Click-Based Asynchronous Mesh Network with Bounded Bundled Data

Anping He
School of Information Science and Engineering
Lanzhou University
Lanzhou, Gansu, China
heap@lzu.edu.cn

Guangbo Feng
School of Information Science and Engineering
Lanzhou University
Lanzhou, Gansu, China
879160522@qq.com

Jilin Zhang
School of Physical Science and Engineering
Lanzhou University
Lanzhou, Gansu, Chian
jlzhang2015@lzu.edu.cn

Pengfei Li
School of Information Science and Engineering
Lanzhou University
Lanzhou, Gansu, China
lipf2017@lzu.edu.cn

Yong Hei
Institute of Microelectronics Chinese Academy of Sciences
Beijing, China
heiyong@ime.ac.cn

Hong Chen*
Department of microelectronics and nanoelectronics
Institute of Microelectronics Tsinghua University
Beijing National Research Center for Information Science and Technology
Beijing, China
hongchen@tsinghua.edu.cn

ABSTRACT

We have implemented an asynchronous mesh network. This paper describes our innovative design using a Click controller. Compared to designs that use other asynchronous circuit families with C-elements and four-phase bundled data, our two-phase Click-based Bounded Bundled Data design is faster, but introduces phase skews when handling concurrent traffic at a single node. Instead of eliminating the phase skews, we use them as computation slots. Our network uses a novel asynchronous arbiter with a queue that can accept data from both the four cardinal directions as well as from a local source, five directions in all. We have implemented our network design in 1×1, 2×2 and 4×4 sizes, larger network could be implemented easier since the isomorphism and modularity of the routing nodes. Our experiments show that an initial data item passes through a node in 157ns v.s. 81ns for non-delay-branch and delay-branch designs separately. Following items take about 65% as long. But for a network, the average latency of a node keeps almost same for different paths. We believe that with the non-delay-branch designs, our asynchronous mesh network could offer 10.1M routes per second for a 1×1 network and 5.33M routes per second for 2×2 or 5.06M for 4×4 networks, and work at the rate of 17.3M, 10.1M and 11.7M with the enhanced delay-branch way. For both cases, its latency is approximately linear with scale.

*Crorresponding author

ICPP 2018, August 13–16, 2018, Eugene, OR, USA

ACM ISBN 978-1-4503-6510-9/18/08…$15.00
https://doi.org/10.1145/3225058.3225118

CCS CONCEPTS

• **Hardware → Asynchronous circuits**;

KEYWORDS

Asynchronous Mesh Network, Click-based Asynchronous Pipeline, Computation Slot, Asynchronous Rotation Priority-based Arbiter, Delay Branch

ACM Reference Format:
Anping He, Guangbo Feng, Jilin Zhang, Pengfei Li, Yong Hei, and Hong Chen. 2018. Click-Based Asynchronous Mesh Network with Bounded Bundled Data. In *ICPP 2018: 47th International Conference on Parallel Processing, August 13–16, 2018, Eugene, OR, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3225058.3225118

1 INTRODUCTION

Future applications running on many-core processors with tens to hundreds of cores on a chip, will require an efficient inter-core communication strategy to achieve high performance [1]. We are capable of monitoring thousands of genes and their interactions, but we lack efficient computing platforms for large-scale (exa-scale) data processing [2]. Currently, the bus structure is a common way that is in charge of the communication between different cores in a multiprocessor System on Chip (SoC), but unfortunately, it will soon become a critical bottleneck in chip performance with its inability to scale its communication network effectively with decreasing feature sizes and increasing number of transistors[3]. The Network-on-Chip (NoC) concept has been proposed to replace conventional bus-based system architectures to create scalable and flexible future SoC designs[4].

Currently, typical application-specific NoC systems often integrate a number of heterogeneous components which have varied functions, sizes and communication requirements[5]. As one of the effected scalable interconnection architectures, the mesh network may significantly improve the performance of a NoC in terms

of scalability, modularity, transport latency, parallelism and many more important metrics. Again, this specific network that equipping a considerable scheduled program, can also balance load of the network and try to reduce the chance of causing a hot spot on the network[6].

Design of a mesh-based NoC consists of processing address, and routing of the traffic traces on the topology such that modules are easy to reuse, power consumption is minimized, and performance constraints are satisfied. Because of the essence of isomorphism and distribution of the topology, a mesh network always reuses router structures that will bring challenges for clock-based designs, such as harmonization for the clock skews and clock distribution; furthermore, requests on a network characterized by events that are unsuitable for a scheduled procedure that potentially slow down the traffic. Nevertheless, routing performance is more sensitive to the router latency[7] not the frequency of a system clock. On the other hand, the physical data-path of a mesh NoC though consumes significant energy[8], the clocked registers would flip-flop continually. Then, it is more feasible to adopt asynchronous mechanism than the clocked mesh network.

In this paper, we introduce an asynchronous mesh structure that is constructed by a Click-based[9] micro-pipeline[10][11]. A Click-based circuit is principally much faster than C-Element-based [12] four-phase bundled data (BD)[13] implementations and NCL-based QDI designs[14], since it follows a two-phase bounded bundled data (BBD) protocol[15–17] that transfers data while communication changes, and eases the control of each data-path by a local clock[18], and its timing analysis is well studied in[19].

Our mesh network is composed of the isomorphic nodes dealing with the routing requests. The Click-based circuit is implemented by a three-layer asynchronous design: the first layer is a configurable asynchronous FIFO that buffers requests of data and addresses from east, west, south, north and local; the second is an asynchronous arbitration in terms of the polling strategy; and the last one is the routing unit that adopts a delay brach technology to process the address and sends data to a specific direction with an innovative enhanced XY algorithm. Each asynchronous module exchanges data and control under a two-phase contract, smoothly and consistently.

Our mesh network introduces phase skews when multi-channel communication and arbitration happens. Instead of eliminating the phase skew we use it as a computation slot to tackle the different components within a chip. In absence of a system clock, the slot sends a request and waits for the acknowledge from a concrete computation, regardless of its latency. This concept enables a higher-speed design.

We implement this asynchronous mesh network with a 1×1, 2×2 and 4×4 node array with a commercial FPGA (Virtex-7 of Xilinx) board, more scales could be obtained by the isomorphic and modular routing nodes. Initial experimental results show that the first route takes more time because of buffering requests, and then overall 65% as long per route. The overall path delay grows linearly in terms of nodes. In short, the asynchronous mesh could offer 10.1M routes per second in a 1×1 network, 5.33M for 2×2 and 5.06M for 4×4, and it behaves functionally with the liner and monotonic growth of its scale. And then we adopt a delay-branch technology that shows more than 40% acceleration for 1×1 network, 47% for 2×2 and 56% for 4×4 with the Xilinx Vivado tools, e.g.,

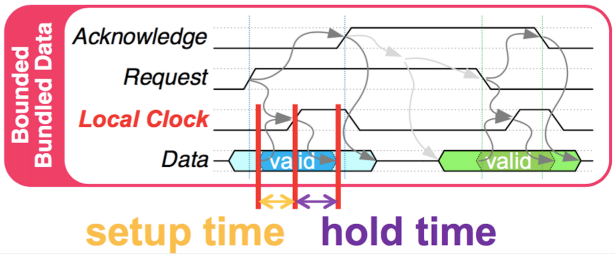

Figure 1: Bounded bundled data protocol[16, 17]

the asynchronous mesh network could work in about 17.3M, 10.1M and 11.7M routes per second, and still keeps the linear property (see Section 5).

The rest of the paper is organized as follows: Section 2 introduces the architecture of the asynchronous mesh network. Section 3 describes the methods dealing with the asymmetric phases. The simulation results are presented and discussed in Section 4. Section 5 concludes the paper.

2 ARCHITECTURE OF ASYNCHRONOUS MESH

The mesh network is one of the widest adopted Networks on Chip (NoCs), because of high performance and easy design. It can route addresses and respond to requests from different directions at any time with event-driving robustness, which makes it suitable be implemented by asynchronous circuits instead of clock-based synchronous circuits. We design a Click-based mesh network. Compared with other asynchronous methods, e.g., four-phase BD implementations and NCL designs, the Click-based two-phase BBD[16, 17] (see Fig. 1) circuit is much faster principally.

According to the topology of mesh network and thanks to the modularity of asynchronous BBD design, all nodes of our design are isomorphic. Structurally, each node contains three layers:

- a configurable buffer;
- an arbiter with queue; and
- a routing processing unit.

The function of each layer's handling adress is harnessed by Click, which compose an asynchronous micro-pipeline. The communication between layers also abide by the BBD protocol (shown in Fig. 2). The addresses and attached data are firstly buffered in the configurable FIFO chains for the different directions, then the arbiter arbitrates between these requests simultaneously, and then the routing unit calculates a concrete destination with an XY-based algorithm[20].

2.1 Asynchronous Communication Crossing Layers

As we mentioned before, the whole routing algorithm is mapped and performed by nodes in our mesh network, e.g., three layers of a node would cooperate to finish a routing task, which requires an elaboration of the accurate communication. Due to the asynchronous BBD mechanism, it is not a heavy work to make this

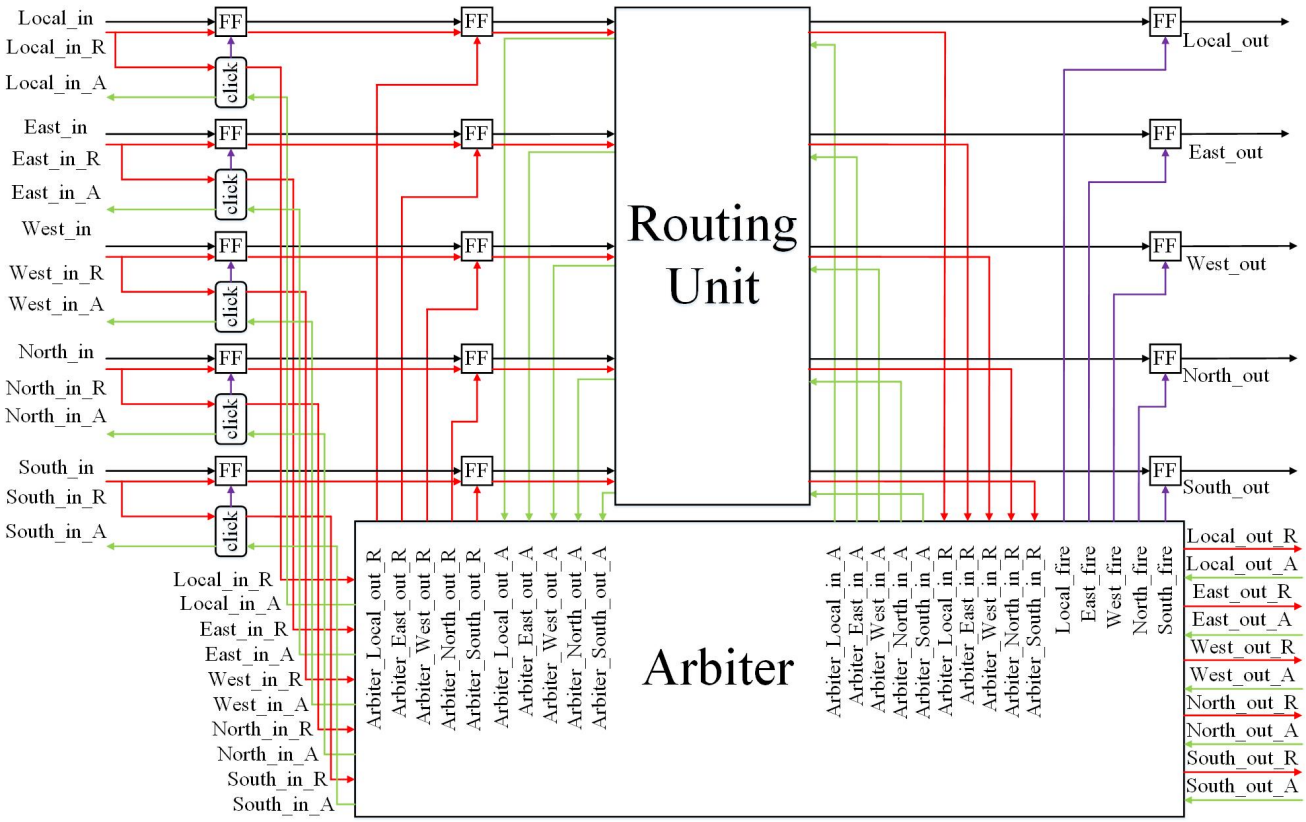

Figure 2: The architecture of our asynchronous mesh network

type of communication run: since the relative timing issues have been well studied and adjusted [19] and BBD protocol splits the communication and control, only thing we focus on is calculating the delay between linked components. Briefly, the buffers are configured by adding or deleting flip-flops clocked by Click, which does not introduce communication hazard because of the asynchronous event driven flow.

In order to describe the design of routing in the mesh network clearly, we focus on the routing of the network and omit the data. Let input and output address and request be $Iaddr$ and $Ireq$, respectively $Oaddr$ and $Oreq$. Let $Iaddr = \{Iaddr_e, Iaddr_w, Iaddr_s, Iaddr_n, Iaddr_l\}$ and $Ireq = \{Ireq_e, Ireq_w, Ireqs, Ireq_n, Ireq_l\}$ be the address and request list from east, west, south, north and local input. Addresses are regarded as binary numbers in the rest of this paper. The configurable asynchronous buffer is composed of flip-flops of n ($n \geq 0$) stages, and formalized as a function buf.

$$\begin{aligned} buf(Iaddr) &= \underbrace{buf(\cdots buf(buf}_{n}(Iaddr))) \text{ and} \\ buf(Ireq) &= \underbrace{buf(\cdots buf(buf}_{n}(Ireq))) \end{aligned} \quad (1)$$

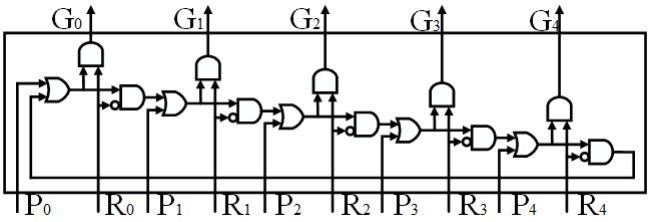

Figure 3: The rotation priority arbitration

2.2 Asynchronous rotation Priority Arbitration

The purpose of the arbiter in a routing node concerns a fair arbitration of buffered requests that may arrive at the same time. In our design, we use the rotation priority arbitration whose highest level access will be changed after each arbitration. This type of arbiter has the advantages of fair arbitration.

The arbiter, grants a request port according to a counter's value. The counter counts between 1 and 5. We use a *mask* for passing

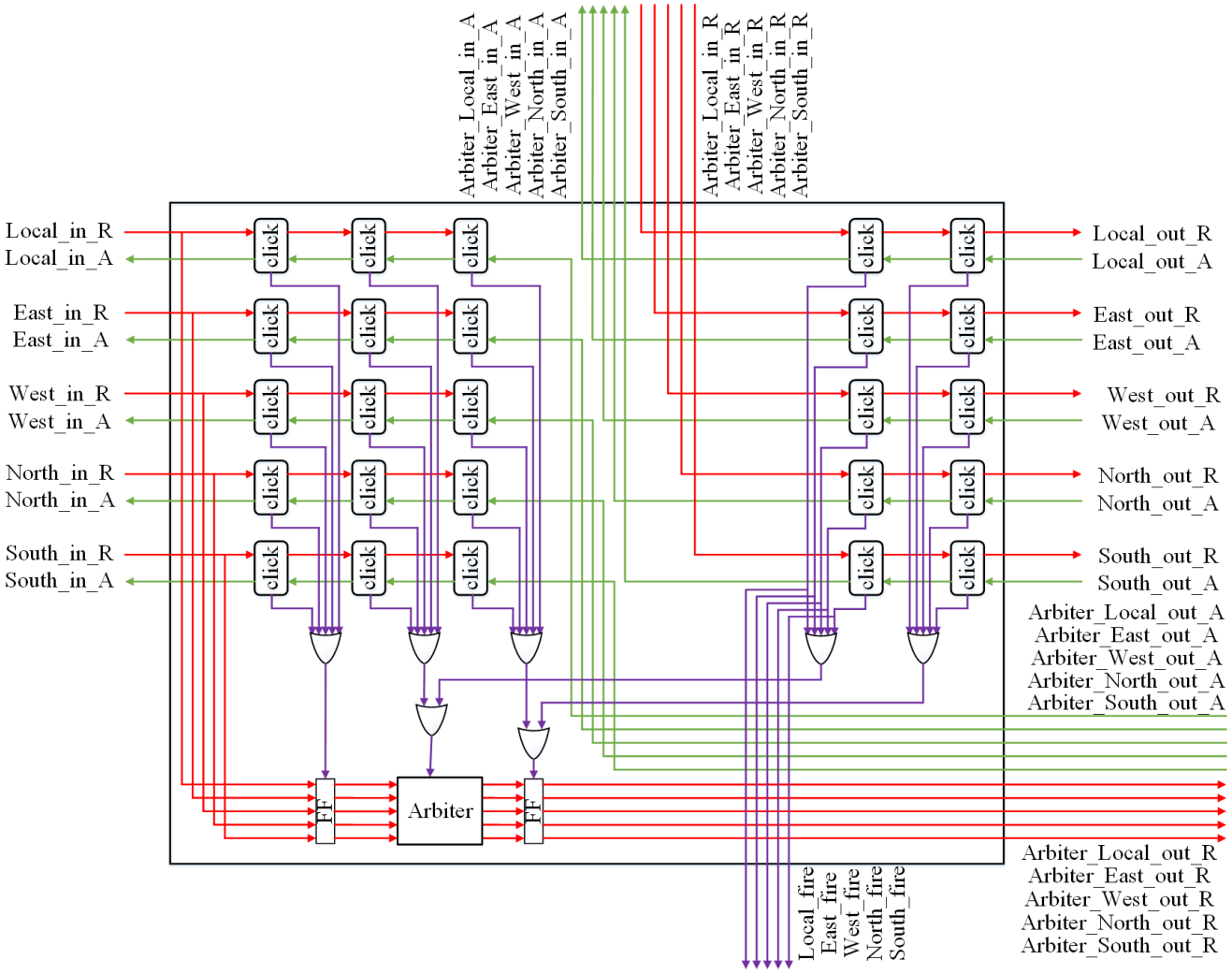

Figure 4: The asynchronous rotation priority arbiter

the priority:

$$mask = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

And $mask(i)$ shows the i-th line of the mask matrix, $0 \leq i \leq 4$. Let c be the times of the abstraction which is implemented by the counter, then the rotation priority-based arbitration is:

$$arbitr = mask(c\%5) \times Ireq \quad (2)$$

With Equation 2, we design the key rotation priority arbitration in Fig. 3, in which P_0 to P_4 are the priority setting ports, R_0 to R_4 are the request inputs. When a P_i goes high, the arbiter selects the i-th *mask* line, then the corresponding R_i captures the highest priority. The arbitration and also the attached data and other functions are managed by the Click pipeline: the control part is composed of five asynchronous micro-pipelines, left three of which do initialization and the first arbitration, then other two drive the arbiter to do the next four arbitrations. Every five arbitrations a group makes up a rotation. As shown in Fig. 4.

2.3 Asynchronous Routing Unit

Routing algorithms are classified in deterministic, random and adaptive. The most common deterministic ones are the E-cube[21] and XY[20] algorithms, which process the routing directions one by one. Since the XY routing algorithm is intuitive, the routing path has good predictability, and its logic structure is clear and easy to implement. We adopt a 2-D XY routing algorithm.

The asynchronous arbiter does not process addresses but just selects between requests. When the address arrives at the asynchronous XY routing unit, the address *addr* for the algorithm will chosen by

$$addr = aMerge(buf(Iaddr), arbitr) \quad (3)$$

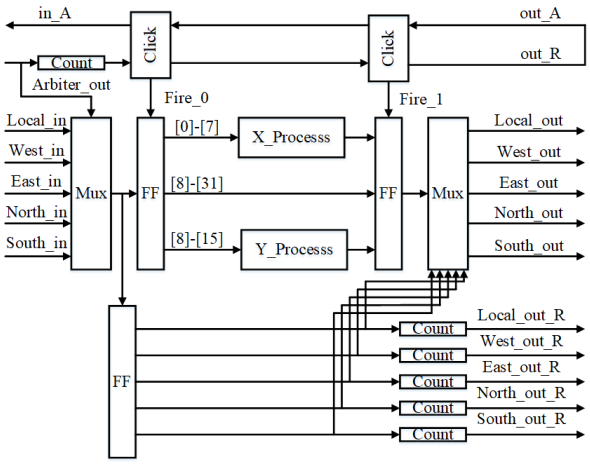

Figure 5: The XY calculation with delay branch

Then four isomorphic modules perform adding or subtracting 1 according to the MSB (symbol bit) of an address, and calculate the target addresses for all 4 directions. The local address is always fixed as 0. Let x and y parse an address into components of X and Y directions separately, MSB be a function that returns the most significant bit of a binary number, $zero$ be a function that returns 1 if a binary number is equal to 0, and cat catenate two strings, then

$$\begin{aligned} addr_e &= MSB(x(addr)) \times cat(x(addr)-1, y(addr)) \\ addr_w &= (1-MSB(x(addr))) \times \\ &\quad cat(x(addr)-1, y(addr)) \\ addr_s &= zero(x(addr)) \times MSB(y(addr)) \times \\ &\quad cat(x(addr), y(addr)-1) \\ addr_n &= zero(x(addr)) \times (1-MSB(y(addr))) \times \\ &\quad cat(x(addr), y(addr)-1) \\ addr_l &= 0 \end{aligned} \tag{4}$$

Regardless of directions, the output of the address is

$$Oaddr = addr_e + addr_w + addr_s + addr_n \tag{5}$$

The request that labels the direction of the data is generated by MSB and address:

$$\begin{aligned} Oreq = \{&MSB(x(addr)), (1-MSB(x(addr))), \\ &MSB(y(addr)), (1-MSB(y(addr))), \\ &zero(Oaddr)\} \end{aligned} \tag{6}$$

All of them are shown in Fig. 5 implemented with delay-branch technology to reduce latency.

Then by combining Equations (1), (2), (3), (4), (5) and (6), we map $Iaddr$ and $Ireq$ into $Oaddr$ and $Oreq$. Each step is implemented by an asynchronous component with Click-based micropipeline[18].

3 ASYMMETRIC PHASES

The control mechanism of Fig. 5 introduces a challenge. There will be five possible requests which share a decision component

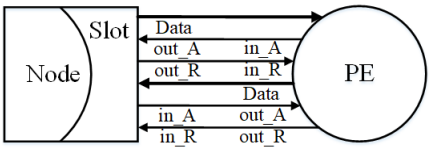

Figure 6: The computation slot and process unit

all arrive parallel. Suppose we choose east first and the communication sets the interface of the decision component high, e.g., in the request going high phase of our two-phase BBD protocol. Consequently, the decision component needs a going low phase of the BBD protocol for processing the next decision, and the east request will communicate secondly and all are well. However, any request from other direction, says west, still tries to push the communication signal high to accomplish its own handshake, while the phase of the decision component has been changed. We call this an asymmetric phase. One solution is to do the decision component communication twice per request, one is the real request for passing and processing data, and the other just a virtual one for resetting the handshake. Instead. we leverage phase skews to introduce computation slots.

Algorithm 1 XY with Computation Slots

```
1: procedure XYCS(addr, data)
2:     if x(addr) ≠ 0 then
3:         XYCS(cat(x(addr) − 1, y(addr)), data)
4:     else if y(addr) ≠ 0 then
5:         XYCS(cat(x(addr), y(addr) − 1), data)
6:     else
7:         return ComputationSlot(data)
8:     end if
9: end procedure
```

3.1 Computation Slots

Principally, the computation slots is a strategy that solves the asymmetric problem and can be seen as a switch of a routing node, e.g., if computation slots is off, the node will only perform routing tasks, and data would input to a local computation unit if open. Then the configuration of the routing becomes a variation of switches among the mesh nodes.

Actually, each node is attached with a specific switch, which communicates with a BBD protocol. The switch supplies a slot called *computation slot* in which any module that follows BBD protocol could be embedded, e.g., a 1-bit request and acknowledge signal, and data . See Fig. 6: the data is output through the *local_out* port, and meanwhile the *Local_out_R* signal triggers asynchronous micro-pipelining in the switch, then one of the two different data paths is enabled in terms of the address. If the local computation is selected, the PE processes data and then responds to the switch with *PE_in_A*, and passes the data through *PE_out* port.

Then the computation slot-based routing algorithm may start a new routing task from a node or send data to a computation slot, as summarized in Algorithm 1.

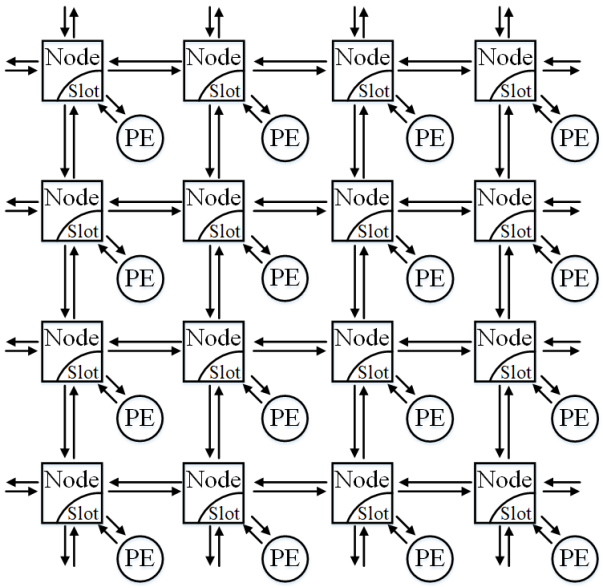

Figure 7: The asynchronous mesh network with computation slot

4 SIMULATION RESULTS AND DISCUSSION

We implemented the about design as a 1×1, 2×2 and 4×4 mesh network in a Xilinx commercial FPGA, using a Virtex-7 evaluation board. The software is the Vivado development tool kit. We simulated the mesh network with input in five directions, each input is a 32 bits containing the first 16-bit address and last 16-bit data, the first 8 bits of address encodes the east-west message, and the last 8 for the north-south, and bit 7 and 15 in address represent the positive and negative direction separately. The data do not participate in the routing procedure.

For the 1×1 mesh network, i.e., a single node mode, we verified the asynchronous arbitration by extreme cases of five direction requests, which shows the results are output in turn after five arbitrations. Waveforms show that when data enter the routing node at the same time, the arbiter arbitrates between the request signals, sorts and outputs the input data information, ensuring the fairness of the routing node to the input data using rotation priority arbitration. The processing time is shown in Table 1. After analyzing the simulation results, we find

(1) the first routing time is longer than the latter ones, since while the address enters the routing node, the arbiter would arbitrate, buffer and sort this request. After buffering, sorting completed, the address is processed directly;
(2) the routing time stays almost the same after the first routing; and
(3) the delay-branch technology improves the throughput dramatically, e.g., from 10.1M to 17.3M routes per second.

Next we simulated multiple routing paths in a 2×2 mesh routing network. The routing time from 00_*node* to 01_*node* is about 373ns, and the average routing time per node is about 187ns. The path from 00_*node* to 11_*node* is the longest one of the 2×2 mesh network, and has a delay of about 563ns. The path is composed of three routing nodes, and the average routing time per node is about 187ns, and the throughput of single node of a path is 5.33MHZ routes per second (See Table 2). However, with the acceleration of delay-branch way, it can be enhanced to 10.1 MHZ.

We then simulated paths in a 4×4 mesh network. The route time from 00_*node* to 01_*node* is about 433ns, its average routing time per node is about 216ns. The route time from 00_*node* to 11_*node* is about 649ns. The route is processed by two routing nodes, and the average routing time per node is about 216ns. The route time from 00_*node* to 22_*node* is about 892ns, whose route is processed by two routing nodes, and the average routing time per node is about 178ns. The route from 00_*node* to 33_*node* is the longest path in the 4×4 mesh network, and the routing time is about 1248ns, which is processed by routing of three routing nodes, and the average routing time per node is about 187ns. From the experimental data, we can derive that the routing time of a single node in a 4×4 mesh network is about 197ns, and the throughput of a single node is 5.06MHZ. Similarly, after applying the delay-branch method, theQ throughput increases to 11.7MHZ (See Table 2).

The three designs are implemented in the virtex-7 series FPGA board, their resource occupation and throughput rates are as shown in table 3. In summary, the resource occupation of our asynchronous mesh structure is a linear increment with the scale of the network, but the throughput keeps almost the same while the nodes compose a network. The results reflect the strong homogeneity and stable structure of mesh networks.

5 CONCLUSION

In this paper, we first proposed an innovative asynchronous design of the mesh network with Click controllers that implements a BBD protocol. We find phase skews when multi-channel communication at one node of mesh happens. Instead of eliminating the phase skews we leverage phase skews to introduce computation slots, which create a design with higher speed. Moreover, we designed a novel asynchronous arbiter equipping a queue that processes addresses from five different directions, and a delay-branch solution of XY routing algorithm, with higher throughput capacity and fairness. All modules are designed and managed with asynchronous pipelines that makes the whole system run smoothly. Since the routing nodes are fully asynchronous and clockless, the whole network could be easily built in a scalable way. We implemented this asynchronous mesh network with 1×1, 2×2 and 4×4 node arrays with a commercial FPGA (Virtex-7 of Xilinx) board. Experimental results show that the asynchronous mesh could offer 17.3M routes per second of a 1×1 network, 10.1M for 2×2, and 11.7M for 4×4, and it behaves functionally with a liner and monotonic growth of its scale.

We believe this type of asynchronous mesh network harnesses the following advantages:

- the BBD protocol enhances the whole design modularity with the isomorphic design of nodes without consideration of clock;
- the absence of the clock make whole design easy;
- the asynchronous arbiter is fair.

Table 1: Routing time of 1×1 network

Times	1*1 (Group1)		1*1 (Group2)	
	Delay-Branch	Tradition	Delay-Branch	Tradition
1	81.199	157.176	81.627	131.963
2	56.305	95.934	56.386	97.045
3	57.226	97.345	55.993	96.108
4	56.178	97.188	58.261	97.371
5	57.831	96.473	56.361	96.864
Average	56.885	96.735	56.750	103.870
Acceleration	41.2%		45.4%	

Table 2: Routing time of 2×2 and 4×4 networks

Path	2*2Mesh		4*4Mesh	
	Delay-Branch	Tradition	Delay-Branch	Tradition
(0,0)-(0,1)	99.328	187.496	85.520	216.456
(0,0)-(1,1)	98.430	187.604	85.265	216.333
(0,0)-(2,2)			85.545	178.402
(0,0)-(3,3)			85.402	178.326
Average	98.879	187.55	85.432	197.379
Acceleration	47.3%		56.8%	

Table 3: Performance List

Resource	1*1 Mesh		2*2Mesh		4*4Mesh	
	Delay-Branch	Tradition	Delay-Branch	Tradition	Delay-Branch	Tradition
LUT(Utilization)	3905/1.29%	1702/0.56%	5899/1.94%	7118/2.34%	22159/7.30%	30995/10.21%
FF(Utilization)	3485/0.57%	955/0.16%	3828/0.63%	3904/0.64%	14568/2.40%	23680/3.90%
Throughput Rate (MHZ)	17.3	10.1	10.1	5.33	11.7	5.06

- the delay-branch method makes the XY algorithm much faster;
- the computation slots increase the latency and configurability.

ACKNOWLEDGMENT

We would thank Ivan Sutherland, Marly Roncken in Asyn- chronous Research Center in Portland State University, USA. They spend more time and energy to introduce and train Anping and his team in the advanced asynchronous circuit design and verification methodologies by a systematic way. The click controller in this paper is directly from ARC, and the asynchronous mesh design, as well as this paper, would never succeed without their help. This work is supported by the Fundamental Research Funds for the Central Universities of Lanzhou University (No. lzujbky-2017-194), Guangxi Science and Technology (No. AB17129012), National Natural Science Foundation of China (No. 61674090), and partly supported by Beijing National Research Center for Information Science and Technology (No. 042003266).

REFERENCES

[1] Meyer M, Okuyama Y, Abdallah A B. A Power Estimation Method for Mesh-Based Photonic NoC Routing Algorithms. Fourth International Symposium on Computing and NETWORKING. IEEE, 2017:451-453.
[2] Xue Y, Bogdan P. User Cooperation Network Coding Approach for NoC Performance Improvement. International Symposium on Networks-On-Chip. ACM, 2015:17.
[3] Menon A, Zeng L, Jiang X, et al. Adaptive Look Ahead algorithm for 2-D mesh NoC. Advance Computing Conference. IEEE, 2015:299-302.
[4] Lee W, Sobelman G E. Mesh-star Hybrid NoC architecture with CDMA switch. IEEE International Symposium on Circuits and Systems. IEEE, 2009:1349-1352.
[5] Duan X, Zhang D, Sun X. Routing Schemes of an Irregular Mesh-Based NoC. International Conference on Networks Security, Wireless Communications and Trusted Computing. IEEE, 2009:572-575.
[6] Ghosal P, Das T S. L2STAR: A Star Type level-2 2D Mesh architecture for NoC. Microelectronics and Electronics. IEEE, 2013:155-159.
[7] Qi S, Li J, Xing Z, et al. A Delay Model of Two-Cycle NoC Router in 2D-Mesh Network. IEEE Symposium on Vlsi. IEEE Computer Society, 2010:316-320.
[8] Park S, Qazi M, Peh L S, et al. 40.4fJ/bit/mm low-swing on-chip signaling with self-resetting logic repeaters embedded within a mesh NoC in 45nm SOI CMOS. Design, Automation & Test in Europe Conference & Exhibition. IEEE, 2013:1637-1642.
[9] Peeters A, Beest F T, Wit M D, et al. Click Elements: An Implementation Style for Data-Driven Compilation. Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits & Systems, 2010: 3-14.
[10] Ivan E. Sutherland and Jo Ebergen. Computers without clocks. Scientific American, 2002, 287(2):62 69.
[11] Ivan E. Sutherland. Micropipelines. Communications
[12] Muller D E, Bartky W S. A theory of asynchronous circuits. Radical Philosophy, 2010, 14(5):204-243.
[13] J. Spars?. Asynchronous Circuit Design - A Tutorial. Microlab.ti.bfh.ch, 2006, 623:1-49.

[14] S. Smith and J. Di, âĂİDesigning asynchronous circuits using null convention logic(ncl),âĂİ Synthesis Lectures on Digital Circuits & Systems, vol. 4, no. 1-96,2009.
[15] M. Roncken, S. M. Gilla, H. Park, N. Jamadagni, C. Cowan and I. Sutherland. Naturalized Communication and Testing. Asynchronous Circuits and Systems (ASYNC), 2015: 77-84.
[16] W.Mallon, âĂİBounded bundled data,âĂİ Asynchronous Research Center, Portland State University, Tech. Rep., 2012. [Online]. Available: http://arc.cecs.pdx.edu/sites/all/uploads/Reports/Mallon
[17] H.Park, âĂİFormal modeling and verifcation of delay-insenitive circuits,âĂİ 2015.
[18] A. He, X. Liu, and Hongchen, âĂİStudy of 8-bit booth asynchronous multiplier based on two-phase handshake protocol with bounded bundle data,âĂİ Chinese Journal of Electronics, 2017.
[19] H. Park, A. He, M. Roncken. X. Song, and I. Sutherland, âĂİModular timing constraints for delay-insensitive systems,âĂİ Journal of Computer Science and Technology, vol. 31, no. 1, pp.77-106,2016.
[20] H. Pan, Q. Hong, J. Du, and P. pan, âĂİComparison of 2d mesh routing algorithm in noc,âĂİ in IEEE International Conference on Asic, 2012.
[21] Draper, Ghosh. Multipath E-cube algorithms (MECA) for adaptive wormhole routing and broadcasting in k-ary n-cubes[C]//Parallel Processing Symposium, 1992. Proceedings. Sixth International. IEEE, 1992:407-410.