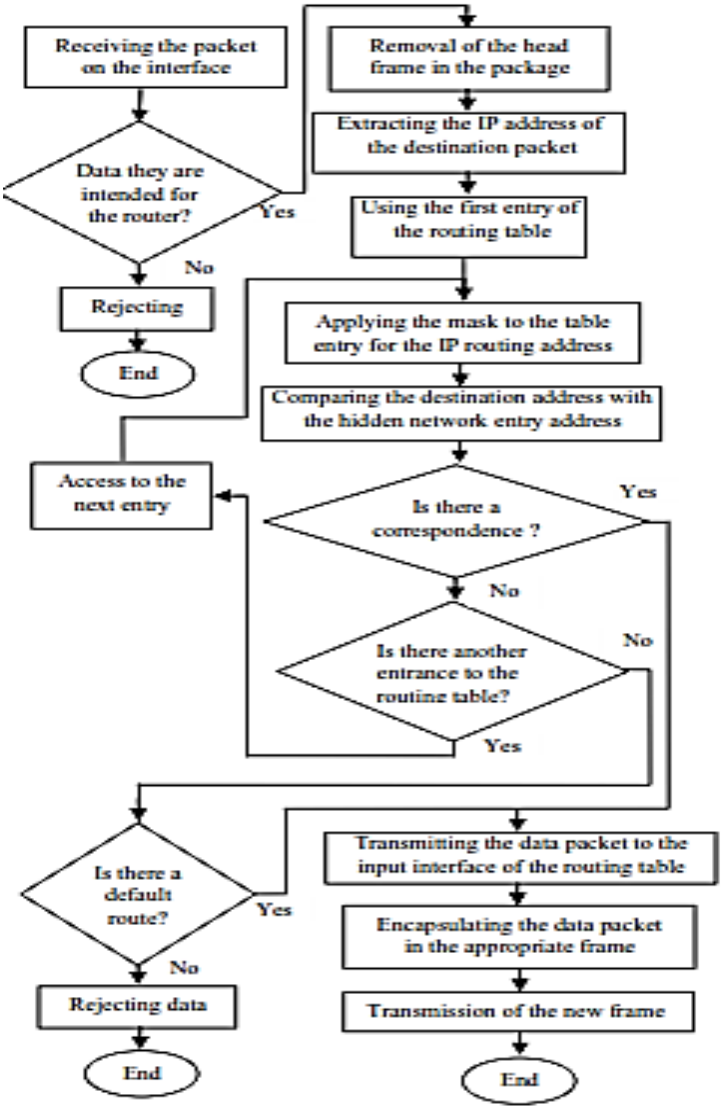


Figure 3: Process routing.

## 4. THE ROUTING PROTOCOL OSPF

Open Shortest Path First (OSPF) is a routing protocol link state developed by the IETF (Internet Engineering Task Force) in 1988 [11].
OSPF protocol is a "link state" category. Each interface on an OSPF router is assigned a metric (or cost) that is considered weight (or state) link this router to another for point to point and links to other links to point to multipoint. OSPF routers exchange statements related to their interfaces with their neighbors [2].
An OSPF router builds a whole view of the network topology. This topology is a connected graph conceptually oriented and weighted in which each vertex represents a router and the edges represent the links between neighboring routers. Based on this graph and Dijkstra algorithm, each router calculates a tree of shortest paths to other destinations including the root itself (Shortest Path Tree-SPT). This tree is then used to build the routing table of the router and the router FIB (Forwarding Information Base-FIB) [11].
We'll talk about this protocol because it is based on the Dijkstra algorithm for determining the shortest path.

## 5. DIJKSTRA ALGORITHM

Following Bellman, Dijkstra was the second, in 1959, to propose a city by any algorithm. It proposes a way to find the path with the lowest weight between two points on a weighted whose edges have a positive or zero weight

undirected graph. Shortly after, in 1960, Whiting and Hillier presented the same algorithm, suggesting that the method could easily be applied to directed graphs [3]. The execution time of the algorithm is O (|V| 2), and may be reduced to O (| E | log | V |) according to the implementation as shown in figure 4.

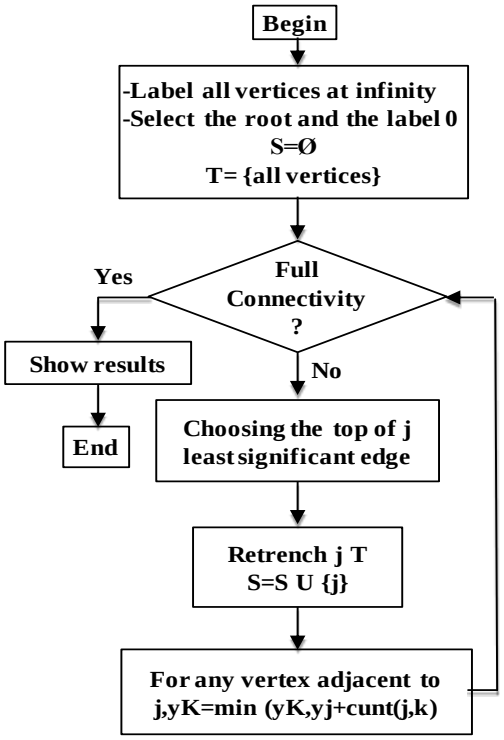


Figure 4: Flow chart of the Dijkstra algorithm.

### a. Description

Are the following considerations:
V: the set of vertices of the graph
E: the set of arcs (or edges) of the graph
T: a temporary set of vertices of a graph
S: the set of vertices processed by the algorithm
Y: a cost table of length |V| (the number of vertex of the graph)
a: the index of the starting node
+C (u, v): provides cost arc vertex u to vertex v belonging to E

### b. Algorithm

**For** ∀ i summits V **do**

Begin
$Y_I \leftarrow +\infty$
End
$Y \leftarrow 0$
$S \leftarrow Ø$
$T \leftarrow V$
**As** $T \neq Ø$ **do**

Begin
Select the vertex j of T smaller value yB
$T \leftarrow T \setminus j$
$S \leftarrow S \cup j$

**For** ∀ k summits adjacent to T j **do**
Begin
$ykB \leftarrow \min (y_k, y_{jB} + c(j, k))$ {c (j, k) is the cost of j to k} dd
End
End

We improved the main algorithm by stopping the search when $S_1 = S_{fin}$ equality is verified, provided of course seek only the minimum distance between $S_{déb}$ and $S_{fin}$. Dijkstra's algorithm can be implemented efficiently by storing the graph as adjacency lists and using a bunch as a priority queue. If the graph has n nodes and m arcs, then the time complexity of the algorithm $O((m+n) \times \ln(n))$ with a binary heap adopting the same time for the weight arcs comparisons. Indeed, the use of the algorithm gives better execution time amortize: $O(m + n \times \ln(n))$.

In a computer network, the routers must be configured to be able to reach all machines. In large networks, it is useful to achieve this configuration via a certain number of algorithms for the automatic calculation of routing, in particular Dijkstra algorithm. For this, we must have the complete map of the network to obtain the cost of the link matrix (Tab. 1). Each router will compute the shortest path between the neighboring routers. This means that it builds a pile whose root is the starting router and the other vertices.

Table1: Matrix of the cost of link.

| **from** / **to** | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| **R1** | 0 | 5 | 0 | 3 |
| **R2** | 0 | 0 | 2 | 0 |
| **R3** | 0 | 0 | 0 | 8 |
| **R4** | 0 | 0 | 0 | 0 |

For example figure 5 shows four interconnected routers. The links between routers represent the existing connections. The numbers placed on each link are the cost of use of the line.

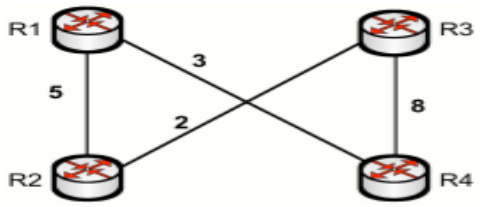


Figure 5: Example Network.

This cost can be related to the use, i.e., the available bandwidth (to designate fast lines), the size (in order to designate less congested lines). Thus, the cost between R1 and R4 is set at 3.
Each router will therefore have to determine the link cost to reach others. This means that each router will be build a tree that is the root and allowing it to reach the other routers. Costs must be minimum. If we take the case of R2, we have:
As shown in figure 6, links in bold are part of the tree whose root is R2 and which connects to the other routers. Thus, the router R2 use R2-R1 link which cost 5 to reach R1. Also, it uses the R2-R3 link which costs 2 to reach R3. Finally to reach R4, router will use the R2-R1-R4 path whose total cost is 8.

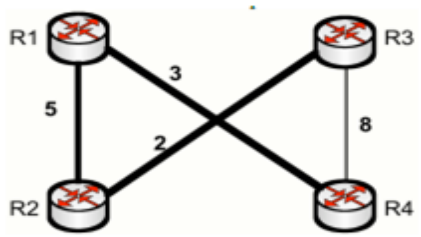


Figure 6: tree for R2.

The algorithm to implement is the one that will build from a particular router, the minimum spanning tree connecting it to all other routers. To put in other way, it is the algorithm that will determine, for a given router, the bonds will be used to achieve other routers with minimal cost.

## 6. PROPOSED ARCHITECTURE

The proposed architecture to calculate the shortest path architecture is shown in figure 7. A feature of the proposed architecture is that it is divided in four parts [8], or on blocks of descriptions. The entire matrix that contains the value of the cost of links between nodes of our network is stored in the ROM which is encoded as a package and is used in the main program to extract the cost of links (data), in the read-only cannot change its contents.

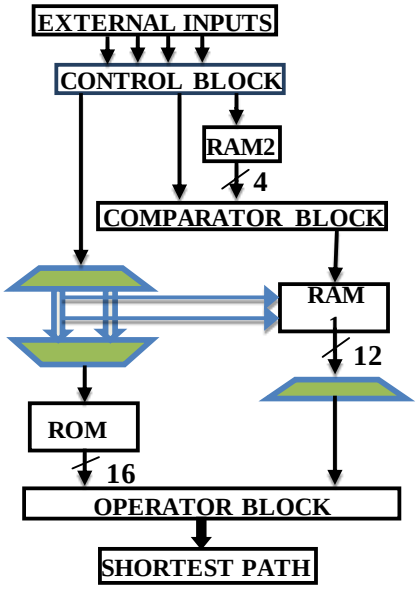


Figure7: Schema of the Dijkstra algorithm.

The data of the ROM are 16-bit vectors. The first 4 bits encode one end of the link, the other 4 followings for the other end of the link and the last 8 are to represent the value of the cost of the link between these two extremes.
RAM1 is an internal component of the program principal. Its data bus consisting of 12 lines, the first 4 are to encode the node and the other 8 for the index of this node. Its address bus has 4 lines to represent all nodes in the network.
The RAM2 is an internal component of the main program.
Its data bus consists of four lines for encoding the unprocessed node. Its address bus has 4 lines to represent all nodes in the network.
In fact we can design the block operator as a full processor and RAM 1 is one of its internal registers, as we watch the block command also as a processor with its internal register as RAM2.
The operator controls blocks and are also internal component of the main program.

## 7. RESULTS AND DISCUSSION

The architecture presented in the previous section (Fig. 7) has been coded in VHDL and synthesized using the Xilinx ISE software. So, they were simulated in modelsim PE 7.0. After the block test, the design has been implemented. After mapping the design to a Xilinx device, placement and routing of FPGA devices have been realized. The specifications of the target devices are given in table 2.

Table 2: Logic elements used in a network of 50 nodes and 72 edges

| Device Utilization Summary (estimated values) | | | [-] |
|---|---|---|---|
| Logic Utilization | Used | Available | Utilization |
| Number of Slice Registers | 3558 | 607200 | 0% |
| Number of Slice LUTs | 22690 | 303600 | 7% |
| Number of fully used LUT-FF pairs | 2916 | 23332 | 12% |
| Number of bonded IOBs | 59 | 700 | 8% |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% |

Figure 8 shows an illustrative diagram of the pre-optimized design in RTL level. It is a representation of generic symbols such as adders, multiplexers, counters and it is generated at the end of the synthesis step. It allows having an overview of the circuit at the beginning of the process implementation.

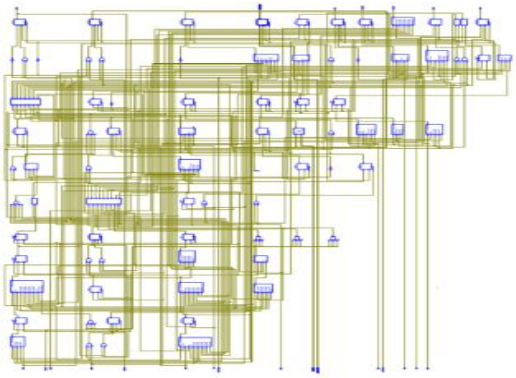

Figure 8: RTL Schema of the implementation of Dijkstra's algorithm on virtex 7.

**Simulation and Test on the Platform Evaluation**

Prior to the final stage of the test on the evaluation platform, we must simulate the operation of the SoC environment including micro blaze. This way of working allows us to see how our implementation in a SoC debug faster and also not only the RTL code of the latter, but the drivers and the test application without going through the time-intensive steps such as placement and routing.

To determine the timing performance of this algorithm and to examine the accuracy of this algorithm simulations at different abstraction levels have been validated. This operation was carried out using a test file that test-bench will be called and was conducted by using the simulation tool Modelsim PE 7.0 [12]. The results of simulation time are shown in figure 9.

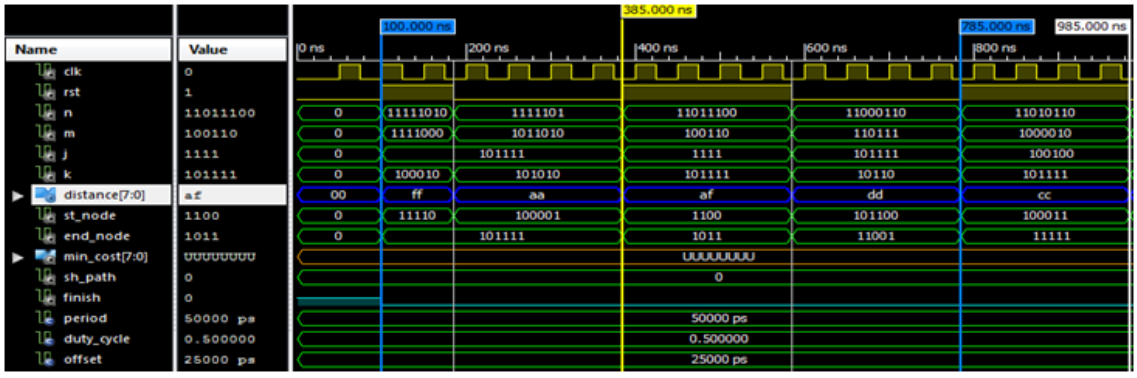


Figure 9: Results of simulation of the implementation of the algorithm FPGA.

The principal entry is a link cost matrix that represents the network topology used. The first node (n (1,1)) of the matrix is the starting node, note that the link costs are represented in hexadecimal numbers. Indeed, the link cost FF represents infinity. Loading and locking of the

cost matrix takes 100 ns. The operator block calculation process takes four iterations to complete the algorithm 785ns. The additional clock cycles are required to extract the routing table information and the total time of loading and extracting calculation is 985 ns. After each iteration, we can observe that the distance vector is updated.Table 3 demonstrates that the energy consumption by the FPGA is increased with the importance of the network which is quite logical [13] because it takes more memory resource (Tab. 2) and operating resource.

Table 3: Energy consumption.

| Environment (nodes and arcs) | Frequency (MHz) | Power consumed (mW) |
|---|---|---|
| 30 nodes (and 54 edges) | 92 | 1875 |
| 50 nodes (and 72 edges) | 97.7 | 3142 |

After completing the simulation successfully, proceed to final test is to bring everything on the evaluation platform. To do this, a bit stream is generated from the HDL files of various components of SOC (micro blaze peripherals ...) and loaded on FPGA with the test application. Table 3 shows the results of implementation on FPGA Virtex7 (XC7VX485T).

For comparison of our results with other studies from literature, we must choose studies with the same network topology (nodes and arcs). However, current computer networks used in most cases, ordinary processors. To show the performance of the use of a FPGA processor, we should make the comparison between the proper functioning of the algorithm implemented on an FPGA and the same algorithm running on a regular processor to validate our choice.

The same proposed architecture in figure 7 was tested separately on each of the two processors. We have used the Xilinx ISE 14.2 tool to implement the program coded in VHDL on FPGA (virtex7) and the same algorithm coded in C running on a computer with an Intel Core i7 processor with 8GB of RAM. All results are presented in table 4:

Table 4: comparing execution times of the Dijkstra algorithm.

| (nodes and arcs) | Executions time Dijkstra (ns) : | | Time Report µP /FPGA (%) |
|---|---|---|---|
| | VHDL sur FPGA | C Sur un µP | |
| **10 nodes (and 30 edges)** | 102.9 | 2665.11 | 25.95 |
| **30nodes (and 54 edges)** | 385.2 | $18{,}93\times10^{3}$ | 49.16 |
| **50 nodes (and 72 edges)** | 985 | $67.28\times10^{3}$ | 68.31 |

The comparison presented in table 4 shows that the execution time of the algorithm increases with the size of the network, also time reports show how the FPGA performs far faster than an ordinary processor.This is due to three main reasons.

In terms of the internal architecture of FPGA, it is designed to perform dedicated tasks very specific so that during his functioning, it performs the task assigned to it, and consequently it is faster. The conventional microprocessor meanwhile performs several tasks at once which are a cause of delay per comparing to the FPGA.Also, the FPGA has the advantage that its programming combines the flexibility of software with the speed of the hardware.

Multiple instructions are executed on variables competitively on the FPGA, and multiple arithmetic operations such as comparisons, are executed in parallel.

To load the bit stream file and the test application is used on FPGA Xilinx EDK (the iMPACT software) and a JTAG cable to connect the PC and evaluation platform. Then the test application runs automatically at the end of loading and communicates the results of the test PC through the RS232 port using HyperTerminal as shown in figure 10.

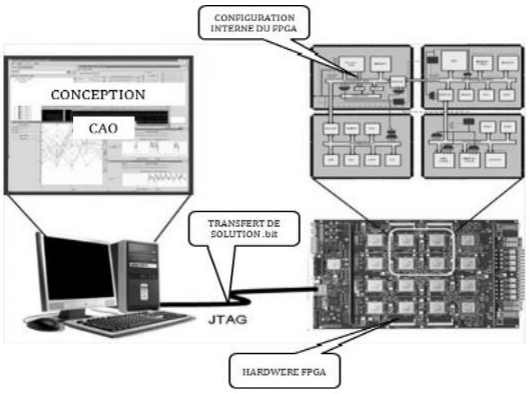


Figure 10: FPGA configuration by a cable JTAG.

## 8. CONCLUSION

This study aimed to propose an implementation approach of one of the most used algorithms in communications networks in an FPGA development board of Virtex7 (XC7VX485T).

The description of the network topology and wireless routing algorithm Dijkstra was presented. The operating characteristic of a routing protocol for a wireless network reduces the number of operations and therefore resources consumed by the FPGA achieved.
Resources used by the FPGA is realized 30% of the overall resources of the target FPGA to an environment of 50 knots and 72 stops, and 3142 mw power consumed FPGA.
Despite their relative slowness, more researchers are interested in the FPGA. Researchers interested in the Hardware free to use a machine that specializes in a task or use of FPGA chips in parallel with a conventional processor. This allows them to accelerate certain tasks materially while keeping a fast machine for any task it is asked for.
The originality of this work relates to the implementation on FPGA then it is to define a methodology for the implementation of the Dijkstra algorithm in a topology of a wireless network by varying the number of node to extract the elements logic used and minimize the energy consumed by FPGA, and the execution of the algorithm in question time and energy consumed by FPGA. For our future work, we are working to improve and adapt the approach proposed in order to perform real-time. In addition, the implementation leads to network congestion, which is why we strive to solve this problem.
Given the development of FPGA technology, it is interesting to ask what routers and even the computer will be made of tomorrow. Will there be always conventional processors like today? These processors will they be supported by FPGA chips loaded to accelerate certain tasks , or is that the FPGA chips will grow enough to compete with conventional processors in terms of speed ?

## REFERENCES

[1] Xiao X., and Ni L., 1999. Reducing Routing Table Computation Cost in OSPF, Internet Workshop, IWS99, 119-125, 18-20.

[2] Medhi D., and Ramasamy K., 2007. Network Routing, Algorithms, Protocols and Architectures, *Elsevier Inc*, 102-113.

[3] Möhring R.H, Schilling H, Schütz B, Wagner D, Willhalm T., 2005, Partitioning graphs to speed up Dijkstra's algorithm, in: Proc. 4th International Workshop on Efficient and Experimental Algorithms, in: Lecture Notes in Computer Science (LNCS), vol. 3503, Springer Verlag, 2005, pp. 189–202.

[4] Xiao B., Cao J., Zhuge Q., Shao Z., Sha E.H.M., 2004. Dynamic Update of Shortest Path Tree in OSPF. In proceedings: 7TH International Symposium on Parallel Architectures, Algorithms and Networks, IEEEXplore, 2004, pp. 18-23.

[5] Rastogi R., Breitbart Y., Garofalakis M and Kumar A. 2003. Optimal configuration of OSPF aggregates, IEEE/ACM Trans. on Networking, 2003, Vol. 11, pp. 181-194.

[6] Xilinx, Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics, Advance Product Specification, DS183 (v1.2) November 7, 2011, available on http://www.xilinx.com/support/documentation/data_sheets/ds183_Virtex_7_Data_Sheet.pdf , 2015.

[7] Fernandez I., Castillo J., Pedraza C., Sanchez C., Martinez J. 2008. Parallel Implementation of the Shortest Path Algorithm on FPGA, *in Proceedings of 4th Southern Conference on Programmable Logic*, IEEEXplore, pp. 245-248.

[8] Benaicha R, Taibi M, 2013., Feb. DIJKSTRA Algorithm Implementation on FPGA Card for Telecommunications, *International Journal of*

*Engineering Sciences & Emerging Technologies*, Volume 4, Issue 2, pp. 110-116.

[9] Tran A.T, Truong D.N, and Baas B., 2010. A reconfigurable source synchronous on-chip network for GALS many-core platforms. *IEEE Trans. Computer*. Aided Design 29(6), pp. 897–910.

[10] Pierre F., 2007. Improving the convergence of IP Routing Protocols. Thèse de Doctorat en Sciences Appliquées Université catholique de Louvain Belgique, 205p.

[11] Fortz B. and Thorup M., 2002. Optimizing OSPF/IS-IS weights in a changing world, *IEEE Journal on Selected Areas in Communications*, Vol. 20, pp. 756-767.

[12] Mentor Graphic, (Modelsim SE User's Manuel, Software, Version 20.1, http://www.mentor.com, 2014.

[13] Xilinx, "Xilinx power tools tutorial" in Xilinx, 2011.