Low-Overhead Error Detection for Networks-on-Chip

Amit Berman and Idit Keidar
Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel
{bermanam@tx, idish@ee}.technion.ac.il

Abstract— **In the current deep sub-micron age, interconnect reliability is a subject of major concern, and is crucial for a successful product. Coding is a widely-used method to achieve communication reliability, which can be very useful in a Network-on-Chip (NoC). A key challenge for NoC error detection is to provide a defined detection level, while minimizing the number of redundant parity bits, using small encoder and decoder circuits, and ensuring shortest path routing.**

We present Parity Routing (PaR), a novel method to reduce the number of redundant bits transmitted. PaR exploits NoC path diversity to reduce the number of redundant parity bits. Our analysis shows that, for example, on a 4x4 NoC with a demand of one parity bit, PaR reduces the redundant information transmitted by 75%, and the savings increase asymptotically to 100% with the size of the NoC. In addition, we show that PaR can yield power savings due to the reduced number of bit transmissions and simple decoding process. Furthermore, PaR utilizes low complexity, small-area circuits.

I. Introduction

Modern device scaling results in deep sub micron noises, which cause interconnect errors to be more dominant and harder to predict [1,2,3,4,5,6,10,11], and also gives rise to new error sources [2,3]. The need for efficient low-power design techniques, along with aggressive voltage scaling and higher integration make interconnects even more susceptible to errors [1,6,11]. In this paper, we focus on efficient solutions for interconnect reliability in the context of Networks-on-Chip (NoCs).

Traditional designs enhance interconnect reliability at the physical layer, using worst-case design margins such as aggressive inter-wire spacing, insertion of repeaters, and shielding of link wires [5,7,10]. Unfortunately, all these techniques incur high area and power costs [3,9]. Moreover, they require knowledge of the circuit layout, thus inflicting design complexity [3,6]. Furthermore, in novel technologies, the efficiency of these techniques decreases because transient errors are becoming harder to predict [10].

A promising alternative to the traditional physical layer solutions is to add reliability at the data-link layer of the NoC, using error detection codes, as suggested in [6,11]. Whereas error protection at the physical layer involves circuit design techniques that rely on specific device parameters, data link solutions are technology-independent [6].

Coding methods add redundant parity bits to the packet, which increase the NoC's cost by requiring either additional wires or extra transmissions. Our goal is to provide any desired level of error detection, while reducing the number of redundant bits, as we specify in Section 2.

In Section 3, we present Parity Routing (PaR), a novel method for error protection in NoC. The main idea behind our approach is to take advantage of the multiplicity of routing paths between nodes. Path diversity was exploited in the past in order to achieve load-balancing, by routing some traffic XY and remaining traffic YX [8]. Here, we use it for the first time for error detection, and achieve better load balancing as a favorable side effect of this approach. For example, if one bit error detection is required, then the traditional approach is to add a single parity bit to the packet. In PaR, we save the redundant bit by selecting the routing path according to the parity of the data. As in [6], errors are detected at every hop: routers along the path can identify parity errors by observing that the packet is on the wrong path. We illustrate this in Fig. 1, where a packet is transferred from a source node, S, to a destination node, D, on a regular mesh NoC. The data parity determines the routing path: 0 for XY routing and 1 for YX. In Fig. 1, the data is 0101, so the parity bit is 0, which indicates XY routing. While transferring the packet from S to the adjacent horizontal node (according to XY routing), one error occurs, changing the data to 0111. At the receiving node, the calculated parity bit is then 1 which indicates YX routing. Since the edge the packet arrives on is not on the expected path, an error is deduced.

A single parity bit can be saved whenever there are more than two available paths between the source and destination nodes. However, this may not always be the case if we wish to employ shortest-path routing: if the source and the destination nodes share one coordinate (either X or Y) there is only one shortest routing path. In such cases, PaR adds an extra parity bit to the packet.

In the general case, where the reliability demand is r redundant parity bits, we expand this method for error protection using the multiple routing paths between S and D. Some of the paths share edges, and therefore we save redundant bit transmissions on some of the edges within the routing paths, but not all. We have verified the correctness of PaR using exhaustive state exploration for all source and destination pairs on NoC grids of up to 5x5 hops, and reliability requirements of 1 to 10 parity bits.

In Section 4, we analyze and simulate the saving achieved by PaR. Our analysis shows that for a reliability demand of

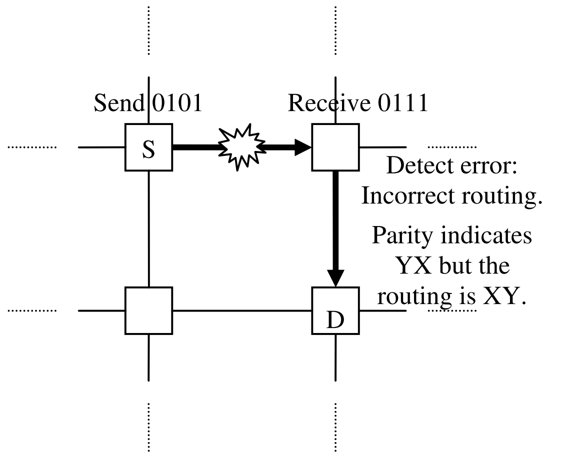
Send 0101
Receive 0111
S
D
Detect error: Incorrect routing.
Parity indicates YX but the routing is XY.

Fig. 1 – Example: bit flip detection.

one redundant parity bit, we save 50% of the redundant bit transmissions on a 2x2 mesh NoC, and 75% on a 4x4 mesh NoC. For a reliability demand of 2 parity bits, we save ~40% on 4x4 mesh NoC, and ~60% on an 8x8 mesh NoC. For any number of desired parity bits, the savings increase asymptotically to 100% with the size of the network. In addition, PaR can yield power saving as it saves bit transmissions and simplifies the error detection decoding process.

II. Goal and Definitions

We tackle the problem of hop by hop error detection. The required reliability level is expressed as the number r, of redundant bits. An externally provided function (or circuit), parity(data), returns r parity bits for protecting data. Any parity function can be used, e.g., CRC [4]. We denote the r redundant parity bits as $p[1], p[2], ..., p[r]$.

A. Problem Definition

Our goal is to design an error detection algorithm, which reduces the transmission of redundant bits, yet with low encoder and decoder circuit overheads and a low design complexity. Consider a packet sent from a source node S to a destination node D, in a regular mesh NoC. We require that the routing from S to D will be on one of the shortest paths. A Coding solution consists of two components, an encoder and a decoder. The encoder and decoder circuits are placed at each node, providing hop-by-hop error detection. We denote the current node where encoding/decoding occurs as H. The encoder and decoder's functions are defined as follows:

1. Encoder: Given H, S, D, and the packet's data, the encoder decides which edge is next on the packet's routing path, and whether there is a need to add redundant parity bits to the packet.
2. Decoder: Given H, S, D, the packet, and the incoming edge, the decoder determines whether an error had occurred.

The flow of information among the different components is shown in Fig. 2. We denote concatenation by commas, e.g., data,p[1] represents the data with one parity bit appeared at the end. If pack=data,p[1] then we denote data=pack − p[1].

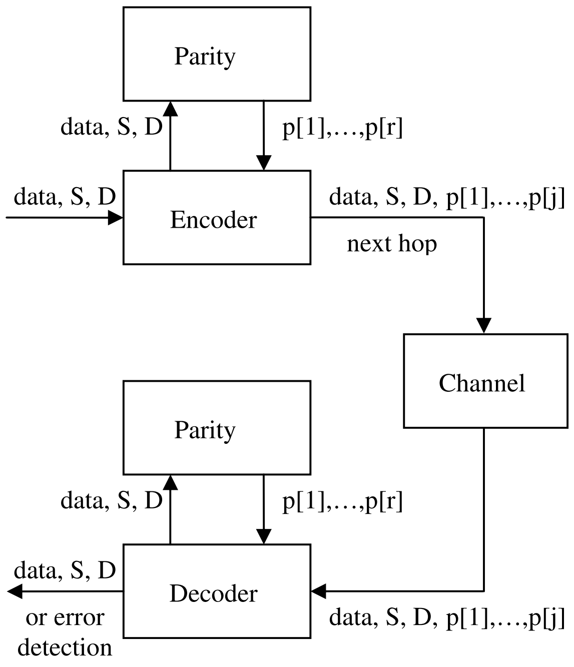

Fig. 2 – Information flow among encoder, decoder and parity circuits.

Note that since the encoder determines the routing path, this approach is applicable for transmission units that carry the source and destination addresses. In case the NoC employs wormhole routing [1], typically only the header flit carries these addresses. In such cases, our scheme can be used either for the entire packet (with checking at the destination node) or only for the header flit, which is the most important flit. For the remainder of this paper, we simply refer to the protected transmission unit as a packet.

B. Definitions

We now introduce some notations that will be used throughout the paper.

We denote $V = (V_x, V_y)$, where V_x and V_y are the coordinates of node V in the NoC mesh, counting nodes from left to right and from top to bottom. For example, the top-left corner node is (0,0), see Fig. 3.

We use $d_x(U,V)$ to denote the vertical distance between nodes U and V, i.e., $|V_x - U_x|$. Similarly, we use $d_y(U,V)$ to denote the horizontal distance between U and V. For example, in Fig. 3, $d_x(U,V) = 1$ and $d_y(U,V) = 2$.

For an edge e, the orientation orient(e) is h if e is horizontal, and v if it is vertical. We define the diagonal distance, $d_d(U,V) \triangleq \min\{d_x(U,V), d_y(U,V)\} + 1$.

For example, in Fig. 3, $d_d(U,V)=2$.

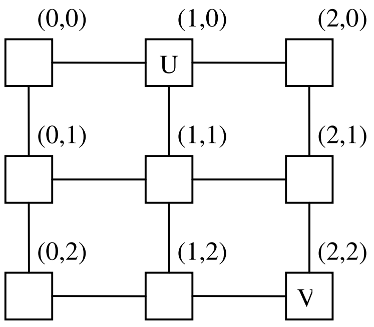

Fig. 3 – Node coordinates in a regular mesh.

III. Parity Routing Algorithm

In this section, we develop the PaR algorithm. For clarity of the exposition, we first present the special case of a reliability demand of one parity bit, called PaR-1, and then expand the algorithm for r redundant parity bits.

A. PaR-1: One-bit Error Protection

Consider the case of a reliability demand of one bit error detection. We use the given parity function to calculate the parity bit of the packet. If the parity bit is 0, PaR-1 routes the packet XY, and in case the parity bit is 1, the routing is YX, as shown in Fig. 4. If S and D are located on the same row or column, then the parity bit is sent along with the packet.

The pseudo-code of PaR-1 encoder is shown in Fig. 5.

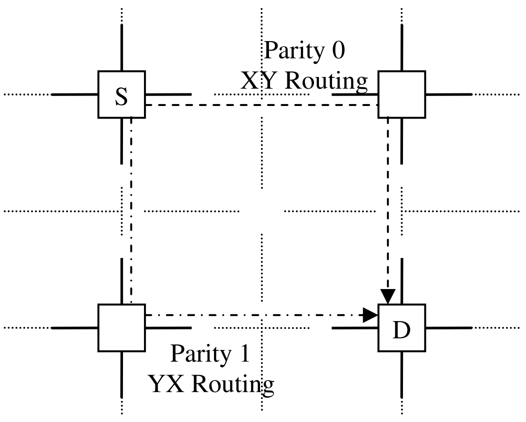

Fig. 4 – PaR-1 concept.

```
(1) PaR-1_Encode (H, S, D, data)
(2)  if $(H = D)$ then return data, no next hop
(3)  next_hop ← next hop on XY route to D
(4)  packet ← data
(5)  p[1] ← parity(data, S, D)
(6)  if $S_x = D_x$ or $S_y = D_y$ then packet ← data,p[1]
(7)  else if p[1] = 1 then next_hop←next hop on YX to D
(8)  return packet, next_hop
```

Fig. 5 – PaR-1 encoder pseudo-code.

The pseudo-code of PaR-1 decoder is shown in Fig. 6.

```
(1) PaR-1_Decode (S, D, packet, incoming_edge)
(2)  error_detected ← false
(3)  p[1] ← extract p[1] from packet
(4)  data ← extract data from packet
(5)  if $p[1] \neq \perp$ then
         /* parity bit was received with the packet */
(6)      newp[1] ← parity(data, S, D)
(7)      if newp[1] $\neq$ p[1] or ($S_x \neq D_x$ and $S_y \neq D_y$)
           then error_detected ← true
(8)  else
(9)      expected_routing ← parity(data, S, D)=0 ?
                        XY : YX
(10)     path ← edges_on_path(S, D, expected_routing)
(11)     if (incoming_edge $\notin$ path)
          or ($S_x = D_x$ or $S_y = D_y$)
           then error_detected ← true
```

Fig. 6 – PaR-1 decoder pseudo-code.

The property that allows us to detect the parity bits according to the routing path is the fact that the XY and YX paths between every S and D that do not share a coordinate are edge-disjoint.

B. PaR-r: r-bit Error Protection

Generally speaking, in order to provide a detection level of r-parity bits without sending redundant bits, we need to distinguish between 2^r edge-disjoint routing paths. Since there are at most 2 edge-disjoint paths between every pair of nodes, PaR-r strives to achieve disjointedness on as many edges as possible, by choosing paths with minimal overlap. For example, in Fig. 7, we see an example of the 2^r routing paths PaR-r uses between S and D for the different values of the r parity bits.

First, note that there aren't always 2^r different shortest paths: if S and D are close to each other, there are fewer paths. In case S and D are sufficiently far from each other, the 2^r routing paths constructed according to the value of the parity bits as a binary number, ParVal, as follows (see Fig. 7): The paths for 0 and 2^r-1 are XY and YX routing from S to D respectively. All other paths go through U1, the next node on the diagonal from S towards D. From there, if ParVal equals to the distance on the y-axis from U1 to the source, i.e. $d_y(U1,S)=ParVal$, then the routing is XY to the diagonal node of D at distance ParVal from D towards S, V1. If ParVal equals the distance on the y-axis from U1 to the destination, i.e. $d_y(U1,D)$, then the routing is YX to V1. Likewise, ParVal values of 2 and 2^r-2 are routed through U2 and the node V2, and so on.

Fig. 8 shows the PaR-r encoder's pseudo code.

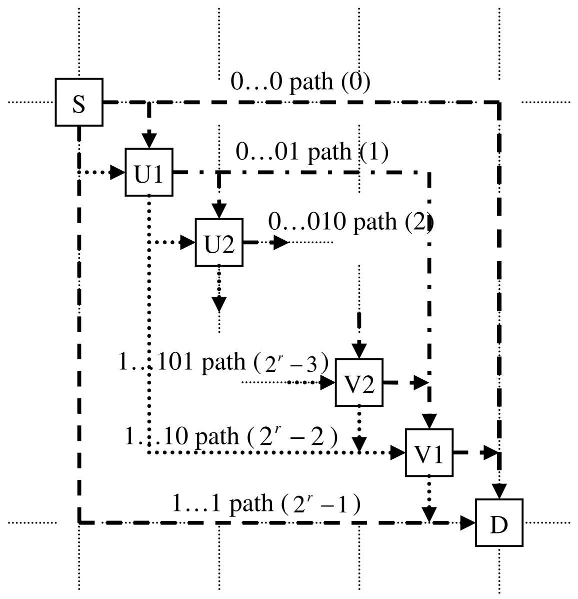

Fig. 7 – The 2^r routing paths for r parity bits.

```
(1) PaR-r_Encode(H,S,D,data)
(2)  p[1]…p[r] ← parity(data,S,D)
(3)  ParVal ← bin2dec(p[1],...,p[r])
(4)  packet ← S,D,data
(5)  if $(H = D)$ then return data, no next hop
(6)  if $(d_y(H,S) = ParVal)$ or
       $\left(d_x(H,D) = ParVal \text{ and } d_x(H,D) < d_x(H,S)\right)$
     then next_hop ← next hop on XY route to diagonal
                     node with the same distance from D
(7)  else if $\left(d_x(H,S) = 2^r - 1 - ParVal\right)$ or
       $\left(d_y(H,D) = 2^r - 1 - ParVal \text{ and}\right.$
       $\left.d_y(H,D) < d_y(H,S)\right)$
     then next_hop ← next hop on YX route to diagonal
                     node with the same distance from D
(8)  else
(9)    packet ← S,D,data,p[2],p[3]…p[r]
(10)   next_hop ← next hop according to PaR-1_Encode
              placing D as the diagonal node from H to D
(11)   return packet, next_hop
```

Fig. 8 – PaR-r encoder pseudo code.

Fig. 9 shows the PaR-r decoder's pseudo code.

If one parity bit is missing (i.e., r-1 are sent with the packet), then H should be either on the same diagonal with S, or one hop away from such a diagonal, or on the same row or column with S or D. In the first case, the missing parity bit is 0 in case a message arrives on a vertical edge, and otherwise it is 1. In the second and third cases, it is 0 for a horizontal edge, and 1 otherwise (see paths in Fig. 8). If additional bits are missing, then they are deduced from the

```
(1) PaR-r_Decode(H,S,D,packet,incoming_edge)
      /* First get the parity bits arrived with the data */
(2)  p[1]…p[r] ← extract p[1]…p[r] from packet
(3)  data ← extract data from packet
      /* if one parity bit is missing */
(4)  if $p[1] = \perp$ then
(5)    p[1] ← decode p[1] according to PaR-1_decode
             placing S and D as the nearest main diagonal
             nodes between S and D
(6)  else if $p[1]\ldots p[r] = \perp$ then
(7)    if orient(incoming_edge)=h then
           p[1]…p[r] ← dec2bin($d_y(H,S)$)
(8)    else p[1]…p[r] ← dec2bin($d_x(H,D)$)
(9)  if parity(data, S, D) $\neq$ p[1]…p[r] then return error
```

Fig. 9 – PaR-r decoder pseudo code.

distance on the Y axis to S for a horizontal edge, or from the the distance on the X axis to D for a vertical edge. When more then one parity bit is missing in the packet then the missing parity bits are deduced according to the binary representation of the distance from H or to S or the distance from H to D, according to the orientation of the incoming edge. When all parity bits had been decoded, we compare them to the parity bits which are calculated from the received data using the given parity(data) function. In case of mismatch between the parity bits, we detect an error.

IV. Analysis

PaR achieves savings in two elements: first, it saves network traffic and interconnects dynamic power due to the reduced redundant bit transmission, and second, it saves dynamic power by avoiding the need to operate the original error protection decoder block (which is likely to grow with exponential complexity while the growth of the parity bits number is linear). We now analyze the savings in redundant bits transmission. We begin, in Section 4.1, by analyzing PaR-1, and then generalize the analysis to PaR-r in Section 4.2. Finally, we present an example of the power reduction archived by PaR-1 in Section 4.3.

For simplicity, our analysis assumes a uniform traffic model, where an equal number of messages are transmitted between all source-destination pairs. We measure the percentage of redundant bit transmissions on an edge-by-edge basis. For example, if a parity bit is sent on two edges in a four-hop path, the savings on this path are 50%. We analyze the average savings over all paths.

A. PaR-1 Analysis

Consider an NxM regular NoC mesh with NM nodes. The number of potential source-destination (S-D) pairs in the NoC is NM(NM-1). Each of the NM nodes has N-1 potential destinations that share the Y coordinate with it. Thus, there are (N-1)NM S-D pairs that share this coordinate. Similarly, there are (M-1)NM S-D pairs that share the X coordinate. The number of S-D pairs between which the transmission of the parity bit is saved is:

$$NM(NM-1)-(N-1)NM-(M-1)NM = (NM-N-M+1)NM$$

We next compute the percentage of savings in terms of edges. The average path length when S and D share the Y coordinate is $\frac{N}{2}$. In a similar way, the average path length when S and D share x coordinate is $\frac{M}{2}$. When S and D are not on same axis, the average path length is $\frac{N}{2}+\frac{M}{2}$. Denote by $CR_1(N,M)$ the percentage of edges on the paths between all the S-D pairs for which the redundant bit is not sent by PaR-1, on an NxM NoC mesh. We get:

$$CR_1(N,M)=1-\frac{(N-1)N+(M-1)M}{(N-1)N+(M-1)M+[NM-N-M+1](N+M)}$$

In case the network is symmetric, i.e., N=M, we get:

$$CR_1(N,N)=1-\frac{2N(N-1)}{2N(N-1)+2N(N-1)^2}=\frac{N-1}{N}$$

For example, in case of a 4x4 network, the cost reduction is 75% of the redundancy bits. We observe that as we increase the network (in both dimensions equally) the CR_1 grows to 100%:

$$\lim_{N\to\infty} CR_1(N,N)=1$$

Similarly, for rectangles with any constant ratio, α, between the width and length, where $M=\alpha N$ this observation is also valid.

In order to show this, we simplify the analysis and prove that the percentage of paths on which no parity bits are sent is asymptotically zero. Since these paths are, on average, shorter than paths where parity bits are sent (as shown above), this simpler result implies that the savings increase asymptotically to 100%. We observe that the percentage of pairs for which we save the redundant bit transmissions is:

$$\lim_{N\to\infty} CR_1(N,\alpha N)=1$$

B. *PaR-r Analysis*

We now analyze the general case of r parity bits, PaR-r. Consider an NxM NoC, a reliability demand of r redundant bits, and two given nodes S and D. Without using the PaR algorithm, we have to transmit r redundant parity bits on all edges in the path, that is, on $d_x(S,D)+d_y(S,D)$ edges. Assume that PaR-r can transmit the packet without the redundant r parity bits, i.e., $\lfloor \log_2(d_d(S,D)) \rfloor \geq r$. According to the PaR-r algorithm, the amount of redundant parity bits depends on the value of the parity bits (ParVal). There are 2 routing paths (0 and 2^r-1) on which no redundant bits are sent on any edge. There are 2 routing paths (1 and 2^r-2) on which there are r – 1 redundant bits sent on 4 edges. For ParVals of 2 and 2^r-3, there are (r – 1) redundant parity bits transmitted on 8 edges (first 4 from S and last 4 to D) and so on, until (r – 1) redundant parity bits are transmitted for a ParVal of (2^r-1). Assuming that ParVal is distributed uniformly, the average redundant parity bits transmitted from S to D is therefore:

$$(r-1)\frac{2\left[0+4+\ldots+4\left(2^r-1\right)\right]}{2^r\left(d_x(S,D)+d_y(S,D)\right)}=\frac{2\cdot(r-1)\left(2^{r-1}-1\right)}{d_x(S,D)+d_y(S,D)}$$

It is easy to see, that as the NoC grows, the percentage of S-D pairs for which PaR-r can choose 2^r paths asymptotically grows to 100%. For such pairs, the average value of the denominator in the equation above (averaging over all relevant S-D pairs) grows to infinity with the NoC size, while the nominator remains constant. Hence, the percentage of parity bits actually transmitted goes asymptotically to zero. In other word, for any constant r, the savings of PaR-r grow asymptotically to 100% with the size of the NoC.

To compute the average savings percentage in a given NoC, we ran a numeric computation, which iterates over all S-D pairs, and sums the savings, and then divides them by the number of pairs. The results for different r requirements are shown in Fig. 10.

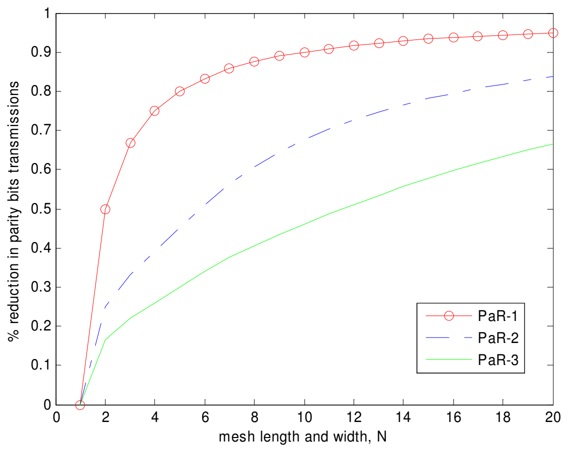

Fig. 10 – PaR cost reduction. If only one redundant bit is required, its transmission is saved 80-90% of the time, even in small NoCs. For 2 parity bits, savings are over 50%, (more than one bit), and for 3 bits, more than 30% on small NoCs.

C. Power Reduction Example

We demonstrate the power saving achieved by PaR-1 with NxN regular mesh NoC with 5mm long, 8-bit width links. Hardware design is implemented on 0.18μ TOWER process, and synthesized by Synopsis's design compiler. Interconnect power consumption is measured by SPICE model, assume random data and traffic patterns.

Measurements of the redundant bits switching power, along with the parity circuits' power and PaR circuits' power are referred as power consumption and shown at Fig. 11. The measurements were made on 2x2, 3x3 and 4x4 regular mesh NoCs. We can observe increased power saving with the size of the NoC. We expect the savings to grow as more parity bits are used because of less redundant network traffic as well as avoiding the need to use more complex error protection decoders.

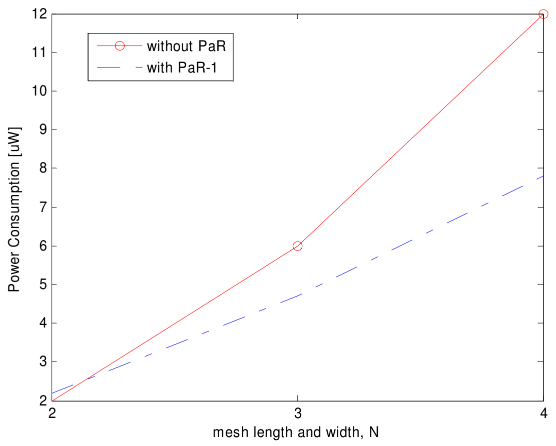

Fig. 11 – Example: PaR-1 power reduction. For 2x2 NoC, the encoder and decoder blocks overhead do not compensate for the power reduction achieved by the reduced redundant bits traffic. For 3x3 NoC, savings are over 25%, and for 3x3 NoC, more than 35%.

V. CONCLUSIONS

Achieving interconnect reliability is already a difficult task facing chip designers and manufacturers today, and can be anticipated to become an even more serious problem in years to come. A key challenge in this context is providing high reliability at a low power cost. While error detection codes provide a promising approach towards achieving reliability, they do expend additional power in redundant bit transmissions. In this paper, we have tackled the problem of ensuring error detection, while reducing the need for redundant transmissions.

We presented PaR – parity routing, a low-overhead error detection solution for networks on chip. PaR can be used to provide any predefined error protection requirement. It exploits NoC path diversity, and selects routing paths based on parity bits. It thus saves actual transmissions of these bits, along with the associated power penalty. PaR uses simple, low-complexity encoding and decoding circuits. We have analyzed the savings achieved by PaR, and have shown that it yields significant savings even on small NoCs, (for example, saving 75% of redundant bit transmissions on a 4x4 NoC mesh), and its savings asymptotically converge to 100% with the size of the NoC. We showed that PaR can yield power savings (for example, saving 35% of redundant power consumption on a 3x3 NoC mesh NoC).

We believe that our novel parity routing approach opens interesting opportunities that may be explored in future work. One such interesting future direction is related to wire (capacity) allocation. By eliminating the need to transmit redundant parity bits most of the time, PaR may allow for wire reductions in NoC design. For example, if a parity bit is sent with every packet, the NoC designer is likely to add a wire for parity bits to all the links in the NoC. On the other hand, if less than 20% of transmissions carry redundant bits (as occurs, e.g., with PaR-1 on a 6x6 NoC), then it might be more cost-effective not to add a parity wire, and transmit the parity bit after the data when needed. A study of the optimal wire allocation for NoCs that use PaR is an interesting topic for future research. Beyond this example, another interesting question for future research is how to extend the PaR approach to also allow for error correction. Though in current day VLSI technology the bit error rates render error detection and retransmission more power-efficient than error correction [2], this situation may change in future technologies, where one may therefore wish to employ error correction.

REFERENCES

[1] A. Dutta and N. A. Touba, "Reliable Network-on-Chip Using a Low Cost Unequal Error Protection Code", 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2007, pp. 3-11.

[2] G. Micheli, L. Benini, "Networks on Chips – Technology and Tools", Morgan Kaufmann Publishers 2006, pp. 75-139.

[3] A. Mogenshtein, E. Bolotim, I. Cidon, A. Kolodny, R. Ginosar, "Micro Modem – Reliability Solution For NoC Communications", ICECS 2004, pp. 483-486.

[4] S. Murali, T. Theocharides, N. Vijaykrishnan, M.J Irwin, L. Benini, G. Micheli, "Analysis of Error Recovery Schemes for Networks on Chips", IEEE Design&Test of Computers 2005, pp. 434-442.

[5] M. Mutyam, "Selective shielding: A Crosstalk-Free Bus Encoding Technique", IEEE ICCAD 2007, pp.618-621.

[6] J. Nurmi, H. Tenhunen, J. Isoaho, A. Jantsch, "Interconnect-Centric Design for Advanced SOC and NOC", Kluwer Academic Publishers 2004, pp. 155-170.

[7] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, C.R. Das, "Exploring Fault-Tolerant NoC Architectures", IEEE DSN 2006, pp. 93-104.

[8] B. Towles and W.J. Dally, "Worst-case Traffic for Oblivious Routing Functions", Computer Architecture Letters, February 2002.

[9] A. Vitkovski, R. Haukilahti, A. Jantsch, E. Nilsson, "Low Power and Error Coding for Network-on-Chip Traffic", Norchip Conference Proc., 2004, pp. 20-23.

[10] P. Vellanki, N. Banerjee, K.S. Chatha, "Quality-of-Service and Error Control Techniques for Mesh-Based Network-on-Chip Architectures", Integration 2005, pp. 353-382.

[11] H. Zimmer, A. Jantsch, "Fault Model Notation and Error Control Scheme for Switch to Switch buses on NoC", CODES ISSS 2003, pp. 188-193.