Fast Transaction-Level Dynamic Power Consumption Modelling in Priority Preemptive Wormhole Switching Networks On Chip

James Harbin, Leandro Soares Indrusiak

Real-Time Systems Group, Department of Computer Science, University of York, UK
{james.harbin,leandro.indrusiak}@york.ac.uk

***Abstract*—This paper specifies an architecture for power consumption modelling integrated within cycle-approximate transaction level modelling for network-on-chip (NoC) simulation. NoC simulations during design validation have traditionally been limited to very short durations, due to the necessity to perform cycle-accurate simulation to represent fully the low level system simulated. Due to the high proportion of overall system power that may be consumed by a busy NoC, high-fidelity NoC power modelling is especially important to accurately assess the effectiveness of link coding and other strategies to reduce NoC power consumption.**

The paper describes the extension of a cycle-approximate TLM methodology to encompass power modelling in NoCs, considering its operation with real application traffic. The proposed scheme avoids modelling of flit-by-flit progress during non-preemptive periods of packet transmission. The simulation performance and accuracy are contrasted with theoretical models and a flit-by-flit scheme (in which each flow control digit passing along a bus wire is simulated). The power consumption reduction delivered by encoding schemes such as bus-invert coding are considered and compared with analytical models to verify the correct performance of the simulation models.

I. Introduction

Network-on-Chip (NoC) concepts have been widely proposed as a solution to the problems of increasing interconnection complexity in system on chip and modern multicore CPU design [1][2]. As such systems scale, the manual routing of custom signalling wires and buses becomes a design area and capacity bottleneck, and performance favours the engineering of a standardised backbone network capable of transporting generic data between processing units. Such data could include memory hierarchy synchronisation transfers, decoded multimedia data, and inter-processor communication.

Power consumption is becoming a major consideration in NoC design, with some early results suggesting that NoCs could consume 30-40% of overall system power [3]. This emphasises the importance of finding strategies to reduce power consumption by NoCs, ideally via physical layer independent techniques that can be applied early in the design flow. Drawing design conclusions from simulation requires simulation flexibility, with execution times sufficiently low as to scale to practical network conditions with realistic traffic. Although such simulations cannot entirely replace the low-level cycle-accurate validations that must be performed during the final design phases of specific NoCs, they provide valuable performance conclusions to allow designers to choose a suitable NoC.

Financial support for this work was provided by the EPSRC, under project 'LowPowNoC', contract EP/J003662/1.

This paper considers the problem of power modelling simulation in NoCs and introduces an algorithm for reducing execution time of NoC power consumption simulations. This algorithm reduces the number of events simulated in the simulator kernel by batching power consumption events, modelling them in cycle-approximate transactions only at particular points of contention in the simulation. By reducing the execution time of NoC power consumption simulations, power reduction techniques such as low-power link coding can be investigated much more quickly to determine their effectiveness under realistic application scenarios.

The paper is structured as follows. Section II considers related work on NoC simulation and presents the power models used. Section III defines the specific problem this paper covers and its fundamental assumptions. Section IV defines in detail our algorithm for TLM power modelling, presenting a pseudocode and visual description and illustrating how it reduces simulation workload. Section V describes the implementation of the algorithm within our simulation framework, and defines parameters used in the implementation. Section VI presents our simulation results, which demonstrate the performance and accuracy of our algorithm, and validate the successful implementation of low-power link codecs. Finally, Section VII considers further work on the simulator and associated power reduction technologies, while Section VIII concludes the paper.

II. Related Work

A. NoC Simulation Methodologies

Transaction level models (TLM) of VLSI systems are characterised [4] as those in which details of communication are provided by channel abstractions, separated from the modelling of the internals of computation. In the general taxonomy of the Gajski-Kuhn Y-chart [5], TLM is at the algorithm level.

Therefore, TLM's abstraction level is considerably higher than the boolean equation models required for systems defined at the gate level, which require time-consuming simulations with circuit analysis tools. TLM also represents a further level of abstraction (and therefore performance improvement) over register transfer level (RTL) models, which distinguish registers (as storage units) and synchronisation from computation.

However, TLM (frequently implemented in the SystemC language [6]) is a very broad concept and systems differ in how closely their transaction interfaces represent reality. A key issue is the timing of the system and the closeness of transaction timings to the real hardware timings. Cycle-accurate TLM seeks to have some level of transaction timing corresponding to the processor or bus cycles experienced in the real system. For example, pin-accurate bus cycle accurate (PA-BCA) models simulate pins and wires cycle-accurate with the hardware system, but modelling inside components is now only cycle-approximate [7]. Transaction-based bus cycle accurate (T-BCA) models discard the abstraction of individual bus signals, instead modelling as cycle-accurate transactions the propagation of signals (e.g. flits) across generic channels which represent a composite bus.

Cycle-approximate Transaction Level Models (TLM) have been developed for on-chip processing elements [8], which speed up computation by abstracting away simulation of individual instructions and using statistics of processing models rather than the models themselves. Similarly, cycle-approximate TLM for NoC simulations such as TLM 2.0 [9] coarsen the abstraction further by allowing simulation of the properties of groups of flow control digits (flits) in a cycle-approximate way during a single transaction. There exists a clear trade-off between simulation granularity and performance, which permits cycle-approximate TLM to be up to four orders of magnitude faster than a flit-accurate model [10]. The approach presented in this paper obtains similar gains, demonstrating that discarding cycle-accuracy and issuing transactions only upon packet injection/release or preemption boundaries (rather than on each flit transmission) permits a performance improvement of almost three orders of magnitude.

B. Power Modelling

Power consumed by elements of a CMOS logic circuit (such as a network-on-chip communication bus and its associated encoders, arbiters, buffers and decoders) can be approximated [11] by Equation 1:

$$P_{total} = AfCV^2 + \tau AfVI_{short} + VI_{leak} \quad (1)$$

The terms in Equation 1 can be defined as follows:

A	Activity proportion of a component
f	Clock-frequency of the circuit or component
C	Capacitance of a circuit component
V	Voltage in the circuit
τ	Gate transition time
I_{short}	Gate short-circuit current
I_{leak}	Leakage current

Total power consumption can be grouped into dynamic power consumption (represented in the first term above), short circuit current (second term), and leakage current (third term). The dynamic power consumption is obtained by considering circuit elements such as the bus wires as capacitors, and represents energy consumed during the charging and discharging of the capacitor. This occurs proportional to the frequency of the network and its activity, hence the emphasis on power consumption reduction as CPUs and SoC operate at higher clock speeds. Although voltage reduction has been useful to reduce power consumption, it has limits due to signal degradation in transit that makes signals vulnerable to error in transmission.

The second term is the short-circuit current, which represents current flowing to ground when a logic gate in a system changes state. Again, this is proportional to the frequency and component activity, and the transition time. The final term is the leakage current, which represents power loss through electron leakage due to quantum tunnelling in the semiconductor. As a result of its independence of dynamic activity, it is considered to not increase with frequency. Overall, however, for a NoC the dynamic transition activity is considered most significant, due to the necessarily long length of the wires which represent a significant capacitance. Although, as components become smaller and governed more by quantum effects, leakage currents become more significant.

In order to compute the power consumption of a bus, a first order model considers the electrical structure of the bus to be composed of independent parallel capacitances. Assuming no connection between the wires, the total capacitance of the bus can be considered as the aggregate of all the individual capacitances of separate bus wires. Since a bus is involved in transmitting coherent signals (grouped in logical flits - flow control units), the power associated with this model can be determined from the number of bit transitions between the bit signals. In coding theory, the number of transitions is equal to the Hamming distance of the bit signals. Therefore, the dynamic power consumption associated with the transmission of flit F_1 after F_0 can be expressed as Equation 2 (derived from [12]):

$$P_{F_1,F_0} = kCV^2HD(F_1, F_0) \quad (2)$$

In Equation 2, k represents a constant depending on physical-layer parameters of the bus, and $HD(F_i, F_j)$ is a function which computes the Hamming distance between two words.

In a real system, bus wires are not exactly independent, as there are electrical effects such as cross-capacitances and inductive effects between the wires. [13] [14]. Therefore, particular bit transmissions within links (or between them when crossing very closely) may affect each other. The architecture proposed in this paper can easily be extended to such coupling power models, although for simplicity the model presented in Equation 2 is used for the implementations and results contained in this paper.

III. Problem Description

The problem considered in this paper is to propose and experimentally verify an algorithm for dynamic power consumption modelling in a cycle-approximate NoC, attempting to retain the power consumption modelling accuracy of a cycle-accurate model through a simplified algorithm that uses coarser transactions. This removes the need to model as a simulator event the passage of each flit through the NoC. A further issue considered is the implementation of low power codecs for NoCs such as transition and bus-invert coding, and the verification of their power reduction performance using our methodology.

The NoC architecture considered combines features from Hermes [15] and QNoC [16]. The arbiter design is modelled upon Hermes, with a structure of four ports to/from neighbouring routers, one port to/from a local processing core, and the use of XY routing to forward data. Processor to router links are considered to be part of the logical structure of routes in the network, and therefore every route modelled has two additional hops for the link to and from a processor. The flow control between the elements is credit-based. The existence of multiple priority levels is based upon QNoC [16], with virtual channels and support for priority preemption. This allows the modelling of realistic applications in which quality of service requirements can be expressed through priority levels, allowing urgent traffic to be transferred over non-essential.

The application layer is abstracted away from the platform model, via a static task mapping which assigns the tasks to specific cores throughout simulation execution. This static mapping keeps tasks upon their same fixed locations throughout simulation execution. It is assumed that even though packets are injected into the network with timings controlled by their application layer, individual flits of a packet are injected into a network link with constant timing (equal to that of the system clock speed).

This paper considers dynamic power consumption, although additional sources of power consumption such as leakage current are not considered. However, it is possible to model these using realistic values for particular NoC routers (from the internal structure of arbiters, buffers etc) from empirical models of representative NoCs. In addition, although the specific simulation uses an independent-bus power model, the algorithm itself is generic in its power tracking and can be adapted to any particular framework. In this paper, power consumption is represented in abstract units consisting of the total number of bit transitions. Since the grid links considered are assumed to be constant length, it is possible to obtain total power consumption by multiplication with a conversion factor.

IV. Algorithm Description

The algorithm presented in this paper is an extension of earlier work on TLM latency modelling [17]. The core of the algorithm is a flow-set sorted in priority order. The algorithm is presented in Listing 1, with grey lines indicating the necessary additions to implement power consumption modelling. The algorithm uses the concept of interference sets (calculated on flow admission) to manage contention for particular links. The interference set of a flow refers to those flows that when active will block a flow from data transfer - those sharing at least one link with a flow, and of a higher priority. The algorithm keeps track of activation times and expected termination intervals, and during update events, changes flow activation status when a flow is blocked by a higher priority flow. When flows are known to not have been interfered with by higher priority flows, their progress over the interval including the flits transmitted and their power impact is updated in a single transaction. Newly admitted high priority flows therefore block others from proceeding until their activation is processed. Traversing the flow list in priority order ensures that the algorithm respects priorities by allocating links to the highest priority flows when they are activated. Therefore, simulation events only occur when flows enter or leave the NoC, or when preemption alters the active status of a flow.

Listing 1: Pseudocode for updating a list of flows with power tracking

```
1  update(currentTime) {
2    for each flow_i in flowlist {
3      if (active_i) {
4        flitstosend_i = flitstosend_i − sentFlits
             (currentTime − t_ai);
5        t_ai = currentTime;
6        if (flitstosend_i == 0) {
7          remove flow_i from flowlist;
8          trackPower(flow_i);
9        }
10       for each flow_n in interference_i {
11         if (active_n) {
12           active_i = false;
13           trackPower(flow_i);
14         }
15       }
16     } else {
17       if (!active_n for all flow_n in
             interference_i) {
18         active_i = true;
19         requestUpdate(currentTime +
               basicLat(flitstosend_i));
20       }
21     }
22   }
23 }
```

The power tracking algorithm employs additional state in the concept of a flit position, which represents the progress of the packet from the source node. Consider the progress of a single packet under transmission from source to destination via intermediate routers in an otherwise idle NoC, as shown in Figure 1. Flit position F_p defines a counter incrementing with time whose valid values are in the range $0 \leq F_p < N + H$,

where H is the hop count of the route and N the total number of flits in the data packet. During $F_p < H - 1$ the route is in a growing phase. During this phase data is being transmitted to the destination, but some of the later links upon the route may be idle as data has not yet reached its destination. The additional flit position values beyond $F_P > N$ account for a shrinking phase in wormhole switching, in which no new data is being injected, but the final flit has not yet reached its destination.

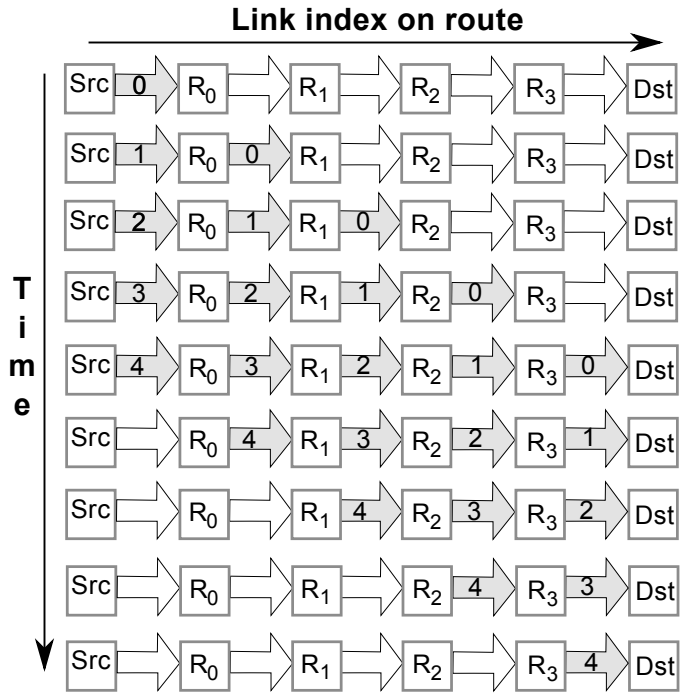


Fig. 1: Structure of the algorithm and flow progression to the destination

The newly introduced *trackPower* function is invoked in lines 8 and 13 of the *update* function, upon the admission of a new flow to the network that preempts $flow_i$, or the completion of $flow_i$. The *trackPower* function (Listing 2) is responsible for sequential link-level modelling of the particular flits involved in this transaction over the interval, within the temporal window in which the flow was identified by the TLM algorithm as having exclusive access to the link. Therefore, rather than the propagation and power impact of these flits being modelled as individual simulator events, they can be processed during a single simulation event in our algorithm. The logic of the *trackPower()* function can be explained as follows:

- Work out the flowing time for this flit by subtracting the current time from its last activation time;
- Work out the number of flits transmitted by this active flow (from its activation time to current simulator time) assuming a constant flit injection rate over time (by default equal to system clock speed);

Listing 2: Pseudocode for TLM power tracking

```
1  trackPower(currentFlow_i) {
2    flowing_time = currentTime − t_ai;
3    endFlitPos = startFlitPos + flowing_time *
         flitTransmissionRate(currentFlow_i);
4    for (flit_index = startFlitPos to
         endFlitPos) {
5      for (link_index = 0 to hopLength(
           currentFlow_i)) {
6        flit_for_link = flit_index − link_index;
7        if (flit_for_link >= 0 and
8            flit_for_link < endFlitLimit(
                 currentFlow_i)) {
9          link = getLink(currentFlow_i,
               link_index);
10         flit = getFlit(flit_for_link);
11         registerTransmission(link, flit);
12       }
13     }
14   }
15   setFlitPos(currentFlow_i, endFlitPos);
16 }
```

- Adding the start flit position to the number of flits injected by the source over the flowing time, to obtain the end flit position;
- Iterating over both the flit indices from start to end flit position, and link indices on the route, look up the flit which has reached this link on the route by this point in the algorithm (flit index minus link index);
- Notify each link of that flit's presence. Inside the function *registerTransmission*, the links apply the chosen encoding and record the power consumption triggered according to a selected model (for example, the independent-wire transition counting specified in Section II-B, Equation 2).

The example in Figure 1 shows the flits that will be applied to particular link indices during modelling for an example with $N = 4$ and $H = 5$, with a starting flit position of 0 and an ending flit position of 8. In this case, under a conventional cycle-accurate model in which each flit transmission is modelled as a simulator token, a simulator event would be required upon every grey arrow. However, the TLM algorithm presented is able to model the entire progression of the route and register its flits in a single simulation event, upon its completion or preemption (in this case it is not preempted by another flow). The horizontal axis from arrow to arrow represents link indices moving along a particular route (modelled in the inner loop of Listing 2) and the rows representing sequential time-steps. Therefore, the link from processing core **Src** to router $\mathbf{R_0}$ would receive in sequence flits $0, 1, 2, 3, 4$. Note that in these case the flow is in its growing phase is still in progress for flit positions 0 to 3, and therefore no link activity is modelled

upon the dashed links in these intervals. Similarly, the route is in its shrinking phase for flit indices 5 to 8.

The algorithm's forward execution therefore enforces a temporal ordering between the preemption points. Since it iterates forward through time in its outer loop and along the links of the route in its inner loop, it is therefore possible to apply cross-coupling power consumption models that depend upon the activity of adjacent network links, e.g. [13] [14]. For power consumption accuracy, the global time in an enclosing simulation model does not need to match the event occurrence times simulated here, but at the link level all the events are applied forwards in temporal order. However, if the flow is pre-empted and a higher priority packet has to be delivered in between the flow of flits, it is possible that some transitions will be mistimed between the boundaries during which preemption occurs.

The model presented here does not model the impact of preemption by higher priority flows, nor does it compute dynamic intersections of routes at particular time intervals. It is possible to imagine a pathological case in which the hop count/flit number per flow is highly variable (some short packets and occasional long ones) which could result in the model giving a substantial under-estimate of power consumption. Assume a very long packet L_p that has been end-to-end encoded for low transition count in its flit sequence is repeatedly preempted by the presence of a sequence of very high priority short packets (i.e. single flits) which share a common link with L_p. Assume further that these packets have inverted bit sequences to L_p, so as to cause transitions on every link wire when traversing the link. Since the high priority packets are so short, they will not occupy their routes fully with flits, but the algorithm presented here treats them as if they do. In the real system, these packets would pass link L interleaved with the long packet (causing additional transitions), but the model would delay the transmission of the long packet entirely until after these have finished. As a result several transitions will be missed out.

Our demonstration results in Section VI-B indicate that in a realistic application case in which packet size range from 16kbit up to 512kbit (representing long lived high priority flows) this theoretical case has only a small impact on overall accuracy. Firstly, in realistic packet sizes and preemption rates, transitions caused by preemption are comparatively infrequent compared to the transitions which occur between flits of the same source packet. Secondly, even if such preemptions did occur, packets are not typically constructed to deliberately produce transitions on preemption.

V. Simulation Implementation

The power modelling framework is implemented upon the *lsi.noc* simulation framework [17] based upon the Ptolemy II simulator [18]. Producer and consumer actors are used to represent data sources and sinks upon the processing cores, which are called by the application model during message generation and transmission.

Parameter	Value
Default flit size (bits)	32
NoC Clock speed (MHz)	1000
Buffer capacity (flits)	7
Number of virtual channels	40
Default data source	XML map model

TABLE I: Simulation parameters for NoC testing simulations used in the results

The cycle-accurate simulation (used as a reference implementation for accuracy validation) uses Ptolemy II actors to represent buffers and arbiters. In the cycle-approximate TLM model simulation, a single central actor represents the NoC, implementing the algorithm defined in Section IV. Therefore, Ptolemy II events that relate to power tracking are only triggered upon admission of a packet to the network or upon scheduled update times to handle contention or process completed flows, greatly reducing total event processing and improving overall performance.

In both the cycle-accurate reference implementation and cycle-approximate implementation of our algorithm, objects are created to hold total dynamic power consumption and process the power impact of data passing over each link. In the reference implementation, power tracking is modelled by monitoring the tokens produced during the processing and decision phase of the arbiter, and assigning them to the object for link power tracking of the output link. In our algorithm, methods upon the link tracking objects are called by *registerTransmission* in Listing 2.

VI. Simulation Results

A. Performance Results

In order to investigate the performance benefits of the TLM algorithm defined in this paper, the simulation execution time of the cycle-approximate TLM power measurement algorithm was compared with the reference cycle-accurate model. Comparisons were made using a 4x4 grid NoC topology, with an application layer modelling the processing layers of an autonomous vehicle, specifically video data generated from on-board cameras, and its navigational and stability control aspects. Parameters for the simulation are given in Table I.

Figure 2 illustrates the differences in simulation execution time for our cycle-approximate TLM power modelling algorithm compared to the reference cycle-accurate model, logging execution times throughout 2 seconds of simulation execution of the same application layer. The graph also shows the execution times for the original implementations of the algorithms, before their modification for power tracking.

It is obvious from these results that the our cycle-approximate TLM power tracking scheme is approximately three orders of magnitude faster than for the cycle-accurate model. Previous work on TLM latency modelling [17] which did not consider power modelling recorded a performance difference of approximately three orders of magnitude for

TLM, as also exhibited here for our results. It is important to note that even cycle-approximate TLM power modelling requires the generation of source data, formatting this data into flits, and loading it into Ptolemy II tokens at least once to reach the interconnect, which is not required in the original TLM modelling without power consumption considerations. Also, power tracking requires bitwise operations and array copying to determine transitions. This accounts for the execution time increase between our original latency modelling algorithm and its modification here with power tracking. The differences between the cycle-accurate reference model with power tracking and without it is comparatively less significant, since execution time in these models is dominated by the processing and propagation of tokens through Ptolemy II actors and by arbiter decisions, rather than data generation and bitwise operations for power modelling.

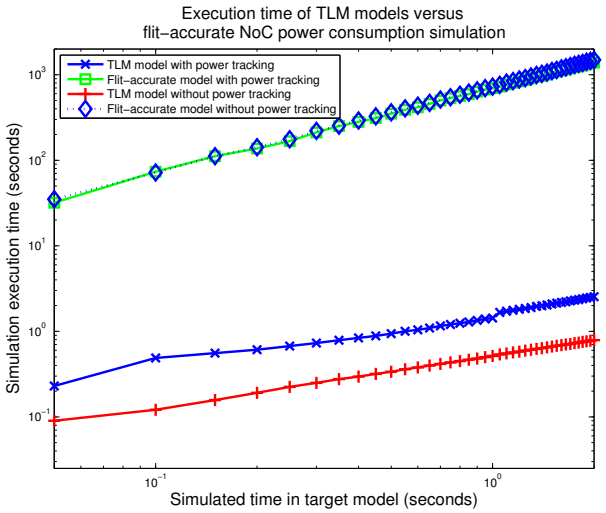


Fig. 2: Simulation performance showing approximately three orders of magnitude execution time for our cycle-approximate algorithm

B. Simulation Accuracy

This section considers the accuracy achieved by the TLM algorithm, and in particular the worst-case accuracies of dynamic transitions recorded upon particular links. The overall accuracy of transitions recorded upon the particular links was -0.24%, that is a very slight under-estimate by our cycle-approximate TLM model. Figure 3 considers the accuracy of modelling for particular links, giving a histogram of error levels for individual links. It is clear that the majority of links exhibit an transition count error less than 1%, although a small proportion of outliers exhibit errors of $1\% - 2\%$ and a small cluster around 2.5%.

Figure 4 shows the structure of the network and the relative errors of different links between the cycle-approximate and cycle-accurate reference model. Figure 4a shows the relative error, while Figure 4b indicates the total power consumption of links and therefore their total traffic carried. The deepest red

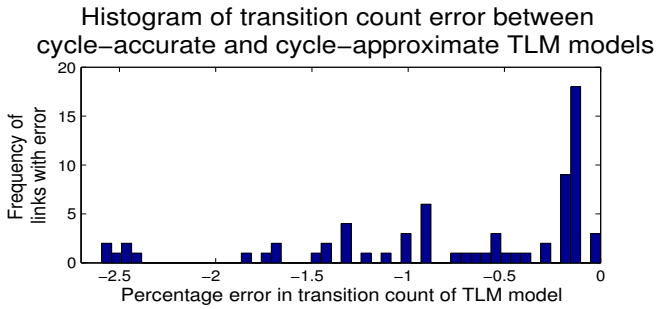
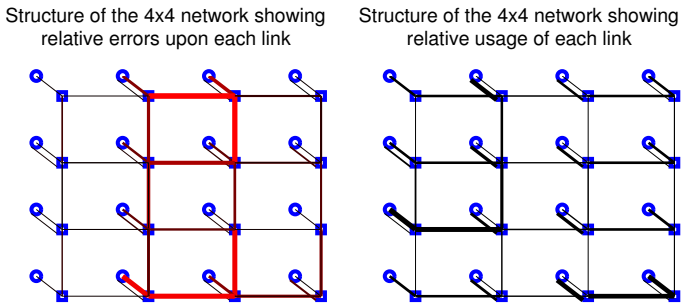


Fig. 3: Histogram of link relative errors for our algorithm against the reference implementation showing relative errors under 3%



(a) Relative error locations (b) Link relative power consumption

Fig. 4: Network errors showing error locations and total power consumption (with default optimal traffic pattern)

in Figure 4a is the maximum error of any link, which as shown in Figure 3 is under 3%. From this, it is clear that the nodes which have the greatest relative errors are those that carry the smallest amount of traffic, and therefore errors upon these links have the overall smallest contribution to overall network power consumption accuracy. In the application traffic model tested, these correspond to links used primarily for the transmission of navigation control information for the autonomous vehicle, which consists typically of shorter packets and more frequent preemptions. However, since the overall impact of their transmissions on the network as a whole is minor to the small amounts of data interchanged upon these links, their adverse effect upon the accuracy of network power consumption as a whole is minor, which accounts for the overall -0.24% deviation between the cycle-accurate and cycle-approximate schemes.

An alternative traffic mapping was considered in Figures 5 and 6, which considers moving traffic sources (particularly those responsible for position tracking and navigation of the autonomous vehicle) into a less optimal location to produce additional contention with the video processing flows. The figures shows again similar results for overall contention, with again heavily used links receiving the smallest inaccuracies. In this model one link had no traffic in either direction and is therefore not displayed. The overall power consumption error increases slightly to -0.29%.

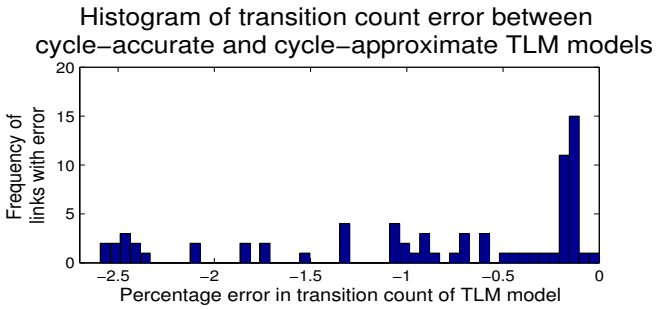
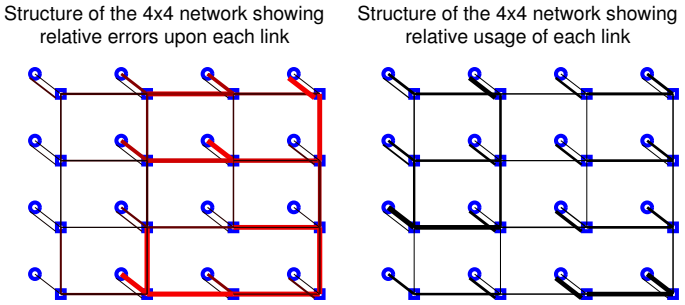


Fig. 5: Histogram of link relative errors for our algorithm against the reference implementation showing relative errors under 3%



(a) Relative error locations (b) Link relative power consumption

Fig. 6: Network errors showing error locations and total power consumption with less optimal traffic pattern (one link not shown due to absence of traffic in either direction upon it)

C. Transition Coding Validation

A physical-layer independent approach to building low-power NoCs focuses upon link coding techniques, which encode data specifically to reduce dynamic power consumption by reducing bit transitions. The following subsections aim to demonstrate the accurate implementation of low-power encoding schemes upon our TLM simulator, verifying that such schemes deliver their intended power savings under our higher performance TLM power modelling. Transition coding [19] encodes bits using an XOR operation from the previous bit sent over a particular wire, favouring data in which bits are frequently alternating. Figure 7 shows the results in cycle-approximate TLM simulation of replacing the application data source with a stream of alternating pairs of 32-bit words **00000000 FFFFFFFF**, showing a reduction in total bit transition events by approximately 90%. This is produced through the ability of the transition codec to convert the alternating bits of the stream to constant values.

D. Bus-Invert Coding Validation

Bus-invert coding [20] is an encoding scheme that reduces bit transitions through additional signalling wires. In its simplest form, activation of a single inversion wire inverts the bus, which indicates that all wires upon the bus carry signals that are the opposite of their usual coding. This is highly beneficial

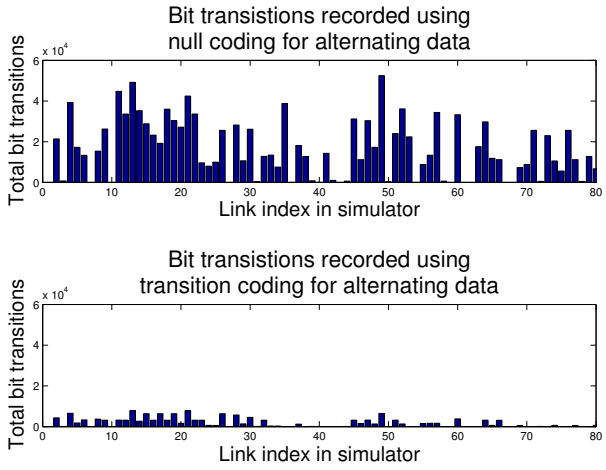


Fig. 7: Transition coding and the reduction in transitions compared to null coding

for patterns which contain a large number of transitions, as signalling the inversion wire allows the entire bus to be inverted, a significant reduction of the worst-case transition activity. In our simulations, activation of the alternating bus wire produced a reduction of 96% of transition activity on a 32-bit bus. In typical data, however, the performance gains of bus-invert coding are not as significant, since transitions are less predictable [21]. For example, in data drawn from a uniform random distribution such as experienced if data is already highly compressed (or otherwise of maximal entropy), the modal case is for transitions upon precisely half the wires; a situation in which bus-invert coding. In [21] the following analytic expression for the expected number of transitions $T(n)$ per flit of uniform-random data under bus-invert coding upon a bus (of even width) n is provided:

$$T(n) = (n+1)\left(\frac{1}{2} - \frac{C^n_{n/2}}{2^{n+1}}\right) \tag{3}$$

Since the expected number of transitions is $n/2$, it is possible to calculate the expected reduction in transitions $R(n)$ for bus width n (as a proportion of uncoded) as:

$$R(n) = 1 - \frac{T(n)}{n/2} = 1 - \left[\frac{(n+1)}{n/2}\left(\frac{1}{2} - \frac{C^n_{n/2}}{2^{n+1}}\right)\right] \tag{4}$$

Figure 8 presents simulated and analytic results for bus-invert coding of uniform random data for bus widths of 8, 16 and 32 bits, demonstrating a very close agreement between theoretically predicted transition reduction and that obtained from the TLM model. The highest error is for 32-bit encoding, where a discrepancy of 11.3% between 11.2% was observed. This error is well within the accuracy observed for TLM modelling in Section VI-B.

VII. Further Work

A potential optimisation exists to avoid the processing overheads of registering bit sequences individually with individual links, depending on the characteristics of the power model and

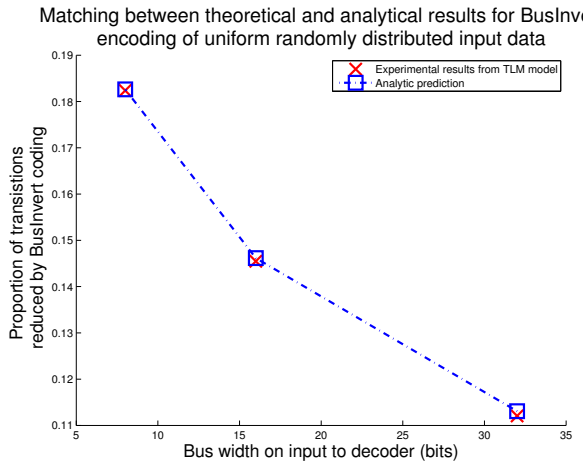


Fig. 8: Match between simulation and theoretical prediction for bus-invert coding

any link encoding that may be in use. If power consumption only depends upon the previous data the link received and the next (as in a transition counting model), and the encodings upon the links of a flow are the same, then is it possible to optimise the algorithm further by computing the total power consumption increment associated with different flit positions in a flow.

In terms of the mechanism of the simulation, one particular artifact of the TLM modelling process is that it allows only one particular route to be activated upon statically intersecting routes simultaneously. Although with long packets the majority of routes will be fully occupied, it is possible that some routes will be partially unoccupied. Therefore, some accuracy gains could be obtained by allowing routes to flow through the growing and shrinking phases of another route (when no data transfer is occurring on this route).

VIII. Conclusions

This paper has defined and investigated an algorithm for power consumption modelling within cycle-approximate transaction level modelling (TLM) upon a priority-preemptive NoC, without the overheads associated with cycle-accurate modelling with a simulator token tracking every flit transmitted throughout the network. The assumption of priority preemption is an advantage for realistic scenarios in which traffic has various levels of importance. The algorithm has been successfully implemented upon our lsi.noc simulation package, and simulation results in a NoC for a real application have demonstrated that simulations can achieve an accuracy of under one percent in overall dynamic power modelling accuracy (via bit transitions), with a speed benefit of between two and three orders of magnitude in comparison to a flit-accurate model. Using the bus-invert coding schemes, close agreement with theoretical predictions upon the simulation platform has also been demonstrated.

Acknowledgements

The authors are grateful to the EPSRC for financial support, and to Alberto Garcia-Ortiz for providing MATLAB code implementations of low-power NoC coding schemes.

References

[1] G. De Micheli and L. Benini, *Networks on chips: technology and tools*. Morgan Kaufmann, 2006.

[2] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3 –21, Jan. 2009.

[3] K. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power network-on-chip for high-performance SoC design," *IEEE Trans. VLSI Syst.*, vol. 14, no. 2, pp. 148 –160, Feb. 2006.

[4] L. Cai and D. Gajski, "Transaction level modeling: an overview," in *CODES+ISSS 2003: 1st Int. Conf. Hardware/Software Codesign and System Synthesis*, Oct. 2003, pp. 19 –24.

[5] D. D. Gajski and R. H. Kuhn, "Guest editors' introduction: New VLSI tools," *Computer*, vol. 16, no. 12, pp. 11–14, Dec. 1983.

[6] A. Rose, S. Swan, J. Pierce, J. M. Fernandez *et al.*, "Transaction level modeling in SystemC," *Open SystemC Initiative*, vol. 1, no. 1.297, 2005.

[7] L. Semeria and A. Ghosh, "Methodology for hardware/software co-verification in C/C++," in *ASP-DAC 2000: Design Automation Conf.*, Jun. 2000, pp. 405 –408.

[8] K.-L. Lin, C.-K. Lo, and R.-S. Tsay, "Source-level timing annotation for fast and accurate TLM computation model generation," in *ASP-DAC 2010: Design Automation Conf.*, Jan. 2010, pp. 235 –240.

[9] J. Aynsley, "TLM-2.0 Language Reference Manual," OSCI, Tech. Rep., 2009.

[10] G. Schirner and R. Dömer, "Quantitative analysis of the speed/accuracy trade-off in transaction level modeling," *ACM Trans. Embed. Comput. Syst.*, vol. 8, no. 1, pp. 4:1–4:29, Jan. 2009.

[11] T. Mudge, "Power: a first-class architectural design constraint," *Computer*, vol. 34, no. 4, pp. 52 –58, Apr. 2001.

[12] W. Fornaciari, D. Sciuto, and C. Silvano, "Power estimation for architectural exploration of HW/SW communication on system-level buses," in *CODES '99: 7th Int. Workshop Hardware/Software Codesign*, 1999, pp. 152 –156.

[13] N. Banerjee, P. Vellanki, and K. S. Chatha, "A power and performance model for network-on-chip architectures," in *DATE 2004: Design Automation Test in Europe Conf.*, vol. 2, Feb. 2004, pp. 1250–1255.

[14] J. A. Davis and J. D. Meindl, "Compact distributed RLC interconnect models - Part II: Coupled line transient expressions and peak crosstalk in multilevel networks," *IEEE Trans. Electron Devices*, vol. 47, no. 11, pp. 2078 –2087, Nov. 2000.

[15] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *Integr. VLSI J.*, vol. 38, no. 1, pp. 69–93, Oct. 2004.

[16] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *J. Syst. Archit.*, vol. 50, no. 2-3, pp. 105–128, Feb. 2004.

[17] L. S. Indrusiak and O. M. dos Santos, "Fast and accurate transaction-level model of a wormhole network-on-chip with priority preemptive virtual channel arbitration," in *DATE 2011: Design, Automation Test in Europe Conf.*, Mar. 2011, pp. 1–6.

[18] C. Brooks, E. A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng, "Heterogeneous concurrent modeling and design in Java (vol .2: Ptolemy II software architecture)," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-29, Apr. 2008.

[19] P. Ramos and A. Oliveria, "Low overhead encodings for reduced activity in data and address buses," in *Proc. Int. Symp. Signals, Circuits and Syst.*, 1999, pp. 21–24.

[20] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. VLSI Syst.*, vol. 3, no. 1, pp. 49 –58, Mar. 1995.

[21] R.-B. Lin and C.-M. Tsai, "Theoretical analysis of bus-invert coding," in *Proc. 43rd IEEE Symp. Circuits and Systems*, vol. 2, 2000, pp. 742 –745 vol.2.