2019 22nd Euromicro Conference on Digital System Design (DSD)

The Conical-Fishbone Clock Tree: A Clock-Distribution Network for a Heterogeneous Chip Multiprocessor AI Chiplet

Tomas Figliolia† and Andreas G. Andreou*
*Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, Maryland 21218
†Xilinx Corporation
Email: tomas.figliolia@gmail.com, andreou@jhu.edu

***Abstract*—The 2.5D nano-Abacus SOC is a neuromorphic CMP architecture for accelerated computing, hardware AI inference and machine intelligence. The SOC consists of a silicon interposer a 3D memory stack, a host FPGA and three (17.47 mm X 14.13 mm) computational chiplets fabricated in the Global Foundries 55nm CMOS technology. The latter employ two networks on chip, a token ring L1-NOC, and a switched circuit L2-NOC as well as a DDR DRAM PHY interface and a general purpose I/O port. In this paper we present a novel clock tree network, named Conical-Fishbone clock tree, that is employed in the low latency energy aware clock distribution networks of the chiplets. The many nets in a Fishbone clock tree consist of conical sections in an inverted cone architecture, with each resulting ring considered to be one clock tree. When a ring is excited at uniform intervals from the ring below, the symmetry in the circular characteristics of the wire makes the effect of reflections be exactly the same along any place in the wire.The designed clock tree allows low clock skew, consumes low power while offering the modularity to support a hierarchical design using standard CAD flows.**

I. Introduction

The 2.5D nano-Abacus SOC is a computer system architecture for a variety of processing algorithms in what we consider today as **Third Wave AI and Machine Intelligence** ranging from deep neural networks to linear and non-linear morphological processing, probabilistic inference using exact and approximate Bayesian methods. High performance energy aware processing is achieved through approximate computing and fixed point arithmetic in a variable precision (6 bits to 18 bits) architecture. The processing pipeline is implemented entirely using event based neuromorphic and stochastic computational primitives. A core "chiplet" is critical in the design of the nano-Abacus 2.5D SOC architecture (Figure 1) which comprises of a 36 mm x 50 mm Si interposer (5 metal layers, four stitched reticles) and "chiplets". The fabricated interposer is shown in in Figure 2 . Two of the chiplets are COTS components: (i) a Xilinx Zynq-7100 die for operating system support and high speed I/O and (ii) a high bandwidth memory stack (Tezzaron GEN4 3D DiRAM). Three additional heterogeneous chip multiprocessor "chiplets" designed in 55nm GF CMOS, implement mixed-signal programmable and reconfigurable processors for energy efficient, streaming processing such as for example wide motion area imagery [1]. The nanoABACUS 2.5D SOC is not an ASIC but rather a processor architecture that can be hardware or software re-configured with three of four "chiplet" processor designs, all with common physical standard footprint and logical interfaces. The floorplan of the chiplet-core and it physical interface are shown in Figure 3 and Figure 4 respectively. The nano-Abacus chiplet-core consists of a high bandwidth memory interface (DRAM Interface) , a Level-1 token ring network on chip (L1-NOC), a Level-2 switched circuit mesh network on chip (L2-NOC), and a general purpose input/output port (GPIO).

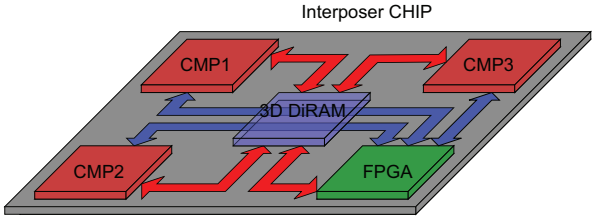

Fig. 1: The 2.5D nano-Abacus SOC depicting three mixed signal heterogeneous CMPs, an FPGA and a 3D-DiRAM stack.

For the communication in between nodes a bufferless mesh network was designed. This network is called the *L2 network* (L2 stands for level two). A standard interface (AMBA like) from the *L2 network* node to each processing unit is designed, one that does not rely on any particular clock (asynchronous interface), and hence the top level designs for the 128 PUs CMPs can be completely abstracted from the content of each of the PUs. Each of the PUs will have its own clock tree completely independent from any other clock in the system. This is the reason why a four phase handshaking interface was designed for the communication of each PU with the *L2 network*. This allows to place "dummy"

DOI 10.1109/DSD.2019.00111

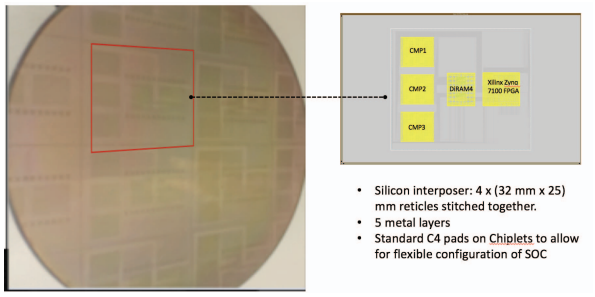

Fig. 2: The nano-Abacus interposer

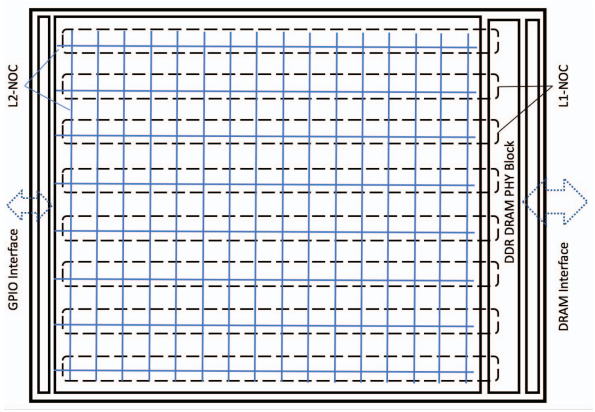

Fig. 3: The nano-Abacus chiplet core architecture

PUs for both main designs, and replace them later with the final desired PUs allowing the top level design to close timing independent of the PU processing engines. The *L2 network* can be seen in Figure 6. The connection to the FPGA is done through a special node on the Mesh network. The communication between the FPGA and its network node uses the same protocol any of the PUs uses with its own node, with the difference that serializers and deserializers needed to be used for the FPGA due to the extremely wide network bus, which is over 300 bits. Additionally, so that throughput to and from the FPGA could be increased, bidirectional pads were used for the communication in between the *L2 network* and the FPGA; this interface we call the GPIO interface.

Access to DDR memory is granted to each of the PU by incorporating an additional network. This network is called *L1 network*, and it allows communication between each of the PUs with the DDR memory through the *DDR DRAM PHY* block. This block translates read and write requests from the PUs to the 3D-DiRAM memory. This network is formed by independent token-ring networks on each of the rows in both designs. Each of these token-ring networks has a dedicated *DDR DRAM PHY* port. A total of eight different token-ring

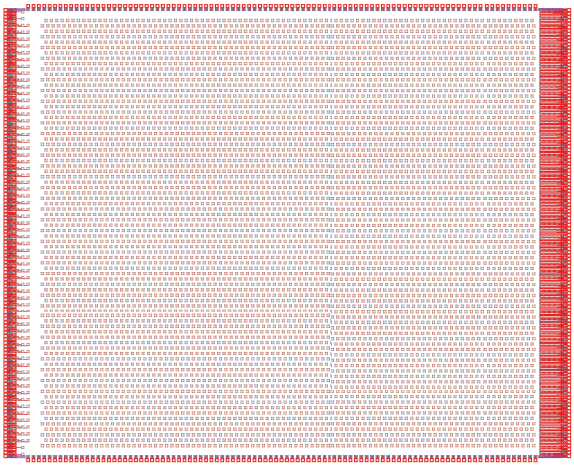

Fig. 4: The nano-Abacus chiplet-core padframe. Each chiplet connects to the interposer through 8024 pads (C4) on the bottom of the chip. Additional wire bonding pads on the periphery of the chiplets allow for probing and using each chiplet without 2.5D integration.

networks are incorporated on this *L1 network*, and each PU communicates with its *L1 network* again through a four phase handshake interface. The *L1 network* can be seen in Figure 3.

When communicating outside of the chips, it can either be done through the DDR memory or through the *L2 network* connecting to the on-interposer FPGA and the GPIO interface (see Figure 1). The *L2 network* has an additional node, in addition to the 128 previously mentioned nodes. This node has access only to the *L2 network*, and the processing unit assigned to this node is the external FPGA to which the *L2 network* connects through the left side pads of the chip. The onboard FPGA is able to send and receive packets to and from the *L2 network* using the four phase handshaking protocol, and also has its own address, making the communication between FPGA and PUs completely transparent. The utilization of this asynchronous protocol in communicating the FPGA with the *L2 network* is very convenient as it does not require the equalization of any of the data bits lines with respect to a received clock.

Each of the CMP chips is $17466\mu m$ by $14133\mu m$ in size. Because of these large dimensions, as indicated earlier, it is impossible to expect the *Place & Route* tool to create clock trees with very low skew and slew. It is for this reason that a custom architecture was designed for the clock trees which is the subject matter of this paper. Long clock tree cells of size $\approx 1500\mu m$ by $\approx 50\mu m$ were designed. These cells take a clock input and generate several clock outputs along one or both long sides, with a skew of only 30ps, allowing clock speeds of up to 1.25GHz to be propagated through these cells.

These cells allow clock trees to be built local to the outputs of these clock tree cells, making these clock trees much smaller and more reliable. In Figure 6 and 5 the different clock cells that allow both networks to be completely in sync can be seen. Similar cells were used for distributing asynchronous reset to the network.

Fig. 5: ***L1 network*** **for the 128 PUs chip.** Eight different token-ring networks communicate with the *DDR DRAM PHY*. The communication between the *DDR DRAM PHY* and the DDR is done through two DDR buses, where each bus is composed of 66 signals, 64 data signals and two complementary clocks. The required pads communicating with the DDR memory are placed on the right side of the chips.

For both networks on chip, the clock frequency is 300MHz, and this clock is distributed from a vertical clock tree cell, to each of the clock tree cells present for every row in the network. Along with the clock, an asynchronous global reset signal (RESET) needs to be distributed to the network on chip (NOC) as well. This reset signal employs the same Conical-Fishbone topology. With respect to the DDR DRAM PHY interface, four different clocks are employed. A $1.25GHz$ clock ($0.8ns$ period) is used to send data from the chip to the DRAM memory. This clock is also sent to the DDR DRAM. Another clock for the DDR DRAM PHY interface is the previously mentioned clock, divided down to a $2.4ns$ period clock. These two clocks need to

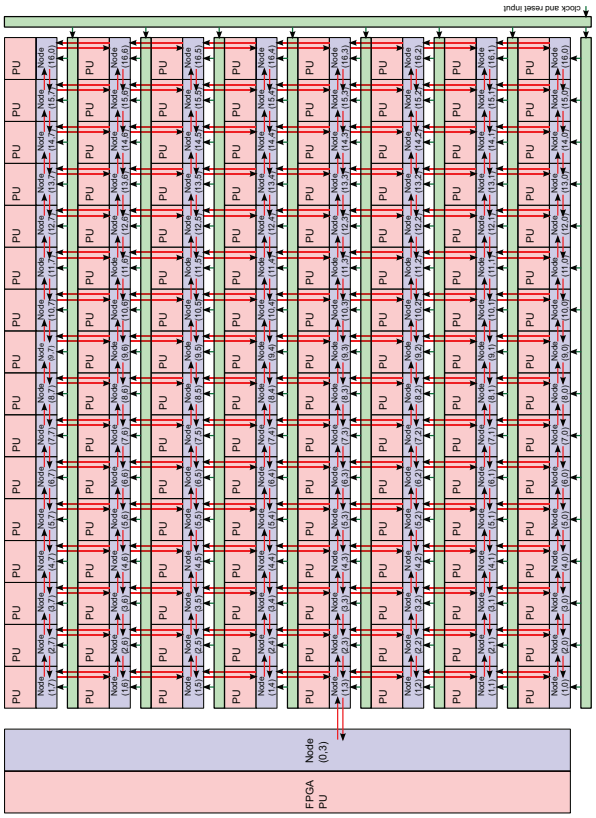

Fig. 6: ***L2 network*** **for the 128 PUs chip.** Communication to the FPGA is done through the $(1, 3)$ node. The communication between node $(0, 3)$ and the FPGA is done through bidirectional pads placed on the left of the chip and the GPIO interface. Each of the packets in the network consists 256 bits of data, making it really difficult to have that same number of pads in that communication. A serializer and deserializer are being used to send and receive between the *N2 network* and the FPGA through the GPIO port. The green blocks distribute reset and clock signals.

be in phase, and it is for this reason that we don't have two clock tree cells for these two clocks, the division is done local to the output of the clock tree cell. The DDR DRAM memory provides the clock for the DRAM to chiplet data transfers. This clock is also a $1.25GHz$ clock and just like in the previous case, it is divided down to a clock with a $2.4ns$ period. Both pair of clocks need to be available on both sides of the clock cell, therefore the output from one side of the clock tree cell is routed to the middle of the cell and then split in two.

II. System on Chip Clock Architectures

Clock distribution networks are crucial sub-systems in general purpose microprocessors in the x86 family and PowerPC [2], [3] as well as application specific processors such as the Anton supercomputer [4]. Skew

and slew are the fundamental tradeoffs in the design of System on Chip (SOC) clock distribution networks. This is especially true for SOCs that occupy a large silicon area where multiple clock tree levels must be employed. High slew is desirable because the relative error $Slew_{error}/Period$ is lower, hence reducing the impact on the maximum operating frequency of the clock. For example, let's assume a $1GHz$ clock tree needs to be synthesized and two cases are analyzed for the required slew, $80ps$ and $200ps$. Robustness in the design can be analyzed using these two slew cases. Let's now consider variability in the clock tree drivers reduces the slew by 50%. The first case is not problematic, the period instead of $1ns$ will become $1.04ns$. The clock speed is reduced by only $\approx 40MHz$. In the second case, where $200ps$ slew is chosen, using the same mismatch, there will be a $100MHz$ speed reduction and the clock. A common engineering solution to this challenge is the implementation of H or Fishbone clock trees (see Figure 7). These clock trees are readily synthesizable using standard CAD tool flows, and they often achieved good skew and slew simultaneously even in large designs.

The latter structures rely on very powerful clock tree drivers and on specific placement for the drivers and wires connecting them. Topology is important to the latter designs, necessitating a "flat" Place & Route, because the positions of the clock drivers and wires are not easily changeable due to geometrical constraints. From Figure 7 it can be seen for the H clock tree that as we increase the number of levels in the clock tree, the areas that can possibly be used as blocks in a hierarchical design become smaller and smaller, making a flat synthesis the only viable option for most cases (area in green). On the other hand the Fishbone clock tree, gives more freedom if modularity is desired in a design. However, with the Fishbone tree design the place and route tools are often not efficient and the dimensions of the rectangular area used for the tree becomes too large. Hence it is more convenient to custom design Fishbone clock trees using the standard CAD tool flows. The Fishbone tree is a good option for the clock distribution network in the nano-Abacus chiplet core, but unfortunately it does not have robust skew properties at the output drivers. In this Fishbone clock tree, every column of drivers has its inputs shorted all together as well as its outputs, but not every driver output sees the same impedance in the line, making it very difficult to achieve a very low skew. Even if terminations were applied at the two ends of each intermediate net to reduce the effect of reflections, the skew problem wouldn't be solved.

III. The Conical-Fishbone Clock Tree

In this section we describe the new clock tree alternative, one that makes sure that the impedance seen at the output of each of the active drivers is exactly the same in the same clock tree level. The architecture we propose is based on the shape of an inverted cone,

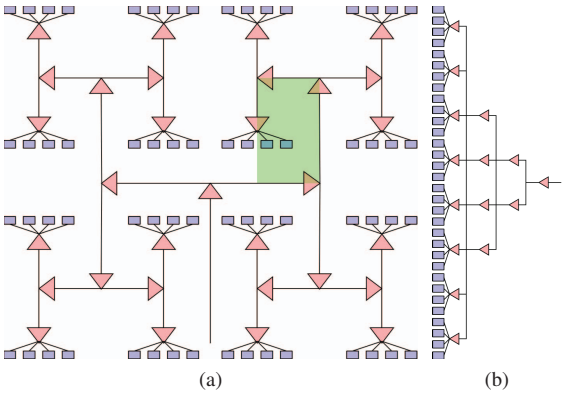

Fig. 7: H-tree for clock distribution (left); Fishbone clock tree (right). All the blocks in blue are all the leaf cells for the clock tree. On green we can see the area in the H tree that can be used as a block in a hierarchical design. As the number of clock tree levels increases, the green area becomes smaller, making it more difficult achieve modular and hierarchical design.

shown in Figure 8. If several cross sections are created in the inverted cone, and we consider each of these resulting rings to be one of the many nets in a Fishbone clock tree, we see that if a ring is excited evenly from the ring below, the circular characteristics of the wire will make the effect of reflections be exactly the same along any place in the wire. This idea is the one that will allow us to achieve both low skew and modular and hierarchically synthesized design.

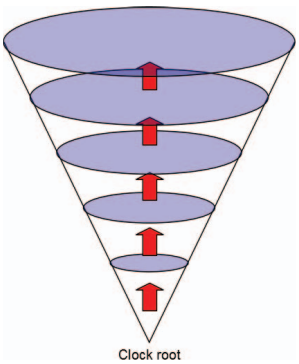

Fig. 8: Inverted cone shape used as inspiration for the design of a new clock tree architecture. Every circle gets excited by the circle below it. The clock root is the tip of the cone.

Inspired by the inverted cone shape, Figure 9 shows the the Conical-Fishbone tree architecture. The resemblance to both an inverted cone and to the Fishbone clock tree is evident. Driven by the clock at the tip of the inverted cone and subsequent levels of the clock tree hierarchy are driven in a geometrical

progression. The first ring of the tree is driven at four equidistant places. If the diameter of *RING 1* is x, then every time a clock tree level is added, the resulting ring increases by x. There is a linear relationship between the span of the tree and the number of levels in the tree. Hence the following *RING 2* will be excited in eight different equidistant places; to maintain symmetry and equivalent load on each point where a ring is excited, two additional drivers are added, the ones in blue. These drivers have their output floating, they are just used to equalize the load along every ring. As the number of clock tree levels increases, the number of active buffers used to excite the following ring is $2.Ring\_n + 2$, where $Ring\_n$ is the ring number. When the layout of this clock tree is done the distances from the ring to the input of the exciting drivers from the same tree level are designed to be exactly the same for all of them. Thus all of the points in each of the rings where the ring is excited and/or read, see exactly the same impedance. The latter property is the main characteristic of the Conical-Fishbone that yields low slew rate.

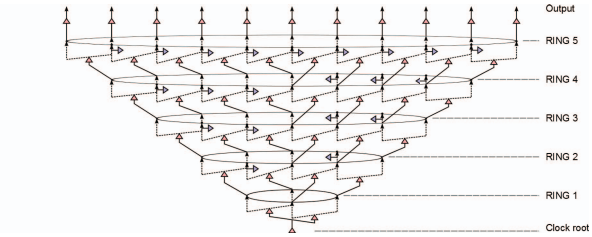

Fig. 9: Inverted cone shape used as inspiration for the design of a new clock tree architecture. Every conical section gets excited by the section below it. The clock root is at the tip of the cone.

A. Clock distribution in the DDR DRAM PHY interface block

DDR DRAM PHY IP interface is a high speed mixed-signal design with strict physical layout and placement constraints. The design of the Conical-Fishbone tree is no exception! Four clocks are needed in the DDR DRAM PHY interface block, and two of them can be generated by the other two. To avoid unnecessary area use he area use the clock divided versions of clocks are generated locally from the output of the high frequency clock tree cell. This allows the reduction by half of the area used for the clock trees in the DDR interface block. Figure 11 shows the augmented clock tree cells used in the DRAM interface block. Along with the two clocks we can find a clock divider and a buffer. The clock divider runs a counter that counts from 0 to 2, and the buffer is placed to compensate for the delay introduced by the clock divider. The reset signal is used to reset the clock dividers to a default state so that all of

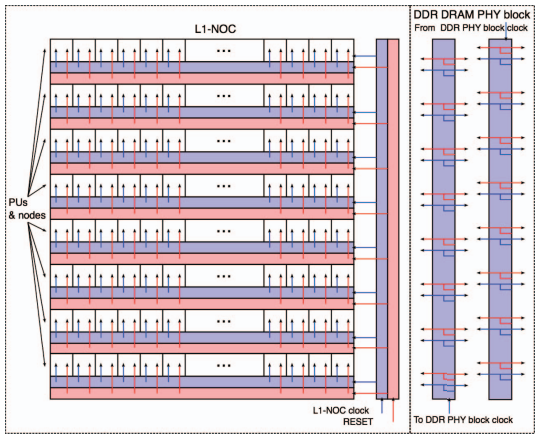

Fig. 10: The Conical-Fishbone clock trees in the nano-Abacus core chiplet. On the NOC side, the blue cells are the ones delivering the $300MHz$ clock, and the red cells are delivering the global reset signal. On the *DDR DRAM PHY interface block* side two clock tree cells are used for the clock used to send data to the DDR memory and for the clock used to read the data back.

the clock outputs are in phase. Because the clock divider counter has a period of 3, from one clock output to the next one, the reset signal is registered three times. This will ensure that after a certain number of clock cycles, all of the clock tree outputs will be completely in phase.

The size of each of the clock tree cells for the DDR DRAM PHY interface block is approximately $13.44mm$ by $50\mu m$. Each of the long sides has 64 clock outputs for both fast an divided clocks. Figure 12 depicts the outputs for the fast clock in the DDR interface block cells. For the distance of $13.44mm$, the entire length of the chiplet-core the maximum (worst case scenario) skew is $31.8ps$.

IV. Conclusions

The chiplets with the Conical-Fishbone clock tree network are fabricated in the Global Foundries 55nm CMOS technology. There is four chiplets in the nano-Abacus family, each of the chiplets is $17466\mu m$ by $14133\mu m$. Because of these large dimensions, it is impossible to expect the place and route tool to create clock trees with very low skew and slew. The Conical-Fishbone clock tree architecture presented in this paper achieves the design objectives. Long clock tree cells of $\approx 1500\mu m$ by $\approx 50\mu m$ were custom designed. These cells take a clock input and generate several clock outputs along one or both long sides, with a skew of only 30ps, allowing clock speeds of up to 1.25GHz. These cells allow clock trees to be built local to the outputs of these clock tree cells, making these clock trees much smaller, hence more energy efficient and more robust.

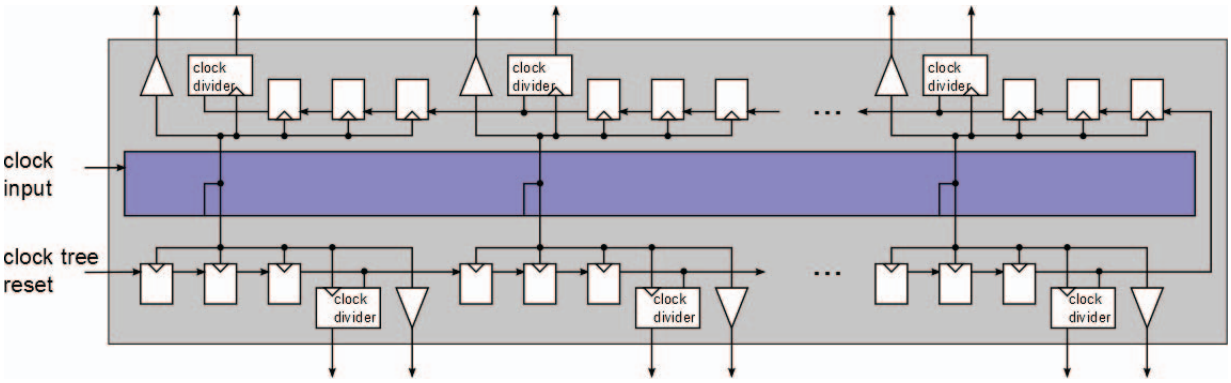

Fig. 11: Circuit details of the Conical-Fishbone clock cell.

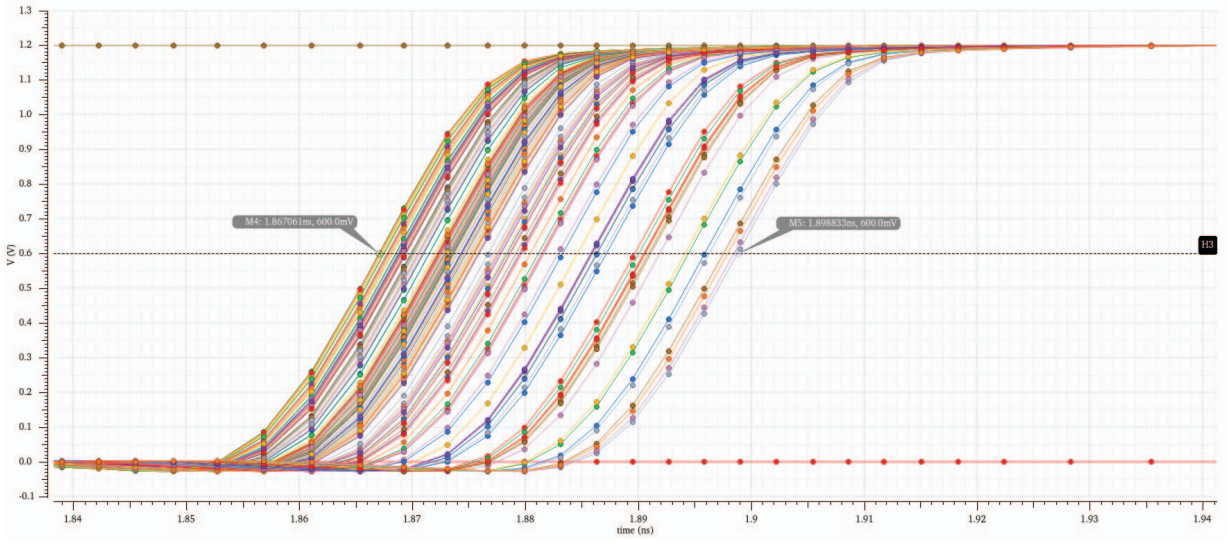

Fig. 12: Simulated outputs of the fast clock propagated through the clock cell in the DDR DRAM PHY interface. For all of the clock outputs along both sides of the cell, extending to $13.44mm$, the worst case skew is $31.8ps$. This simulation has been done considering all the parasitics of the clock tree layout.

ACKNOWLEDGMENT

This work was supported by DARPA UPSIDE project HR0011-13-C-0051 through BAE Systems; we are grateful to Dr. Mike Graziano and Dr. Louise Sengupta who managed the project the first two years while at BAE systems, for their support and encouragement.

REFERENCES

[1] A. G. Andreou, T. Figliolia, K. Sanni, T. S. Murray, G. Tognetti, D. R. Mendat, J. L. Molin, M. Villemur, P. O. Pouliquen, P. M. Julián, R. Etienne-Cummings, and I. Doxas, "Bio-inspired System Architecture for Energy Efficient, BIGDATA Computing With Application to Wide Area Motion Imagery ," in *Proceedings of the 2016 IEEE 7th Latin American Symposium on Circuits and Systems (LASCAS)*, 2016, pp. 1–6.

[2] J. Schutz and C. Webb, "A scalable X86 CPU design for 90 nm process," in *2004 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2004, pp. 62–513.

[3] J. Warnock, J. Keaty, J. Petrovick, and J. Clabes, "The circuit and physical design of the POWER4 microprocessor," *IBM Journal Of Research And Development*, vol. 46, no. 2, pp. 27–51, 2002.

[4] R. O. Dror, J. P. Grossman, K. M. Mackenzie, B. Towles, E. Chow, J. K. Salmon, C. Young, J. A. Bank, B. Batson, M. M. Deneroff, J. S. Kuskin, R. H. Larson, M. A. Moraes, and D. E. Shaw, "Exploiting 162-Nanosecond End-to-End Communication Latency on Anton," in *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2010)*. IEEE Computer Society, Nov. 2010.