A Network-based Communication Platform for a Cognitive Computer

Mostafa W. Numan, Jesse Frost, Braden J. Phillips, and Michael Liebelt

Centre for High Performance Integrated Technologies and Systems (CHiPTec)
School of Electrical and Electronic Engineering
The University of Adelaide
Adelaide, Australia
{mostafa.numan,jesse.frost,braden.phillips,michael.liebelt}@adelaide.edu.au

Abstract. *Street* is a reconfigurable parallel computer architecture. It executes a production language directly in hardware with the aim of realising advanced cognitive agents in a more energy efficient manner than conventional computers. Street requires frequent communication between many processing elements and to make this communication more energy efficient, a network-based communication platform, *StreetNet*, is proposed in this paper. It maps the processing elements onto a 2D mesh architecture optimized according to the data dependencies between them. A deadlock-free deterministic routing function is considered for this platform along with the concept of sleep period, analogous to human sleeping, to reorganize the placements of processing elements based on runtime traffic statistics. These mechanisms serve to reduce total network traffic and hence minimise energy consumption.

Keywords: Cognitive computer, computer architecture, networks-on-chip, mapping

1 Introduction

Street is a reconfigurable, flat, parallel architecture designed for symbolic cognitive workloads [6]. The goal of Street is to find a new computer architecture that can take advantage of the huge number of transistors in modern integrated circuits to achieve advanced cognitive computation in real time, but with much lower power consumption than current computers. It is designed to use in real time embedded implementations of artificial general intelligence, exemplified by the plethora of potential autonomous robotics applications. The new machine is very different from conventional computers, consisting of many simple processing elements executing and communicating in parallel. A bus-based interconnect performs well in production systems with a small number of processing elements, or when groups of dependent productions are mapped to the same processor [1], however it does not scale well for more frequently interacting processing elements. For chips with a large number of processing elements, network based

communication provides better scalability, and is seen as the most efficient solution [13]. In this paper, a network-based communication platform, *StreetNet*, is proposed for efficient communication among the processing elements of Street.

2 Street

Street executes a parallel production language directly in hardware. This language, which we call Street Language, is inspired by Forgy's OPS5 [5] and the languages used in the Soar [10] and ACT-R [3] cognitive architectures. However Street Language is different from all of these. Street is asynchronous, with no global match-select-act cycle as found in traditional production systems. This asynchronous model provides the best opportunity to parallelise traditional production systems in application level [2].

2.1 Street Language

An intelligent system is implemented using a set of production rules written in Street Language [6]. Each production rule is an *if-then* statement: *if* a specified pattern exists in working memory, *then* the rule makes some changes to working memory. Working memory is a set of tuples called working memory elements (WMEs). Each WME has one or more elements called attributes. For instance, the WME `(ID17 source ID2)` has 3 attributes: `ID17`, `source`, and `ID2`. Here is a simple example of working memory of just 3 WMEs:

`{(ID17 name Torrens), (ID17 source ID2), (isCounted ID5)}`

Each production rule consists of a left hand side (LHS) of one or more condition elements (CEs), and a right hand side (RHS) of one or more actions. In the example in Fig. 1, `(<p> type dog)` is a CE and `(<p> isOld)` is an action. A complex cognitive agent would consist of thousands of production rules operating on symbolic and numeric data in working memory.

```
st {oldDogs
    (<p> type dog)      // condition elements
    (<p> age (<a> > 7))
-->
    (<p> isOld)         // actions
}
```

Fig. 1. A Street Language production rule

A subset of working memory that satisfies all of the CEs in a production rule with consistent variable assignments is called an instantiation of the rule. So instantiations of the rule above will be pairs of WMEs in working memory such as: `{(pet1 type dog), (pet1 age 8)}`. Note that the first attributes must be

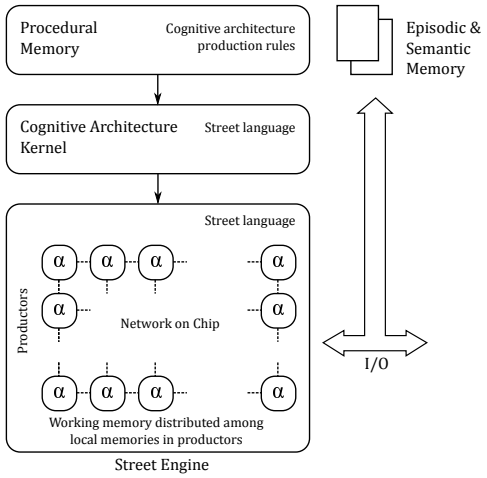

Fig. 2. Hardware/software stack on *Street* (adopted from [6])

the same as they were specified by the same variable `<p>`. The WMEs must join on any shared variables. The actions of a production rule are performed for each new instantiation of the rule. An action such as `(<p> isOld)` adds a WME to working memory. Actions can also remove WMEs from working memory. This change in working memory may cause other production rules to instantiate, and all new instantiations are executed.

2.2 Street Architecture

The Street architecture consists of a large number of identical and simple microcoded processing elements (PEs) with a single production rule assigned to each. A PE contains a controller, a block of content-addressable memories (CAMs) and an Arithmetic Logic Unit (ALU). The PEs communicate using tokens to notify each other of changes to working memory. The local memory of a PE stores just the subset of working memory that may lead to instantiations of its rule. Each PE matches the associated production rule against its own local memory with an algorithm similar to TREAT [12]. For each incoming token, a PE updates the contents of its local memory, finds new instantiations (*match*), and outputs tokens corresponding to the rule's actions (*act*). The controller coordinates the PE's match-act cycle. Fig. 2 shows Street architecture executing an agent using a symbolic cognitive architecture.

3 StreetNet: The Communication Platform

When a PE produces tokens these are transmitted to other PEs and may cause new rule instantiations and yet more tokens. There may be a large amount of token traffic between PEs or small clusters of PEs so efficient data interconnect is

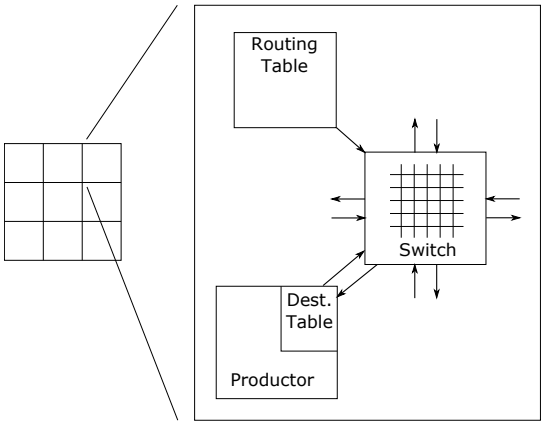

Fig. 3. A 3×3 StreetNet structure

required. For a device with a large number of PEs a network based interconnect, named the *StreetNet*, is proposed in this paper.

3.1 Network Architecture

A regular tile-based 2D network architecture is considered for StreetNet. Each tile contains a single PE and router, however adjacent tiles may be linked to share memory resources (discussed below in 3.6). Every PE has a *destination table* generated from the dependency graph. It lists the desired destinations and corresponding tokens to those destinations. Each router is connected to its local PE and four neighbouring tiles. Each router also has a routing table that is checked for destination. The tokens are broken into packets and forwarded to the neighboring tile towards the destination. A crossbar switch is used as the switching fabric in the router. Fig. 3 shows the structure of a StreetNet.

3.2 Dependency Graph

The mapping of PEs onto network architecture is based on a *dependency graph*. A dependency graph is a directed graph, where each vertex p_i represents a PE. Directed arcs represent non-zero communication paths between two PEs and are assigned a weight characterising the communication rate between the PEs. Fig. 4 shows an example of a dependency graph of nine PEs. This graph is used to map the PEs onto the network so that the most dependent PEs are placed close together in the expectation this will reduce communication latency and power consumption. The PEs are sorted by total incoming and outgoing traffic that was recorded during runtime, and mapped in this order. This is useful since the positions of the PEs with high traffic requirement have higher impact on the overall energy consumption. This dependency graph is updated during a sleep period (described in subsection 3.5) depending on runtime traffic statistics.

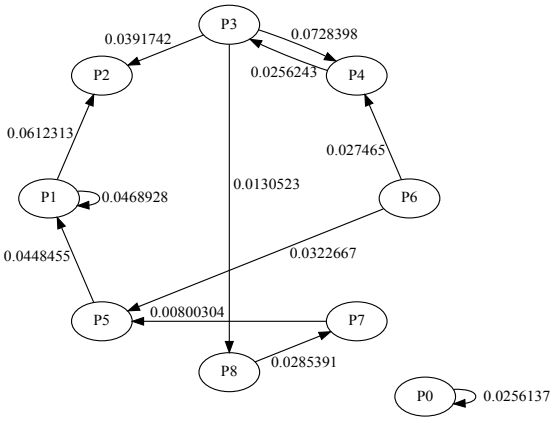

Fig. 4. An example of dependency graph with nine PEs

3.3 Mapping Techniques

In [8], an energy aware mapping technique is proposed for networks-on-chip (NoC) with a regular architecture. This technique is adopted in the StreetNet. The average energy consumed in sending one bit of data from a tile to a neighboring tile is calculated as

$$E = E_S + E_B + E_L \tag{1}$$

where E_S, E_B and E_L are the energy consumed at the switch, buffer and link. Since the energy consumed for buffering is negligible compared to E_L [8], (1) becomes

$$E = E_S + E_L \tag{2}$$

Now, if the bit traverses n hops to reach tile t_j from tile t_i, the average energy consumption is

$$E_{t_i,t_j} = n \times E_S + (n-1) \times E_L \tag{3}$$

For a system that involves a large number of processing elements, it is important to adopt efficient mapping and routing techniques so that total energy consumption is minimized and communication traffic does not exceed available bandwidth. Two different mapping techniques have been considered in this work: one is based on Simulated Annealing (SA) and the second is based on Branch-and-bound (BB) technique.

Simulated Annealing based Mapping Simulated Annealing (SA) [9] is a well known technique for solving optimization problems. It effectively optimizes solutions over large state spaces by making iterative improvement. It has a concept of *temperature* which is initially very high and keeps reducing in every step until it reaches the minimum temperature. For each temperature, it starts with

```
current_temp=MAX_TEMPERATURE;
previous_cost=INITIAL_COST;
current_mapping=randomMapping();
current_cost=cost(current_mapping);
do{
    while(attempts<MAX_ATTEMPTS){
        current_mapping=makeRandomTileSwap(current_mapping);
        new_cost=cost(current_mapping);
        ΔC=new_cost - current_cost;
        if (random(0,1)≤exp(-ΔC/current_cost×current_temp))
            current_cost=new_cost;
        else
            current_mapping=rollbackTileSwap(current_mapping);
    }
    if(toleranceTest(previous_cost,current_cost)‖current_temp≤MIN_TEMPERATURE)
        done=1;
    else{
        previous_cost=current_cost;
        current_temp=getNextTemperature(current_temp);
    }
}while(!done);
```

Fig. 5. Simulated annealing algorithm for PE mapping

a random feasible solution and searches for better solutions with lower cost. This is a greedy algorithm. A tolerance test is done in every iteration to check if the cost is changing insignificantly over the last few temperatures or the temperature reaches a certain limit. Eventually, when the temperature goes below the minimum limit, it defaults to the greedy algorithm only. Fig. 5 illustrates the SA algorithm for PE mapping.

Branch-and-bound based Mapping In this mapping technique, a search tree is generated that represents the solution space. The root node corresponds to the state where no PEs are mapped. Each internal node represents a partial mapping and each leaf node is a complete mapping of PEs onto tiles. Fig. 6 shows the search tree of the solution space. For example, the node labelled $t_0 t_{n-1} t_1 ... t_{n-2}$ represents the placement in which PEs P_0, P_1, P_2, ... P_{n-1} are mapped to tiles t_0, t_{n-1}, t_1, ... t_{n-2} respectively.

The branch-and-bound (BB) mapping finds the solution node which has the minimum cost. The cost of mapping is calculated by the total energy consumed by all the PEs that are already mapped. The PEs are initially sorted based on their traffic demand obtained from the dependency graph. As the PEs with higher traffic demands dominate the overall energy consumption, they are mapped first to the unoccupied tiles to generate new child nodes. Each node has a table that stores the routing paths between its occupied tiles. When a child node is generated, the table from its parent node is inherited, and the routing

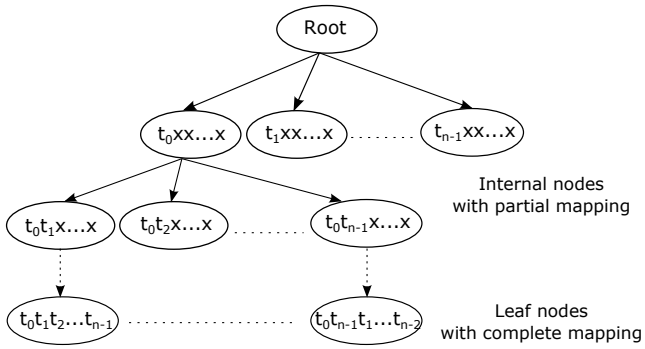

Fig. 6. Search tree of solution space

path to the new tile is added to the table. Then, each of the newly generated child nodes are examined to see if it is possible to generate the best leaf node later. The upper and lower bounds of the nodes are calculated to detect candidate optimal nodes. The upper bound of a node is the value that is no less than the minimum cost of its leaf nodes; the lower bound is defined as the lowest cost that its descendant leaf nodes can possibly achieve. If the cost or lower bound of a node is higher than the lowest upper bound that is already found so far, it is deleted without any expansion, because it is guranteed that the node cannot lead to the best mapping solution. The lower and upper bounds are updated after every step. All the nodes are traversed this way, and finally the node with minimum cost is accepted as the best mapping.

3.4 Routing

Wormhole routing is considered in this work because of limited buffering resources. In wormhole routing, packets are broken down into flow control digits (*flits*) and the flits are routed over the network in a pipelined fashion. The header flit contains routing information and leads the packet to the destination. Deterministic dimension-ordered routing is chosen in this work. In comparison to adaptive routing, deterministic routing requires less buffering space since no ordering is required for received packets [4]. Moreover, deterministic routing algorithms are livelock free. We use the west-first turn model [7] that prohibits north-to-west and south-to-west turns to make it deadlock-free.

3.5 Sleep Period

Street stores traffic statistics during runtime which are used periodically to update the dependency graph. The assignments of rules to PEs and their location within the network are refined based on this so that total traffic energy consumption is minimized. During this period, the execution is paused, analogously to human sleeping [11], and the memory contents as well as destination tables

are re-arranged between PEs. However, the network structure and routing table remain unchanged as they are the fixed components of StreetNet. After the sleeping period, it continues to operate as before, but with improved performance.

3.6 Clustered PEs

If memory content exceeds the capacity of a PE, the flat architecture of Street's tiles allows the PE to use the memory resources of adjacent unused tiles. One of the PEs in the group of tiles acts as a *master PE*, and the destination table attached to it remains active. This master PE acts as source or destination of packets, the other routers of the cluster are used to forward the packets only. The PEs inside the cluster communicate through a local bus. If there are no unused adjacent tiles, the rule is moved to a new PE with free adjacent tiles and the contents are transferred during a sleep period.

4 Experiments

StreetNet was tested over network architectures ranging from 4 to 196 processing elements. As we have not yet developed any large-scale agents, dependencies were artificially created. For every architecture, 10 random dependency sets were generated. Each dependency set was used for mapping using both simulated annealing and branch-and-bound techniques. Fig. 7 shows the total energy consumption comparison between the mapping techniques. This shows that SA based mapping performs slightly better than BB based mapping, but when compared in terms of computation time, the latter significantly outperforms the former, as seen in Fig. 8. This indicates that BB mapping works much faster than SA mapping but the energy savings at execution time from the SA mapped solution may warrant the extra mapping time. StreetNet creates routing tables for all the routers as well. Since deterministic dimension-order routing has been considered in this work, routing tables do not change over time. As a result, ordered packet delivery and simplicity are ensured.

5 Conclusion

In this paper, a network-based communication platform, StreetNet, is proposed for the Street cognitive computer in which the PEs are mapped onto a 2D mesh architecture. The mapping is derived from a dependency graph that is obtained from runtime traffic statistics. This work introduces the concept of a periodic sleep period, during which the placement of the PEs is updated to improve overall energy efficiency. Branch-and-bound and simulated annealing based mapping techniques are discussed here. Experiments indicate that branch-and-bound mapping significantly outperforms simulated annealing based mapping when compared in terms of computation time. Clustered PEs are implemented in this work to accommodate large memories. The number and orientation of

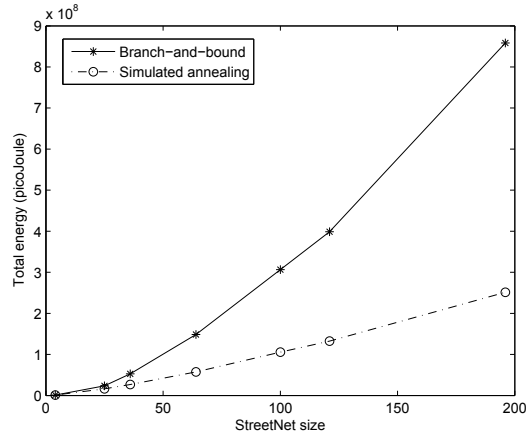

Fig. 7. Energy comparison between two mappings

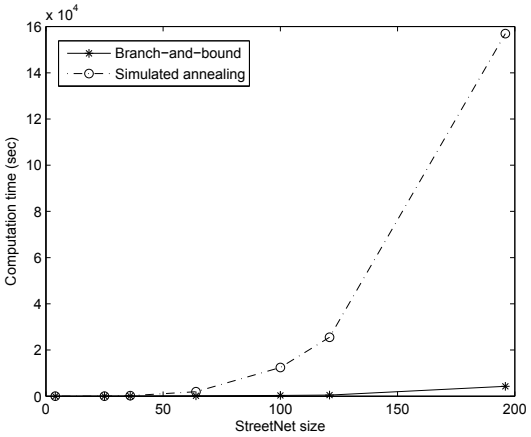

Fig. 8. Computation time comparison between two mappings

the tiles of a clustered PE will affect the overall energy consumption of the system to an extent that is yet to be investigated. Moreover, we plan to implement the mapping algorithm using Street Language so that Street can update PE mapping using its own resources.

References

1. Amaral, J.N.: A parallel architecture for serializable production systems. Ph.D. thesis, Citeseer (1994)

2. Amaral, J.N., Ghosh, J.: Parallel Processing for Artificial Intelligence 1, chap. Speeding up production systems: From concurrent matching to parallel rule firing, pp. 139–160 (1993)
3. Anderson, J.R.: ACT: A simple theory of complex cognition. American Psychologist 51(4), 355 (1996)
4. Bhattacharyya, S.S., Deprettere, E.F., Teich, J.: Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation (2003)
5. Forgy, C.L.: OPS5 user's manual. Tech. rep., Computer Science Department, Carnegie-Mellon University (Jul 1981)
6. Frost, J., Numan, M.W., Liebelt, M., Phillips, B.J.: A new computer for cognitive computing. In: 14th IEEE International Conference on Cognitive Informatics & Cognitive Computing. Beijing, China (July 2015)
7. Glass, C.J., Ni, L.M.: The turn model for adaptive routing. SIGARCH Comput. Archit. News 20(2), 278–287 (Apr 1992), http://doi.acm.org/10.1145/146628.140384
8. Hu, J., Marculescu, R.: Energy- and performance-aware mapping for regular NoC architectures. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 24(4) (April 2005)
9. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (May 1983)
10. Laird, J.E.: The Soar Cognitive Architectures. The MIT Press, Cambridge (2012)
11. Landmann, N., Kuhn, M., Piosczyk, H., Feige, B., Baglioni, C., Spiegelhalder, K., Frase, L., Riemann, D., Sterr, A., Nissen, C.: The reorganisation of memory during sleep. Sleep Medicine Reviews (0), – (2014), http://www.sciencedirect.com/science/article/pii/S1087079214000264
12. Miranker, D.P.: TREAT: A New and Efficient Match Algorithm for AI Production Systems. Morgan Kaufmann (2014)
13. Richardson, T., Nicopoulos, C., Park, D., Narayanan, V., Xie, Y., Das, C., Degalahal, V.: A hybrid SoC interconnect with dynamic TDMA-based transactionless buses and on-chip networks. In: VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design., 19th International Conference on (Jan 2006)