

Power optimization of interconnection networks for transport triggered architecture

Xuemi Zhao and Zhiying Wang

(Computer School, National University of Defense Technolog, 410073, P. R. China)

Abstract-Transport triggered architecture (TTA) has been shown to provide an efficient way to design application specific instruction set processors. However, the interconnection network of TTA is based on simple-bus, which consumes much extra power for specific data transport. In this paper, we employ segmented-bus to solve this problem. How to partition the buses lies on the placement of macro blocks, and the manual creation of an optimal placement would be a labor-intensive process, leading to high design costs. Instead, we propose automatic approach to place the macro blocks specialized to given applications. For several real life cryptographic applications, a factor of 1.14 in bus power reduction can be achieved while maintaining the same performance.

Keywords: Application specific instruction-set processor; TTA; Segmented-bus; Low power design

1. Introduction

A variety of factors is making it increasingly difficult and expensive to design and manufacture traditional application specific integrated circuits (ASICs) as feature size decreases. Design tools are finding it difficult to handle the complexity and electrical design challenges posed by each new technology generation. The next consequence is lowered design productivity and rising manufacturing costs. This has started a significant move towards the use of programmable solutions. For the platform manufacturer, programmability provides higher volume to amortize design and manufacturing costs. For the application implementer, programmability provides a lower risk and shorter time-to-market implementation path. An application instruction set processor (ASIP) is the solution [1]. Of course, the flexibility provided by ASIP comes with area and power overhead.

Very long instruction word (VLIW) architectures have been widely used in ASIP design due to their modularity and scalability. VLIW architectures exploit

the instruction level parallelism by executing operations in parallel in concurrently operating functional units [2, 3]. There are even VLIW architectures, which support customized, application-specific function units [4]. However, the data path of VLIW is complex when scaled to very high performance levels as required by many digital signal processing applications. The hardware required for bypassing values becomes too complex when adding many functions. And VLIW architectures have also been criticized for their requirements for read/write ports in the register file.

An alternative architecture where the drawbacks of VLIW architecture can be avoided is transport triggered architecture (TTA) [5] where a program describes only the operand transfers between the computational resources. This architecture supports splitting conveniently to overcome the bottleneck caused by the register file. In TTA, interconnections of functional units are multi-bus based, which solve the drastically increasing complexity of bypass. In addition, such a mirrored programming paradigm allows new scheduling and allocation techniques to be used in high level language compilers. TTA processors can be designed with a MOVE framework, a toolset that provides a semi-automatic design process [6].

However the multi-bus network of TTA is constructed with simple buses. Every function unit connects with the buses with the so-called sockets. A simple bus is slow and consumes much more extra power due to large capacitive load caused by the long physical length. A solution to the above mentioned problems is a reconfigurable segmented-bus network. Contribution [7] compared different bus structures for TTA, which includes tri-state bus, AND/OR bus, multiplexer bus and segmented multiplexer bus. But their evaluation results were only from synthesis tools, the actual data transports for application and physical characters were not considered, so its conclusion was not accuracy. In this work, we will revisit the segmented-bus network.

Specialized reconfigurable segmented buses, while beneficial in theory, would be impractical in practice if they had to be created by hand for each groups of applications. Each of these optimized reconfigurable interconnections may be quite different, depending on the application set or sets desired. Unfortunately, this contributes significantly to the design costs of the hardware. Therefore, this work automatically generate these custom reconfigurable segmented buses based on an input set of applications, and by doing so, greatly decrease the cost of new architecture development.

This paper is organized as follows. We start by introducing transport triggered architecture and its design framework in Section 2. In Section 3, the idea of reconfigurable segmented bus network is demonstrated. Section 4 describes the automatic generation flow and the algorithms used. Section 5 gives experiment results, and Section 6 draws some conclusions.

2. Transport Triggered Architecture

TTAs are a lot like VLIW architectures in that they can perform multiple operations per cycle. The principle difference is the way in which operations are programmed and executed. Whereas in VLIWs instructions specify RISC type operations, in TTAs they specify data transports. Operations are triggered as a side effect of these data transports: the destination of a transport implicitly specifies the kind of operation that will be performed on the data.

TTAs are organized as a set of functional units

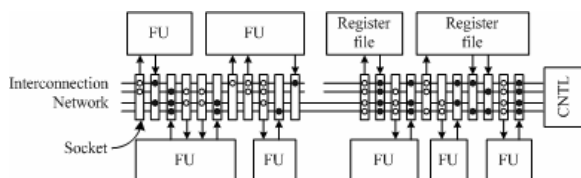


Figure 1. TTA concept

(FUs) and register files (RFs), which are connected by some kind of interconnection network. RFs contain general purpose registers. Fig 1 shows a TTA, containing seven FUs, two RFs and six transport buses. FUs and register files are connected to the interconnection network with one or more input and output sockets. Input sockets contain multiplexers, output sockets de-multiplexers. The network controller (CNTL) controls the pipelining of the network.

A TTA program only specifies transports, so it supports only one type of operation: the *move* operation from source to destination. Sources and destinations are any register, and any FU input and

output. In general a three-address operation translates into three *moves*, e.g.:

$$\text{add } r3, r2, r1 \Rightarrow r1 \rightarrow O1\text{add}; r2 \rightarrow O2\text{add};$$

$$R\text{add} \rightarrow r3$$

The notation used is: O1add and O2add are the operand registers of the adder, and Radd denotes the result of this adder. TTAs can be easily made fully programmable. This requires support for control flow operations and conditional execution. Control flow operations can be implemented by making the program counter a visible source and destination location of the instruction fetch unit. Compared to VLIW, TTA has such hardware advantages as trivial decoding, transport efficiency, register efficiency and FU splitting etc. From compiler's view, TTA has some specific code generation optimizations, i.e. software bypassing, dead result move elimination, operand sharing, socket sharing and more scheduling freedom.

The MOVE framework is an environment containing a set of software tools for designing ASIPs. It provides a semi-automatic design process shortening the design-time. The framework exploits the scalability, flexibility, and simplicity of TTA. The design space explorer searches for a processor configuration, which yields the best area/performance ratio for a given application. Hardware resources of the processor, such as the number and type of buses, FUs and RFs and there connectivity, are described in an architecture description file. The design space explorer optimizes first the hardware resources. An architecture configuration fulfilling the cost and performance requirements is then chosen by the designer for connectivity optimization where unnecessary connections are removed. A simple-bus interconnect topology can simplify the design, but it consumes much more redundant power. Segmented buses are widely used to overcome this shortcoming.

3. Segmented-bus networks

A segmented bus is a bus which is partitioned to two or more segments. Each segment acts as a normal bus between modules that are connected to it and operates parallel with other segments. Segments can be connected dynamically to each other in order to establish connection between modules located in different segments. The concept of segmenting the buses has been proposed in the past, mainly for multi-computer architectures [8, 9, and 10]. More recent approaches [11, 12, and 13] place a segmented bus in the context of a single-chip device. Especially, contribution [14] discussed segmented buses systematically in theory and implementation.

The first step in such a design effort is to organize a communication scheme which will allow the components of a system efficiently transfer data over the shared resource, the bus. Due to the segmentation of this resource, parallel transaction can take place, thus increasing the performance. When parts of the segmented bus are not involved in transactions, the passive segments, they are isolated from the rest of the bus. Hence the active segments, the ones that currently transfer data, offer a faster operation and reduced energy per transferred bit. This is because the capacitance of the bus is reduced only to the sum of parasitic capacitance caused by the physical length of the currently used bus segments.

3.1 Applying segmented buses to TTA

Previous works on TTA have organized communication scheme. How to applying segmented-buses to TTA can be illustrated as Fig.2 shows. The interconnection of Fig.2 (a) is a simple buses network, and the sockets have two possible configurations: connected or unconnected. We partition the buses with bus connectors at the position of sockets and get Fig.2 (b), and there are four possible conductivities at every

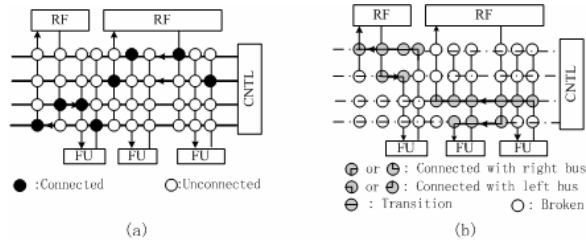


Figure 2. Applying segmented buses to TTA: (a) Simple-bus network; (b) Segmented-bus network

socket: connected to the left bus, connected to the right bus, transition and unconnected. When there are four data transports in a specific cycle. Full lengths of buses are active in a simple buses network, but only essential parts of the buses are active in a segmented buses network, and the remains are idle, which are dashed in Fig.2 (b). We can get that the active length of these four data transports is reduced highly, and the energy consumed by the buses can be reduced correspondingly.

The major overhead of this methodology comes from bus connectors. Connectors are implemented using a few gates, the overhead they will impose will be small taking into account the increasing dominance of interconnect in power and delay as technology scales. Also, the impact of the control wires for the connectors is marginal compared to the buses which are usually composed of 16 or 32 wires. Although it is clear that some overhead is introduced by the addition of

connectors on the bus, this overhead can be very limited. Next we will discuss how to route data transports on segmented buses.

3.2 Routing problem

Because each move slot of one instruction corresponds to a bus, routing for a special move is to open the bus connectors between the source socket and the destination socket. If two moves in the same instruction have the same source, they can share one bus, and the length is the maximum length. The routing can be done by the instruction decoder in hardware, and the software tool chain of TTA to generate the parallel code does not need any modification.

4. Automatic generations of the optimal segmented-bus network

We segment the bus at the position of sockets, where the ports of FUs connect to the network. The actual segmented bus structure lies on the physical placement of macro blocks. Our target is, according to specific application, the total data transport length is minimal. And the total length can be calculated as Equation (1):

$$L_{TT} = \sum_{i=1}^{N_i} \sum_{j=1}^{N_{im}} l_{ij} \quad (1)$$

where N_i is number of instructions executed for the application, N_{im} is the corresponding number of moves for the i -th instruction, l_{ij} presents the data transport length of the j -th move in the i -th instruction. The l_{ij} can be achieved as Section 3.2 describe and can be recorded by the architecture simulator.

All the macro blocks are organized on two rows, every macro block have the same height, and its width is proportional to its area. Finding the optimal placement manually is labor-intensive and will lead to high costs. Here we utilizes a simulated annealing algorithm [15] to find the optimal placement automatically, the algorithm is commonly used in FPGA placement to assign netlist instances to physical computation units, and standard cell placement to determine locations for actual physical cells.

This algorithm operates by taking a random initial placement of elements, and repeatedly attempts to change the location of a randomly selected element. The change is accepted if it improves the overall cost of the placement are sometimes accepted. The probability of accepting a non-improving change is governed by the current ‘temperature’. At the beginning of the algorithm, the temperature is high,

allowing a large proportion of bad changes to be accepted. As the algorithm progresses, the temperature decreases, and therefore the probability of accepting a bad move also decreases. At the end of the algorithm almost no bad moves are permitted. Here, a ‘change’ can be either changing the position of a macro block or change the port position of a block. The cost metric is based on the total data transport length L_{TT} .

The guidelines presented for VPR [16], a place and route tool for FPGAs, govern the initial temperature calculation, number of moves per temperature, and cooling schedule. These values are based on N_{blocks} , the number of macro blocks. The initial temperature and number of moves per temperature are derived from this value. The cooling schedule specified by VPR is also used, where the new temperature T_{new} is calculated according to the percentage of moves that were accepted at the old temperature T_{old} .

5. Experiment results

In order to compare these two interconnect structures, a bottom-up evaluation methodology is developed based on a combination of accurate physical models with a cycle-based architecture simulator.

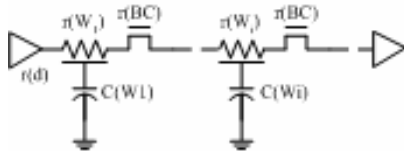
5.1 Energy and delay model

First, we derive analytical models for both the energy-consumption and network-delay. A universal energy model for the interconnect network is easily derived as in equation (2):

$$E = (C_{wire} + C_{BC}) \cdot V^2 = K_L \cdot L + K_{BC} \cdot N_{BC} \quad (2)$$

where K_L is the energy per unit wire length, and K_{BC} is the energy of a bus connector, and N_{BC} is the number of bus connectors. In simple-bus network, L is the whole wire length, and N_{BC} is zero. In segmented-bus network, L and N_{BC} represent active wire length and active bus connectors number separately.

We use the Elmore model [17] to estimate the delays.



$r(d)$: resistance of the driver
 $r(W_i)$: resistance of the i -th wire segment
 $C(W_i)$: capacitance of the i -th wire segment
 $r(BC)$: resistance of the switch
 $C(i)$: subtotal capacitance after the i -th switch

Figure 3. Delay model of the segmented bus

Equation (3) is employed to calculate the simple-bus network.

$$t_{p(simbus)} = r(in) \cdot C_{total} + 0.5 \times r(wire) \cdot C_{total} \quad (3)$$

where $r(in)$ is the equivalent resistance of the input driver, C_{total} is the total capacitance due to wire and load, and $r(wire)$ is the wire resistance.

For the segmented-bus structures, the Elmore delay model results in the following equation,

$$t_{p(segbus)} = r(d) \cdot C_{total} + \sum_{i=1} \{r(W_i) \cdot [0.5 \cdot C(W_i) + C(i)] + r(BC) \cdot C(i)\} \quad (4)$$

Symbols used in Equation (4) are explained in Fig.3.

Using the analytical models presented above, delay for the networks of specific processors can be calculated directly. The actual power consumption still lies on the statistic information by the simulator.

5.2 Evaluation results

For each macro block used in the target architectures, the reference area was obtained by synthesizing the design using a 0.18um 1P6M stand cell library with 1.8V supply voltage. The maximum delay of these blocks was used as the requested clock period in the estimations of the MOVE design framework. Wire parameters about resistance and capacitance are calculated on average of the six metal layers.

Three different applications were selected to compare the power results of the segmented-bus networks to the simple-bus interconnections, and they are listed in Table 1. The three applications are all from cryptography in the real world. The first application is security hash which composes MD5, SHA-1, and SHA-256, each algorithm makes digest of 1Mbit raw data separately. The second application is symmetric cipher algorithms which compose DES, 3DES, AES and RC6, and they encrypt 512Kbit raw data separately. The third application is a public key algorithm, i.e. RSA, which is a widely used public key algorithm, and here it does a 1024-bit signature with Chinese remainder theory. The dominating operation in the first and second application is bit manipulating, and multiplication is the key operation in RSA.

Table 1: Three applications used to test segmented-bus networks

Application	Algorithms
Security hash	MD5, SHA-1, SHA-256
Symmetric cipher algorithms	DES, 3DES, AES, RC6
Public key algorithm	RSA-1024

The initial un-segmented architectures were all achieved by the MOVE framework with bus width = 32. The simulator was modified to route transports on segmented buses and record active transport length and

active bus connectors' number for each move operation. Macro blocks' height h is an alterable parameter, and the width of each block can be calculated by area and h . In addition, we assume that all the sockets of modules are evenly distributed.

For the three cryptographic applications, the power consumption of simple-bus network and segmented-bus network are calculated according to Equation (2) and the statistical information from simulation. The results are shown in Fig.4 (a). Note that power consumptions of security hash and symmetric cipher are both the average value of the algorithms included. The absolute power consumption value of the network lies on three elements: the application, the height h and network type. For special application, when h steps up from 150um to 250um, the absolute power consumption value decreases as the bus wire length is shortened. For the same application and the same h , a

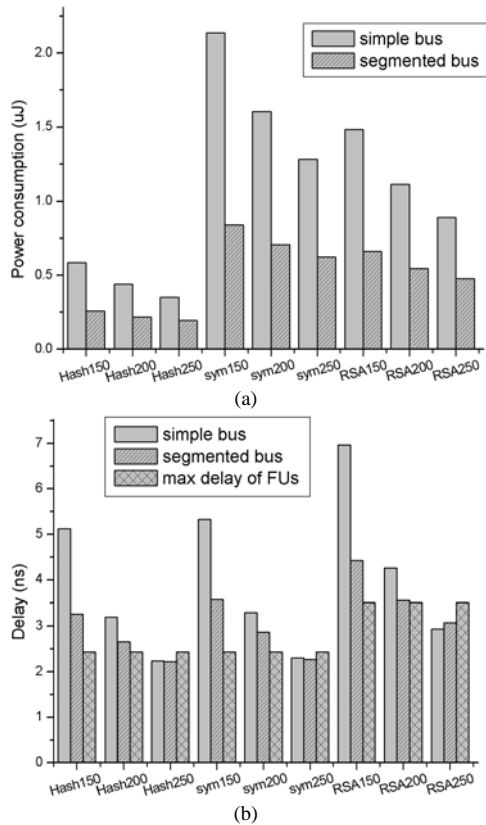


Figure 4. Evaluation results: (a) Power consumption; (b) Delay

segmented-bus network consumes much smaller power than a simple-bus network. But the consumed power ratio of a simple-bus network to segmented-bus network decreases with h steps up, the reason is that power consumed by the bus connectors of segmented buses is constant. And that ratio varies for different

application. On average, a factor of 1.14 in bus power reduction can be achieved.

For every ASIP, the delay of each interconnection network is calculated with Equation (3) and (4), and Fig.4 (b) shows the results. For the same application, the delay of simple-bus networks decreases with $O(1/h^2)$ as h increases, but the decreasing speed of segmented-bus networks is much smaller because the delay incurred by the bus connectors is constant. For RSA application, when h equals 250um, the delay of a segmented-bus is bigger than the corresponding delay of a simple-bus, but it is smaller than the maximum delay of function units (FUs). And in other cases, the segmented-bus networks are all faster than the corresponding simple-bus networks. So employing segmented buses will not affect the performance.

Above all, by using a segmented-bus network, a factor of 1.14 in power reduction of interconnections of TTA can be achieved on 0.18um 1P6M standard cell technology, while maintaining the same performance.

6. Conclusions and future work

This paper described the low power enhancement of transport triggered architecture through a segmented-bus network. We broke the bus at the position of sockets. The routing problem is simple and can be done in the decoder, and the design framework especially the software tool chain does not need any modification. The segmented buses are concerned with the placement of macro blocks. To get the minimum power consumption of the network for specific application, we employed a heuristic algorithm, i.e. simulated annealing, to place the macro blocks automatically. Experiment results with several practical applications show that an optimized segmented bus network saves 53% in power consumption, while maintaining the same speed compared to the simple-bus network.

However, this work assumes that ASIPs are implemented with standard cells, and the geometry character of computational components can be adjustable. ASIPs often employ such IP modules as multipliers, register files to get higher hardware efficiency. But the geometry character of IP module is irregular. For this requirement, our future research will concentrate on hierarchical segmented-bus network.

References

- [1] K. Keutzer, S. Malik, and A. R. Newton. From ASIC to ASIP: The Next Design Discontinuity. In IEEE

- International Conference on Computer Design, pp. 84-90, September 2002.
- [2] P. Faraboschi, G. Brown, J. A. Fisher, G. Desoli, and F. Homewood. Lx: A technology platform for customizable VLIW embedded processing. In Proceedings of the 27th Annual International Symposium on Computer Architecture, pp. 203~213, 2000.
- [3] B. Middha, V. Raj, A. Gangware, A. Kumar, M. Balakrishnan, and P. Jenne. A Trimaran based framework for exploring the design space of VLIW ASIPs with coarse grain functional units. In Proceedings of the 15th International Symposium on System Synthesis, pp.2~7, 2002.
- [4] D. Jain, A. Kumar, L. Pozzi, and P. Jenne. Automatically customising VLIW architectures with coarse grained application-specific functional units. In Proceedings of the 8th International Workshop on Software and Compilers for Embedded Systems, pp.17~32, 2004.
- [5] H. Corporaal. Transport Triggered Architectures: Design and Evaluation. PhD thesis, Delft Univ. of Technology, September 1995.
- [6] H. Corporaal, M. Arnold. Using transport triggered architectures for embedded processor design. Journal on Integrated Computer-Aided Engineering, Vol.1998(1), pp.19~37, 1998.
- [7] R. Makela, J. Takala, and O. Vainio, "Analysis of Different bus Architectures for Transport Triggered Architectures", in Proceedings of 21st Norchip Conference, pp.56-59, 2003.
- [8] C. Katsinis. A Segmented-Shared-Bus Multicomputer Architecture. Ninth International Conference on Parallel and Distributed Computing and Systems (PDCS'97), 1997.
- [9] R. Krishnamurti, E. Ma. An Approximation Algorithm for Scheduling Tasks on Varying Partition Sizes in Partitionable Multiprocessor Systems. IEEE Transactions on Computers, Vol.41(12), pp.1572-1579, 1992.
- [10] C. H. Yeh, B. Parhami. Design of High-performance massively parallel architectures under pin limitations and non-uniform propagation delay. Proceedings of the 2nd AIZU International Symposium on Parallel Algorithms / Architecture Synthesis, pp. 58-65, 1997.
- [11] J. Y. Chen et al. Segmented Bus Design for Low-power Systems. IEEE Transactions On Very Large Scale Integration Systems, Vol.7(1), pp.25-29, 1999.
- [12] S. Lee and K. Choi. Partitioned-bus architecture synthesis based on data transfer model. The Sixth Asia Pacific Conference on Chip Design Language, Fukuoka, Japan, 1999.
- [13] J. Jeon, K. Choi. An Effective Synthesis Algorithm for Partitioned Bus Architecture. Institute of Electrical Engineer, Electronics Letters, Vol.35(6), pp.440-441, 1999.
- [14] W. B. Jone, J. S. Wang, H. Lu, L. P. Hsu and J. Y. Chen, Design theory and implementation for low-power segmented bus systems, ACM Transactions on Design Automation of Electronic Systems, Vol. 8(1), pp. 38~54, 2003.
- [15] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, Optimization by simulated annealing, Science, May 13, 1983, pp. 671-680
- [16] V. Betz, J. Rose, VPR: A new packing, placement and routing tool for FPGA research, International Workshop on FPGA, pp. 213-222, 1997.
- [17] J. Rubenstein, P. Penfield, and M. A. Horowitz, Signal delay in RC tree networks, IEEE Transactions on Computer-Aided Design, Vol. CAD-2, pp. 202-211, July 1983.