OVNI-NET: A flexible cluster interconnect for the new OVNI Real-Time simulator

Tom G. De Rybel Jorge A. Hollman José R. Martí

The University of British Columbia
Department of Electrical and Computer Engineering
2356 Main Mall, BC. V6T1Z4 Vancouver, Canada
Phone: +1(604)822-2364, Fax: +1(604)822-5949, e-mail: tomr@ece.ubc.ca

Abstract - To accommodate the MATE power system solution scheme in the OVNI real-time simulator, a new cluster interconnect is proposed. The system presented combines both high performance and flexibility, allowing real-time operation. Using an FPGA-based approach, the hardware can be fully reconfigured and optimised to very diverse interconnect architectural demands without physical hardware modifications. Also, due to the fully-parallel and self-contained operational structure, the communication overhead introduced is very low while still maintaining a high bandwidth. Through the use of a shared-memory topology, switching is avoided, further enhancing the system.

Keywords - clustering, real-time, network interconnect, low-latency, OVNI real-time power system simulator, FPGA

1 Introduction

The original OVNI [1],[2],[3] real-time simulator presented one of the lowest latency network solutions available today. However, with the implementation of the general MATE algorithm[4],[5], a new, more flexible interconnect solution became necessary. Also, the previous hardware solution[3] was limited in bandwidth and tied to the IDE[1] hard-disk interface. The IDE technology is soon to disappear, in favour of serial ATA[2].

The new hardware system developed circumvents any of the normal PC standard ports and is build directly on top of the industry-standard PCI[3] bus. Also, the parallel connections between nodes are replaced by high-speed 0.5 Gbps serial links, allowing for a system of larger physical size with a more manageable wiring solution. These enhancements yield an even lower latency while significantly boosting the throughput compared to the old system. It also allows the hardware to be used on non-PC platforms such as Sun, Apple, HP, and IBM workstations.

A major change to accommodate the flexibility of MATE was the incorporation of an intelligent hub, avoiding many of the latency issues introduced by switched packet network solutions. To further boost the performance, no microprocessors were used in the network system. Instead, the required intelligence was distributed throughout the system and implemented locally in logic. This allows for a highly parallel operation of the interconnect with an absolute minimum overhead.

The integration of the new interconnect system and the MATE solution scheme is now so tight that the network system itself has become part of the solution algorithm. Nonetheless, due to the highly modular design approach, and the extensive use of FPGA[4] technology, the system can be modified at will.

Very different solution schemes and enhancements such as hardware error control, digital and analog I/O[5], and other features can be implemented without electrical modifications, allowing the same physical hardware to be used in both a research and a production environment. The result is an economic, obsolescence proof, flexible, high-speed low-latency cluster interconnect system build around industry-standard components.

In this paper we will demonstrate the much improved performance of the new interconnect and compare it with the previous solution.

2 Background

In order to achieve high communications efficiency, one must first take a closer look as to how the system is represented in MATE. This formulation is unique in that it can be almost transparently mapped to a customised interconnect system.

2.1 MATE (Multi-Area Thévenin Equivalent)

At the hart of the OVNI simulator lies the MATE system representation. MATE segments a large simulation into smaller chunks (areas or sub-systems) which can be solved on individual solver nodes. These areas are then represented as Thévenin equivalents to the outside world.

The connections between these sub-systems are represented as the lines impedance matrix[5]. Due to this scheme the internal complexity of the sub-systems can be hidden from the overall simulator. To solve for the entire system, the central machine only has to solve for the links impedance matrix connecting the different Thévenin

[1] Integrated Drive Electronics
[2] Advanced Technology Attachment
[3] Peripheral Component Interconnect
[4] Field Programable Gate Array
[5] Input/Output

equivalents, which is independent of the internal complexity of the individual subsystems.

Using matrix notation, the sub-systems and links are formulated as a large (sparse) matrix. The sub-systems are organized around the diagonal and the links in columns (Fig. 1). Looking at the high degree of regularity, we can readily identify the areas that contain the information we want to communicate in the system.

The design objective is to map these areas (containing the solved data) into the physical solver nodes joined by physical interconnects.

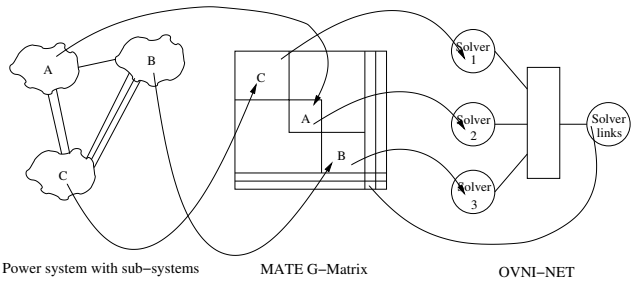

Figure 1: Subsystem to Mate to OVNI-NET Mapping

2.2 ΔT: Simulation clock

Besides communicating the Thévenin equivalent data, the simulation requires a steady clock to not only synchronise the communications but also the simulation itself. In MATE, this all important Δt signal is not necessarily the same for all individual sub-systems. Using to the concept of latency [6], [7],[8],[5], some solver nodes can be running at an integer multiple of the fastest reference Δt.

As such, the simulation clock system poses some extra requirements on the network system. It is agreed that the system runs at the fastest Δt in the simulation and that slower nodes skip a number of cycles before again taking part in the simulation. All of this happens without introducing extra software-overhead since it is implemented as state-machines in the networking system logic.

In itself, the simulation interval timing system is a good illustration of the power of the FPGA design approach. A clock of this type is uncommon in normal computer network systems, yet on the OVNI-NET platform we can add something as invasive as this without much effort (Fig. 2).

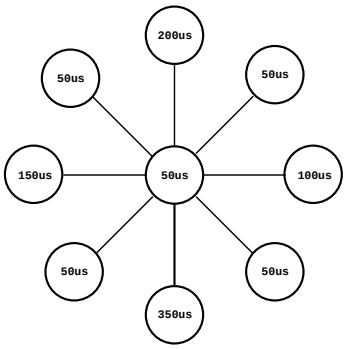

Figure 2: Multiple Δt's

3 Implementation

To function as a research platform the system was designed with a high amount of flexibility in mind. Part of the solution was found in a fully modular design approach. But that in itself is not enough to accommodate the mutability of an environment where new concepts requiring very different networking layouts are frequently tested. Also in a production system the ability to drastically modify the machine easily with little cost protects the investment as new techniques can be implemented with a mere software update.

The result of these concerns is a design split into self-contained sub-systems, each employing their own local control hardware with the ability to fully re-configure the circuits. For these reasons one could call OVNI-NET a distributed network processor: distributed as the intelligence is spread-out throughout the system where it is needed locally; network processor due to the autonomous handling of the communication tasks.

Throughout the design all efforts have been made to prevent hard-wiring of any functionality. In the cases where that could not be avoided, the entire sub-system was constructed on a separate board that plugs into the respective master board. If modifications are required, the necessary investment, and system down-time are confined to this board. However, despite the rigorous choice for flexibility, a classic software approach using microprocessors was avoided.

A solution to meet the extremes between total changeability but slowness of software and total rigidity but high speed of wired hardware was found in the FPGA technology. This technology offers hardware-like speed and latencies with software-like flexibility and changeability.

When designing the system, the power and inherent segmentationalist approach of the MATE concept surfaced throughout the network design itself. Indeed, the divide-and-conquer philosophy almost dictated the layout of the system, tying principle and design closely together. This reliance upon one another is part of the key to achieving high performance while maintaining low cost. However, due to the flexibility of the design, it can be readily modified to suit an entirely different system as well. This polymorphic property is the strength of the OVNI-NET design concept.

3.1 Principle of operation

MATE is a two-step solution process. First the subsystems, mapped onto the solver nodes, compute their local results in parallel. When they are done, those results are communicated to the central solver which solves the reduced system represented by the links subsystem. The updated link currents are then communicated back to the sub-systems and the whole process starts again. This cycle has to be completed within the smallest Δt used in the system.

When looking at the block diagram of the network one can see that its topology follows the general concept. Each of the node computers has an interface card that connects

to a hub. This hub concentrates, or separates, the data to the central solver. The link to this machine is parallel for additional speed. The whole network system runs synchronous at the clock of the central solver.

The hub design is the key to the latency performance of OVNI-NET. For MATE, we only need to communicate small amounts of data, but with strict latency requirements. Ideally, the central solver should be able to grab whatever data it needs from any of the node computers memory with zero delay and vice-versa. This, however, is impossible, if nothing else due to the limitatons imposed by the speed of electrical signals in the medium.

The next best thing would be a multi-processor design with a shared memory. Although certainly possible, this brings us in the realm of large servers and other super computers. Their cost is prohibitive and it also defeats the whole purpose of MATE, i.e., the ability to run, and expand, on cheap commodity hardware.

Limited by the PC platform, a possible solution is the use of PCI cards and a shared-memory (the hub). This is what OVNI-NET does.

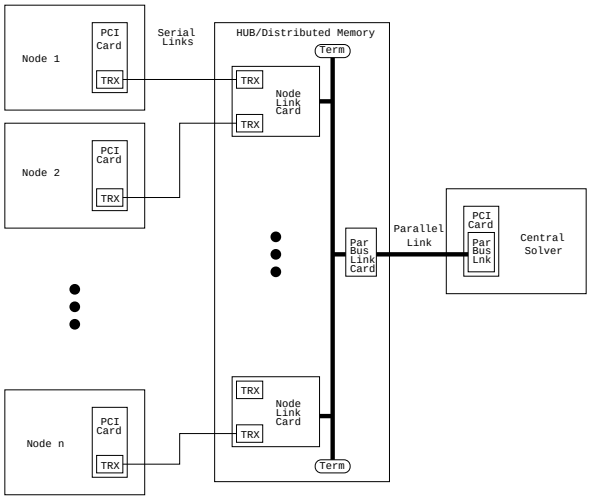

Figure 3: OVNI-NET Hardware Overview

The design functions not so much as a classic network, but rather as a shared memory pool. The moving of data in the system is handled through autonomous memory-to-memory copy operations. Autonomous in the sense that the PCI cards employ their own intelligence and are capable of handling all network communication details at hardware speeds. As a result, we already have savings in latency as there is less need for from-and-back communication cycles between the networking system and the system CPU.

For illustration, let us walk through the communication cycle as it happens during operation. First, all nodes in parallel transfer the just completed sub-system solution. The system's CPU memory-maps this data onto the card's local memory. This is merely an abstraction as the card will immediately transfer this data to the hub. All nodes can do this in parallel. The transfer takes place over their 528Mbps serial links to the hub, the shared memory pool. This memory is accessible in full parallel form.

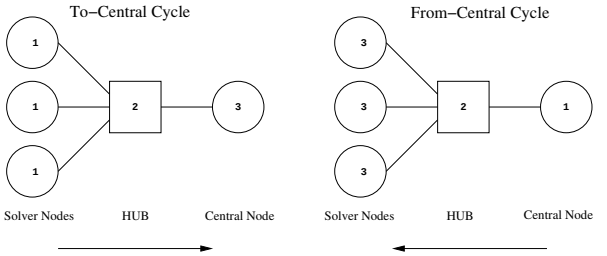

Figure 4: Cycle Data Flow

The main savings in latency are found in this hub part of the design. First, the links are always on. This means that the PLLs of the serial link serialisers/deserialisers are constantly in lock and do not need to be relocked. This implies for the parts we used about 1μs per link.

Second, there is no switching. The central node does not have to poll each machine individually, request data, and address the next. This means that the latency of the system is NOT additive in this stage. With Myrinet networks and the likes, one has to access each machine sequentially, adding up the latencies.

Myrinet is designed for a wholly different purpose where each machine has to be able to talk to each other machine directly. Hence the use of switch fabrics and switched hubs. This adds complexity, latency, and much cost to the system that is not required in our case. Our shared-memory approach avoids this complexity altogether and is perfectly tailored to the MATE concept, allowing it to work at far better performance.

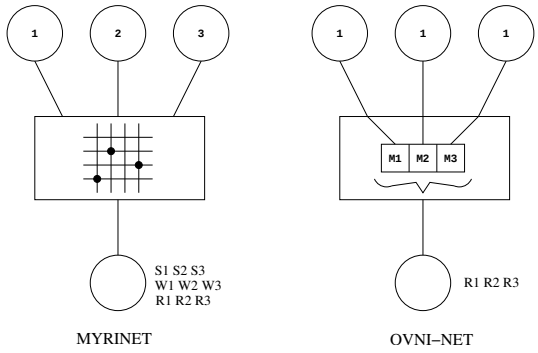

Figure 5: MYRINET vs OVNI-NET

In OVNI-NET the network system infers almost all required operations from the data and the state of the network system itself, allowing even dynamic change of data sizes without program intervention.

This aspect, and the latency savings explained above, gives our design superior latency performance due to the inherent specialised structure of the whole. Yet, we are not locked-in to this one custom application as the design can be tailored to other situations by re-programming the FPGA's. This way our design is quite universal while still being highly specialised. If one were to design a switch in addition to the hub, virtually all common network topologies would be possible.

After the data copy to the hub, the second stage of the first half of the communication cycle can start. Here all results are copied directly into the central solver's memory. Since all data has already been gathered, it can then be moved in one single block without interruptions at the full PCI bus speed.

Here lies the bottleneck of the MATE concept when implemented on real hardware. There is only so much data one can move in a certain amount of time, restricting the otherwise unlimited expandability. This is a common issue with all systems based on a single central solver. However, as we will show later, this is not a paramount obstruction since with modern machines this limit is so high that even substantial networks can be handled without problems. Also, the advent of the new PCI-X architecture will alleviate the problem even more when it becomes more readily available.

For our second stage of the communiations (from central to nodes) the system is symmetric in reverse. Here the symmetry in the hardware serves another important purpose. When interfacing with real-world devices it is imperative that all of these run at exactly the same clock rate to prevent ADC/DAC sample time mismatches. When care is taken to make all interconnects of equal length, the inherent symmetry and the centrally-controlled Δt logic will assure a very accurate synchronisation between all parts of the system. In addition, in most cases the PCI cards can be modified and expanded with an analogue interface card enabling stand-alone data-acquisition and interfacing, further reducing cost by eliminating a PC as mediator for real-world interfacing.

3.2 Dynamic Discontinuous Address Translation

The underlying architecture of OVNI-NET, the shared memory concept, invites a fully memory-mapped architecture. However, for efficiency and the desire to dynamically allocate the data-sizes required per node by inference, a special addressing scheme had to be devised.

The situation is as follows: the central solver has a block of memory with node data. To optimise the operational speed, all node data blocks have to be lumped together in one continuous chunk. Each block has its length written in the first word and the board knows the number of nodes (the only parameter that has to be actively configured).

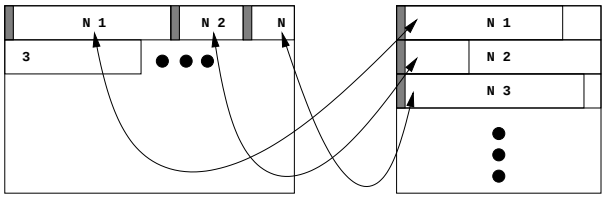

Figure 6: Dynamic Discontinuous Address Translation

Now the central machine needs to map all of these node data blocks on the correct solver node. To speed things up even further, by making the data blocks dynamic we do not have to transmit the full data size (1K words) if this is not required and thus save communication time and bandwidth.

For maximum efficiency, we need to do this with the least processor intervention possible. This means that we will have to communicate this data continuously, without further processing (eg. lookup table-based translation) to the shared memory. This allows us to use block operations on the PCI which save time, and also enables DMA techniques, when implemented.

This introduces the problem that the network hardware itself has to determine where exactly it has to map parts of this data stream into its shared memory. Each node card (which has two solver node shared buffers) is address-mapped in 1K word chunks. Thus while operating on these memory segments, the system has to jump to the next block when all data for the previous one has been delivered.

A special addressing scheme is thus needed to handle these dynamic and discontiniuous operations. In our case we can make one simplification, however, and that is that all operations will always be sequential, never random access. Although this changes little on the principle, it does make the implementation easier.

By capturing the logic in state machines instead of programs, this dynamic allocation comes at virtually no extra overhead cost while we can keep all communications optimised during the operation itself and never [6] have to actively set parameters in the boards, omitting the overhead these individual PCI bus operations would incur.

The only protocol convention in the system finds its origin here. To infer the data length, the very first word of a data segment has to be the length of that segment. This word is read by the hardware and used to configure all the buffers and address translation logic. As this happens at runtime while the data is passing through the system, the cost of this convenience is at most one bus cycle. At 33MHz this translates to a mere 30ns. Given the tremendous savings on transmission time and making the system self-configuring, this small extra delay is well justified.

3.3 MATE Connectivity

OVNI-RTDNS[3] employed point-to-point connections. Within the implementation, this allowed a maximum of two links per machine. As a result, the potential network connectivity was severely limited.

In OVNI-NET however, the links between subsystems are abstracted. As such there is no more limit (except for the maximum data size) on the number of links between sub-systems, allowing a full graph connectivity if so desired. For example, the following illustration shows the largest possible connectivity of OVNI-RTDNS with four nodes next to OVNI-NET.

It has to be noted that several lines between subsystems are also possible with MATE, but here only single lines are shown for clarity. (Fig. 7)

[6] Except for one, the total number of nodes. Although also this could be inferred from the data as well but since it never changes at runtime in MATE, we choose not to.

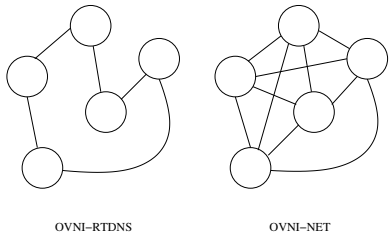

Figure 7: OVNI-RTDNS Point-To-Point vs OVNI-NET MATE Abstracted Mapping

4 Results

In the current implementation, the maximum number of three-phase links per solver node is set at 84 (6 32 bit words per link) and can be readily expanded to twice as many, pending optimizations. It has to be noted that the total number of links in the system is limited by the bandwidth of the central node.

Although ample, this does pose a limitation on the maximum system size in the current implementation of OVNI using MATE. However, future use of the full capabilities of PCI (66MHz, 64bit vs 33MHZ, 32bit used now) or upgrading to PCI Express can drastically alleviate this limitation.

Table 1 shows the performances of different interfaces, including OVNI-RTDNS and OVNI-NET, for the particular case of point to point communication.

I/O Interface	Dell P-MMX 233 MHz	Bus Width [bits]	Time needed for 3-phase Transmission Line
EPP/ECP	$0.4\,MB/s$	8	$57.6\,\mu s$
PCI-EPP/ECP	$1.0\,MB/s$	8	$24\,\mu s$
OVNI-RTDNS	$3.3\,MB/s$	16	$7\,\mu s$
OVNI-Net	$133\,MB/s$	32	$3.36\,\mu s$

Table 1: Point to point data transfer performances

Even tough there is already an important advantage in the communication time for the point to point case, the full advantage of the new hardware interface is seen when full connectivity is used between the solver nodes of the simulator, and when the full bandwidth of the hardware is exploited.

The comparison of available bandwidth is seen in Table 2. It has to be noted that all the OVNI-NET transmission times in these tables are for the complete OVNI-NET system, including the hub, running on a non-real-time version of Linux[7] in user-space. The OVNI-RTDNS system ran on the real-time PharLap RTX operating system.

transmission lines	OVNI-RTDNS	OVNI-NET
1 3-phase	$7\mu S$	$3.36\,\mu s$
2 3-phase	$14\mu S$	$3.72\,\mu s$
84 3-phase	$not\,possible$	$33.24\,\mu s$

Table 2: point to point bandwidth OVNI-NET advantage

The total transmission time in OVNI-NET is dependent on the total number of links in the system[8] plus an additional 30ns overhead for each solver node.

OVNI-NET system transmission time:

- Latency from solver node to PCI card
  1.5 μs
- Largest data block in the system
  33Mwords/s $\times$ data size
- Sum of all data in the system
  33Mwords/s $\times$ total data size
- Protocol overhead
  30ns per solver node
- Latency from hub system to central solver
  1.5 μs

As such, these point-to-point tables also reflect the communication time for a full-fledged MATE system since they only exclude the protocol overhead for the multiple solver nodes involved.

5 Conclusion

From the above results and considerations, we can state that OVNI-NET greatly enhances the flexibility of the OVNI Real-Time Power System Simulator while still maintaining its outstanding performance and full real-time capabilities. The addition of a shared-memory hub and distributed intelligence allowed the MATE concept to be directly mapped to the cluster interconnect itself, providing the ideal environment for this network solution.

Due to the adaptability of the system, it can be readily adapted to the older OVNI-RTDNS system if so desired[9], allowing it to profit from the enhanced interconnect performance.

REFERENCES

[1] J.R. Martí, L.R. Linares, Roberto Rosales, and H.W. Dommel, "Ovni: a full-size real time power system simulator," in *International Conference on Digital Power System Simulators*, Montreal, Quebec, Canada, 1997, pp. 1309–1317, ICDS'97.

[2] J.A. Hollman, J.R. Martí, and Jesús Calviño Fraga, "Implementation of a real-time distributed network simulator with pc-cluster," in *Parallel Computing in Electrical Engineering, 2000. PARELEC 2000. Proceedings. International Conference on*, 2000, pp. 223–227.

[3] J.A. Hollman and J.R. Martí, "Real time network simulation with pc-cluster," *Power Systems, IEEE Transactions on*, vol. 18, no. 2, pp. 563–569, 2003.

[7] Standard RedHat 9 kernel
[8] With a maximum of 84 per node and a maximum of 243 nodes in the entire system.
[9] Omitting the hub for even lower latency

[4] J.R. Martí, L.R. Linares, J. Calvino, H.W. Dommel, and J. Lin, "Ovni: an object approach to real-time power system simulators," in *Power System Technology, 1998. Proceedings. POWERCON '98. 1998 International Conference on*, 1998, vol. 2, pp. 977–981 vol.2.

[5] J.R. Martí, L.R. Linares, J.A. Hollman, and F. Moreira, "Ovni: Integrated software/hardware solution for real-time simulation of large powersystems," in *14th Power Systems Computer Conference*, Seville, 2002, PSCC'02.

[6] A. Semlyen and F. de Len, "Computation of electromagnetic transients usign dual or multiple time steps," *IEEE Transactions on Power Systems*, vol. 9, pp. 1274–1281, 1993.

[7] L.R. Linares and J.R. Martí, "Sub-area latency in a real time power network simulator," in *International Conference on Power System Transients*, Lisbon, 1995, pp. 541–545, IPST'95.

[8] F. Moreira, J.A. Hollman, L.R. Linares, and J. R. Martí, "Network decoupling by latency exploitation and distributed hardware architecture," in *International Conference on Power System Transients*, Rio de Janeiro, 2001, IPST'01.