# The Role of Petri Nets in Generating the Logical Circuits

Huda Dheyauldeen Najeeb

Department of public relations / College of Media/University of Al Iraqia

**ABSTRACT**

**This project presents an architecture of a Petri net based framework for modeling and signal transition graph used to derive speed-independent asynchronous logical circuits and also present the PN Editor module which is a necessary part in this application as well as it discusses how to export the network to a file , how to load a Petri net applications from an external file and set the necessary parameters and necessary conditions for Petri net which must be followed, otherwise it will not generate a logical circuit. It focuses on the generation of logical circuits by the Petri net which is using as the main model behavior specification. This study is depending on two researches named "Petri Flow: A Petri Net Based Framework for Modeling and Control of Workflow Processes" and "Hardware and Petri nets: Application to synchronous circuit Design" as I used them for the sake of generating logical circuits and its application by using Petri nets. In this work, the use of Java which derives asynchronous logical circuit using the behavioral patterns of circuits, after Petri net designed and created by the application of PNEditor.**

**Keyword: Petri Nets, PNEditor, Transition Graph (STG), State Graph (SG), Next-state functions.**

## INTRODUCTION

Since 20 years ago, the asynchronous circuits have experienced a renaissance as a possible solution for several problems caused by the design of these circuits. Seeing asynchronous circuits as a system of concurrent make events computational model more suitable for analysis and synthesis. Due to the increased interest of developers to the procedural algebra, for example, Petri nets. In the early 1960 Petri Net was introduced by Carl Adam Petri which one of his main charisma, such as the formal modeling is how to identify the fundamental aspects of concurrent systems both mathematically and conceptually[1].

Petri nets are finite automata and are usually computational model of sequential circuits. In this model system in each state it loads the inputs, outputs writes and moves to the next state. Time is measured in cycles, one cycle is the time of transition from one state to another. This cycle is determined by a timer. Structurally, a Petri Net (PN) is a directed bipartite graph contains transitions and places [2].

Whereas the transitions are drawn in the form of rectangles which is used to describe the events that may modify the system state. While the places are drawn in the form of circles, objects or model conditions. There are tokens inside them which are drawn in the form of black dots for representing the certain value of an object or condition. The theory of Petri nets used on a large scale for the execution of a variety of evaluation and modeling tools. Mostly can be found them in the database named a Petri Net Markup Language (PNML) which is coordinating the use of an XML-based interchange format [3].

PNEditor is used to design models which is essentially a place / transition net that is distinguish between dynamic and static places. As well as it can be modeling with subnets that are only representing visual tool designer ,for this reason, it was chosen to design a Petri net.

## PETRI NETS

Petri nets are types of network used mathematical as a model for modeling processes non-intersecting which getting at the same time (in parallel). This network considers the generalization of automata theory. They are kind of directed graphs that represent systems graphically and mathematically where Petri net components based on three main elements[4]:

PN = (P, T, F) It symbolizes the building of a Petri net.

1- Places and symbolized "P" (expressed graphically in a circle).

$P = \{p_1, \ldots, p_m\}$.

2- Transitions and symbolized "T" (expressed graphically in the form of a rectangle).

$T = \{t_1, \ldots, t_s\}$.

3- Arcs oriented symbolized "F" (expressed graphically in the form of flow relations).

$F \subseteq (P \times T) \cup (T \times P)$,

$P \cap T = \emptyset$.

Each arc up between the transmission and places, or vice versa thus we distinguish two types of arcs:

Input arcs: Sets of input places of a transition ($t_s \in T$) or transitions of a place($p_m \in P$).

$\bullet t_s = \{p_m \in P : (p_m, t_s) \in F\}$,

$\bullet p_m = \{t_s \in T : (t_s, p_m) \in F\}$.

Output arcs: Sets of output places of a transition ($t_s \in T$) or transitions of a place($p_m \in P$).

$t_s \bullet = \{p_m \in P : (t_s, p_m) \in F\}$,

$p_m \bullet = \{t_s \in T : (p_m, t_s) \in F\}$.

## PNEDITOR

PNEditor displays the habitual characteristics of a graphical editor for the design of place/transition nets. Place/transition nets can be drawn easily by PNEditor, for example transitions, labeled places, markers with multiple tokens per place and weighted arcs. Other functions is to save the net to the file, define subnets and roles, save the predetermined subnets to files, import and export file and further standard characteristics, like unbounded redo and undo actions[5].

## CREATING SIGNAL TRANSITION GRAPHS

Petri nets like timing diagrams model, which specifies asynchronous interface as signal waveforms which explicitly show the causality and concurrency relationship between signal transitions. We will show it on a small example VME bus controller. First, we need to know the behavior of the circuit, and then we model the behavior in PNEditor as a Petri net state transition graph.

Figure (1) shows a block diagram of VME Bus Controller. Input and output signals can be divided into three groups by function: signals cooperate with the bus controller (DSw , DSr and DTACK), signals interact with devices connected to the VME bus controller (LDTACK and LDS ) and signal (D) which is connect between a bus and a device via the data transceiver.

Controller behaves as follows:

- Request to write or read data from the device is received by signals ( DSw or DSr).
- Write and read request through signal( LDS).
- Transceiver must be opened by controller for transmitting data to the VME bus controller(signal D)when the data has been ready by device(LDTACK).
- Each transaction must be terminated by resetting all interface signals.

Figure (2) Timing diagram shows the read cycles.

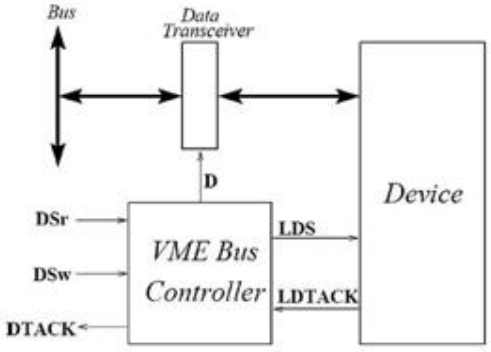



**Fig. 1: VME bus Controller**

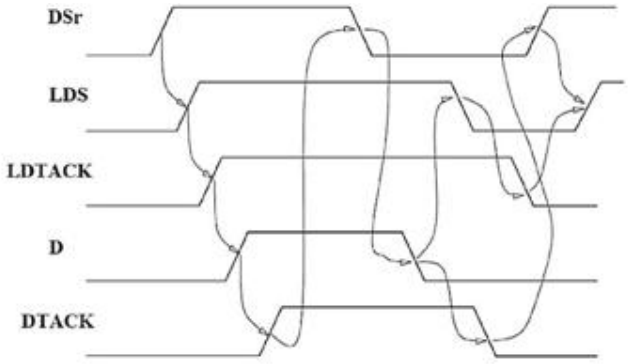



**Fig. 2: Read Cycle**

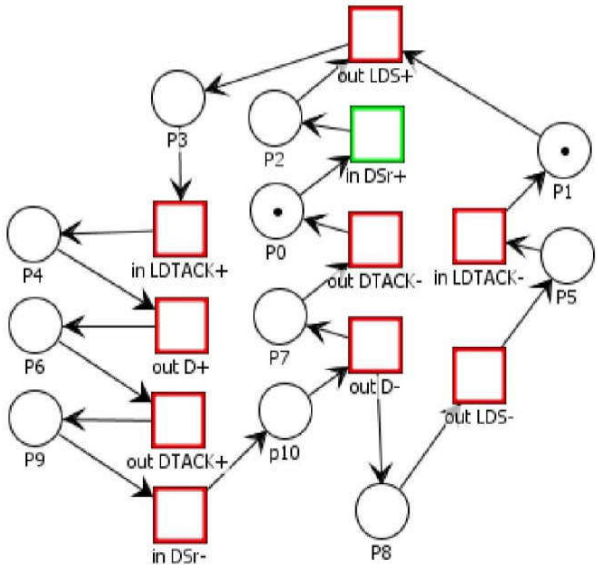



**Fig. 3: Signal Transition Graph**

Figure (3) of Petri nets is showing the same behavior. Interpret events with an increase (+) and decrease (-) signal transmission. Petri net with this interpretation is named Signal Transition Graph(STG)[5,6].

In STG, we can distinguish three kinds of signals: Input, internal and output. Where the behavior of input signals DSw, DSr and LDTACK isdefined by environment but the behavior of the internal signals and output signals LDS, DTACK, and D is defined by the system and must be implemented in circuit [6,7].To design a model we use PNEditor. It can be described as follows:

- Squares represent Transitions.
- Circles represent places.
- Arrow determines the direction to fire the token.
- Label contains the type, name of STG and direction of the signal (+/-). [8,9]

## STATE GRAPH

State Graph (SG) is depending on the specific transition system acquired from the analysis of reachability of the signal transition graph. Every state corresponds to a marking and the firing of the signal transition graph.

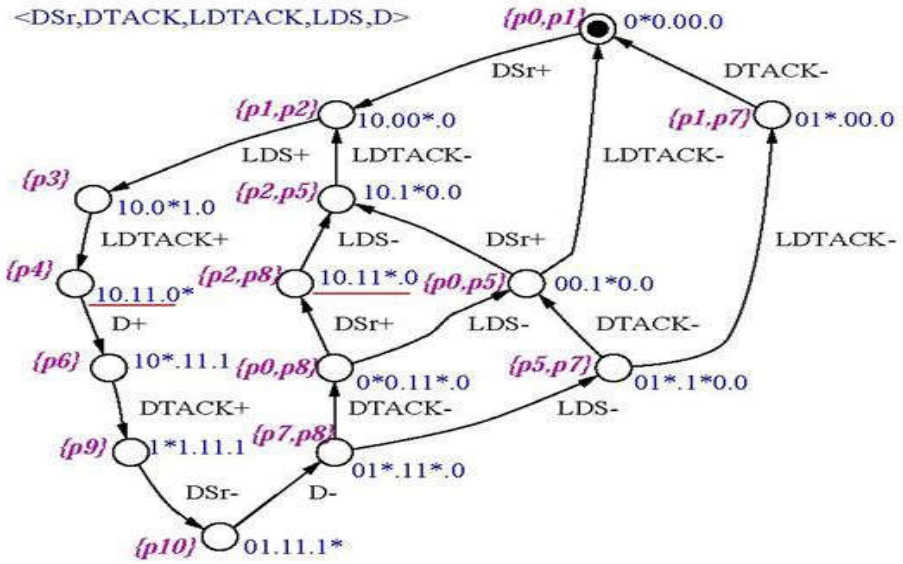



**Fig. 4: State graph for the Read cycle**

In figure.4,the corresponding state to the marking{P 7, P8}havethe code(01∗.11∗.0) , which means that( DTACK, LDTACKand LDS are at 1 )but DSr and D are at 0 .As well as both of signals LDS and DTACK are enabled.

Desired characteristics to be enforceable as a speed independent circuit:

- Consistency means STG consists ascertained that the increase and decrease are signal transmission in all possible runs.
- Persistency of signal transitions.
- Boundedness which ensures that the SG is limited Boundedness.
- Completeness of state encoding means not exist the same code with various internal or output signals behavior in the two various states.

Figure (4) satisfies consistency, persistency and boundedness but doesn't satisfy "Completeness of state encoding". The corresponding states for signs P2, P8} and P4}which having the same code but different output signals behavior. Where the event LDS− is enabled in the state P2, P8}while the event D+ is enabled in the state P4}. From this we conclude that the information of signals value isn't sufficient for determining the future system behavior. For this reason, we should learn the next-state functions [6].

## NEXT- STATE FUNCTIONS

When the state graph meet all of the features that can be derived implementability next-state function for internal and output signals. The states of the GS can be split into four groups ,when the signal *n* is gotten, these groups are: positive, negative excitation region(XR n+ and XRn− )and quiescent region(UR n+ and UR n− ).
A state belongs to XR($n^+$) if $n^+$ is enabled and n equal to "0" in this state. In this case, the signal value is signed with $0^*$ in the state graph. A state belongs toUR n+ if z is in stable 1. Those definitions are analogous forXRn− and URn− .

We can define the next-state function for a signal *n* as follows:

$$z = \begin{cases} 1 \text{ if } z \in XR\ n+ \cup UR\ (n+) \\ 0 \text{ if } z \in XR\ n- \cup UR\ (n-) \\ -\text{otherwise} \end{cases}$$

In the state graph: z indicates the binary code.

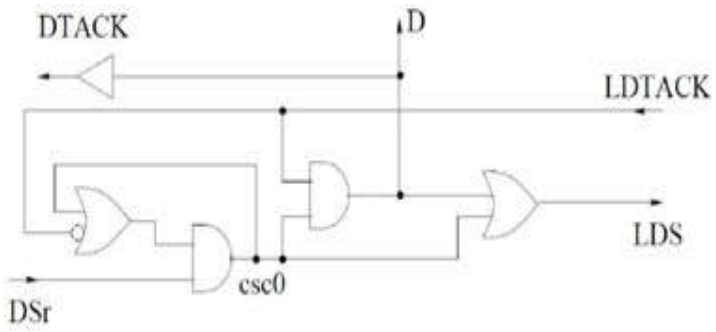
$f_s$ (z) = – means there is no state with this code , therefore, *z* became unimportant condition for minimizing Boolean[10].

For the example:

$D = LDTACK \wedge csc0$; $LDS = D \vee csc0$

$DTACK = D$; $csc0 = DSr \wedge (csc0 \vee \neg LDTACK)$

The application of the following equation was obtained logical circuit described below




## EXPERIMENTAL RESULTS

For applying and generating an asynchronous circuit, first we must design Petri net by using PNEditor module, second we must differentiate between the input signals and output signals, as well as internal transitions and the changes in the signal. Petri nets and individual signals are mapped to its passage, and it shall be checked whether the network is bounded, then whether the signals are consistent. It also verifies the persistence of each signal transition.

The applications need to store and retrieve data, therefore it was necessary to implement the following operations:

a-loading:

- pnml => Petri net
- pflow => Petri net
- xml => loading and storage of Petri nets

b- Saving:

- Petri net=>pnml
- Petri net=>pflow
- loading and storage of Petri nets =>xml

Petri net cooperates with the format "pnml" and "pflow" and asynchronous circuits with the format "xml" ("pnml" and "pflow" are also written in the format "xml", which are standardized formats recognized in PNEditor also).


**Fig. 5: depicts the label of STG**

The signal either falls from 1 to 0 or rises from 0 to 1 and thus, the Symbol ( - )refers to the fall of the signal while the symbol (+)refers to the rise of it.

After finishing from the model, we save it in PFLOW format, or export it in PNML format, now both formats are supported by the application.

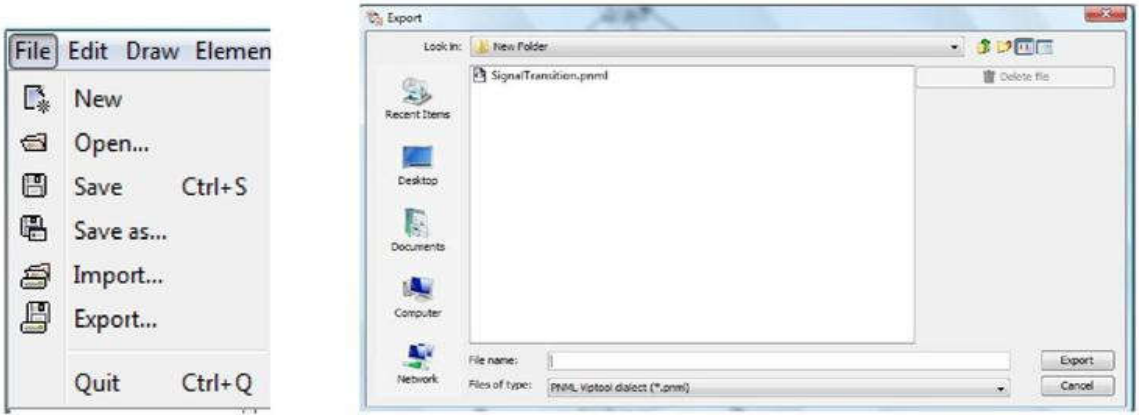



**Fig. 6: depicts the model was exported in PNML format**

**Fig. 7: Open file with STG net from PNEditor**

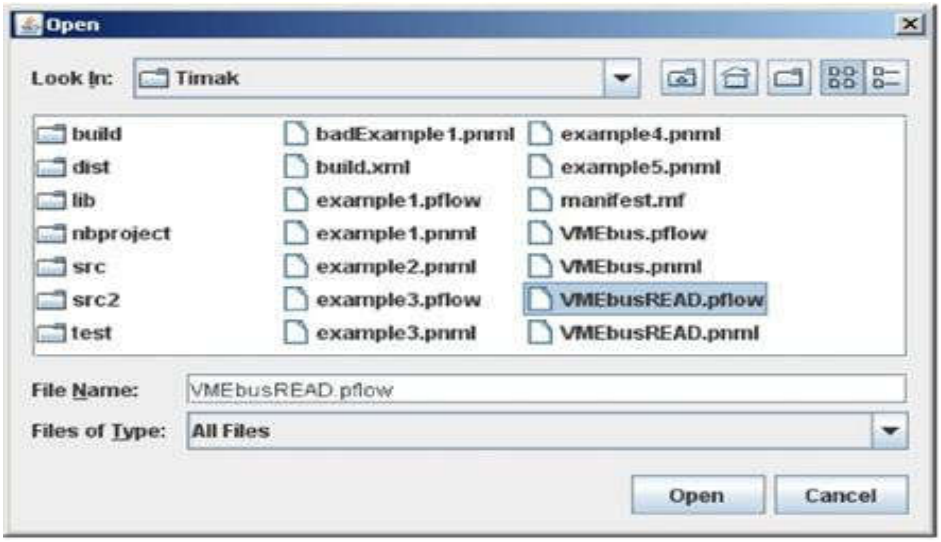



**Fig. 8: Choose file**

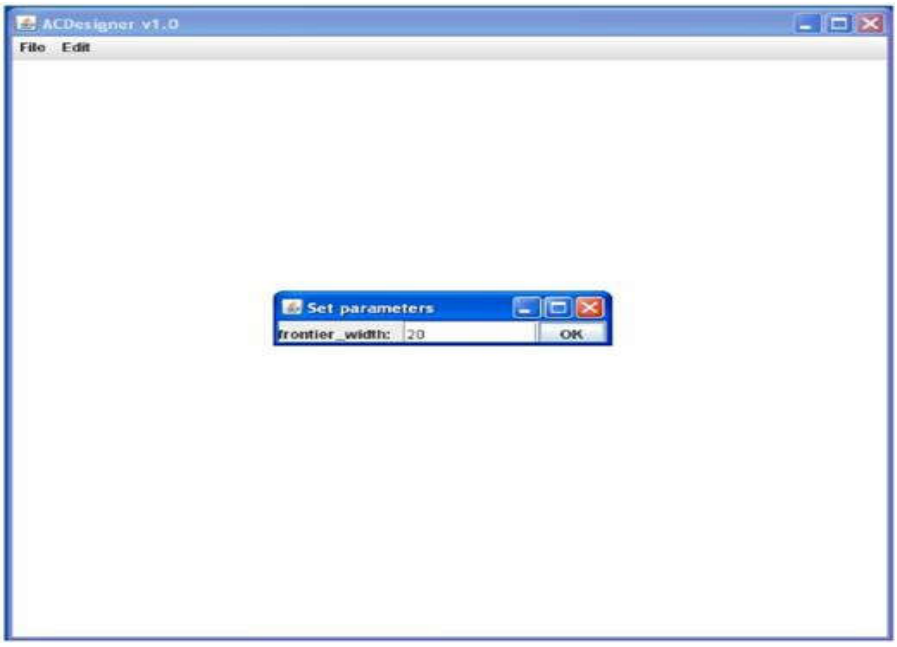



**Fig. 9: Choose Frontier Width and click OK**

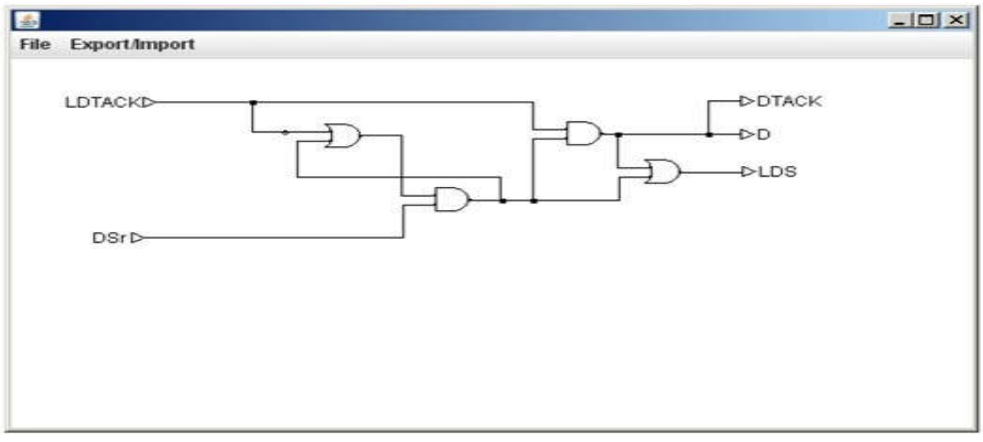
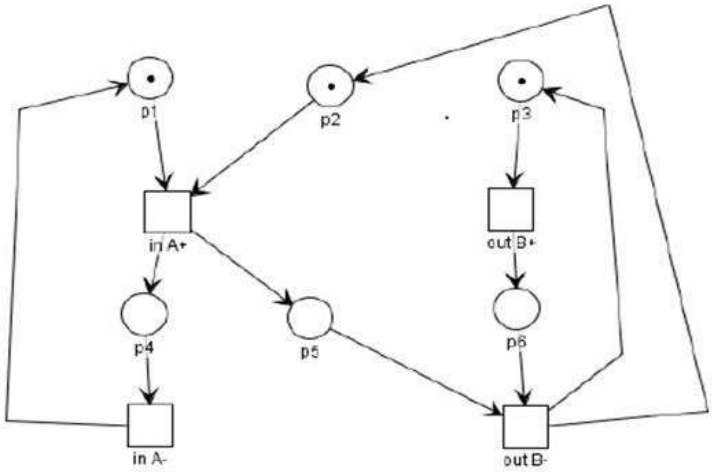


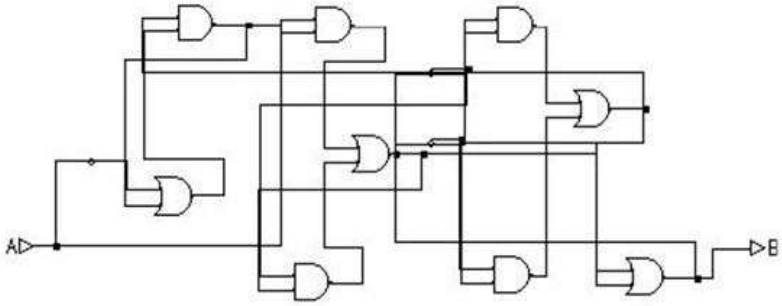

**Fig. 10: Generated logical circuit**

Logical circuit is shown on the main windows. Sometimes, resize of window is necessary to see the results. You can drag the gates and signals to reposition them. Now the circuit can be exported to xml file which can be imported it back for review or it can be exported as JPEG picture.
Some of the outputs(circuit generated by Petri net)described below in figure(11):

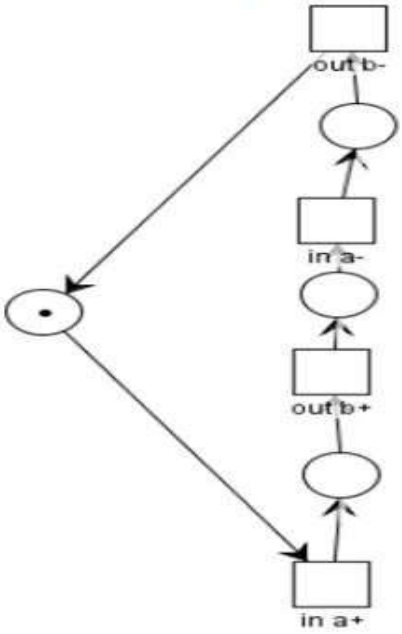
**Input1**



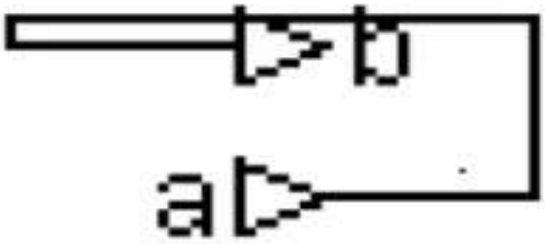

**Output 1**



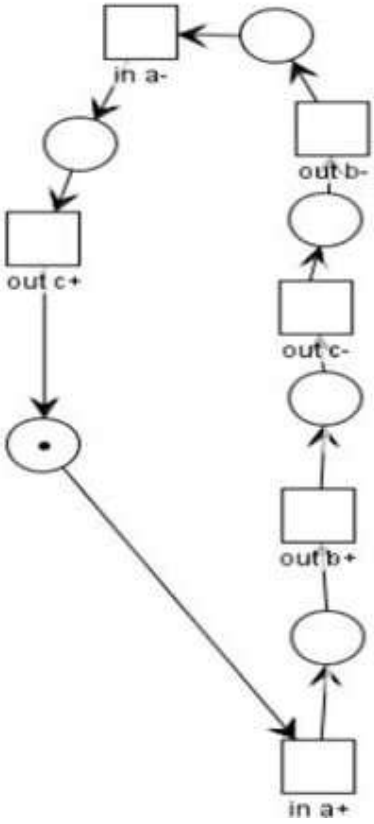

**Input 2**



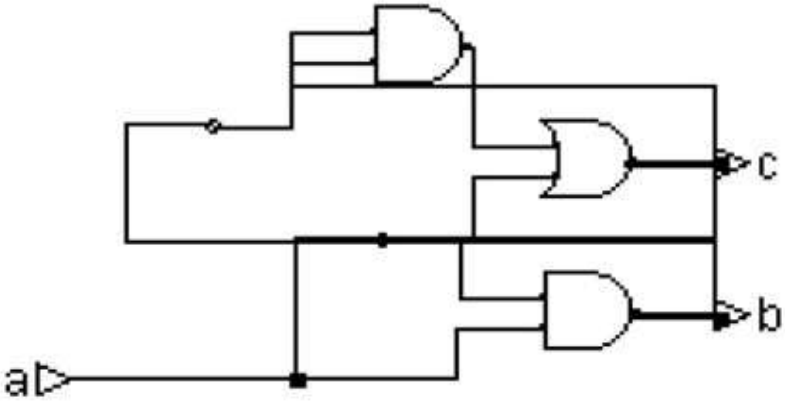
**Output 2**




**Input 3**




**Output 3**



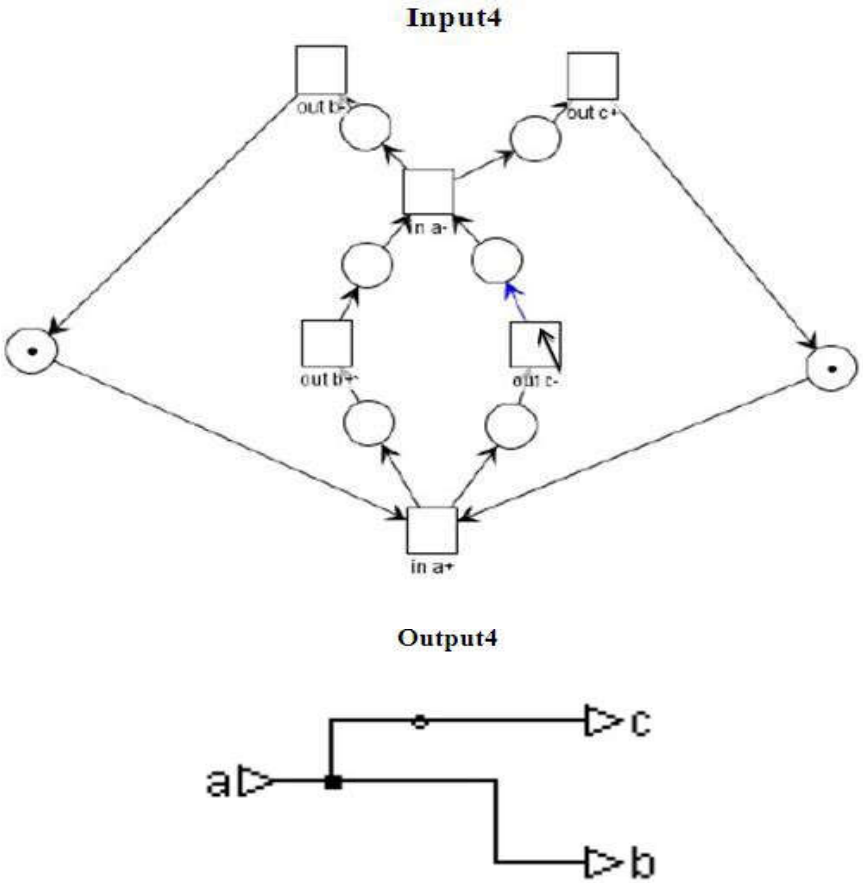



**Fig. 11: Circuit generated by using Petri Net**

## CONCLUSION

Petri nets can be used on a large scale for implementing variety of evaluation and modeling instruments. Its theory to help designers for ensuring correctness and implementation at design time. Petri nets are easy to calculate and manipulate of space of the state, for this reason we can use this technique more effectively to encode the state and logical design. In this paper we have been successfully generated logical circuit by using Petri nets, with the help of PNEditor we can design Petri nets, and therefore we conclude that it can generate any logical circuit with the help of Petri nets which must be followed with necessary parameters and conditions.

## REFERENCES

[1]. R.A. Dev, "Petri, C. Communication with Automata". Tech. Rep. RADC- TR-65-377, Center. New York, 1966.
[2]. A. Marsan, M. Balbo, G. Conte, G., Donatelli, and G. Franceschinis, "Modelling With Generalized Stochastic Petri Nets", Wiley, 1995.
[3]. J. Billington, et al. "The Petri Net Markup Language", Technology, and Tools, March 2003.
[4]. B. Arkadiusz, T. Jacek, A. Marian, "Transition Based Synthesis with Modular Encoding of Petri Nets into FPGAs", advances in electrical anelectronic engineering, (p435-p437),2014.
[5]. M. Riesz, M. Baláž and G. Juhás, "Petri Flow: A Petri Net Based Framework for Modelling and Control of Workflow Processes", Faculty of Electrical Engineering and Information Technology Slovak University of Technology,2010.
[6]. L.Y. Rosenblum and A.V. Yakovler, "Signal grahs :from self-timed to timed one". In proceedings of internal workshop on the timed petri Nets, pages 199-207, Torino, Italy, July 1985 IEEE computer society press.
[7]. T.A. Chu and L.A. Glasser, "synthesis of self-timed control" , In proc. International Conf. computer Design (ICCD) circuts from group , pages. 565-571, 1986, IEEE comp. society press.
[8]. P. Bonet, C. M. Llado, R. Puigjaner "PIPE v2.5: a Petri Net Tool for Performance Modeling", Department of Cadencies Mathematics and Informatics,07071, Palma de Mallorca, Spain,2008.
[9]. E. Kindler,"Proceedings of the Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets",ATPN 2004,25th international conference on application and theory of petri nets Bologna, Italy, June 21-26, 2004.
[10]. J. Cortadell, M. Kishinevskyi, A. Kondratyev, L. Lavagno, A. Yakovlev "Hardware and petri nets: Application to synchronous circuit Design",2001.