Proceedings of the 1st ICEENG conference, 24-26 March, 1998

MILITARY TECHNICAL COLLEGE
CAIRO-EGYPT

ICEENG 98

FIRST INTERNATIONAL CONF. ON
ELECTRICAL ENGINEERING

INCREMENTAL BEHAVIORAL SIMULATOR FOR DIGITAL CIRCUITS ON PCs

Ebrahim Zakaria *, Hamdy M. Kelash ** , Samir M. Hassan *** ,
Hassan A. Shehata ***

Abstract

Verification before fabrication is one of the main objectives of computer aided design. The highly complex designs implemented in VLSI, ULSI and WSI technologies need more sophisticated tools for verifications on the same level of complexity. Simulation as a process of modeling a real system on computer is an imperative tool especially in the earlier phases of design, where a lot of modifications can be done on the design to get a free error design. The latest development in the capabilities of personal computers makes it possible to develop many applications on PC platforms. In this paper we propose an incremental behavioral simulator for digital systems to run on PC platforms. The simulator is used to evaluate the performance of digital circuits without hardware realization of such systems. The simulator implements the event-driven mode of operation. The incremental facility of the simulator reduces the overall simulation time where the part affected by a change is only resimulated. Different case studies have been evaluated on the proposed simulator. Results of simulation time as a function of circuit size are presented.

1. Introduction

As the technology progresses the complexity of the systems designed using this technology increases. Design of digital systems is one of these areas in which the technology of production proceeds rapidly. The revolution of silicon technology made it possible to have more than one million transistors on a single chip, which is equivalent to more than a hundred thousand gates. The production of such custom prototypes is a very expensive and a time consuming procedure, so the development of such complex designs without verifying their works before fabrication is a big risk. A single error in the design means extra money and a time delay for the reproduction. Verification before fabrication is one of the main objectives of computer aided design. The highly complex designs implemented in VLSI, ULSI and WSI technologies need more sophisticated tools for verifications on the same level of complexity.

*Shebein El-Koom University, Faculty of Eng.
**Menoufia University, Faculty of Electronic Eng., Dept. of Comp. Science and Eng
*** Signal Department, Egyptian Armed Forces

Simulation as a process of modeling a real system on computer is a vital tool for verification which assists the designer to predict the performance of the designed system without physical realization. It is an imperative tool especially in the earlier phases of design, where a lot of modifications can be done on the design to get a free error design. The latest development in the capabilities of personal computers makes it possible to develop many applications on PCs platforms. The incremental behavioral simulator is one of these powerful tools which can simulate such digital systems at different levels. Simulation permits several designers to evaluate different parts of a design simultaneously . It can be used during the debugging of the prototype of a new system. It can also be used to evaluate the quality of test programs. It allows precise control of the timing of asynchronous events (e.g. interrupts). Simulation helps the designer to optimize system behavior. It also permits checking error conditions. With simulation, the simulated circuit can be started with any desired initial state to study its performance at different conditions.

2. Simulation Hierarchy

Digital system can be simulated at different levels of abstraction, ranging from the behavioral level to the geometric level. At the behavioral level a system can be described in terms of the algorithms that it performs. At the functional level which is also called a register transfer level model (RTL), the design can be used to describe the flow of data and control signals within and between functional blocks. At this level the design is made up of building blocks such as flip-flops, registers, multiplexers, counters, encoders, arithmetic logic units and elements of similar level of complexity. The logic level simulation describes a system as an interconnection of switching elements or gates. At this level the designer's interest is in logical correctness of the design and timing of signals at different nets. The circuit level of simulation is used on individual gates and functional devices to verify their behaviors. The circuit is described in terms of interconnection between resistors, capacitors, inductors, transistors and current sources. The designer is interested in knowing what kind of switching speeds, voltages and noise margins are expected. Geometric level model describes a circuit in terms of its physical shapes. In this paper we stress on the behavioral level where the design is described in terms of black boxes. These black boxes are expressed in terms of their algorithms without going deep to their constructions.

3. Simulation Approaches

Models are activated by their input signals, which could be digital or analog. Accordingly simulation approaches can be categorized as time driven and event driven. Time Driven Simulation approach is proper for analogue circuit [1], which is beyond the scope of this work. In this paper we stress on event driven simulation approach.

3-1 Event Driven Simulation

In event driven simulation, simulated time is incremented from one event time to the next, where an event represents a change in state. Thus, event driven simulation may have greater potential speed up than time driven simulation [2]. When a signal change occurs on a primary input or the output of any circuit element, then an event is said to have occurred on the net driven by that primary input or output of an element. When an event occurs on a net, then all elements driven by that net are simulated. If a signal change on the input of a device does not

cause a change on the device output, then simulation is terminated along that signal path Since the amount of activity in a time frame is minimal, why bother to simulate only those elements which are involved in signal changes. This strategy, employed at global level, requires a simulator that can operate on tables of information which include a list of the elements in the circuit and their interconnections. The interconnection list includes a list of the inputs and a list of the destination elements for each circuit device. The list is used to link elements for each circuit device and link elements together for the purpose of passing simulation results between elements and facilitating scheduling of elements for simulation. Event driven simulation can be performed in either nominal delay or zero delay conditions. The nominal delay simulator assigns delay values to logic elements based on manufacturers recommendation or measurements with precision instruments. The zero delay simulator totally ignores delay time within a logic function performed by the element. The nominal delay simulator gives precise simulation results but on the cost of CPU time. The zero delay simulator usually runs faster but does not indicate when events occur. The work done in this paper deals with digital systems. The digital system simulator will run in an event mode of operation.

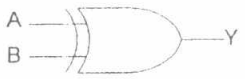
4. Modeling of Blocks

Modeling is an important stage prerior to simulation. At this stage the simulated blocks are to be expressed at the required level. Models are written in Pascal language because of its simplicity and availability. In our work we describe the models in a standard form as shown below.

PROCEDURE <module name>(set of inputs : type,set of outputs : type, input time, delay, output time : type);

As an example EX-OR is described according to its truth table shown in Fig.1,

Logic Gate



Truth table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Fig.1. Logic symbol and Truth table of EX-OR gate

Accordingly the model of the Exclusive_OR can be expressed as :

```
PROCEDURE XOR(a,b,y,Tin,delay,Tout : integer);
  begin
    if a=b then y:=0
           else y:=1;
    Tout := Tin + delay;
  end;
```

5. Structure of Simulator

The infrastructure of the simulator, shown in Fig.2, consists of :

- **Schematic Library:** It contains the graphic representation for each component and its database.

- **Schematic editor:** The user edits his design using his own components from the ready made library or create his own components.
- **Translator:** This program It converts the schematic file to a text file written in Structured Design Language format (SDL), describing the simulator system, into a form suitable for simulator.
- **Behavioral Model Editor and Library:** It contains the procedures describing the algorithm of each block and also the designer can edit new procedures for new blocks to be simulated.
- **Stimuli file:** It contains the input signal required for executing the models. these signals are written in a certain format containing information about the excited net, its value, its time.
- **Simulator Core:** It contains two sub modules; the scheduler module and the component status module.

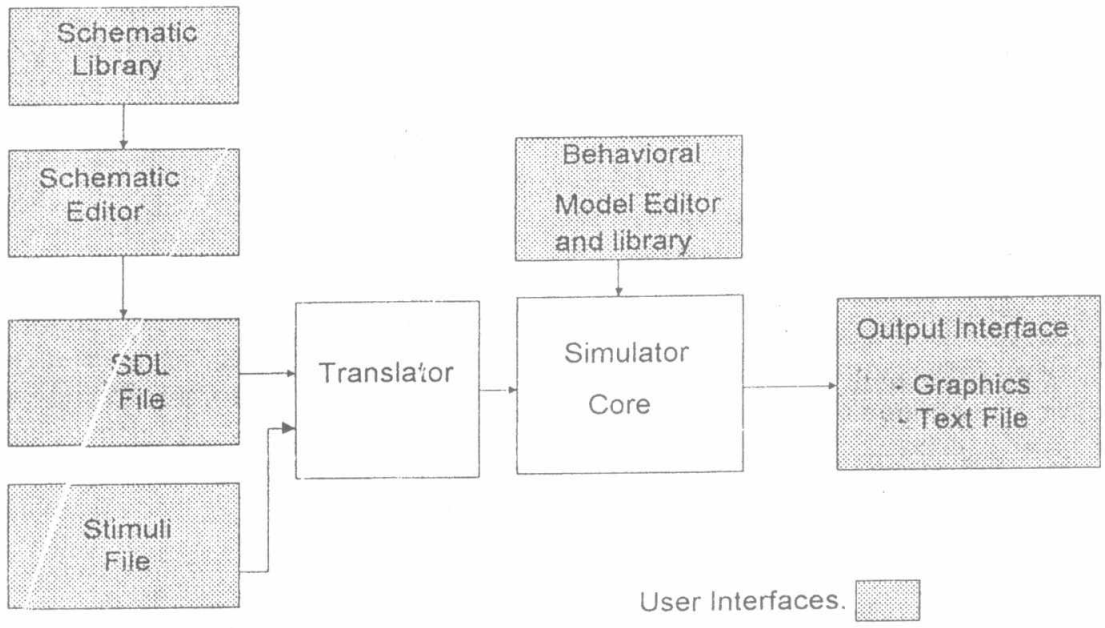


Fig.2. Structure of simulator

5-1 Scheduler

The digital simulator implements the event driven mode of operation. In event driven simulation when a signal change occurs at a certain net, then an event is said to have occurred at this net. When an event occurs on a net, then all elements driven by that net are simulated. If a signal change on the input to a device does not cause a change on the device output, then simulation is terminated along that signal path. During simulation; the events corresponding to primary inputs as well as to any other nets in the design should be arranged on a time priority. This operation is carried out by the scheduler. The scheduler is constructed from a two dimensional array called the scheduler array containing net name, value and time. The scheduler implements the algorithm shown in Fig.3, where The scheduler array receives input stimuli either from input interface or from the component status array. The scheduling

procedure sorts the contents of scheduler array by time. It transmits the nets of the lower time stamp at the top of the scheduler array, to the component status array. After that it shifts the contents of the array to delete the transmitted nets to be ready for receiving the calculated nets from the next phase of simulation.

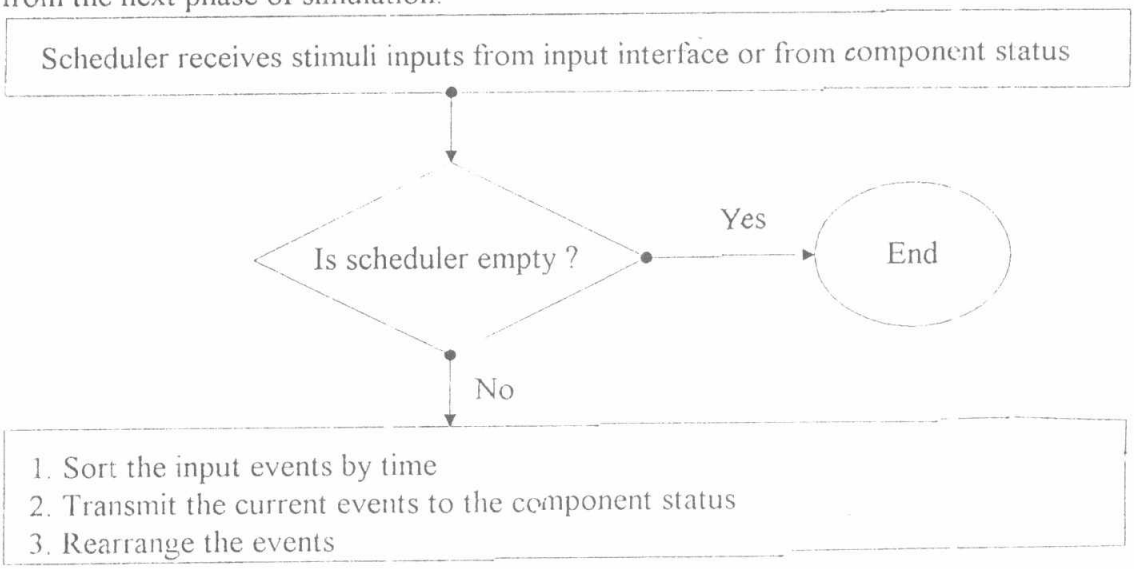


Fig.3. Algorithm of scheduler array

5-2 Component Status :

Each simulated component in the design has a group of inputs and another group of outputs. These nets change their status from a phase of simulation to another. The value and time of activation of each net is stored for the previous and current time. If there is an event(s) at the input of a component, its behavior is executed to calculate the corresponding output(s). All these activities of storage, updating and calculations are carried out by component status stage. This stage has an array called the component status array shown Fig.4, It is a three dimensional array containing names of input nets, old value of the input nets, new value of the input nets, names of output nets, old value of the output nets, new value of the output nets at the previous

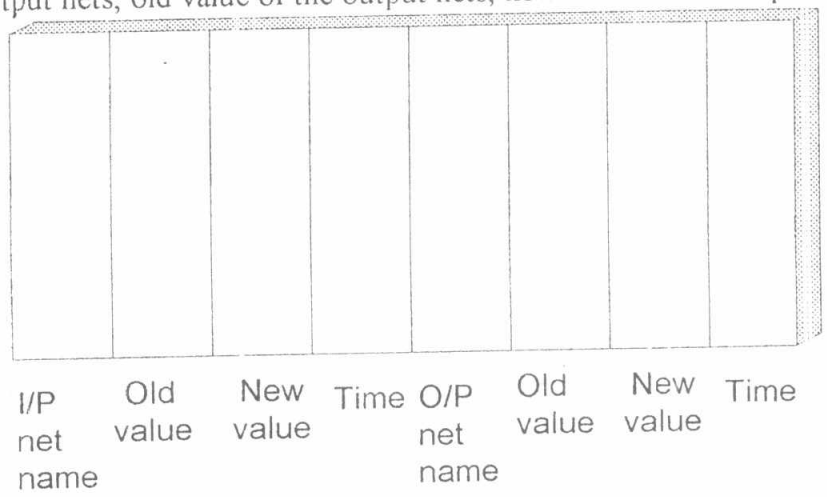


Fig.4. Component status array

and current time stamps. After building the component status array with the connectivityinformation it receives the transmitted nets from scheduler array and update the component status array at the event time. Then it calculates the output at the event time and update the component status array, and transmits the changed outputs to the scheduler array.

6. Incremental Simulation

When there is an error in the design (behavioral or schematic), or it is desired to make a change in the design (inserting or deleting a block) to get the required specifications or to enhance the design. Normally, simulators deal with such situations as a completely new design, that should be simulated from the beginning. The incremental simulator seeks to reduce the time delay from which the conventional simulator is suffering from when it resimulates a design after making some changes on it. The simulator reduces the computational effort by using the previously evaluated nets outside the cone of disturbance and reevaluates only the nets inside that cone.

The incremental simulator should have the following characteristics:

a) Simulation continues instantaneously after a pause in which changes in interconnections between the subsystems (models) or logical blocks can be made.
b) Minimum delay to introduce new blocks into the model.
c) Not to recompile the complete model when making some changes in the behavioral model of some blocks.

To explain the concept of incremental operation consider the circuit shown in Fig.5, which is the first phase of design.

Assuming that the circuit has been simulated, it means that all the nets values have been calculated for the complete simulation period and stored in the following form :

net number net value time

Now as a result of this simulation result it is realized that an inverter has to be incorporated into the path connecting the output of AND1 and the input of NAND3.

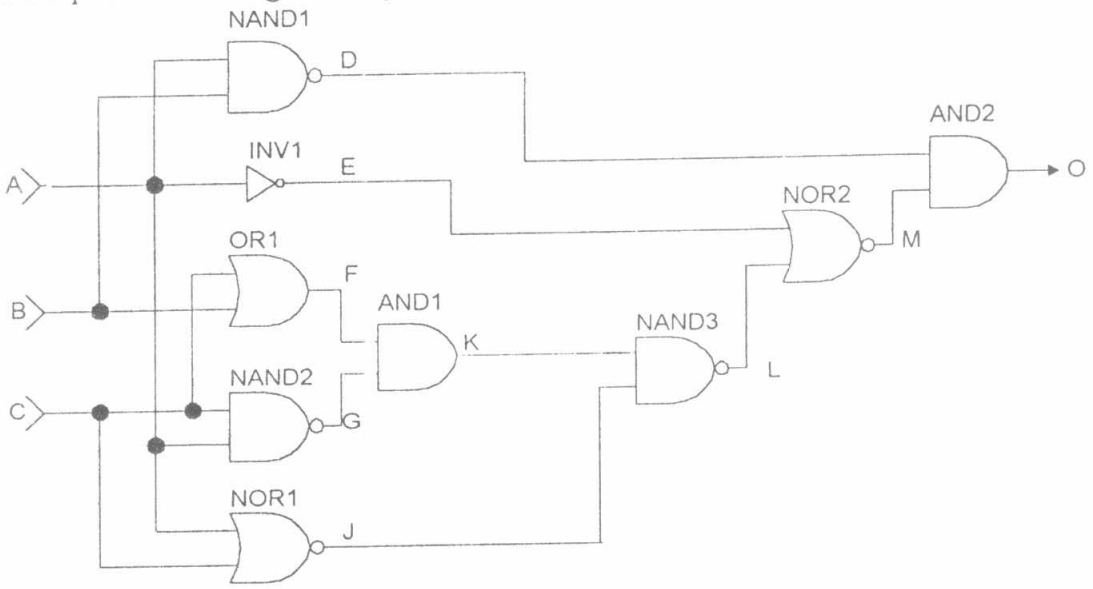


Fig.5. A circuit under simulation (at the gate level).

The circuit after inserting the inverter INV2 is shown in Fig.6,

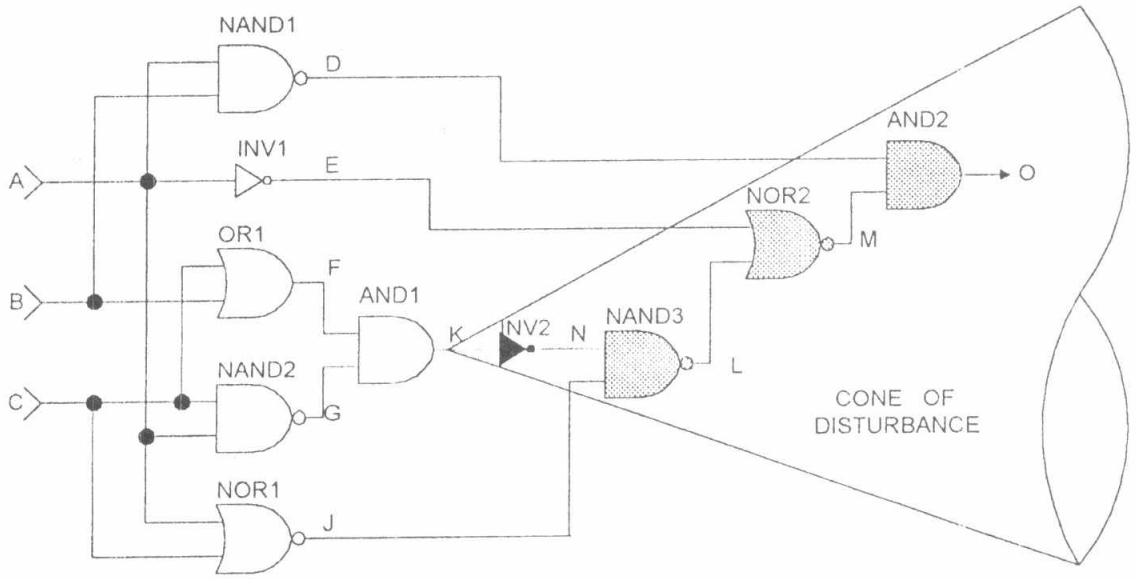


Fig.6. Cone of disturbance after inserting an inverter

What the incremental simulator will do with this circuit is:

1) Determination of the cone of disturbance: This cone contains the gates that are affected by the insertion of INV2. In this case these gates are NAND3, NOR2, AND2. These gates only should be reevaluated in simulating the new circuit.
2) Determination of the cone driving signals: Signals D, E, J, and K which enters the cone from outside are considered as drivers for the components inside the cone of disturbance.
3) Simulation of the components inside the cone of disturbance.

The circuit which will be simulated again is shown in Fig.7.

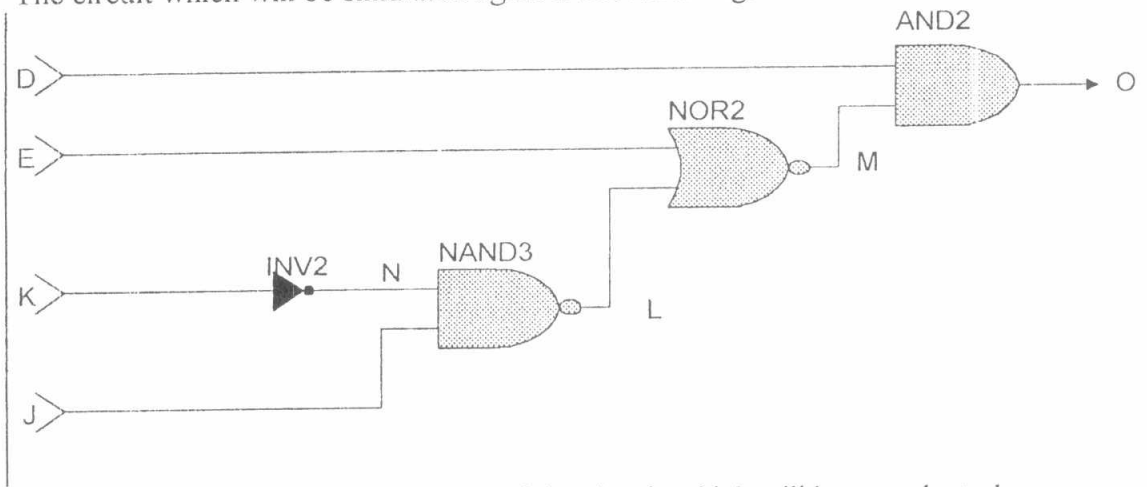


Fig.7. The portion of the circuit which will be reevaluated.

The simulator does not reevaluate all the entire circuit but only the gates inside the cone of disturbance. The history of the signals D, E, J, and K is available from the last simulation run.

Thus if these values are accessed again and applied to these inputs, we can get the new results of this new phase of design[3]. Incremental simulator presents a very fast simulation tool which responds very quickly to the design changes occur frequently in the earlier stages of design. This simulator achieves speed up ranges from 3 to 20, where the running time is dependent on the relative locations of design changes on the incremental simulator. This speed up of simulation process helps the designer in making his changes on the design with minimum time delay. The simulator presented in this paper has the following features :

1. The simulator can detect glitches caused by hazards.
2. The simulator is designed to detect instances where two or more tri_state devices are simultaneously active.
3. Display of signals at different points in time domain.
4. Incremental mode of operation.

7. Results

The circuit shown in Fig.8, has been simulated on the proposed simulator. It represents a Pseudo Random code (PN) generator. It consists of 3 stage shift register and an EX_OR gate. It produces a code of length $2^3 - 1 = 7$. The clock is running at a frequency of 1 MHz. The output wave forms of simulation are shown in Fig.9-a,b, for two different initial conditions (1, 1, 1) and (1, 0, 1) stored in shift registers. To evaluate the performance of the simulator a design with different sizes has been simulated. The design is a two word full adder with 1-bit, 2-bit and 3-bit words. Simulation time for these three cases are shown in table 1, where the simulator runs on a Pentium133 platform.

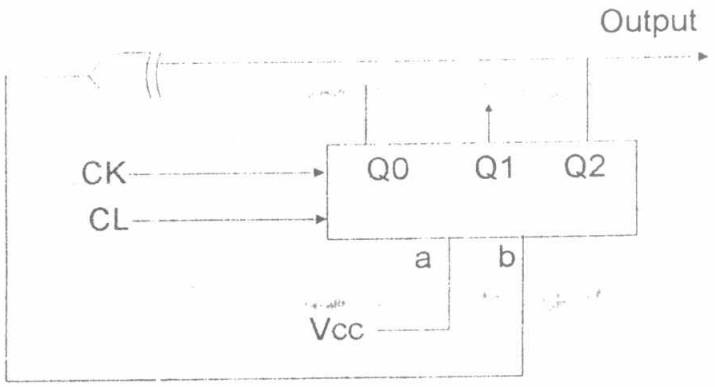


Fig.8. Pseudo Random code generator

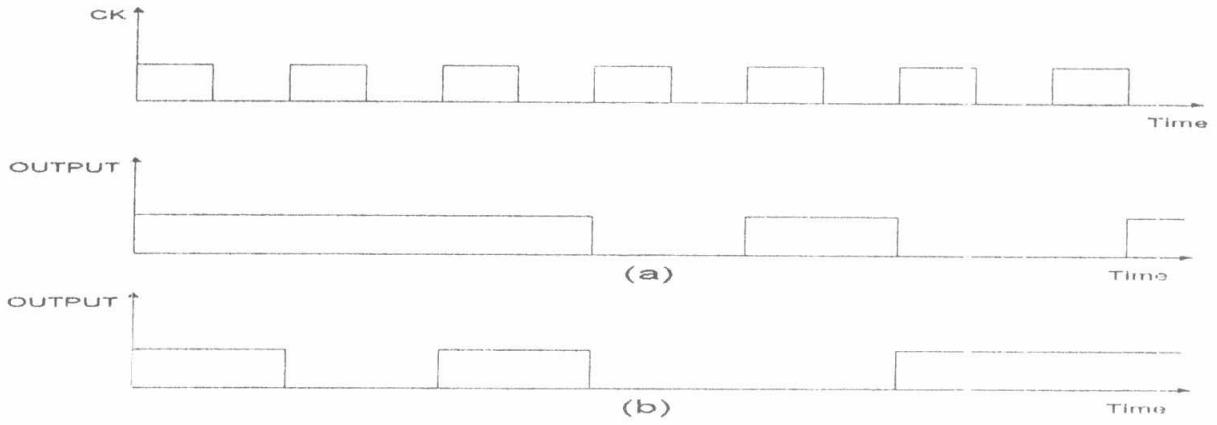


Fig.9. Output waveforms of simulation

Table 1. Simulation time of full adder circuits

Circuit Name	No. of Gates	Simulation time
1-bit word full adder	5	0.9 msec
2-bit word full adder	10	3.96 msec
3-bit word full adder	15	9 msec

8. Conclusion

The simulator presented in this paper is dedicated to digital circuits. It implements the event driven mode of operation. The simulator has incremental feature. It runs on PCs where the simulator software is written in PASCAL. Many circuits have been tested on the simulator and verified. PC platforms show powerful capabilities in running behavioral simulators for digital circuits.

9. References

[1] R. Righter, J. Walrand, 'Distributed Simulation of Discrete Event Systems' , Proceeding of the IEEE, Vol. 77, No. 1, PP.99-113., Jan.1989.

[2] Alexander Miczo,'Digital Logic Testing And Simulation' , John Wily & Sons, 1987.

[3] K Dimond, S Hassan, 'An Incremental Functional Simulator Implemented on a Network of Transputers', the Proceedings of the European Design Automation Conference, Glasgow, Scotland, pp296-300, 1990.

[4] Drago M., Rihard k., Borut Z.,'Simulation And Modeling Of Continuous', 1992.

[5] Miron A., Melvin A. B., Arthur D. F., 'Digital System Testing And Testable Design', Computer Science Press, 1990.