An energy-efficient Network-on-Chip for a heterogeneous tiled reconfigurable Systems-on-Chip

Nikolay Kavaldjiev, Gerard J. M. Smit
Department of EEMCS,
University of Twente, the Netherlands
{nikolay, smit}@cs.utwente.nl

Abstract

This paper proposes a Network-on-Chip architecture that offers high flexibility and performance. It is used in a System-on-Chip platform for future multimedia mobile devices. The network is packet switching wormhole network with virtual-channel flow control and source routing. The initial implementation results for a network router show its feasibility and size comparable with other available solutions.

1. Introduction

Future wireless multimedia terminals must meet several, often conflicting, requirements: high performance, excellent energy efficiency, high flexibility, support for Quality of Service (QoS), small size, and low cost. In addition to that, the time to market is a crucial factor. In the CHAMELEON[1] [1] project we defined an architecture that tries to fulfill most of these requirements. We propose a dynamically reconfigurable tiled heterogeneous System-on-Chip (SoC) platform for the future multimedia mobile devices. Three types of tiles can be distinguished on this architecture, depending on their level of configurability: bit-level reconfigurable tiles (e.g. FPGAs), word-level reconfigurable tiles (e.g. MONTIUM tiles [2]) and general-purpose programmable tiles (DSPs and microprocessors). The programmability of the architecture enables the system to be targeted at multiple applications. Achieving a high clock speed is not a primary goal for energy-efficient architectures. Instead, massive parallelism is used to achieve a high performance. Reconfigurable systems offer the flexibility and adaptiveness needed for future QoS based systems.

[1] This research is supported by the PROGram for Research on Embedded Systems & Software (PROGRESS) of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the technology foundation STW.

1.1. Reconfigurable architectures for wireless terminals

Until recently only a few reconfigurable architectures have been proposed for wireless devices. There are a few exceptions, for example, the Maia chip from Berkeley [3]. Most reconfigurable architectures were targeted at simple glue logic or at dedicated high-performance computing. Moreover, conventional reconfigurable processors are bit-level reconfigurable and are far from energy efficient.

However, there are quite a number of good reasons for using reconfigurable architectures in future wireless multimedia terminals:

- New emerging multimedia standards such as JPEG2000 and MPEG-4 have many adaptivity features. This implies that the processing entities of future wireless terminals have to support the adaptivity needed for these new standards.
- Although reconfigurable systems are known to be less efficient compared to ASIC implementations they can have considerable energy benefits. For example: depending on the distance of the receiver and transmitter or cell occupation more or less processing power is needed. When the system can adapt - at run-time - to the environment significant power-saving can be obtained.
- Standards evolve quickly; this means that future systems have to have the flexibility to adapt to slight changes in the standards. By using reconfigurable architectures instead of ASICs costly re-designs can be avoided.
- The cost for manufacturing chips is growing rapidly and in particular the mask cost increases with each technology generation. It is expected reconfigurable designs to have longer live compared with ASIC and thus the manufacturers can save on mask costs.
- Reconfigurability reduces risks: these systems can adapt to standards that may change during and after product development, and the time to market can be

shortened. It will enable systems with true multi-standard capability, which extends the product's life cycle and when a good architecture is chosen at design time it is possible to enhance the system later for applications it was not originally designed for.

Reconfigurability also has another more economic motivation: it will be important to have a fast track from sparkling ideas to the final design. Time to market is crucial. If the design process takes too long, the return on investment will be less.

We expect that the combination of *high-level design tools* and *reconfigurable hardware* architectures will enable designers to develop highly flexible, efficient and adaptive systems and applications for future multimedia terminals.

This paper is organized as follows: Section 2 introduces the architecture we foresee for future mobile devices and discusses the benefits of such architecture, Section 3 discusses the organisation of the inter-tile communication in the proposed architecture and gives details on a design of a network router, Section 4 refers to related projects dealing with reconfigurable architectures and on-chip networking, Section 5 gives initial results for the implementation of the on-chip network router and finally Section 6 gives some conclusions.

2. Chameleon tiled organization

Today's semiconductor manufacturing techniques provide a huge transistor budget and allow putting different types of functions on the same chip. As a result it is possible to build a complete system, including processors, memory and analogue parts on a single chip. The latter is called a system-on-chip (SoC) and usually is organised as an array of equally sized areas called tiles.

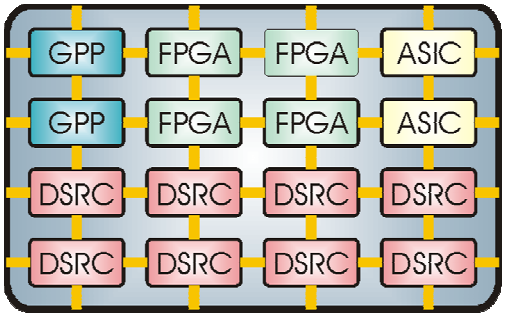

Figure 1. Heterogeneous tiled organization
(DSRC – domain-specific reconfigurable)

Each of the tiles in the system contains a computation unit and hence we have a parallel system, which correlates with the demand for high processing power. If using only general purpose computation units we may fail in fulfilling the energy-efficiency requirement. Although capable to handle virtually every task the general purpose processor (GPP) is far from energy-efficient. To perform efficient computations we need units specialised in certain application domains or for certain tasks. Such a unit could show much higher performance for the tasks it is optimised for while consuming much less energy than a general purpose processor. But the specialisation of the computational units could make the system quite inflexible. The solution could be to make some of the tiles reconfigurable. This approach is thought to be a good compromise between efficiency and flexibility.

Flexible and adaptive SoCs can be realized by integrating reconfigurable hardware parts of different granularities into heterogeneous reconfigurable SoCs (HRSoCs). Reusable HRSoCs are ever more appealing to industry due to the exponentially increasing mask costs for an ASIC. Such a HRSoC with reconfigurable hardware parts of different granularities is in line with the idea to match an algorithm with hardware that can execute it efficiently. In our view, such a heterogeneous HRSoC for mobile devices should contain domain specific processor tiles connected by an on-chip interconnection network.

When a processing entity is replicated a number of times on a chip, a tiled architecture is obtained. A feasible template for a future-proof architecture is constructed from processing tiles that do not grow in complexity with technology. Instead, as technology scales the number of tiles on the chip grows. An on-chip communication network then combines the tiles into a HRSoC. An on-chip network that routes packets has a higher bandwidth than an on-chip bus, as it supports multiple concurrent communications. The well-controlled electrical parameters of an on-chip interconnection network enable the use of high-performance circuits that result in significantly lower power dissipation, higher propagation velocity and higher bandwidth than is possible with conventional circuits.

An architecture in which processor tiles are connected by an on-chip network has a very modular design. The modularity reduces design complexity and makes it possible to employ the enormous transistor budget that will be available in future integrated circuits. The design of a single tile is relatively simple and therefore a lot of effort can be put in power optimisations on the physical level of integrated circuit design. A tiled approach also eases verification of an integrated circuit design, since the design of identical tiles only has to be verified once. The computational power in a tiled architecture scales linearly with the number of tiles. The more tiles there are on a chip, the more computations can be done in parallel (providing that the network capacity scales with the number of tiles).

References to memory display a high degree of locality. Temporal locality of reference relates to the observation that referenced data is often referenced again in the near future. Spatial locality of reference relates to the

observation that once a particular location is referenced, a nearby location is often referenced in the near future. Accessing a small and local memory is much more energy-efficient than accessing a big and far distant memory. Transporting a signal over a 1mm wire in a 0.05mm technology will require more than 50 times the energy of a 32-bit operation in the same technology. (The off-chip interconnect will consume more than a 1000 times the energy of a 32-bit operation!) A tiled architecture intrinsically encourages the usage of small and local in-tile memories. Exploiting the locality of reference principle extensively will improve the energy-efficiency substantially. The processor tiles in a tiled architecture can be viewed as a pool of resources.

Some algorithms might fit in a single processor tile, while others might require multiple tiles to achieve the required performance. Although tiles operate together in a complex system, an individual tile operates quite autonomously. In a tiled architecture every processor tile is configured independently. In fact, a tile is a natural unit for partial reconfiguration. Unused tiles can be configured for a new task, while at the same time other tiles are performing other tasks. That is to say, a tiled architecture can be reconfigured dynamically.

A tile processor might not need to run at full clock speed to achieve the required QoS at a particular moment in time. Also, when a task is pipelined on multiple processor tiles it is plausible that not all processor tiles in the pipeline require running at the same clock speed. An energy advantage can be gained when the processor tiles each have a configurable clock frequency. The energy consumption of a circuit varies linearly with its clock frequency, since energy consumption is the time rate of power consumption. A first order approximation of the dynamic energy consumption P in a circuit is given by the formula: $P = \frac{1}{2} CV^2 f$ where f is the operating frequency of the circuit, C the total capacitance and V the power supply voltage. Energy consumption varies linearly with clock speed and by the square of voltage. Reducing the supply voltage is far more lucrative than reducing the clock frequency. The supply voltage of a circuit cannot be reduced arbitrarily. The delay of a circuit increases significantly when the supply voltage is reduced. When the clock frequency is lowered, the supply voltage can be lowered as well. A configurable clock in combination with dynamic voltage scaling produces cubic reductions in dynamic energy consumption.

A thorough understanding of the algorithm domain is crucial for the design of an (energy) efficient reconfigurable architecture. The architecture should impose little overhead to run the algorithms in its domain. Inter-processor communication is in essence also overhead, as it does not contribute to the computation of an algorithm. Therefore, there needs to be a sound balance between computation and inter-processor communication.

To summarise, the system we propose for a platform for future personal mobile devices is a tile-based Heterogeneous Reconfigurable System-on-Chip (HRSoC) – it consists of multiple heterogeneous computational units, some of them reconfigurable and some of them optimised for a certain application domain, and all of them interconnected through a general on-chip communication network.

3. Communication architecture

An instance of the tiled architecture discussed in the previous section is shown in Figure 2.

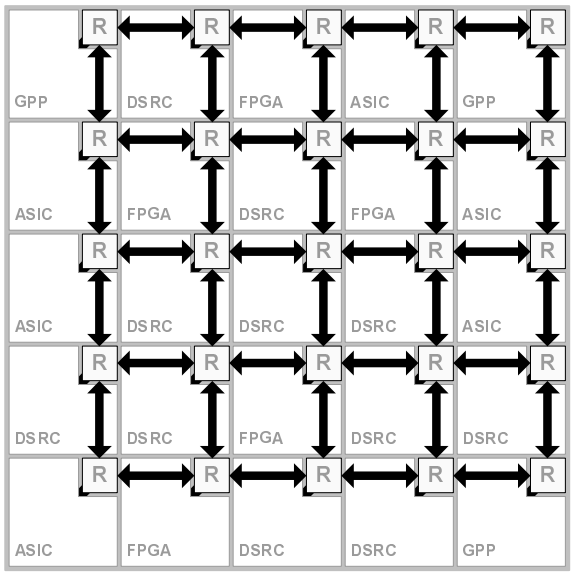

Figure 2. Tile-based Heterogeneous Reconfigurable System-on-Chip (HRSoC)
(DSRC – domain-specific reconfigurable)

A tiled organization needs a mechanism that allows the tiles to communicate with each other. Such a mechanism is implemented by an on-chip network. It consists of routers (denoted as R boxes on Figure 2) interconnected through communication channels. Each tile in the system has its own router to which it sends all data it wants to transmit and from which it receives all data it needs. The network is in charge to transport this data from the sending tiles to the receiving tiles.

We make the following assumptions for the on-chip communications:

- Communication is stream-based: there are 'streams' of data-items transmitted between processes. A stream-item can be as small as a byte, but also larger blocks (e.g. OFDM symbols).
- Processes as well as communication streams 'live' for a considerable time (seconds to hours). The

streams only change when a user selects another setting of his system.

- We focus on rather deterministic semi-static wireless/multimedia processing tasks. (Although there can be some non-determinism e.g. data-dependent runtimes etc). The communication of such systems can be characterized as block-based streaming data.
- Communications might have QoS properties (real-time constraints, minimum throughput, etc.).

For inter-tile communications in the HRSoC architecture we propose an on-chip packet-switching network. Compared with on-chip bus it has higher bandwidth as it supports multiple concurrent communications.

Based on a survey of the existing network solutions [4] we chose for a mesh-like topology for our network-on-chip. It keeps the global on-chip wires short and well structured and gives wires with well-controlled electrical parameters which simplifies their optimization for low power and higher speed on circuit level [5]

Currently we have implemented a network router in VHDL. This gives us the opportunity to perform chip-area, performance and energy estimations. The structure of the router is given on Figure 3. It is a wormhole router with a virtual-channel flow control [6] which offers high flexibility and possibility to have network traffic of different granularities. Wormhole is the only switching technique that allows minimizing the router's buffers size, because it makes the buffers size independent of the packet length. The router has 5 input and 5 output ports – 4 for connection with the neighbor routers and one for connection with the local tile. There are 4 virtual channels per physical channel which are buffered at the inputs and after multiplexing go to a crossbar.

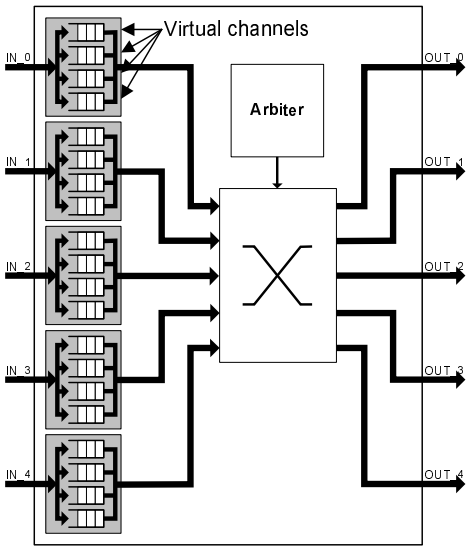

Figure 3. Organization of the 5x5 wormhole router

The wormhole router controls dataflow on a flit level, where flit (flow control digit) is an atomic data item of size one or more data words. Data in the network are transported through packets; a packet consists of two or more flits. The first flit of the packet is always a header flit and carries the destination address and the last flit is always a tail flit. The flits between the header and the tail flits are data flits and transport the data. The amount of flits in a packet can be of any number.

When a header flit of a packet arrives on a certain input virtual channel (VC) the router examines its destination address in order to find to which output channel the packet has to go. The flit is stored in the respective buffer for the input VC and a request is send to the arbitration unit. The arbitration unit is requested to allocate a VC on the output channel to which the packet is destined. After allocation is done all the flits that arrive on the input VC will automatically be sent to that output VC. At the end of the packet the tail flit will clean up this connection and frees the VCs.

Using that mechanism the header flit reserves VCs in all routers it passes from the source to the destination and thus creates a virtual path between them.

The destination address carried by the header flit determines the route the packet will take in the network. This address is given to the sending process by a central coordinating node (see section 3.1). We use source routing where the destination address completely determines the route the packet takes in the network. In a centralized system this will allow the central coordinating node to avoid communication deadlocks, to optimize the traffic in the system and to give some traffic guarantees.

For arbitration in our router we use the iSLIP arbiter [7]. It is a simple dynamic arbiter based on a round-robin algorithm, which can be implemented efficiently in hardware and provides fair arbitration for all VCs. The main task of the arbiter is to decide on a cycle by cycle basis which of the received flits advance to their destination outputs while avoiding collisions and providing fairness.

3.1. Central Coordination Node

In view of the tiled approach described above we assume that one node, which we call Central Coordinating Node (CCN), performs system coordination functions. This node can be a General Purpose Processor (e.g. an ARM9) with an RTOS (real-time operating system) running on it. The main task of the CCN is to manage the resources of the system and to allocate them to the applications that have to be run.

We believe that there are a number of good reasons to go for a centralized (CCN) approach:

- tiles might be simple (e.g. FPGA, ASIC, Montium), i.e. not equipped to run a part of a decentralized task scheduler.
- we advocate a heterogeneous system, so there should be an entity that has an overview of the capabilities and utilization of all the tiles.
- the CCN has an overview of the entire system i.e. all tiles utilization and all inter-tile communication streams. This simplifies a number of issues, e.g. :
  - it can give QoS guarantees to the communication streams. This would be much harder in a decentralized solution
  - the latency overhead can be minimized
  - the CCN can find the most (energy-) efficient mapping of processing and communication while utilizing locality of reference
- the CCN can avoid misbehaving/inversatile tiles/communication links. This possibility improves the reliability.
- the CCN can run on a GPP (e.g. ARM); this means that this processor can be reused for other tasks while not performing coordination tasks. In a decentralized approach each tile needs some intelligence; and this adds to the chip-area of the overall system.

From a theoretical/scientific point of view a central node has a number of drawbacks: most notably scalability and reliability issues. The scalability drawback can be solved by introducing hierarchy in the system. A CCN might be responsible for number of tiles (e.g. 16), this is realistic because we assume that the CCN only has coordinating (control) functionality, the main data-streams do not pass through the CCN, but are dealt by the routers and communication channels between the individual tiles.

The reliability issue (CCN single-point of failure) might be addressed by having more than one GPP in a HRSoC.

3.2. Operation

Our network supports both streaming and non-streaming applications. For streaming applications the CCN can reserve a virtual path between the different tiles that constitute this application. A virtual path is a concatenation of virtual channels, as explained in the previous section. The sender receives from the CCN to which VCs the virtual path should be allocated. Each sending tile sets up a virtual path to the receiver, and keeps this path as long as the application is running. This virtual path can be considered as a "virtual circuit switched" communication path.

For non-streaming applications we also support packet-switched communications. A sending tile sends a packet containing a header flit, data flits and a tail flit, along the path defined by the CCN. This virtual path is only allocated during the actual transfer of the packet. When the entire packet has been transported, all the virtual channels that were used for the virtual path are freed and can be used for other communications. The CCN can decide, when QoS permits, to share virtual channels by more communications. This, however, might lead to a temporary blocking of a message and thus leads to best-effort characteristics.

Below we give two examples of how our system can be used.

Example 1

Suppose in our system, shown again on Figure 4, we choose for a CCN on the GPP tile, let say, in the upper left corner.

After system start we load a multitasking operating system (RTOS) on the GGP and it starts one or more tasks (represented by the top-left circle in Figure 4). Suppose one of these tasks wants to perform some computationally complex function, for example a FFT. It calls a system function, which is handled by the OS. The OS checks which available resources (tiles) can perform the requested function, selects one and configures it, if this is necessary, by sending a message to it through the network. Finally it sends the data to be processed, again by using the network. The task that has requested a FFT function will stay blocked on an input communication channel waiting for the result and during that time the GPP is free to run other tasks. The processing unit committed to the FFT computation (represented in Figure 4 by the circle labelled FFT) will do its job and will return the result by sending a message back, which will unblock the waiting task and make it ready.

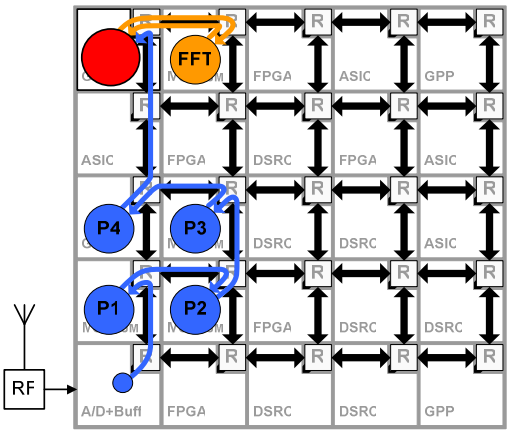

Figure 4. Two examples of processes mapping on HRSoC

The procedure above is, in fact, an application of well known principles from operating systems. A difference is, may be, that here the terms function and resource are, in a certain extend, merged and that the access to the resources is always made through network communication channels.

When another FFT is requested, the OS will know that there is already such a function mapped and will directly send data to it. In case a higher performance is required, the OS can map the FFT function on several tiles which work in parallel.

We can generalise this example for every function for which there is a suitable resource in the architecture. We can generalise it not only for a single function but also for composite functions consisting of multiple basic functions interconnected following a certain process-graph. To map such a composite function the CCN has to map all the basic functions onto tiles and to allocate the communications between the tiles. Data we want to process are sent to the one of the basic functions which is an input for the composite function. During the processing the basic functions exchange data between them using the network and at the end when the processing is finished the basic function which is an output for the composite function returns the final result. We will illustrate this through an example showing how the system can handle the processing needed for supporting a wireless communication channel.

Example 2

We designed such a composite function for a HiperLAN2 wireless channel [8] using three MONTIUM tiles [2] where the functions that the three tiles perform are as follows:

P1: Prefix removal and Frequency offset correction

P2: Inverse OFDM

P3: Equalisation, Phase offset correction and De-mapping.

The fourth process P4, running on a GPP, can do the processing needed in the higher layers of the network protocol stack and provides the pure data packet to the consumer task. In Figure 4 we see four processes (P1 to P4), represented with circles labelled P1 to P4, mapped on separate tiles and interconnected through communication channels in a chain fashion. This group of interconnected processes can be seen as one composite function with input point P1 and output point P4. It accepts the data received by the antenna of the wireless channel, does all necessary processing to extract the user data transported on the wireless channel and sends this data to a consumer process that runs on the coordinating node CCN (represented by a dark circle).

The analogue front-end of the antenna is connected to an interface tile, in our example in the bottom left corner, which does the A/D conversion, provides buffer space for the converted data and implements some simple control logic. When a complete radio packet is received the interface tile sends it to P1, P1 processes it and sends it to P2 and so on. P1 to P3 implement the physical layer of the wireless channel.

4. Related work

The ÆTHEREAL project [9] proposes a network-on-chip architecture. The router designed there also uses dynamic SLIP arbitration but combined with time-division arbitration. The later allows their router to give certain bandwidth guarantees but also increase the complexity and require common notion of time.

In [10] a network on chip for a tiled architecture is proposed. The tiles there are interconnected in a folded torus topology. For flow control mechanism also virtual channels are used. The router has different topological placement on the tile than ours as it is aimed at exploiting the large amount of wiring resources and uses very wide communication channels.

The RAW architecture [11] is a tiled architecture but it is homogeneous and aims in exploiting instruction level parallelisms. For inter tile communications 4 on-chip networks are used there: 2 static networks using time-division and 2 dynamic networks

The Pleiades project [12] proposes a heterogeneous architecture combining components of different programming granularity. The interconnection topology used there is a two level hierarchical mesh and the network is implemented using self-timed circuits. Our current implementation is fully synchronous.

5. Results

The router presented in Section 3 was modelled in VHDL and synthesized for a Xilinx Virtex-II FPGA and standard 0.5 µm ASIC technologies. Table 1 gives the synthesise results. For size indication we took the number of utilized CLB slices in the FPGA and the number of utilized gates in the ASIC. The table presents the total amount of utilized resources as well as the distribution of these resources among the main router's components. The same results are represented graphically in Figure 5 and Figure 6.

Table 1. Implementation results

	Virtex-II	ASIC 0.5um
	CLB slices	Gates
Buffers	764	25467
Arbiter	733	21719
Crossbar	335	5782
Total	1832	52968

To get a rough indication about the size of the router we make a relative comparison with the implementation results of the MONTIUM tile. Synthesized for Virtex-II the tile utilizes 12755 CLB slices, which is 7 times more than the CLB slices utilized by the router. On the other hand, when synthesized for 0.13 µm process the tile takes an area

of 1.8 mm^2. So we could expect our router, when synthesized for 0.13 um process, to take roughly one seventh of this area, which is $1.8mm^2 / 7 = 0.26mm^2$. That size is comparable with the size of the on-chip router presented in [9], which is also 0.26 mm^2, again for the same 0.13 µm technology.

The maximal clock frequencies at which the router can operate when synthesized for Virtex-II and ASIC 0.5 um are 66 MHz and 68 MHz respectively.

Our next step is to synthesize the router for a 0.13 µm process technology and to make area, performance and energy estimations. Since the design was made generic we can change the basic parameters of the router: physical channels width, number of virtual channel per input and size of the buffers. This will allow experimenting with multiple versions of the router and observe what role the router's components play in the area and energy consumption.

In the next phase we also plan to do energy estimations and a detailed performance analysis of non-streaming communications. Furthermore, we see ample possibilities to optimize our design.

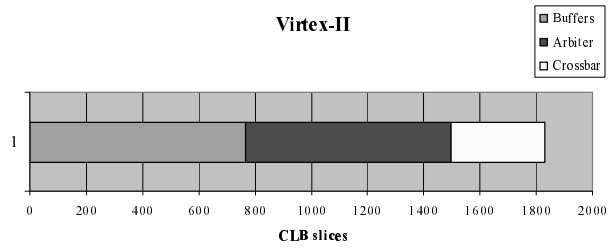

Figure 5. Resources distribution for Virtex-II

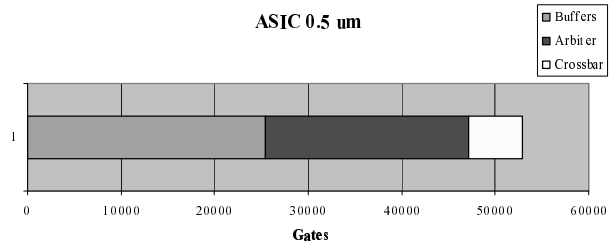

Figure 6. Resources distribution for ASIC

6. Conclusion

This paper presented the system-on-chip architecture we foresee for future wireless multimedia terminals, focusing on the on-chip communication architecture. We proposed a network-on-chip solution based on packet switching wormhole network with virtual-channel flow control. The initial implementation results for an on-chip network router showed that it is feasible and comparable in size with other proposed solution. In the same time it provides the performance and flexibility required by the targeted application domain.

7. References

[1] Smit G.J.M., Havinga P.J.M., et al.: "Dynamic Reconfiguration in Mobile Systems", *Proceedings of Field-Programmable Logic and Applications*, pages 171-181, Montpellier, France, September 2002.

[2] Heysters P.M., Smit G.J.M. & Molenkamp E.: "A Flexible and Energy-Efficient Coarse-Grained Reconfigurable Architecture for Mobile Systems", The Journal of Supercomputing, volume 26, number 3, Kluwer Academic Publishers, Boston, U.S.A., November 2003, ISSN 0920-8542.

[3] Abnous A.: "Low-Power Domain-Specific Processors for Digital Signal Processing", *Ph.D Dissertation*, University of California, Berkeley, USA, 2001.

[4] Nikolay Kavaldjiev, Gerard J.M. Smit, "A survey of efficient on-chip communications for SoC", *PROGRESS 2003 Embedded Systems Symposium*, October 22, 2003, NBC Nieuwegein, Netherlands, ISBN 90-73461-37-5.

[5] H. Zhang and V. George and J.M. Rabaey, "Low-swing on-chip signalling techniques: effectiveness and robustness", *In Transactions of Very Large Scale Integration(VLSI) Systems*, pp 264--272, Vol.8, No.3, June, 2000.

[6] W. J. Dally, "Virtual-channel flow control", *IEEE* Transactions *on Parallel and Distributed systems*, vol. 3, no. 2, pp. 194-205, March, 1992.

[7] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188--201, April 1999.

[8] Paul M. Heysters, Gerard K. Rauwerda, Gerard J.M. Smit, "Implementation of a HiperLAN/2 Receiver on the Reconfigurable Montium Architecture", *11th Reconfigurable Architecture Workshop (RAW 2004)*, Santa Fé, New Mexico, USA, April 26-27, 2004.

[9] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander, "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip", *Proceedings of Design Automation and Test Conference in Europe*, March 2003.

[10] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", *DAC*, June 2001, pp. 684-689.

[11] Michael Bedford Taylor, Jason Kim, Jason Miller, David Wentzlaff, Fae Ghodrat, Ben Greenwald, Henry Hoffman, Jae-Wook Lee, Paul Johnson, Walter Lee, Albert Ma, Arvind Saraf, Mark Seneski, Nathan Shnidman, Volker Strumpen, Matt Frank, Saman Amarasinghe and Anant Agarwal, 'The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs", *IEEE Micro*, March/April 2002.

[12] Hui Zhang, Vandana Prabhu, Varghese George, Marlene Wan, Martin Benes, Arthur Abnous, "A 1V Heterogeneous Reconfigurable Processor IC for Baseband Wireless Applications", *Proceedings of ISSCC2000.*