A New Attitude based on Real Time Operating System for NoC in Hotspot Traffic Model

Seyyed Amir Asghari, Hossein Pedram and Hassan Taheri

Abstract—The reduction in the size of transistors, leads to the increase in the numbers of transistors to more than several billions on a chip. Therefore, new techniques have to be carried out to manage this large quantity of transistors on a single chip. Network on Chip (NoC) is an implementation technique to resolve this problem. But this NoC management is a challenging job and the communication management need regular scheduling and configuration. One attitude towards NoC management is making use of Real Time Operating System (RTOS) for scheduling, task introduction, and dynamic assigning priorities to the tasks and message passing. Therefore in this paper, MicroC/OS-II RTOS is used. This RTOS is ported in Motorola ColdFire microprocessor. This microprocessor is located in the core of a node of mesh topology based NoC. The traffic model in this paper is hotspot.

Index Terms—MicroC/OS-II, Motorola ColdFire Microprocessor, Network on Chip, Real Time Operating System.

I. INTRODUCTION

THE System on Chip (SoC) can include different components such as processor, I/O unit and various types of memories. Each of these components can have different communication protocols [1].

Generally, Interconnection processing elements in NoC is carried out by ports, whereas, in multiprocessor SoC (MPSoC) with numerous processing elements, it is expected that these ports in the case of latency, scalability and energy consumption, are turned into bottlenecks.

Therefore, the idea of NoC that includes the routers which are connected by the means of links is introduced. But the communication management in NoC is a challenging job. So, utilization the RTOS will be in charge of managing this challenge. This OS can be ported on NoC node microprocessor. In this paper, MicroC/OS-II RTOS is ported in the central node of NoC mesh topology based on hotspot traffic model. However, the OS can be ported in the all nodes. The idea of applying NoCs also has been used in the previous works such as [9].

In this paper MicroC/OS-II is used in an innovative way that is making use of the RTOS. This OS, in contrast with similar OSs such as Windows and Linux is not monolithic and application program do not effect on kernel. Also, it has a few number of code lines for kernel that it has a willing impact on power computing to usual OSs. As NoCs are power constrained, this is considered a privilege feature [11].

In the 2nd part of this paper, NoC structure and its components are introduced.

In the 3rd part, MicroC/OS-II RTOS and its privilege features are introduced.

In the 4th part, different types of traffic models are explained. A specific traffic model which is being taken into account is hotspot traffic model.

In 5th part, Motorola ColdFire processors are introduced. In the implementation of OS based NoC, the MCF5484 ColdFire processor is used.

In the 6th part, microprocessor programming and debugging tools are introduced.

In the 7th part, two different attitudes, one based on using OS, the other one without using OS are compared and the advantages of OS based NoC are brought up. Also, in this section, a PrioRout routing algorithm is introduced.

In the 8th part, the carried out implementation is presented and the last part the final conclusion is brought up.

II. NOC STRUCTURE

A NoC has been formed of routers and links. The IP blocks have been connected to each other by means of the network interfaces (NI). Also the routers communicate to each other over links. A router distinguishes packet paths in network. The router has been concluded of some buffers, a routing function unit, a selection function unit and a switch for packet transmission to packet destinations [2] [10].

Network Interfaces justifies IP block communication protocol and packet transmission protocol by means of the router. Each network interfaces can connect several IP blocks to the routers.
III. MICROC/OS-II RTOS

MicroC/OS-II is a RTOS that has been applied to embedded application. If we have a toolchain (A system concluded compiler, assembler and linker), we can add an OS to it. MicroC/OS-II has a full preemptive and real time kernel which means OS runs the high priority tasks which are ready to running. Many traditional kernel acts on format of preemptive, but the MicroC/OS-II is much better than them.

Analysis of OSs with monolithic kernel (such as Windows and Linux) which is consisting of millions of line of code when they encounter problem is difficult and nearly these OSs would not bug free.

The kernel of MicroC/OS-II has only 5000 lines of code and we can confirm that it reached to a level that will be bug free [3].

A. Multitasking feature

MicroC/OS-II can manage up to 64 tasks. However MicroC/OS-II reserves the four highest priority tasks and the four least priority tasks for its uses. So it leaves the 56 free tasks.

B. Multitasking feature

For MicroC/OS-II task managing capability, first we need to be creating a task. For creating the task, we can use one of these functions:

- OSTaskCreate
- OSTaskCreateExt()

OSTaskCreateExt() is an extended version of OSTaskCreate() that it has some extra features. For a creating multitasking, at least we need to create one task. We can not create the task with Interrupt Service Routine (ISR). In the figure 2 we can see the segment code of OSTaskCreate function:

```
INT8U OSTaskCreate (void (*task)(void *pd), void *pdata, OS_STK *ptos, INT8U prio)
```

As you see above, need four arguments:

**Task**: A pointer to task code.

**Pdata**: It is a pointer to the argument. This argument passed to the wanted task of the beginning moment.

**Ptos**: It is a pointer to the top stack. This pointer should be assigned to the task.

**Prio**: It is the priority of the wanted task.

IV. TRAFFIC MODELS

The traffic model is one of the important parameters in evaluating the latency time of interconnection networks.

These models are produced according to the application programs which are run on the machine. In different application, different models are used. Traffic models are defined according to three parameters [4]:

- The entrance time to networks
- Message length
- Address distribution type

A. The uniform traffic model

Uniform traffic model is the simplest traffic model which used in most of evaluations. In this model, each node sends message to the other nodes in network with equal probability. For example in a $6 \times 6$ mesh topology, each nodes sends message to the other nodes with the probability of %2.85.

All source or destination nodes are selected with equal probability. The selection of source and destination node for each message will be independent from other messages [4].

B. Hotspot traffic model

In hotspot traffic model, the numbers of messages which are sent to special node as the hot node are more than the other nodes. Usually the one node is considered as a hot node.

Because of sending some packets of the created messages in network to this spot, the traffic around this node is more than the other spot.

Equalizing protocols and OS functions are the instances which lead to the production of this kind of traffic. The most colorful node in figure 3 is the hot node and the traffic congestion is clear around it.
C. Permutation traffic model

Permutation traffic model is another traffic model that a lot of parallel programs like FFT, matrix problems, and fault tolerant routing algorithms have behavior like it.

In this model, the destination address is found by placing the source address in a permutation function. So for each source address there always is a destination address. Bit reversal, First (second) matrix transpose, shuffle and butterfly traffic models are some examples of the permutation model. For instance the traffic model of matrix transpose explained; if we consider M and N as the dimension size of the 2-D network and \((i,j)\) as the source node address, the destination address is produced as follow:

\[
(i, j) \rightarrow (M \times N - 1 - j, M \times N - 1 - i)
\]  

(1)

The destination address in second matrix transpose is produced as follow:

\[
(i, j) \rightarrow (j, i)
\]  

(2)

D. Local Traffic model

Local traffic model is similar to application program. In this model, each node sends special volume of its created message to its neighbor. The number of neighbors is related to the distance between neighbor nodes (called neighbor radius). Radius one is shown in figure 4. In that the block nodes are the neighbors of node.

In all explained traffic model, some percentages messages are distributed as per mutative, local or one sent to the hotspot and the other messages are distributed in another way which is usually uniform.

V. ColdFire Microprocessor Introduction

Motorola corporation is one of pioneer in producing 8, 16, 32 bit microprocessors and microcontrollers. ColdFire microprocessor family is the most famous and successful production of its company. These processors have m68000 architecture that which are suitable to be used in real time system. To meet this purpose of this paper MCF5484 is used.

VI. BDM Module as a Debugging and Programming Tool

The figures 5 show the interface of this module with processor core and its other interfaces. As you see, debug module is connected to the main bus of the microprocessor and so in some cases if can work with ColdFire CPU core in a parallel form.

The capabilities of this module are divided into three groups:
A. Real time trace support
It has the ability to dynamic calculation of the running path, which is useful for debugging. ColdFire has the ability to place 8 bits of parallel data on emulator. This data shows the microprocessor status and memory data.

B. Background Debug Mode (BDM)
This capability provides low level debugging for ColdFire. In this module we can access the memory without stopping the microprocessor. But changing amount of registers needs to halt the microprocessor.

C. Real time Debug Support
With use of Debug Interrupt Routine Time, in this mode, the amount of registers and variable data are saved fast and the systems returns to normal stopping the main program.

BDM mode is useful for the following reasons:
- BDM is always accessible for debugging and firmware upgrading
- It is used for programming external flash
- It provides the entire control of the microprocessor and so the whole system.

These features lead to debugging the microprocessor by the use of those tools, which are used for programming the microprocessor.

Although, most of BDM commands don’t lead to halt stopping and they are capable to be run with a program concurrently. Some conditions which lead to microprocessor stopping are available as follow:
- Fault occurrence in BDM system
- Breakpoints
- Halt command that can be activated with 'Go' from BDM

VII. PERFORMANCE COMPARISON IN TWO STATUSES: WITH AND WITHOUT OF OS

In this section, we want to compare two different attitude in a mesh topology based NoC. For this comparison, we use of 3×3 mesh topology based on hotspot traffic model. The use of OS in topologies with limited nodes is worth if nodes communication complication or the number of defined tasks are a lot. In the attitude that OS is not used, the pass traffic in the central node of hotspot traffic model is a lot. So as a result there is the probability of the congestion of the packets when input packets are assigned the output. In order to remove this problem, we use the virtual channel. However these virtual channels increase many overhead. For each channel which is added the power consumption increases and results to the increases of power in this attitude.

If virtual channel are used, the router needs to use the MUX and DEMUX for he selection of the packets. The figure 6 shows the packet placing in virtual channel and also the selection of packet from virtual channel.

As you see in this attitude, some components such as MUX, DEMUX and buffers are necessary. These components lead some complication like a buffer management and packet selection from buffers. In the attitude based on the use of OS, we define task s based on I/O ports (Local port is negligible). As a result there are four tasks: North, South, East and West.

Now, OS assigns one task priority for each port. Based on the assignment of priorities to these tasks, we can manage the routing of the input packet to input port easily.

OS is responsible for scheduling and task management. In this trend, priority assigning is programmed in the way: each time the output port is busy, the free ports based on PrioriRot are used.

A. Deterministic:
Execution time of all MicroC/OS-II functions and services are deterministic. This means that you can always know how much time MicroC/OS-II will take to execute a function or a service. Furthermore, except for one service, execution time of all MicroC/OS-II services does not depend on the number of tasks running in your application.

B. Task stacks:
Each task requires its own stack. However, MicroC/OS-II allows each task to have a different stack size. This allows you to reduce the amount of RAM needed in your application. With MicroC/OS-II’s stack checking feature, you can determine exactly how much stack space, each task actually requires.

C. Services:
MicroC/OS-II provides a number of system services such as mailboxes, queues, semaphores, fixed-sized memory partitions, time related functions, etc.

D. Interrupt Management:
Interrupts can suspend the execution of a task and, if a higher priority task is awakened as a result of the interrupt, the highest priority task will run as soon as all nested interrupts complete. Interrupts can be nested up to 255 levels deep.
E. Critical section of code

The critical section of code, briefly named critical section, is a code which should be atomic and run as a basic block necessarily. So the segment code is uninterruptible when placed in this section. To assure that, all interrupt are disabled before critical section to be ran and after that they will be able again.

VIII. Prior Rout routing algorithm

In this algorithm, input packet will choose a different output port based on the selected input port and its destination. In the figures 7, we can see a 3×3 mesh topology of a NoC. In this topology, OS has been ported on the router which has special color.

![Mesh topology based NoC](image)

Figure 7. Mesh topology based NoC

![Central router ports](image)

Figure 8. The Central router ports

The number of router ports depends on the location. For example the router which situated in the north east has three ports: Eastern port, Southern port and Local port. The router which the OS has ported on it is located in the central of the mesh topology and it has five ports which are: Southern port, Northern port, Eastern port and Local port. The figure 8 shows the number of ports in the central router.

In Prior Rout routing, if the input port is the northern one and output port is the eastern one and eastern port is free, output port is the eastern port. If the eastern port is busy, the output port is would be the southern port and the southern port would be also, in the worst situation, and the western port would be the output port. So the task priority in this example would be:

\[ TP_{\text{NorthToEast}} = 3 \]
\[ TP_{\text{NorthToSouth}} = 2 \]
\[ TP_{\text{NorthToWest}} = 1 \]

As a result, there is need neither for saving nor buffering. In the same manner, for all packets which their destination is neighbor port, higher task priority belongs to this port. The next priority would be toward the frontal port and the lower priority belongs to the output port. In Prior Rout routing, if the input port, is eastern one and the output port is the eastern one and also be free, the output port would be the western one. If the western port is busy, the output port would be either the northern port or the southern port. That in this case, we choose the free port in clockwise. So we should have:

\[ TP_{\text{EastToWest}} = 3 \]
\[ TP_{\text{EastToNorth}} = 2 \]
\[ TP_{\text{EastToNorth}} = 1 \]

The table 1 shows the packet routing according to use of the OS.

<table>
<thead>
<tr>
<th>Task</th>
<th>Routing</th>
<th>Best Output Case</th>
<th>Mean Output Case</th>
<th>Worst Output Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>North to South</td>
<td>South</td>
<td>East</td>
<td>West</td>
</tr>
<tr>
<td></td>
<td>North to East</td>
<td>East</td>
<td>South</td>
<td>West</td>
</tr>
<tr>
<td></td>
<td>North to West</td>
<td>West</td>
<td>South</td>
<td>East</td>
</tr>
<tr>
<td>South</td>
<td>South to North</td>
<td>North</td>
<td>West</td>
<td>East</td>
</tr>
<tr>
<td></td>
<td>South to East</td>
<td>East</td>
<td>North</td>
<td>West</td>
</tr>
<tr>
<td></td>
<td>South to West</td>
<td>West</td>
<td>North</td>
<td>East</td>
</tr>
<tr>
<td>East</td>
<td>East to West</td>
<td>West</td>
<td>South</td>
<td>North</td>
</tr>
<tr>
<td></td>
<td>East to North</td>
<td>North</td>
<td>West</td>
<td>South</td>
</tr>
<tr>
<td></td>
<td>East to South</td>
<td>South</td>
<td>West</td>
<td>North</td>
</tr>
<tr>
<td>West</td>
<td>West to East</td>
<td>East</td>
<td>North</td>
<td>South</td>
</tr>
<tr>
<td></td>
<td>West to North</td>
<td>North</td>
<td>East</td>
<td>South</td>
</tr>
<tr>
<td></td>
<td>West to South</td>
<td>South</td>
<td>East</td>
<td>North</td>
</tr>
</tbody>
</table>
IX. EXPERIMENTAL RESULTS

A packet from north to east has been analyzed based on PrioRout routing and MicroC/OS-II features.

There is one task for each port; therefore, there are four tasks altogether. There are 12 paths as shown in table 1.

Creation of four tasks in MicroC/OS-II is shown in figure 9:

<table>
<thead>
<tr>
<th>Task</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>West</td>
</tr>
<tr>
<td>2</td>
<td>South</td>
</tr>
<tr>
<td>3</td>
<td>East</td>
</tr>
</tbody>
</table>

Consequently, the input packet follows this priority assignment once it reaches the task (North port). Four Boolean global variables are defined during task implementation and creation to show whether or not the ports are busy. The next higher priority (south port in this example) will be selected. Also the other tasks follow this routing. To sum up, based on using OS, when packet flits are going to pass the router, they do not need to be stored in buffers. Therefore the power consumption is lowered in comparison the case without using OS. In the worst case (figure10-b), if all the ports are busy, the packets can be stored in input buffers and task stacks. This means that we do not need virtual channel. In the case without using OS (figure10-a), higher priority packets may be waited for lower priority packets. But in attitude with using OS, sent packet based on their priorities send according to their importance. But we should notice that the new output packets can not interrupt until the all flits of previous packets are sent. Also, in attitude with using OS, we are able to message passing. As when two ports reach the different ports, once one packet based on critical section feature is been selected. This section can be priority assigning. For example forward path has higher priority than the neighbor path. So the table III shows the comparison two attitudes (with and without using OS). Horizontal axis shows the task priorities and vertical axis the packet transmission time. As a result, transmission time of higher priority packet is lower.

Figure 9. Creation of four tasks in MicroC/OS-II

```c
#define TASK_STK_SIZE 512 // Size of each task's stacks (# of WORDs)
#define TASK_START_ID 0 // Application tasks IDs
#define TASK_1_ID 1
#define TASK_2_ID 2
#define TASK_3_ID 3
#define TASK_4_ID 4
#define TASK_START_PRIO 4 // Application tasks priorities
#define TASK_1_PRIO 1
#define TASK_2_PRIO 1
#define TASK_3_PRIO 1
#define TASK_4_PRIO 1

// Create the first task
OSTaskCreate(TestTask1,(void*)11,&TestTaskStk1[TASK_STK_SIZE], 11);
// Create the Second task
OSTaskCreate(TestTask2,(void*)11,&TestTaskStk2[TASK_STK_SIZE], 11);
// Create the Third task
OSTaskCreate(TestTask3,(void*)11,&TestTaskStk3[TASK_STK_SIZE], 11);
// Create the Forth task
OSTaskCreate(TestTask4,(void*)11,&TestTaskStk4[TASK_STK_SIZE], 11);
```

Figure 10. a) Worst case in without using OS b) Normal Case in using OS state.
Table II Packet transmission status with northern source

<table>
<thead>
<tr>
<th>Source Port that sends 1000 packets</th>
<th>Other Port Status</th>
<th>Destination Port and the number of received packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>East is Free</td>
<td>East-1000</td>
</tr>
<tr>
<td>North</td>
<td>East is Busy and South is Free</td>
<td>South-1000</td>
</tr>
<tr>
<td>North</td>
<td>East and South are Busy and West is Free</td>
<td>West-1000</td>
</tr>
</tbody>
</table>

In our simulation, phytech evaluation board is used that the MicroC/OS-II has been ported in it. We reach to these result that have been shown in table III.

X. CONCLUSION

In this paper, the usage of a real time OS, in a NoC framework based on hotspot traffic model has been analyzed. Communication management in NoC, needs a precise planning, scheduling, resource allocation, message passing. Satisfy these parameters, needs efficiently. In this paper a RTOS has been used. Since the NoC is power constrained and the OS which is used has the a few line of code, this selection (a RTOS) has a significant effect on minimizing the power consumption. Based on the implementation, RTOS features can be used in NoC.

REFERENCES


Seyyed Amir Asghari was born in Lashteh Nesha in Guilan province of Iran, on June 26, 1984. He received his BS degree in Computer Engineering from Amirkabir University of Technology in 2007. He graduated from the Amirkabir University of Technology in MS. He is a research assistant of Asynchronous Design Laboratory in the same school.

Hossein Pedram Received his BS degree from Sharif University in 1977 and MS degree from ohio State University in 1980 in Electrical Engineering. He received his PhD degree from Washington State University in 1992 in Computer Engineering.

Dr Pedram has served as a faculty member in the Computer Engineering Department in Amirkabir University of Technology since 1992. He teaches courses in computer architecture and distributed systems. His research interests include innovative methods in computer architecture such as asynchronous circuits, management of computer networks, distributed systems, and robotics.

Hassan Taheri Received his BS degree from Amirkabir University of Technology in 1975 and MS degree from University of Manchester Institute of Science and Technology (UMIST) in 1978 in Electrical Engineering. He received his PhD degree from UMIST University in 1988 in Electrical Engineering.

Dr Taheri has served as a faculty member in the Electrical Engineering Department in Amirkabir University of Technology. He teaches courses in Data Communication Network, Computer Communication, Teletraffic Engineering, Electronic Switching, Digital Communications, Telephone Switching, Probability and Statistics.