ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Encoding and Decoding of T-OCI Codes on NoC Router

G. Sravanthy
Department of Electronics & Communication Engineering,
HITAM, Hyderabad, Telangana-502401, India.

K. Anil Kumar
Department of Electronics & Communication Engineering,
HITAM, Hyderabad, Telangana-502401, India.

Abstract

On-chip interconnects are the achievement aqueduct in latest system-on-chips. Code-division multiple access (CDMA) has been proposed to apparatus on-chip crossbars due to its anchored latency, decreased adjudication overhead, and large bandwidth. In CDMA, average administration is enabled in the code amplitude by allotment a bound amount of N-chip length erect overextension codes to the processing elements administration the interconnect. In this paper, we improved overloaded CDMA interconnect (OCI) to enhance the accommodation of CDMA network-on-chip (NoC) crossbars by accretion the amount of accessible overextension codes. Consecutive and alongside OCI architectonics variants are presented to attach to altered area, delay, and power requirements. Compared with the accepted CDMA crossbar, on a Xilinx Artix-7 AC701 FPGA kit, the consecutive OCI batten achieves 100% high bandwidth, 31% low power utilization, and 45% power saving, while the alongside OCI batten achieves N times higher bandwidth compared with the consecutive OCI batten at the amount of added length and power consumption. A 65-node OCI-based brilliant NoC is implemented, evaluated, and compared with an agnate amplitude analysis assorted admission based torus NoC for assorted constructed traffic patterns. The appraisal after-effects in agreement of the power appliance and throughput highlight the OCI as a able technology to design the physical layer of NoC routers.

INTRODUCTION

System on chip (SOC) is a complex interconnection of various functional elements. It creates communication bottleneck in the gigabit communication due to its bus based architecture [1]. Thus there was need of system that explicit modularity and parallelism, network on chip possess many such attractive properties and solve the problem of communication bottleneck. It basically works on the idea of interconnection of cores using on chip network.

The communication on network on chip is carried out by means of router, so for implementing better NOC, the router should be efficiently design [2]. This router supports four parallel connections at the same time. It uses store and forward type of flow control and Fsm Controller deterministic routing which improves the performance of router. The switching mechanism used here is packet switching which is generally used on network on chip [3].

In packet switching the data the data transfers in the form of packets between cooperating routers and independent routing decision is taken. The store and forward flow mechanism is best because it does not reserve channels and thus does not lead to idle physical channels. The arbiter is of rotating priority scheme so that every channel once get chance to transfer its data. In this router both input and output buffering is used so that congestion can be avoided at both sides.

FOUR PORT ROUTER ARCHITECTURE

Router Architecture:

The Four Router Design is done by using of the three blocks .the blocks are 8-Bit Register, Router controller and output block. the router controller is design by using FSM design and the output block consists of three fifo's

Cite this article as: G. Sravanthy & K. Anil Kumar, "Encoding and Decoding of T-OCI Codes on NoC Router", International Journal & Magazine of Engineering, Technology, Management and Research, Volume 6 Issue 1, 2018, Page 70-80.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

combined together the fifo's are store packet of data and when u want to data that time the data read from the FIFO's [4]. In this router design has three outputs that is 8-Bit size and one 8_bit data port it using to drive the data into router we are using the global clock and reset signals, and the err signal and suspended data signals are output's of the router .the FSM controller gives the err and suspended_data_in signals .this functions are discussed clearly in below FSM description.

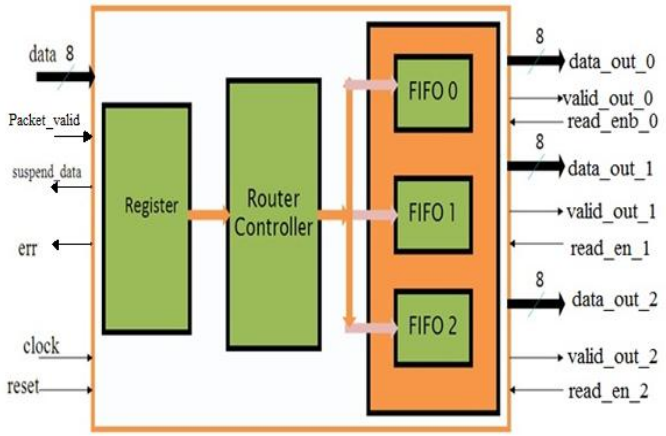

Figure 1: Four Port Router Architecture

The router_reg module contains the status, data and parity registers for the Network router_1x3.
These registers are latched to new status or input data through the control signals provided by the fsm_router [5].
There are 3 FIFO for each output port, which stores the data coming from input port based on the control signals provided by fsm_router module.
The fsm_router block provides the control signals to the fifo, and router_reg module. The Router blocks Diagram shown below fig…
Router blocks are

- Register
- Router controller(FSM)
- FIFO Output Block

Register Block:

This module contains status, data and parity registers required by router. All the registers in this module are latched on rising edge of the clock.

Data registers latches the data from data input based on state and status control signals, and this latched data is sent to the fifo for storage [6]. Apart from it, data is also latched into the parity registers for parity calculation and it is compared with the parity byte of the packet [7]. An error signal is generated if packet parity is not equal to the calculated parity.

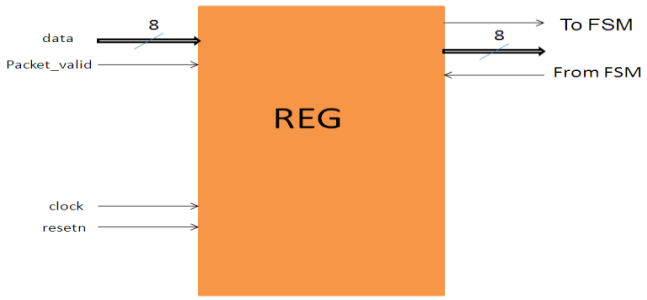

Figure 2- Register Block

If reset is low then output (dout, err, parity_done and low_packet_valid) are low.
The output parity_done is high
When the input ld_state is high and (fifo-full and packet_valid) is low or when the input laf_state and output low_packet_valid both are high and the previous value of parity_done is low. It is reseted to low value by reset_int_reg signal.

The output low_packet_valid is high.
When the input ld_state is high and packet_valid is low.It is reseted to low by reset_int_reg signal.

First data byte i.e., header is latched inside the internal register first_byte when detect_add and packet_valid signals are high, So that it can be latched to output dout when lfd_state signal goes high.
Then the input data i.e., payload is latched to output dout if ld_state signal is high and fifo_full is low.
Then the input data i.e., parity is latched to output dout if ld_state signal is high and fifo_full is low.
The input data is latched to internal register full_state_byte when ld_state and fifo_full are high; this full_state_byte data is latched inside the output dout when laf_state goes high.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Internal parity register stores the parity calculated for packet data, when packet is transmitted fully, the internal calculated parity is compared with parity byte of the packet. An error signal is generated if packet parity is not equal to the calculated parity.

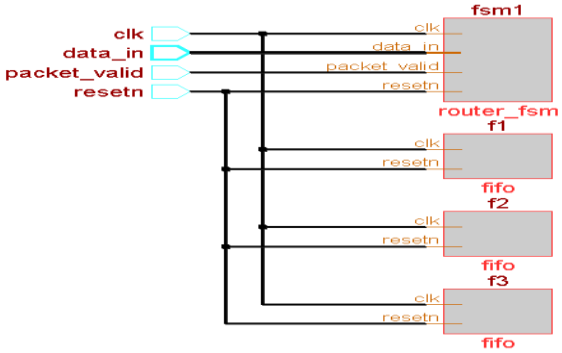

Figure 3-Register block synchronization

In the above figure register block is synchronize with the fsm to latch input data to it. Here, clk, resetn signals are synchronous with the entire module.

Eg: We are giving packet data as input to it and making read single (re1, re2, re3) as high w.r.t input first data byte of the packet. The receiving data is driven to the Router Controller for reaching its destination port. Which has 11 input pins (data_in [7:0],packet_valid , clk, reset).

Eg: data_in=8'b10101010, clk, reset, packet_valid are HIGH

Router Controller (FSM):

This module generates all the control signals when new packet is sent to router. These control signals are used by other modules to send data at output, writing data into the fifo.

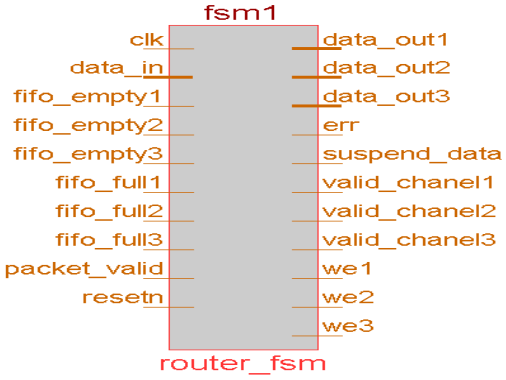

Figure 4- Router Controller Block

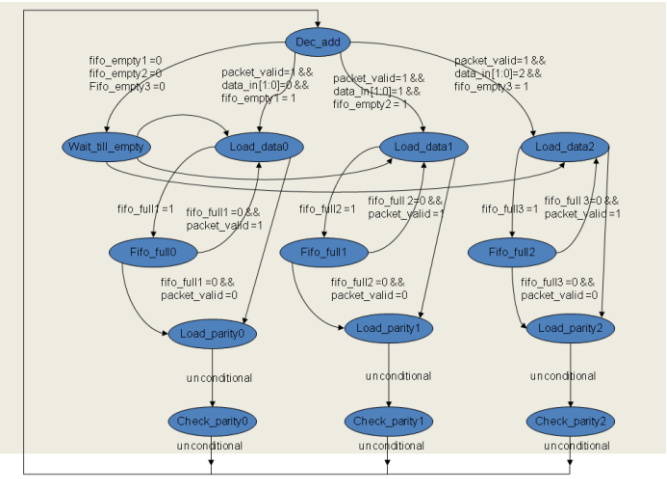

Figure 5 - Router Controller State diagram

STATE – CHECK_PARITY_ERROR1

In this state **reset_int_reg** signal is generated, which resets the status and parity registers inside the **router_reg** module. Neither any data is latched nor any input data is accepted. Router_reg compares the data parity from packet with calculated parity during this state.This state changes to default state DECODE_ADDRESS with next clock edge.

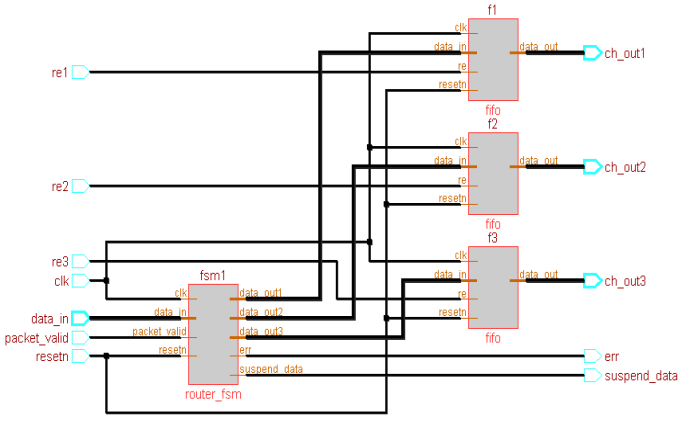

Figure 6 FSM synchronization block

Fsm block will synchronize register and fifo modules. The function of fsm is it taken data from data register and input data is latched to respective output based on header address which controls the function of design. So, it is called Router controller.

It has 17 inputs and 32outputs. For designing it we having

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

states(decode_address,wait_till_empty,load_data0,load_data1,load_data2,fifo_full0,fifo_full1,fifo_full2,load_parity0,load_parity1,load_parity2,check_parity0,check_parity1,check_parity2).By this it takes independent decision for the data to reach its destination port.

Eg: INPUT: data_in from Registerblock=8'b10101010, fifo_full0, fifo_full1,fifo_full2 are LOW and fifo_emp0, fifo_emp1, fifo_emp2 are HIGH.

OUPUT:data_out1,data_out2,wr_enb1,wr_enb2,suspend_data,err,valid_ch1,valid_ch2 are LOW and data_out3,we_enb3,valid_ch3 are HIGH.

Router Output Block:

There are 3 fifos used in the router design. Each fifo is of 8 bit width and 16 bit depth.

The fifo works on system clock. It has synchronous input signal reset.

If resetn is low then full =0, empty = 1 and data_out = 0

The FIFO has doing 3 deferent operations

- Write Operation
- Read operation
- Read and Write Operation.

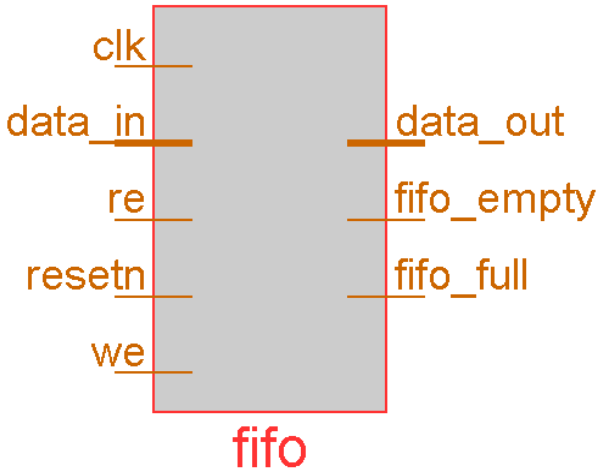

Figure 7 FIFO Block

The functionality of FIFO explain below:

Write operation:

The FIFO write operation is done by when the data from input data_in is sampled at rising edge of the clock when input write_enb is high and fifo is not full.in this condition onaly FIFO Write operation is done.

Read Operation:

The FIFO Read Operation is The data is read from output data_out at rising edge of the clock, when read_enb is high and fifo is not empty.

Read and Write operation can be done simultaneously.

Full – it indicates that all the locations inside fifo has been written.

Empty – it indicates that all the locations of fifo are empty.

The Output Block of Network Router conisistes of three FIFO.Each FIFO is a 8-Bit data Width and 16 bit data depth .the strcture of OUTPUT Block is shown in below fig..

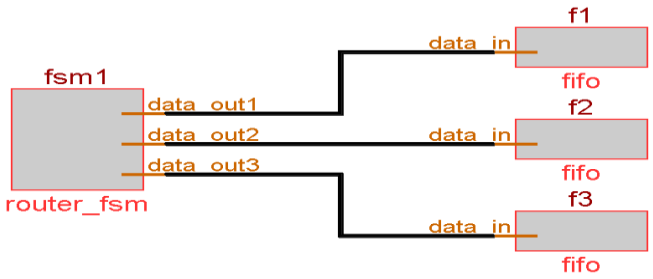

Figure-8 Synchronization of Output Block with FSM

This module provides synchronization between fsm and fifo modules. It provides faithful communication between single input port and three output ports.

It will detect the address of channel and will latch it till packet_valid is asserted, address and write_enb_sel will be used for latching the incoming data into the fifo of that particular channel.A fifo_full output signal is generated, when the present fifo is full, and fifo_empty output signal is generated by the present fifo when it is empty.The output vld_out signal is generated when empty of present fifo goes low, that means present fifo is ready to read(vld_out_0 = ~empty_0,vld_out_1 = ~empty, vld_out_2 = ~empty_2).

The write_enb_reg signal which comes from the fsm is used to generate write_enb signal for the present fifo which is selected by present address.

Eg: INPUT: clk,reset,read_enb are HIGH ,write_enb are LOW and data_in=8'b10101010.

OUTPUT:full is LOW,empty is HIGH and data_out=10101010.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

DESIGN ASPECTS AND APPROACH

Register:

It holds 8-bit values.Writing verilog code it should be declared as 8-bit width.

Router Controller:

For designing it we need finite state machine which controls all signals and we need states for controlling.The states should be for initializing data,wait for data until empty, loading data to respective port,if fifo is full w.r.t port it have to wait until empty,for parity byte loading and parity calculation.For initializing decode_state,for data waiting till empty wait_till_empty,for loading data load_data0, load_data1, load_data2,for checking fifo is full fifo_full0, fifo_full1, fifo_full2,for loading parity load_parity0, load_parity1, load_parity2 and for parity calculation check_parity0, check_parity01,check_parity2 totally we need 14 states having 4-bit width.But 4-bit it will give 16 states i.e full case.So, we are using only 14 states i.e parallel case. States represent as parameter keyword by this state value can't change throughout the design.For this design block we written in verilog code in behavioral model.

First we verifying whether it is reseting or not and when packet_valid signal is the data(10101010) is driving to w.r.t output port(fifo2) when it making low data is loading to load parity for parity calculation.And also verified whether it is switching to another port with another data(10101000) when reseting and also driven some bytes of data to the port.Testbench is written in verilog code.

FIFO:

It is 8-bit width and 16-bit depth.For fifo full or empty we are taking fifo_full and fifo_empty signals.For the status of full or empty of fifo we need a internal counter for counting it locations upto 16 locations it mean it is 4-bit wide.Input signals.

They are data_in(8-bit),were,clk,resetn and output signals are data_ou(8-bit),t,fifo_empty,fifo_full.Data is driven when write and not fifo full and it read when read and not fifo empty.RTL code it is written in verilog code in behavioral model.

It is verified by giving 16 bytes of data in data_in ,we is high then fifo_full becomes high.When it is high data can't be written into it.We get output in data_out and re is high it given all 16 bytes of data which we had driven after that fifo_empty is high then we can't read data and we also verified when both we and re signals are high it is written in verilog code.

Top module:

In this module we synchronized register, router controller, fifo blocks for that we calling all the modules with .name instance declaration and developing a design plan with these blocks as data_in is driven to register block, routing data to respective port decision taken by router controller and we are considering three fifo as output port. Same clock is given to all blocks i.e clock is synchronous. Code it is written in verilog code.

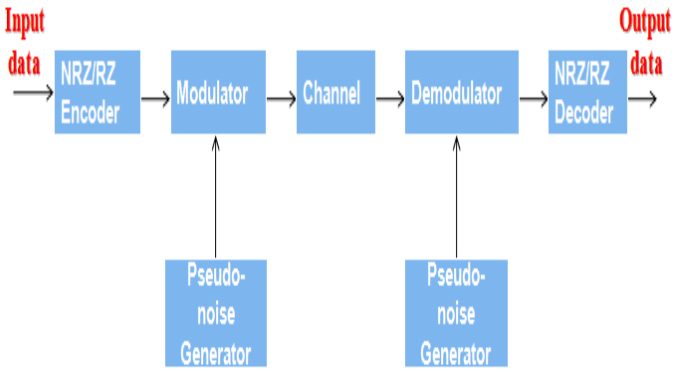

Fig 9: Block diagram of top module

CDMA Transmitter

The eight bit input data corresponding to a particular user is converted into serial form by an eight bit PISO. The PISO is clocked by Fmaster divided by 15 clock where Fmaster is 0.5GHz. Then it is spreaded by the 15 bit PN code. The PN code generator is clocked by Fmaster. Spreaded data of all the four users are summed up and generated the signal to be transmitted.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

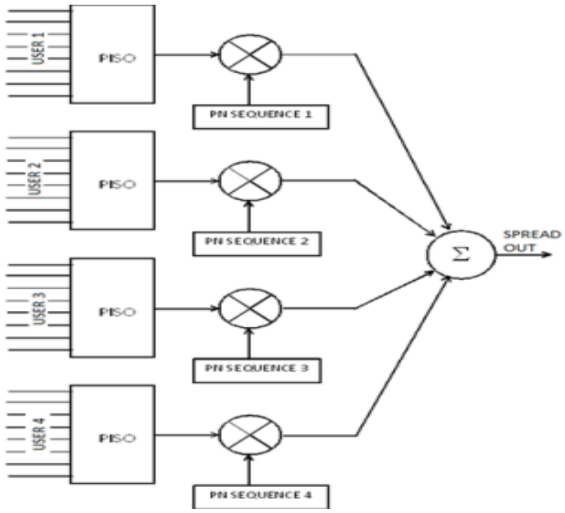

Fig 10: CDMA Transmitter

CDMA Receiver:

After de spreading the received signal with the corresponding code, it is compared with the same PN code, which is converted into parallel, using an 8 bit comparator. The comparator uses 0.33GHz clock frequency. If the actual transmitted data was a high then the de spread output will be same as that of the PN sequence. So the comparison function is performed in such a way that, it compares the de spread output with PN sequence. If it is same, then it can be concluded that the data send is a high and if it is not, then the data will be a low. So the comparator output corresponds to the actual transmitted data of a particular user. Thus it is able to reconstruct the original data from the spreaded output.

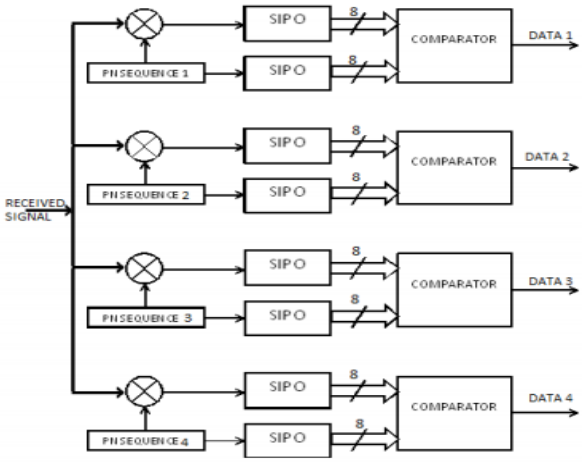

Fig 11: CDMA Receiver

PN sequence generator

Linear feedback shift registers are used for generating PN sequences. Components of D ip ops are used for this since structural modeling is used. To generate the sequence, rst it is necessary to initialize the ip ops to a particular value. Since 15 bit long PN sequence is being used, four ip ops are required and these four ip ops are required to be initialized. For that purpose, init signals are used. After the initialization, the xor feedback logic will provide a method to generate a PN sequence. Orthogonal sequences are required in this system. Time shifted versions of a PN sequence will be nearly orthogonal. So to shift the sequences, shift registers are used in which the sequence is given as input to the registers. The outputs from intermediate ip ops are taken which will be time shifted. So at the output of PN generator four PN sequences are obtained.

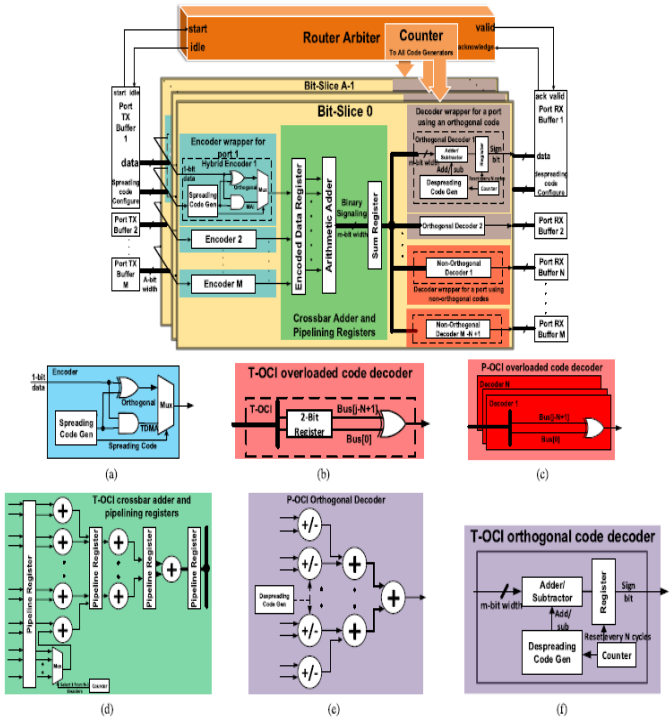

Fig. 12. High-level architecture and building blocks of the OCI crossbar. (a) TOCI/P-OCI hybrid encoder. (b) T-OCI nonorthogonal decoder. (c) P-OCI nonorthogonal decoder. (d) T-OCI pipelined crossbar tree adder, in which the adder is replicated *N* times for P-OCI crossbar. (e) P-OCI orthogonal decoder. (f) T-OCI orthogonal decoder.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The CDMA router has *M* transmit/receive ports. The main difference between the overloaded and classical CDMA routers is that $M > N - 1$ for the former due to channel overloading. Each PE is connected to two network interfaces (NIs), transmit and receive NI modules.

During packet transmission from a PE, the packet is divided into flits to be stored in the transmit NI first-input first output (FIFO). The router arbiter then selects *M* winning flits at most from the top of the NI FIFOs to be transmitted during the current transaction.

The selected flits must all have an exclusive destination address to prevent conflicts, and a winner from two conflicting flits is selected according to the router's priority scheme. The employed priority scheme is the fixed winner that takes all priority schemes; only one of the transmitters is given a spreading code and is acknowledged to start encoding. Once done, the router assigns CDMA codes to each transmit and receive NI. NIs with empty FIFOs or conflicting destinations are assigned all-zero CDMA codes such that they do not contribute MAI to the CDMA channel sum. Afterward, flits from each NI are spread by the CDMA codes in the encoder module.

The data are spread into *N* chips, where *N* is the CDMA code length that equals the number of clock cycles in a single crossbar transaction. Spread data chips from all encoders are summed by the CDMA crossbar adder and the sum is sent out serially to all decoders. The encoding/decoding process lasts for *N* clock cycles synchronized via a counter. At each decoder, the assigned code is cross correlated with the received sum to decode the data from the summed chips. The decoded flits are stored in the receive NI FIFOs until they are read by the PEs. In this paper, we focus on the high-level architecture and implementation details of the overloaded CDMA crossbar represented by the gray block in Fig. 1(a). A store and forward flow control and a deterministic routing algorithm are employed in the OCI router. The routing algorithm lies at the network layer, which is a higher layer than the physical layer containing the crossbar switch. According to the OSI model design principles, each layer of the model exists as an independent layer. Theoretically, one can substitute one protocol for another at any given layer without affecting the operation of layers above or below. Thus, using the same flow control protocol and routing algorithm enables comparing the OCI-based router with SDMA- and TDMA-based routers.

A. OCI Crossbar High-Level Architecture

The main objective of this paper is increasing the number of ports sharing the ordinary CDMA crossbar presented, while keeping the system complexity unchanged using simple encoding circuitry and relying on the accumulator decoder with minimal changes. To achieve this goal, some modifications to the classical CDMA crossbar are advanced. Fig. 2 depicts the high-level architecture of the OCI crossbar for a single-bit interconnection. The same architecture is replicated for a multibit CDMA router.

M TX-RX ports share the CDMA router, where spread data from the transmit ports are added using an arithmetic binary adder having *M* binary inputs and an *m*-bit output, where $m = \_\log 2\ M\_$. The adder is implemented in both the reference and pipelined architectures.

A controller block is used for code assignment and arbitration tasks. Each PE is interfaced to an encoder/decoder wrapper enabling data spreading/despreading. Unlike orthogonal spreading codes, which are XORed with the binary data bit, an AND gate is utilized to spread data using nonorthogonal spreading codes. The AND gate encoder works as follows: if the transmitted data bit is "0," it sends a stream of zeros during the whole spreading cycle, which does not cause MAI on the channel; if the transmitted data bit is "1," the encoder sends a nonorthogonal spreading code. Therefore, the additional MAI spreading code will either contribute an MAI value of one or zero each clock cycle because the encoder is an AND gate.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The XOR encoder of the ordinary CDMA crossbar cannot be used to encode the OCI codes because it only complements the spreading code chips, so an XOR gate will cause MAI to the crossbar whether the data bit is "0" or "1." A hybrid encoder is developed for both orthogonal and nonorthogonal spreading with an XOR gate, an AND gate, and a multiplexer unit, as shown in Fig. 2. Two decoder types are implemented for orthogonal and nonorthogonal data. More details about each component of the OCI crossbar will be presented.

B. OCI Code Design

The Walsh–Hadamard spreading code family has a featured property that enables CDMA interconnect overloading. The difference between any consecutive channel sums of data spread by the orthogonal spreading codes for an odd number of TX-RX pairs M is always even, regardless of the spread data. This property means that for the $N - 1$ TX-RX pairs using the Walsh orthogonal codes, one can encode additional $N - 1$ data bits in consecutive differences between the N chips composing the orthogonal code. Thus, exploiting this property enables adding 100% nonorthogonal spreading codes, which can double the capacity of the ordinary CDMA crossbar.

In this section, the code design methodology, mathematical foundations, and the decoding details of the OCI codes are providedS. An AND gate encoder is used to encode data with nonorthogonal spreading codes as shown in Fig. 2(a).
Therefore, for a nonorthogonal encoder, if data to transmit are one, a single spreading chip at a specific time slot in the spreading cycle is added to the channel sum, which causes the consecutive sum difference to deviate. The nonorthogonal codes imitate the TDMA signaling scheme as each code is composed of a single chip of "1" sent in a specific time slot.

The encoding/decoding scheme presented in this paper provide a novel approach that enables coexistence between CDMA and TDMA signals in the same shared medium. Therefore, the developed encoder is called TDMA overloaded on CDMA interconnect (T-OCI). Fig. 3 shows an encoding/decoding example of two T-OCI codes for a spreading code of length $N = 8$. An odd number of orthogonal codes must be used simultaneously to preserve the even difference property of Walsh codes.

where S is the N-cycle waveform of the channel sum, $dC(j)$ is the orthogonal CDMA data bit sent by the j th user, $dT(j)$ is the nonorthogonal TDMA data bit sent by the j th, $Co(j)$ is the orthogonal code assigned to the j th user, and $T(j-N+1)$ is the TDMA code assigned to the j th user. The TDMA code $T(i)$ is a single chip of "1" assigned at the i th time slot.

The TDMA term of the equation is the sum of products of TDMA chips and their corresponding data bits. This term can be viewed as another N-chip spreading code added to the orthogonal spread data represented by the first term of the equation. It should be indicated that the first chip of the TDMA MAI code is always set to zero ($T(1) = 0$), and the remaining $N - 1$ chips are assigned according to the encoded data bits; this note is the key to properly decode both orthogonal and nonorthogonal spread data.

C. OCI Crossbar Building Blocks

Two variants are realized for each OCI crossbar, reference and pipelined architectures. The pipelined architecture is implemented to increase the crossbar operating frequency, and consequently, bandwidth by adding nonfunctional pipelining registers to reduce the crossbar critical path. The OCI crossbar shown in Fig. 2 is basically composed of three main building blocks: 1) the encoder wrappers; 2) the decoder wrappers; and 3) the crossbar adder blocks, which are described in the following.

1) Crossbar Controller: At the beginning of each crossbar transaction, the controller assigns spreading codes to different encoders.

The assignment of orthogonal dispreading codes to receive ports is fixed, i.e., does not change between the

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

crossbar transactions. Therefore, for a router port to initiate the communication with the receive port .

Encoder must be assigned a spreading code that matches the destined decoder. If two different ports request to address the same decoder, the controller allows one access and suspends the other according to a predefined arbitration scheme. This code assignment scheme is called receiver-based protocol. In this paper, a static allocation scheme that allocates fixed spreading codes to all encoders is used. To interconnect a large number of PEs, a torus, star, or hybrid NoC topology can be realized where the assignment of spreading codes is local to each router. Consequently, each new packet arriving at a router is assigned a spreading code corresponding to its exit port decoder. The crossbar controller issues handshake signals to the transmit and receive ports with matching spreading codes to enable the transmitter encoders and receiver decoders.

2) Hybrid Encoder: The encoder is hybrid, it can encode both orthogonal and nonorthogonal data. A transmitted data bit is XORed/ANDed with the spreading code to produce the orthogonal/nonorthogonal spread data, respectively. A multiplexer chooses between the orthogonal and nonorthogonal inputs according to the code type assigned to the encoder as depicted by Fig. 2(a). The encoder is replicated N times for the P-OCI crossbar.

3) Crossbar Adder: For a spreading code set of length N, the number of crossbar TX-RX ports is equal to $M = 2(N - 1)$. In the T-OCI crossbar, sending a "1" chip to the adder is mutually exclusive between nonorthogonal transmit ports according to the T-OCI encoding scheme. This indicates that among the $2(N-1)$ inputs to the adder, there are guaranteed $(N - 2)$ zeros, while the maximum number of "1" chips is N. Therefore, a multiplexer is instantiated to select only a single input of the nonorthogonal TDMA encoded data bits and discard the remaining bits that are guaranteed to be "0." Thus, the adder has only N-bit inputs, $N-1$ from orthogonal encoders, and 1 from the multiplexer, as shown in Fig. 2(d). The sum produced by the adder circuit needs $(\log_2 N)$ wires. The number of needed stages of registers to pipeline the adder is $(\log_2 N)$, as depicted in Fig. 2(d). N replicas of the crossbar adder are instantiated for the parallel encoding adopted in the P-OCI crossbar.

4) Custom Decoder: There are four decoder types for different CDMA decoding techniques: the orthogonal T-OCI and P-OCI decoders and the overloaded T-OCI and P-OCI decoders.

The orthogonal T-OCI decoder is an accumulator implementation of the correlator receiver. $N - 1$ accumulator decoders are instantiated in all CDMA crossbar types for orthogonal data despreading. Instead of implementing two different accumulators (the zero and one accumulator), an up–down accumulator is implemented and the accumulated result is the difference between the two accumulators of the conventional CDMA decoder as shown in Fig. 2(f). The accumulator adds or subtracts the crossbar sum values according to the despreading code chip and resets every N cycles. The sign bit of the accumulated value directly indicates the decoded data bit, where the positive sign is decoded as "1," while the negative sign is decoded as "0." The P-OCI orthogonal decoder shown in Fig. 2(e) differs from the T-OCI orthogonal decoder in receiving the adder sum values concurrently not sequentially; therefore, the accumulator loop is unrolled into a parallel adder.

The T-OCI overloaded decoder depicted in Fig. 2(b) is composed of a 2-bit register to store the LSBs of two sum values, first of which is $S(0)$ and the second is $S(j - N + 1)$, where j is the number of the T-OCI decoders $(N \leq j \leq 2N - 2)$. The two bits are fed to the XOR gate, which decodes nonorthogonal spread data. The T-OCI decoder is replicated N times to implement the P-OCI decoder of Fig. 2(c). The 2-bit register is not needed anymore because the $S(0)$ and $S(j-N+1)$ values exist in the same cycle. The T-OCI and P-OCI crossbar architectures contain $(N - 1)$ orthogonal decoders and $(N - 1)$ overloaded decoders.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

SIMULATION AND SYNTHSIS RESULTS

OUTPUT WAVEFORM:

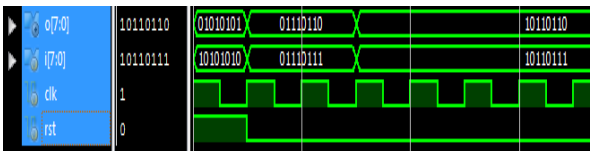

Fig 13: Simulation results of scheme1

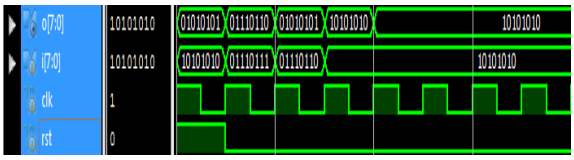

Fig 14: Simulation results of scheme2

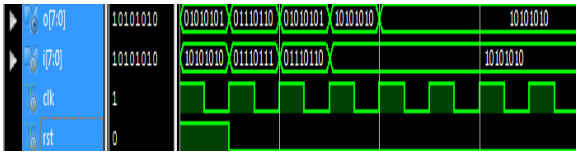

Fig 15: Simulation results of scheme3

CONCLUSION

The designed system can be used to implement a secure centralized data transfer in a local network. The transmitter receiver module to use can be RF, Bluetooth, zigbee or any other techniques which can be selected on the basis of range requirement. Also the system can be used to transfer s ensed data inside an industry, where the environment is comparatively noise free and insecure. Since the design and hardware can be improved easily with low cost the idea of this project is very much suitable for such situation. The security and capacity of the system can be improved by using the techniques such as dynamic CDMA and code hopping. Thus this system can be easily implemented, improved and used for secure networks.

In this paper, we introduced the concept of overloaded CDMA crossbars as the physical layer enabler of NoC routers. In overloaded CDMA, the communication channel is overloaded with nonorthogonal codes to increase the channel capacity. Two crossbar architectures that leverage the overloaded CDMA concept, namely, T-OCI and P-OCI, are advanced to increase the CDMA crossbar capacity, we exploited featured properties of the Walsh spreading code family employed in the classical CDMA crossbar to increase the number of router ports sharing the crossbar without altering the simple accumulator decoder architecture of the conventional CDMA crossbar. Generation procedures of non-orthogonal spreading codes are presented along with the reference and pipelined architectures for each crossbar variant. The T-/P-OCI crossbars were implemented.

Many future work directions are inspired by this paper including exploiting the mathematical properties of the code space to find additional nonorthogonal codes and boost the CDMA interconnect capacity and exploring more architectural optimizations of the OCI crossbar. Moreover, we plan to investigate using the OCI-based routers in different network topologies, evaluate their performance using standard benchmarks, and study their suitability for various applications.

REFERENCES

[1] K. Asanovic et al., “The landscape of parallel computing research: A view from berkeley,” Dept. EECS, Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2006-183, 2006.

[2] P. Bogdan, “Mathematical modeling and control of multifractal workloads for data-center-on-a-chip optimization,” in Proc. 9th Int. Symp. Netw.-Chip, New York, NY, USA, 2015, pp. 21:1–21:8.

[3] Z. Qian, P. Bogdan, G. Wei, C.-Y. Tsui, and R. Marculescu, “A trafficaware adaptive routing algorithm on a highly reconfigurable network-onchip architecture,” in Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign, Syst. Synth., New York, NY, USA, Oct. 2012, pp. 161–170.

[4] Y. Xue and P. Bogdan, “User cooperation network coding approach for NoC performance improvement,” in Proc. 9th Int. Symp. Netw.-Chip, New York, NY, USA, Sep. 2015, pp. 17:1–17:8.

[5] T. Majumder, X. Li, P. Bogdan, and P. Pande, “NoC-enabled multicore architectures for stochastic analysis of

biomolecular reactions," in Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE), San Jose, CA, USA, Mar. 2015, pp. 1102–1107.

[6] S. J. Hollis, C. Jackson, P. Bogdan, and R. Marculescu, "Exploiting emergence in on-chip interconnects," IEEE Trans. Comput., vol. 63, no. 3, pp. 570–582, Mar. 2014.

[7] S. Kumar et al., "A network on chip architecture and design methodology," in Proc. IEEE Comput. Soc. Annu. Symp. (VLSI), Apr. 2002, pp. 105–112.