INTEGRATION, the VLSI journal 43 (2010) 188–201

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Wave-pipelined intra-chip signaling for on-FPGA communications

Terrence Mak [a,*], Pete Sedcole [a], Peter Y.K. Cheung [a], Wayne Luk [b]

[a] Department of Electrical and Electronic Engineering, Imperial College London, Exhibition Road, London, UK
[b] Department of Computing, Imperial College London, Exhibition Road, London, UK

ARTICLE INFO

Article history:
Received 14 August 2009
Received in revised form
17 November 2009
Accepted 19 January 2010

Keywords:
FPGAs
On-chip communication
Wave-pipelining
Interconnect model
Throughput
Source synchronous

ABSTRACT

On-FPGA communication is becoming more problematic as the long interconnection performance is deteriorating in technology scaling. In this paper, we address this issue by proposing a novel wave-pipelined signaling scheme to achieve substantial throughput improvement in FPGAs. A new analytical model capturing the electrical characteristics in FPGA interconnects is presented. Based on the model, throughput and power consumption of a wave-pipelined link have been derived analytically and compared to the conventional synchronous links. Two circuit designs are proposed to realize wave-pipelined link using FPGA fabrics. The proposed approaches are also compared with conventional synchronous and asynchronous pipelining techniques. It is shown that the wave-pipelined approach can achieve up to 5.7 times improvement in throughput and 13% improvement in power consumption versus conventional delay-based on-chip communication schemes. Also, trade-offs between power, throughput and area consumption between the proposed and conventional designs are studied. The wave-pipelining approach provides a new alternative for on-FPGA communications and can potentially become a promising solution to mitigate the future interconnect scaling challenge.

1. Introduction

The continuous scaling of process technology presents a challenge on interconnects design, as predicted by the International Technology Roadmap for Semiconductors (ITRS). The gap between interconnection RC delay and the gate delay will be increasing exponentially (9:1 with the 65 nm technology according to ITRS 2006 report [1]). This is especially true for global interconnections, which will dominate the overall speed performance of a chip. In addition, the interconnect capacitances account for a large proportion of the total chip capacitances. As a results, the energy consumed by interconnects can even be higher than that of the logic circuits.

The interconnect challenge is exacerbated in FPGA with its reconfigurable interconnect architecture. Interconnections in FPGAs are generally constructed from segments of interconnect fabrics which are slower and dissipating more energy when comparing to ASIC custom designs. This is especially the case for global interconnections, which span a large physical distance across the chip. Although more components and silicon can be fitted into a single chip, intra-chip communication would introduce significant performance hindrance and alternative signaling technique may be needed to mitigate the interconnect challenge.

Despite of the irregularity and idiosyncratic nature of FPGA long interconnections, buffers were embedded at switches to speed up the signal propagation. With the help of these buffers, the long interconnections can be modeled as multiple stages of RC transmission line, which facilitate the realization of wave-pipelined signaling [2]. In wave-pipelined signaling, multiple bits are allowed to traverse simultaneously along the line, thus a significant throughput improvement can be obtained. For many complex applications, bandwidth is the main concern for communication between modules or processors. A conventional interconnect has its bandwidth dictated by the RC time constant (or characteristic impedance) of the wire, thus limiting the data throughput. Using wave-pipelined signaling, as will be shown later in this paper, the data throughput can exceed the limit imposed by the RC time constant.

In this paper, we propose a pulsed wave signaling (or namely wave-pipelining) design strategy to mitigate the interconnect challenge in FPGAs. The contributions of this paper are:

1. Propose a new interconnection model for global routings in FPGAs. The new model generalizes the irregular interconnection circuits as multiple buffered interconnect stages which can be applied in FPGA architectural evaluation and prediction

* Corresponding author.
E-mail addresses: t.mak@ic.ac.com, t.mak@ic.ac.uk, t.mak@imperial.ac.uk (T. Mak), pete.sedcole@gmail.com (P. Sedcole), p.cheung@ic.ac.uk (P.Y. Cheung), wl@doc.ic.ac.uk (W. Luk).

0167-9260/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2010.01.002

for interconnect performance for different technology processes (Section 3).
2. Derive the fundamental delays and throughput for FPGA interconnects. The expressions can be applied to analyze delay, throughput and power consumptions for different signaling strategies, such as delay-based signaling, wave signaling, register pipelined link. This analytical model provides the basis for studying performance benefits and shortcomings for different signaling approaches in FPGAs (Section 4).
3. Propose two circuit designs for implementing analog wave signaling in FPGAs. These two approaches improve the signaling throughput using phase-adaptation and oversampling techniques. The new methods are evaluated through actual implementations on a Xilinx FPGA device. Trade-offs between power, speed and area are studied and comparisons with conventional synchronous and asynchronous pipelining techniques are also investigated (Section 5).

2. Related work

Wave-pipelined logic circuits was originally proposed in 1969 [3] and it has been continuously developed for new techniques and applications throughout the years [4]. The original idea was to allow combinational logic to process new data set before the previous data set reached the registers, such that few combinational logics sit idle. Wave-pipelining, thus, suggests simultaneous existence of multiple data bits in a combinational logic data path. Realization of wave-pipelining circuit on FPGA has been reported in [5] by Boemo et al. A wave-pipelined multiplier circuit has been realized by constructing a well structured network topology of LUTs. Various techniques for implementation of wave-pipelining circuits on FPGA can be found in [6]. However, due to variability of logic delay and the design complexity, wave-pipelining circuit is difficult to put into real practice.

On the other hand, interconnects with repeaters are fairly regular circuits. Recently, the focus of wave-pipelining had shifted from the logic to the interconnection circuits and a number of interconnect wave-pipelining design for ASIC has been proposed [7–10] in order to achieve a higher throughput of interconnections. On the wave-pipelined interconnect, a new data bit is sent before the previous data bit reaches the sink, the maximum bit rate is not limited by the wire delay. Instead the minimum data pulsewidth that can be sustained on the wave-pipelined interconnects, which is smaller than the interconnect latency, determines the maximum interconnect throughput. As a result, there can be a significant enhancement in the interconnect throughput through simultaneously presence of multiple bits on the interconnect.

While these new signaling techniques are promising to improve the interconnect throughput, it is more challenging to implement analog wave signaling in FPGA than in its ASIC counterparts. Since the FPGA architecture is a pre-fabricated digital platform with highly constrained flexibility, it prohibits a straightforward realization of analog circuits or modifications of the physical architecture. Also, specially designed interconnects and analog-digital interfacing circuits are required for realizing the wave-pipelined links.

In this paper, we are aiming to exploit the interconnect throughput in FPGAs and to develop reliable high throughput on-chip communication links. To our knowledge, this work is the first to propose using wave-pipelined signaling for on-FPGA communication and present new design methodologies for its realization in FPGAs.

3. Global interconnect model for FPGAs

In order to exploit the interconnect throughput in FPGA, it is important to develop a detail model to characterize the electrical properties and predicting the throughput performances. Previous FPGA-based interconnect models, such as in [11,12], focuses on delay estimations and architectural explorations. These models adopted a simplified electrical characterizations and are not able to be extended to analyze the wave-pipelined throughput. In this section, we present a new FPGA-specific global interconnect model. This model generalizes FPGA interconnections into multiple stages of buffered line and provides an analytical solution for studying wave-pipelined signaling.

3.1. Multiple-stage model

Consider an typical island-style FPGA architecture [12,13] which comprises of a 2D array of logic blocks or slices that can be interconnected via programmable routing. A global interconnection comprises of a combination of programmable interconnects that is physically spanning over a long distance from the source to the sink. Particularly, the interconnection in FPGAs is much more complicated than conventional long line found in ASIC design. Fig. 1(a) shows a typical example of an interconnection. It starts from a source, which is the output of a logic block, and is connected to a short wire segment via interconnect switch. There are a number of ways to realize the interconnect switch. The simplest way is by using a pass transistor [14,12]. Other approaches such as stages of multiplexer [15] and transmission gates can also achieve the same functionality with a higher speed. Although a pass transistor is shown in the figure, it can be replaced by other types of logics, such transmission gates and the closed-form analytical solution remains the same with different resistance and capacitance values. Most modern FPGAs have wire segments of different lengths that can be used to implement interconnections with different requirements. For example, in Xilinx Virtex-4 series FPGAs, there are three types of wire segments, which are with length 1, 3, 6 (Hex-line) and 24 units. Through programmable interconnect switches, wire segments can be connected to form a long range route. Since constructing long range interconnection by aggregating multiple short wires segments increases energy dissipation, long wire segments are frequently being used for realizing global interconnection. Typically these long wire segments will span a long distance (for example 24 tiles in Xilinx Virtex-4 [16]).

In addition, different interconnect routing architectures have been proposed. Examples are the tapped branching lines from long wire, known as early-turns [15]. They are proposed to provide flexible routing to other channels. Also, various short-cut for the interconnect switches are proposed in [14], such as buffers inserting into the long wire to reduce the delay.

FPGA interconnect is a fairly complex circuit, which comprises of multiple buffers, switches, pass-gates and multiplexers, whereas interconnect in ASIC is rather simple. In order to obtain an analytical model for these long line, the FPGA interconnect structure has to be abstracted and simplified.

The interconnection depicted in Fig. 1(a) can be generalized as a network of resistance and capacitance (RC) pairs, with buffers in between. This is depicted in Fig. 1(b). A simple lumped model can be used to model the short interconnect segment. More complicated models, such as β and T models [17] can also provide a good approximation with higher accuracy. The switching logics and routing branchings can be modeled by RC circuits as in [18] and equivalent transistor parasitics can be found by testing circuits. To further generalize the model, we can divide the whole

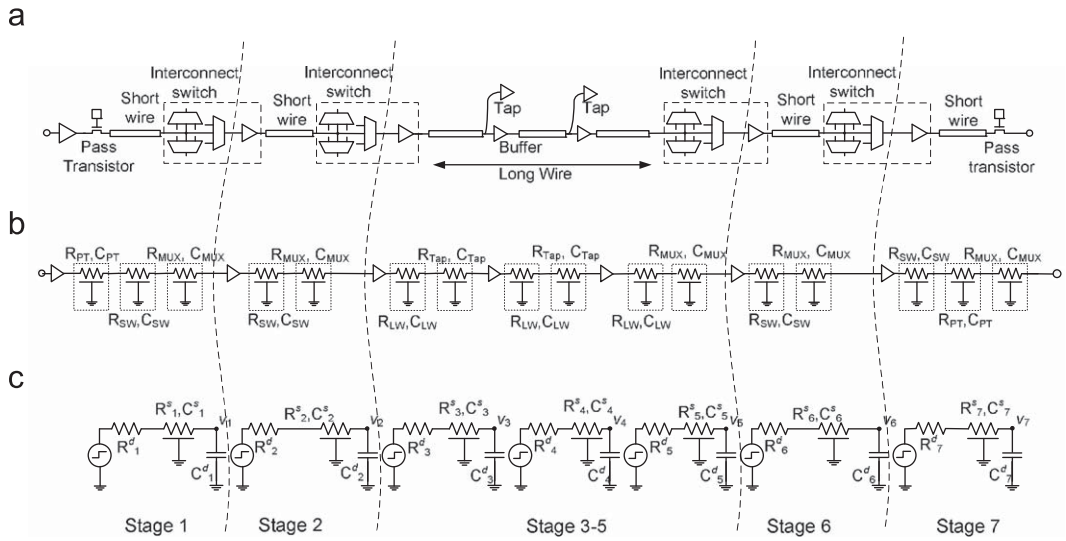

Fig. 1. A model of a typical global interconnection in FPGAs. (a) The schematic of an interconnection comprises of short and long wires and which are connected through switching points. (b) Circuit model of the corresponding interconnection. (c) Switch-level RC circuit for the interconnection as a chain of segments driven by drivers.

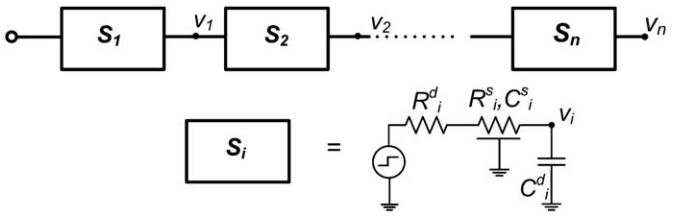

Fig. 2. A general n-stage block diagram of an interconnection model.

interconnection into n stages. Each stage is either driven by a driver at each programmable switch or by a buffer in the long wire. By Thevenin's theorem, the combinations of RC pairs at the i-th stage can be summarized into R_i^s and C_i^s, respectively. Following [17,19], we model the driver or buffer by a switch-level RC circuit and, thus, the input resistance and load capacitance of the driver are denoted as R_i^d and C_i^d, respectively.

Fig. 2 shows a general n-stage model which generalizes the complex interconnection into a simple multiple-stage link. The stage unit, S_i, models the i-th buffered interconnect with the electrical parameters, R_i^d, C_i^d and R_i^s, C_i^s, for characterizing the input buffer and interconnect parasitic. The detailed mathematical expression will be shown at Section 3.2. By resolving the waveform at each of the stages, we can study and evaluate the whole interconnection analytically. In the following subsection, a simple closed form approximation will be reviewed and applied to approximate the waveform at each of the stages.

3.2. Waveform approximation at each stage

The waveform at each stage of the interconnection can be modelled by RC lines. Step response for RC lines is well studied in the literatures [17]. In general, there are two types of model can be used, which are the distributed and the lumped models. The lumped RC model is a pessimistic model for a RC line and that a distributed RC model is more realistic. It is well known that for 1.0RC, the V_{OUT} is only charged up to 63% of V_{dd} for the lumped model versus the distributed model yield a 90% of V_{dd} [17].

The two RC models for buffered lines are presented in the following. Suppose, variable v_i is denoted as the fraction of the supply voltage at the far end of the i-th stage. Thus $v_i = V_i(t)/V_{DD}$. The response of a lumped RC network is

$$v_i(t) = \frac{1-e^{-t/R_i^s C_i^s}}{V_{dd}} \tag{1}$$

The response of the distributed network is harder to calculate and there is no closed-form solution exists for this equation. Only approximation such as the formula presented in the following [17]:

$$v_i(t) = 2erfc\left(\sqrt{\frac{R_i^s C_i^s}{4t}}\right), \quad t \ll R_i^s C_i^s \tag{2}$$

$$= 1.0 - 1.366e^{-2.5359t/R_i^s C_i^s} + 0.366r^{-9.4641t/R_i^s C_i^s}, \quad t \gg R_i^s C_i^s \tag{3}$$

These equations are difficult to use for ordinary circuit analysis. Alternative approximation given by Sakurai in [19] provides a closed-form expression for distributed RC line, which is more convenience to be applied for interconnect modeling. In Sakurai's approximation, the output of a RC line for a step function is given by

$$v_i(t) = 1 - \sum_{j=1}^{\infty} k_{i,j} e^{-t/\sigma_{ij}} \approx 1 - k_{i,1} e^{-t/\sigma_{i,1}} \tag{4}$$

To simplify the expression, we drop the index j, thus we have

$$v_i(t) = 1 - k_i e^{-t/\sigma_i} \tag{5}$$

where k_i is a coefficient and σ_i is time constant for charging or discharging the buffered line [19]

$$\sigma_i = R_i^d C_i^d + R_i^d C_i^s + R_i^s C_i^d + 0.4 R_i^s C_i^s \tag{6}$$

and

$$k_i = 1.01 \frac{R_i^d C_i^s + R_i^s C_i^d + R_i^s C_i^s}{R_i^d C_i^s + R_i^s C_i^d + \frac{\pi}{4} R_i^s C_i^s} \tag{7}$$

Also a more general expression that suits for FPGA interconnects can be provided, which is applicable for some stages that has a pass transistor. It is well-known that a pass transistor implemented with a NMOS device is not effective at pulling a node to V_{DD}. When the pass-transistor pulls a node high, the output only charges up to $V_{DD} - V_{Tn}$ [20]. Therefore, we let γ_i be the discount factor for the i-th stage, $\gamma_i = (V_{DD} - V_{Tn})/V_{DD}$. Note that using pass transistor has been abandoned in many commercial FPGA architectures. Instead, multiplexer and transmission gate

are used in modern architecture for switch and connection block design. For these cases, we can simply let the discount parameter $\gamma = 1$. Then, we can have the general approximation for the rise and fall of v_i at the i-th stage as

$$\text{(Rise)} \quad v_i = \gamma_i - \gamma_i k_i e^{-t_i/\sigma_i} \tag{8}$$

$$\text{(Fall)} \quad v_i = \gamma_i k_i e^{-t_i/\sigma_i} \tag{9}$$

The complex interconnection in FPGA has been generalized as a series of buffered segments. Methodologies that was used for ASIC to study global interconnects can be applied here to analyze the overall delay and throughput in an FPGA interconnects. With this model, the delay and throughput for the overall link can be derived which will be discussed in the following section.

3.3. Delay of long lines in FPGA

Interconnection is modeled as a multiple-stage link and by resolving the waveform at each of the stage. The delay of the link can be derived analytically. Consider an interconnection that is partitioned into n stages of buffered line. The Thevenin's equivalent RC at the i-th stage are denoted as R_i^s and C_i^s, and the input resistance and load capacitance of the driver are denoted as R_i^d and C_i^d, respectively.

It is assumed that delay in each stage is given by the 50% rise time (or the fall time)[1] of the interconnect segment. The delay of a driver at the i-th stage is denoted as δ_i. Particularly, the delay δ_i will also include the delay of logics at the interconnect switch. Thus, the parameter δ_i can be unique to each stage as all the interconnect switches can be different, subjected to the routing of the interconnections. The time t_i for the i-th stage to reached v_i can be derived from Eq. (8) as

$$t_i = \sigma_i \ln\left(\frac{\gamma_i k_i}{\gamma_i - v_i}\right) \tag{10}$$

where t_i is the time required for the output of the i-th stage to reach a fraction v_i of the full-scale voltage. Alternative delay models are available in the literature, such as Elmore Delay [21]. The Elmore delay model is a RC lumped network model, provides a pessimistic estimation. The Bakoglu's delay model [17] is very similar to the equation derived here based on Sakurai's model. The total delay for the voltage at the n-th stage reached v_n is given by

$$T_n^{\text{Delay}} = \sigma_n \ln\left(\frac{\gamma_n k_n}{\gamma_n - v_n}\right) + \sum_{i=1}^{n-1} \sigma_i \ln\left(\frac{\gamma_i k_i}{\gamma_i - 0.5}\right) + \sum_{i=1}^{n} \delta_i \tag{11}$$

The delay-based throughput Γ_n^{Delay} for an interconnection with n stages is given by the inverse of the propagation delay as

$$\Gamma_n^{\text{Delay}} = \frac{1}{\sigma_n \ln\left(\frac{\gamma_n k_n}{\gamma_n - v_n}\right) + \sum_{i=1}^{n-1} \sigma_i \ln\left(\frac{\gamma_i k_i}{\gamma_i - 0.5}\right) + \sum_{i=1}^{n} \delta_i} \tag{12}$$

From the above equations, delay has a linear relationship with time constant σ. With technology scaling, coefficient k converges to 1 and δ decreases as the reduction of gate delay. However, time constant σ will increase significantly with the new technology processes, simply because the explosive growing of wire resistance [22].

[1] It is common to use rise time to refer to both rise and fall times. This nomenclature will used in the rest of the paper.

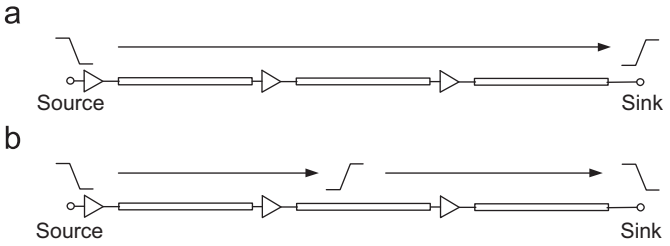

Fig. 3. Signal transmission using (a) delay-based and (b) wave-pipelined schemes.

4. Wave-pipelined signaling

Despite of the large delay of long interconnection in FPGAs, it has been shown that the intrinsic interconnect architecture benefits the realization of wave-pipelining, which allows multiple bits simultaneously traversing along the line [2]. Pre-fabricated buffers have been inserted into the switches and long interconnect which facilitates and preserves the pulse waveform and propagation throughout the lines. The overall throughput (or data rate) of the line can therefore be significantly increased. In the following, the fundamental wave throughput and power dissipation models will be presented.

4.1. Throughput of wave-pipelined links

Conventional delay-based signaling uses a synchronous mechanism that a new bit of data will be transmitted from the source only if the last bit of data has arrived the sink. Therefore, the throughput of interconnect for this approach will be the reciprocal of the interconnect delay, as shown in Eq. (12). Fig. 3(a) shows the mechanism of delay-based signaling. A new data bit, as the falling edge in the figure, is started to be transmitted when the previous data bit has arrived at the sink.

The throughput of an interconnect can be increased by increasing the data injection rate. The source injects a new data bit into the link even the previous data bit has not arrived at the sink (as shown in Fig. 3(b)). In this way, the throughput of the link can be greatly improved, as multiple bits are traversing along the line simultaneously. However, there exist an upper bound for the throughput, which guarantees that data bits do not interfere and corrupt each other along the line and data will be transmitted reliably. Particularly, when a pulse signal propagates along a line, the amplitude of the pulse attenuates. This is due the discrepancy between the charging and discharging time of an interconnect. For example, Fig. 4(a) shows a stimulus waveform that is injected into the interconnect. This input is then transformed into a waveform in Fig. 4(b), which is observed at the end of first interconnect stage. It has a smaller pulse width and pulse amplitude is reduced to v_1, where $v_1 < Vdd$. When the pulse propagates through n stages, the pulse width and amplitude are further reduced, as shown in Fig. 4(c). It is important to ensure that the amplitude of the pulse at the sink is large enough,[2] so that a valid data can be registered at the output.

Mathematically, this can be determined by evaluating the *maximum data rate* which is the reciprocal of minimum pulse width from the source, such that the amplitude of this pulse at the end of n-th segment can be pull up to v_n. If the data rate is higher than that of the maximum data rate, the amplitude of the pulse at the sink will be less than v_n, which will not be registered correctly and may cause signaling error.

[2] It has been suggested that v_n equals to 90% of Vdd in [23], which is a pessimistic assumption to ensure reliable data transfer and a transistor switch is fully turned off until the waveform reaches.

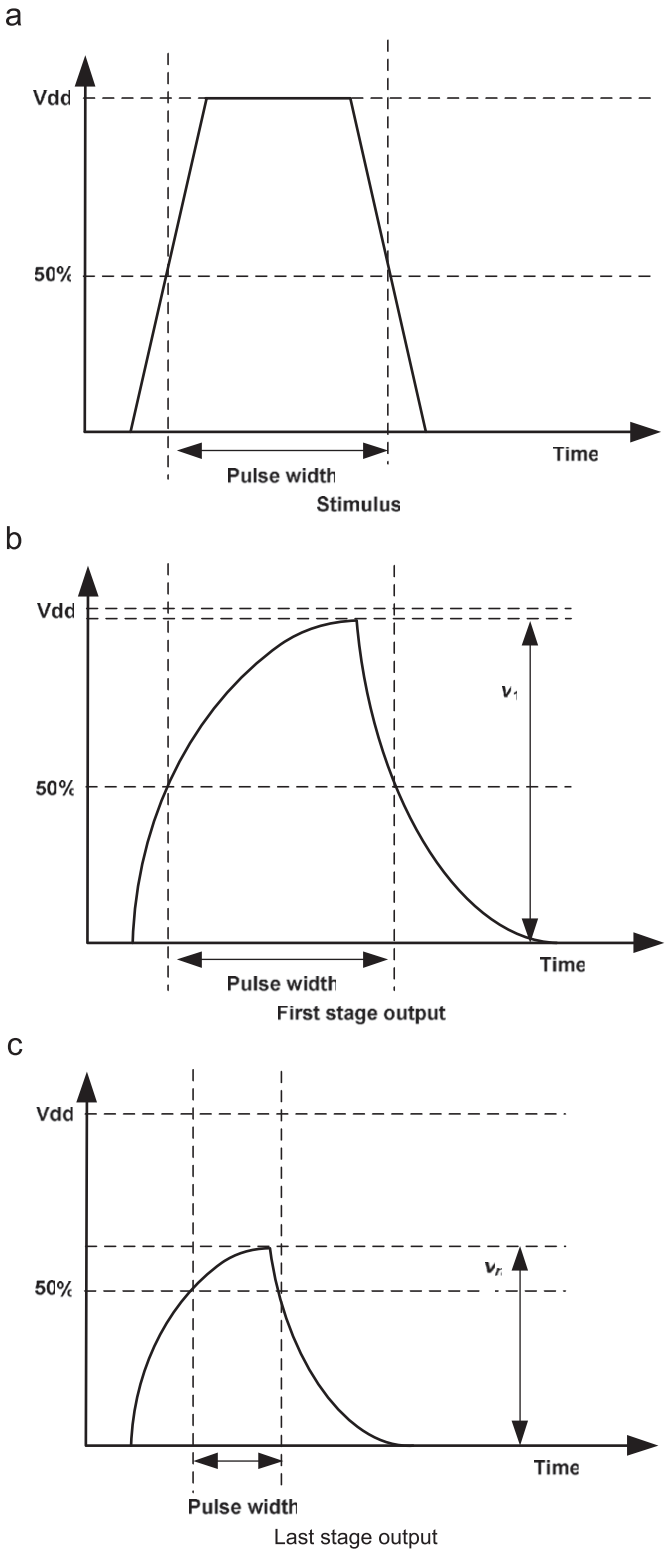

Fig. 4. Signal transmission using: (a) stimulus pulse at the source, (b) waveform of the pulse observed at the output of the first stage and (c) attenuated waveform observed at the output of the last stage.

The minimum pulse width for an interconnection with n stages is denoted as T_n^{Wave} and let v_i to be the amplitude of the pulse at the i-th stage in the link. Using the buffered interconnect model in Eq. (10), the time required for voltage at the first segment reached v_1 becomes

$$T_n^{\text{Wave}} = \sigma_1 \ln\left(\frac{\gamma_1 k_1}{\gamma_1 - v_1}\right) + \delta_1 \tag{13}$$

where δ_1 is the delay of the buffer at the 1-st stage. Therefore, maximum data rate (or throughput) Γ_n^{Wave} for an interconnections is given by

$$\Gamma_n^{\text{Wave}} = \frac{1}{\sigma_1 \ln\left(\frac{\gamma_1 k_1}{\gamma_1 - v_1}\right) + \delta_1} \tag{14}$$

Note that v_1 is an unknown variable and can only be computed backward from v_n. The computation of v_1 is presented as follows.

Computing v_1 backward from v_n: In [23], a recursive relationship between v_{n-1} and v_n is given by

$$v_{n-1} = \frac{1}{2 - v_n} \tag{15}$$

This expression is obtained based on the assumption that all stages are equal and thus a technological independent recursive relationship of v_i and v_{i-1} can be derived [23]. However, it is not applicable to the FPGA interconnect model, as all stages will have their own corresponding parameters and bound to be different. Thus, a more general expression for the v_i and v_{i-1} relationships is required.

Since the rise time for the $(i-1)$-th segment reaches 50% of the supply voltage is $T_i^{\text{Delay}} - T_{i-1}^{\text{Delay}} - \delta_i$, we have the following expression:

$$v_{i-1}\gamma_{i-1}k_{i-1}e^{-(T_i^{\text{Delay}} - T_{i-1}^{\text{Delay}} - \delta_i)/\sigma_{i-1}} = 0.5 \tag{16}$$

By substituting Eq. (11) into Eq. (16), it can be further reduced to

$$v_{i-1}\gamma_{i-1}k_{i-1}e^{-\sigma_i/\sigma_{i-1}\ln\gamma_i k_i/(\gamma_i - v_i) - \ln((\gamma_{i-1} - v_{i-1})/(\gamma_{i-1} - 0.5))} = 0.5 \tag{17}$$

which leads to the recursive relationship of v_i and v_{i-1},

$$v_{i-1} = \frac{\gamma_{i-1}}{\gamma_{i-1}k_{i-1}(2\gamma_{i-1} - 1)\left(\frac{\gamma_i - v_i}{\gamma_i k_i}\right)^{\sigma_i/(\sigma_{i-1})} + 1} \tag{18}$$

As can be seen from Eq. (18), it comprises of the parameters σ_i, k_i to provide a specific modeling of each stages in a long interconnection in FPGA. Eq. (18) can be converted back to Eq. (15) by letting $k_{i-1} = k_i$, $\sigma_{i-1} = \sigma_i$ and $\gamma_i = 1$ for $i = 1,2,\ldots,n$. These conditions imply that an equivalence of each interconnect stage is required in order to use Eq. (15). Note that v_{i-1} from Eq. (18) does not depends δ_i, which is the delay of buffer at stage i. This is a counter-intuitive result. It implies that the throughout is independent of buffer delay. An example is presented in the following to illustrate the calculation for the interconnect throughput.

Example 1. Let an interconnection with three identical buffered stage, each is of length 0.12 mm. Let the resistance, $R_i^s = 489\,\Omega/\text{mm}$ and capacitance C_i^s = 187 fF/mm where i=1,2,3. Also, let the output resistance of the buffer, $R_i^d = 245\,\Omega$ and the output load of the buffer, C_i^d = 201 fF. Therefore, the time constant $\sigma = 0.23 \times 10^{-9}$ s and coefficient k=1.10 can be obtained using Eqs. (6) and (7). Suppose the signal at the end of the interconnection will be registered as HIGH when the voltage is larger or equal to 90% of Vdd. Let $v_3 = 0.9, \gamma_i = 1, i = 1,2,3$ and using Eq. (18), we can have v_2=0.901. Similarly, we can have v_1=0.9167, which implies that the first segment has to be driven up to 91.67% of Vdd, so that the signal can be captured at the sink. Finally, using Eqs. (13) and (14), we can have the minimum pulse width equal to 0.643 ns and the maximum throughput becomes 1.56 Gbit/s.

In the above discussion, it can be seen that the relationship between throughput and the electrical parameters of an

interconnect is complex. The important result from this analysis is that throughput does not merely depend on time constant σ, but the time constant ratio σ_i/σ_{i-1} between the i-th and $(i-1)$-th stages, as this ratio determines the v at previous stage. It means that a wave-pipelined link is less sensitive to wire scaling, but rather the interconnect lengths of the segments would determine the fundamental throughput of the link. This indicates an important opportunity to mitigate the large RC delay challenge from technology scaling.

4.1.1. Power

Since there are no reconfigurable logic elements along a wave-pipelined link, the power consumption equals to the power to drive the signals along the line. Let f_{sw} be the average toggling frequency of the line and the capacitive load at each stage equals to the sum of the wire capacitance C_i^s and the buffer output load C_i^d. The dynamic power consumption for a wave-pipelined link with n-stage buffered interconnect can be computed using the following equation:

$$P_{Wave} = 0.5V_{dd}^2 f_{sw} \sum_{i=1}^{n} (C_i^s + C_i^d) \tag{19}$$

where f_{sw} is the average bit toggling frequency of the link, and C_i^s and C_i^d are the interconnect and driver capacitance, respectively. Note tat the power expression here only captures the power consumption of the data lines. Wave pipelined signaling requires additional circuits for source synchronous clock reference and data sampling which require additional power consumption, as will be discussed in Section 5.

4.2. Register-based pipelining link

Apart from wave-pipelining, inserting registers into a long line is a well know technique [24] to improve circuit throughput. Long interconnections can be partitioned into shorter segments, each is driven by a register and, therefore, the throughput increases, but at the expense of high power dissipation and larger latency.

4.2.1. Throughput

The throughput of a register-pipelined link can be approximated as follows. Suppose there are k registers to be inserted into the long line and the long line can be equally divided into $k+1$ short segments. Unlike the wave-pipelining link, extra interconnects are required to route the line into a slice block and reconnect the output of the slice back to the routing tracks. Furthermore, there is an extra propagation delay which is attributed to register itself. Assuming a same structure and network topology for each slice in an FPGA, the additional delay to the line for adding one register is denoted by ζ_{Reg}. The overall delay of the link becomes

$$D_{Reg}^k = (k+1)T_k + k\zeta_{Reg} \tag{20}$$

where T_k is the delay of each short line segment and ζ_{Reg} is the delay of each register. The throughput of the link will then be the reciprocal of the delay of each segment and register, thus

$$\Gamma_{Reg}^k = \frac{1}{T_k + \zeta_{Reg}} \tag{21}$$

4.2.2. Power

The overall power of the link is similar to the wave-pipelined link but with additional capacitances, C_{Reg} introduced by the registers (and local interconnects in a slice or tile). The expression is as follows (Table 1):

$$P_{Reg} = 0.5V_{dd}^2 f_{sw} \left[kC_{Reg} + \sum_{i=1}^{n} (C_i^s + C_i^d) \right] \tag{22}$$

Table 1
Notations used in the derivation.

R_i^d	Input resistance of the driver at the i-th segment
C_i^d	Load and intrinsic capacitances of the driver at the i-th segment
R_i^s	Thevenin equivalence resistance of the i-th segment
C_i^s	Thevenin equivalence capacitance of the i-th segment
k_i	The approximation coefficient for the i-th segment
σ_i	The time constant for i-th segment
v_i	The voltage of the i-th segment
t	Time
γ_i	The discount factor for i-th segment
T_n	Propagation delay for the n-segment interconnect
Γ_n	Throughput for the n-segment interconnect

Table 2
Comparison between register-based pipelining and wave-pipelining based on the theoretical analysis.

	Register-based pipelining	Wave-pipelining
Technology (nm)	65	65
Length (mm)	6.45 (75 tile)	6.45 (75 tile)
Registers/stages	5	14
Throughput (Gb/s)	1.4	1.4
Latency (ns)	6.12	4.12
Power (mW)	10.08	7.98

Example 2. Table 2 compares the throughput and power performance of register-based and wave-pipelined signaling on two identical links with same configuration and electrical parameters and using Eqs. (14)–(22). For the same technology, length and throughput, the register link requires 26% more power and 49% longer delay, when comparing to the wave-pipelined link. These results demonstrate a significant benefit to wave-pipelined link for global interconnection. However, the register-pipelined link is much more straightforward to implement in a synchronous system, such as FPGAs. Wave-pipelined link applies analog signaling for data transmission, which requires new design of on-chip communication components and these techniques will be presented in the following section.

4.2.3. Summary

The maximum data rate (throughput) of a wave-pipelining link has been derived. It is interesting to find that this throughput is independent of buffers or interconnect RC delay, but the electrical characteristics of the link. Especially, the interconnects in FPGA presence the required characteristics for realizing wave-pipelined signaling and, thus, the wave-pipelining approach provides an opportunity to mitigate the delay problem attributed to the large RC time constant in FPGAs. The results from an example show that the wave-pipelining link is more energy efficient and has smaller latency than the register-based link. In the following section, new FPGA-based circuit for realizing wave-pipelined link are presented.

5. Wave-pipelining circuit design

Consider that the long interconnection introduces additional delay to the link, the key is to synchronize the phase difference from the incoming data and the receiver clock. A simple approach can be adding delay element in the clocking path of the receiver as

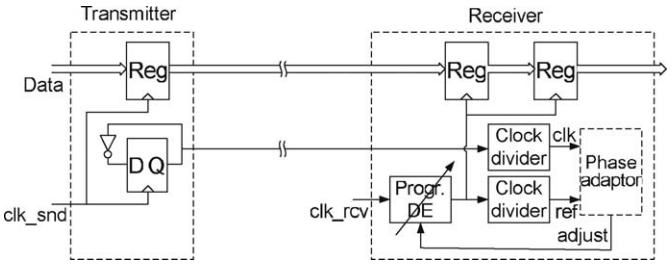

Fig. 5. Schematic of transmitter and receiver for the phase adaptation design.

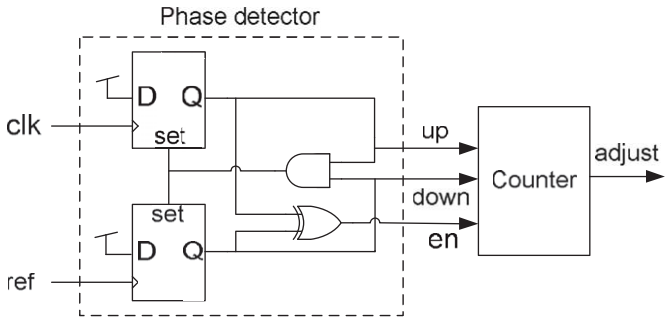

Fig. 6. Design of a phase adaptor.

to match the skew of the incoming data. However, the skew is unknown to an FPGA design before the place-and-route, an intelligent approach is required, such that the circuit can adaptively adjust the delay to match the delay of the line.

Fig. 5 shows the schematic of a phase adaptation circuit for a wave-pipelining link. Data is transmitted source synchronously with a clock signal generated by the transmitter. The clock signal will provide a reference signal to adjust the receiver in order to register the incoming data at the correct timing. Utilizing a digital lock loop or phase lock loop, the received data can be sampled with variable amount of delay relative to the transmitter clock signal as shown in Fig. 5. Fig. 6 shows the phase detector circuit implemented in this design. The XOR gate performs phase difference detection and the counter output is used to adjust the programmable delay element in order to modify the phase of the lock clock. A more detailed timing diagram for the phase adaptation circuit has been detailed in the Appendix, as not to interrupt the flow of the thesis.

The phase detector shown in Fig. 6 identifies the differences in phase between two input clock signals. The design is digital adaptation of the phase detector design in a conventional phase-lock-loop (PLL), except that the output of the XOR-gate becomes an enable signal to the counter to adjust the delay. The output of the counter will adjust the programmable delay element in order to modify the phase of the local clock.

Design of an accurate and high resolution programmable delay element is important for the phase adaptation circuit. This is a challenging task to FPGA designers, as there is no available circuits to implement delay elements.[3] In here, a chain of LUTs are used in our design for the delay circuits. Fig. 7 shows the design of delay element using LUTs in FPGA. The circuit can provide an arbitrary delay by selecting the appropriate combination of LUTs. Each LUT implements two invertors and thus the resolution of the delay circuit can achieve around 50 ps. Alternative methods, such as using interconnects which may provide delay element with

[3] For some recent FPGA models, delay elements are implemented at the I/O modules for inter-chip chip communication circuit implementations.

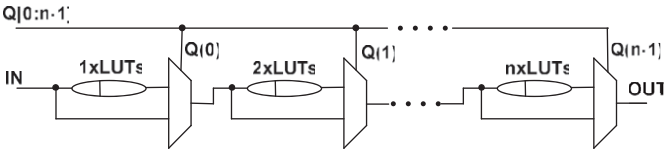

Fig. 7. Implementation of Delay Element in FPGA.

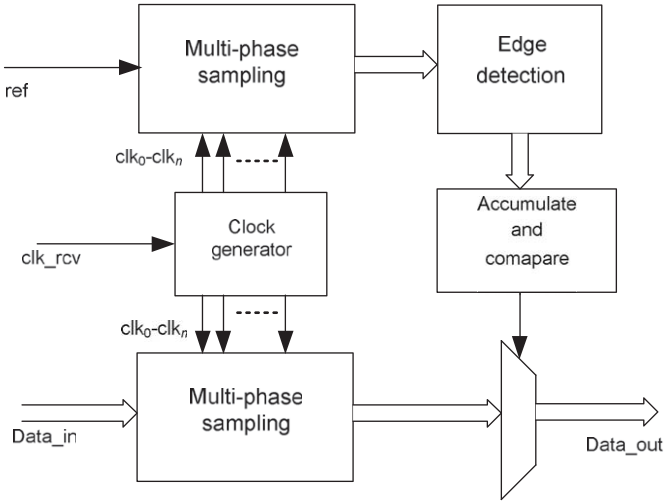

Fig. 8. The design of an oversampling receiver.

higher resolution, can also be used to realize the delay element in FPGA. However, it is challenging to manage interconnect and routings in FPGA without appropriate tools from the FPGA vendors.

Another issue is the alignment of the data path with the transmitter clock. If the clock skew of the data is large, the register may fail to capture the data. When the data link has a large bit-width, variance of the interconnection length will become significant and thus increases the skew [25]. Interconnection length will be lengthened because of competitions for the limited routing resources in an FPGA. Manual routing is conducted trying to minimize the variation of interconnection length for the source synchronous link. It is regarded as future work to develop specific routing algorithm for variance minimization in communication links.

5.1. Multi-phase oversampling

An oversampling receiver captures incoming data multiple times within one clock cycle with the multi-phase clocks. A decision has to be made in order to pick the right sample as the data output. The schematic of a multi-phase oversampling circuit is presented in Fig. 8. A number of clock outputs at difference phases (clk_{0-n}) are produced from the clock generator. These clock signals are used to sampled the incoming data at different time instance within a clock cycle. The sampling process is realized within the multi-phase sampling block. The edge detection block analyzes the registered data and by using XOR-gate, the bit 0–1 transition point can be located. The accumulate-and-compare block decides which sampled data is valid and outputted for later processing. The valid data point is usually located between two 0–1 transition edges. Fig. 9 shows a typical example of data recovery using oversampling. In this case, three samples are obtained by the multi-phase clocks. The bit transitions between samples s_3 and s_1 are detected by the XOR-gate. The statistics of

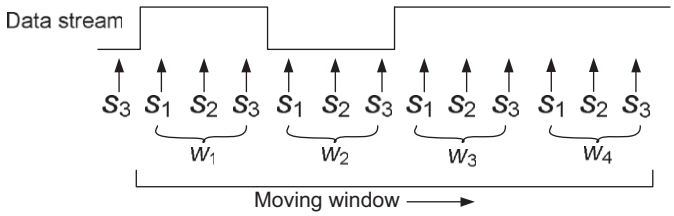

Fig. 9. An illustration of data sampling and moving windows for data recovery.

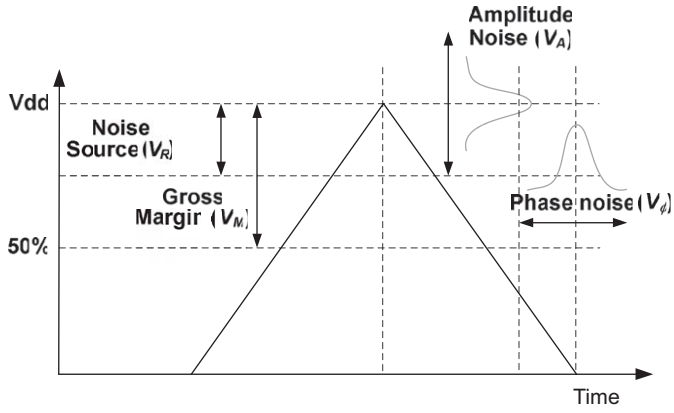

Fig. 10. Timing diagram for error computation.

the bit transitions will be accounted over the moving window, which is four phases in this case. Thus, data sample at s_2 will be selected as the valid data.

The oversampling clock requires a much higher clocking frequency in order to sample the data multiple times within a clock period. An effective way to achieve this is to use delay elements as shown in Fig. 7. By inserting delay elements into the clocking path, multiple clocking edges within a clock period can be realized.

Normally, the edge detection and valid data selection would apply to every single incoming line. If such a circuit were to be implemented for communication link with large bit-width, tremendous hardware area will be required. A source synchronous approach can also be applied here to reduce hardware area consumption, as the edge detection and valid data selection only apply on the reference signal. The output of the accumulate-and-compare block will be used to select valid data for all other lines. As a result, the area overhead for oversampling can be greatly reduced.

5.2. Bit error rate computation

In the source synchronous circuit, unbounded Gaussian noise sources account for a significant fraction of signaling noise. There is always some probability that the Gaussian noise source will exceed the margin and that a probabilistic analysis is required to compute the bit error rate for evaluating the reliability of the wave-pipelined communication link. The network noise margin, V_M, is the amount of voltage margin available to tolerate noise. In general, there are error sources can be described as amplitude error and phase error, as depicted in Fig. 10. Noise sources are usually approximated as Gaussian sources, with a normal distribution, and described by their standard deviation or root mean square value.

The throughput limit for a single line is limited by its minimum pulse width, as have discussed in Section 4. When communication comprises of multiple lines as a bus, pulses may arrive at the receiver at different time stamp because of various on-chip dynamic noise and data dependent crosstalk. The skew between the data path and the reference clock (see Fig. 1) would results an error sampling wrong data. Whether the communication link is susceptible to error depends on the variance (σ_k) in the interconnection lengths in the bus. A smaller throughput receiver is needed to compensate the large skew in order provide a large enough margin to register the correct samples. This is in contrast to the delay-based of register-based pipelining, of which the throughputs depend on the longest interconnection length of the bus.

Furthermore, skew error also contributes to the bit-error as well. Sampling away from the peak of the waveform results in a sampled valued that is less than the peak signal, thus $V_{dd}-V_\phi$. The slewing portion of the signal effectively translates phase noise into amplitude noise so timing noise translating into signal noise. For the clarify of analysis, a triangular waveform is considered, the phase noise translates linearly into amplitude noise by a linear equation. Thus, we have the phase noise

$$g(\sigma_\phi) = V_{dd}\left|\frac{2\sigma_\phi}{T}\right| \tag{23}$$

The effect of jitter has a similar effect as static phase error except the error depends on the phase-noise characteristics and its probability distribution.

The root-mean-square (RMS) voltage of the i-th amplitude noise source is denoted by V_{Ai} and the i-th phase noise source is denoted by $V_{\phi i}$. The standard deviation of the amplitude noise is denoted as σ_A. We can model multiple Gaussian noise sources as a single combined source with rms voltage V_R by summing the variation of each source and taking the square root as follows:

$$V_R = \sqrt{\sum_i (\sigma_{Ai}^2 + g(\sigma_{\phi i})^2)} \tag{24}$$

The effective voltage signal-to-noise ratio is computed from V_M and V_R as [26]

$$VSNR = \frac{V_M}{V_R} \tag{25}$$

Since the noise is Gaussian, the probability density function of the noise voltage is given by

$$p(x) = \frac{1}{V_R\sqrt{2\pi}}\exp\left(-\frac{x^2}{2V_R^2}\right) \tag{26}$$

where V_R is the standard deviation of the combined noise. The probability that the noise exceeds noise margin V_M, the threshold voltage, is given by the error probability

$$P(x > V_M) = 1 - \mathrm{erf}(V_M) \tag{27}$$

and, thus, the upper bound of the probability gives the bit error rate [26],

$$P(\text{error}) < \frac{V_R}{V_M\sqrt{2\pi}}\exp\left(-\frac{V_M^2}{2V_R^2}\right) \tag{28}$$

5.3. Register pipelining: synchronous versus asynchronous

In conventional synchronous system, register pipelining can be easily realized by inserting registers into the data path to increase the clock frequency. This can also be applied for communication link design. Registers can be inserted into the long wires, thus increasing the throughput of the link. However, this approach assumes a synchronous clock domain between the two communicating ends. Such an assumption is not scalable as large systems generally contains multiple clock domains. Furthermore, if the

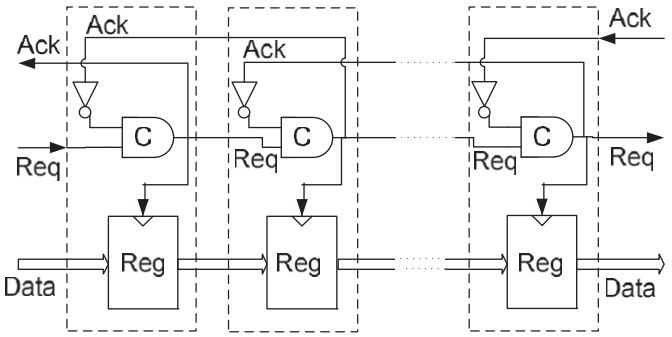

Fig. 11. 4-phase bundle data asynchronous communication link (figure adapted from [27]).

clock spans a large area, issues such as clock skew and jitter will becoming problematic.

Therefore, an asynchronous design will be required in such scenario and provide more reliable data transfer between modules by replacing clock circuits with handshake circuits. A number of asynchronous design has been proposed and thoroughly studied in the last two decades. Asynchronous communication link has yet to be explored in FPGAs.

One of the most popular asynchronous handshake protocol is the 4-phase bundled data protocol [27]. At each of the stages, there is a register to pipeline the data and the register is enabled by the handshaking logics (see Fig. 11). The drawback of this 4-phase protocol is that register will be written only at the rising edge of the request signal. The speed of the pipeline can be improved by introducing protocol that operates at both the rising and falling edges of control signals.

A 2-phase bundle protocol, originated by Sutherland in [28], provides such an improvement. At each of the stages, there are two registers, one will be written at rising edge while the other one will be triggered at falling edge. There are two registers operating in one cycle. The change of a stage of the request and acknowledgement signals symbolizes the read and write process of the registers.

Although the register based pipelining is reliable, several sources might introduce errors into the link and especially when operating at very high speed. The most obvious source of error is when the data signal travels faster than the control signal. Register will capture undesirable data. An effective approach to resolve this problem is by adding delay elements into the request and acknowledgement line to guarantee the control signals lag behind the data. In FPGA, this can be achieved by adding delay elements, which is a chain of LUTs, into the control path. However, this will be at the expense of throughput performance.

5.4. Summary

The five on-chip communication strategies are summarized and compared in Table 3.

6. Experimental results

6.1. Comparing analytical results with SPICE simulations

Consider a global interconnection comprises of buffers, interconnect segments and switches that can be modeled in SPICE as shown in Fig. 12. We compared the wave-pipelining signaling with the conventional delay-based and register-based pipelining signaling in the Cadence Virtuoso design environment and simulated with SPECTRE simulator. The interconnection circuits were modeled in distributed RC network and different technology parameters are based on the Predictive Technology Model (PTM).[4] Also, interconnects of single, double, length-3 and length-6 were considered in the experiments to construct long interconnections.

Experimental results on comparing the analytical model against SPICE simulation are reported in the following. Fig. 13 shows the SPICE simulation results for interconnections of different lengths, in units of tile lengths. Interconnects, in this case, are constructed solely by using single and double lines. Stages of multiplexers are used in connecting these short wire segments. Experimental results for interconnects constructed by using single, double, hex and long lines are shown in Fig. 14.

An identical trend between the analytical and SPICE simulation results can be found for throughput achieved by both wave-pipelining and delay-based approach are shown in Fig. 13(a) and (b), respectively. The average relative error for the wave-pipelining and delay-based throughput are 38.8% and 51.9%, respectively. The analytical results are generally in line with the experimental results. For simplicity, on-resistance of the driver transistor is assumed to be a constant in the buffer model. An approximation method that is based on calculating an average transistor resistance [17] is employed to estimate the on-resistance of the drivers. In reality, the resistance varies according to the input gate voltage [20], and the discrepancies on the resistance estimation may result errors in the throughput prediction. A more comprehensive buffer model can be adopted in our multi-stage model and this will be our future work.

Similarly, an identical trend between the analytical and SPICE simulation is found for the case of multiple-length segments in Fig. 14. The average relative error for the wave-pipelining and delay-based approaches are 30.7% and 41.7%, respectively. Note that there are two spikes at tile 8 and 24. This is because hex lines are used for tile 8 and long lines are used for tile 24 instead of short wires.

6.2. Comparing wave-pipelining with delay-based signaling

Fig. 15 compares the throughput between delay-based and wave-pipelining approaches with different interconnection length. In general, throughput decreases with increasing interconnection length. Specifically, throughput for the delay-based approach drops 85% with the length increasing from 32 to 150 tiles whereas the throughput for wave-pipelined approach drops 35%. Also, the wave-pipelined approach consistently outperforms the delay-based approach. A throughput gain of 21% can be obtained with a 32-tile length interconnection and a 566% gain in throughput with a longer interconnection length, e.g. 150-tile.

Fig. 16 compares the throughput of the two approaches with different technologies. As the technology continuous to scale, both signaling approach will constantly gain in throughput performance. Despite of the fact that performance gap between the two approaches is getting narrower (from 303% to 216% for the 130 nm and 32 nm technology nodes, respectively) over next few technology generations, the wave-pipelining approach still significantly outperforms the delay-based signaling.

It is well know that interconnection throughput can be increased by inserting registers into the long line. However, this approach is at the expense of the power consumption and latency of the link. In the following, we present the results that examined the power and delay overhead of the register-pipelining with comparing to the wave-pipelined link. Specifically, for a typical long interconnection (with length of 72-tile), three registers are

[4] http://www.eas.asu.edu/ptm/

Table 3
A brief summary and comparison between the different on-chip communication approaches.

	Register pipelining			Wave pipelining	
	Async. (4-phase)	Async. (2-phase)	Synchronous	Phase adaptation	Oversampling
Cross clock regions	Yes	Yes	No	Yes	Yes
Regions	Yes	Yes	No	Yes	Yes
Data rate	Slow	Moderate	Fast	Fast	Fast
Hardware area	Moderate	Large	Small	Moderate	Large
Power consumption	Low	High	High	Moderate	High
Reliability	High	High	High	Moderate	Moderate
Design complexity	Moderate	Moderate	Easy	Complex	Moderate

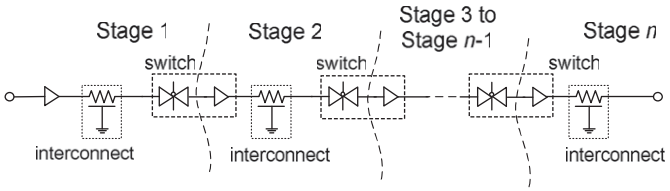

Fig. 12. SPICE circuit model for the experiments.

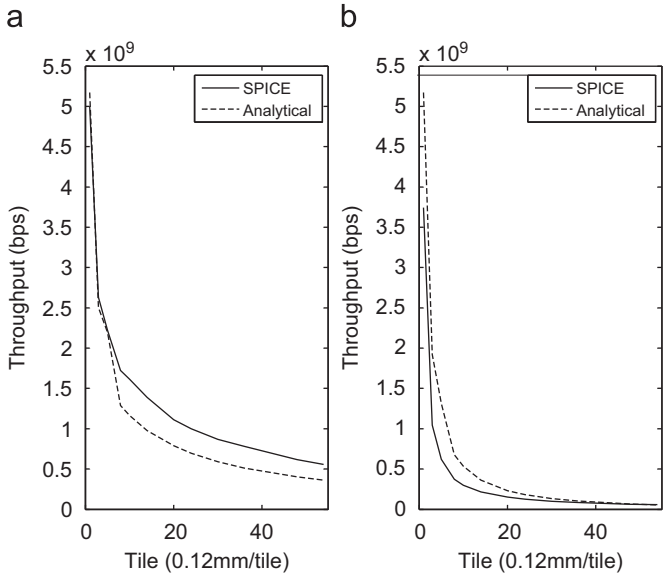

Fig. 13. Analytical model versus SPICE simulation for interconnection comprises of single and double lines only. (a) Interconnect wave-pipelining throughput and (b) delay-based throughput.

required in order to provide the same throughput as the wave-pipelined link.

Fig. 17 shows the improvement for using wave-pipelining over the register-pipelined link for different technologies. Throughout the technology scaling, the power consumption decreases for both approaches and improvement for using wave-pipelining increases from 7% to 13%. The results for the latency comparison are shown in Fig. 18. Latency for both approaches decreases as the technology scales down. But, the delay difference between wave-pipelined and register-pipelined approaches increases from 20% to 35%. In summary, both approaches will be benefited from the technology scaling and the wave-pipelining approach appears a better signaling scheme in terms of power dissipation and delay versus its register-pipelined counterpart.

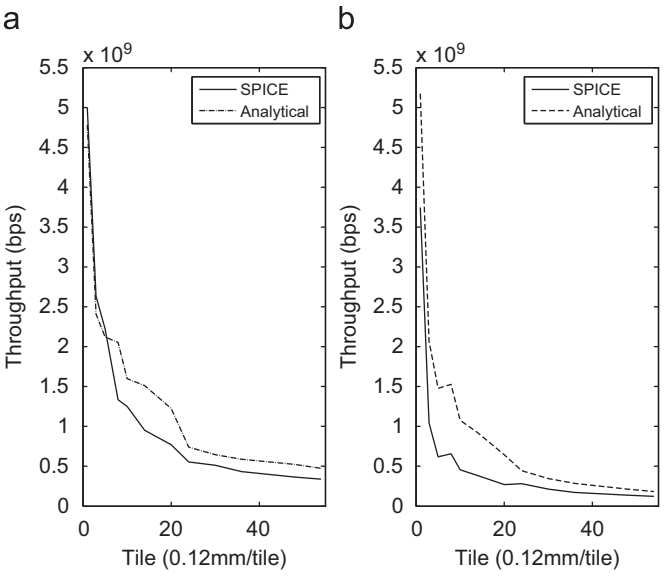

Fig. 14. Analytical model versus SPICE simulation for interconnection comprises of single, double, Hex and long lines. (a) Interconnect wave-pipelining throughput and (b) delay-based throughput.

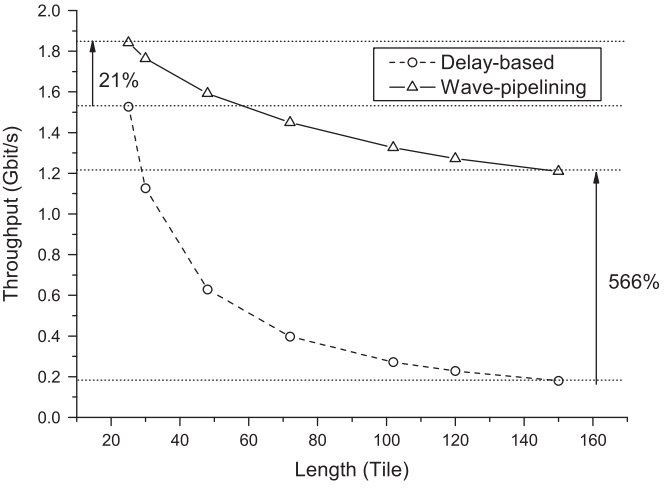

Fig. 15. Throughput comparison between delay-based and wave-pipelined signaling for different interconnection length.

6.3. Evaluation of wave-pipelining in real FPGA

Consider a typical communication link, which comprises of a transmitter, a receiver and two RAMs, as shown in Fig. 19. The register bank will be only applied on the register-pipelining

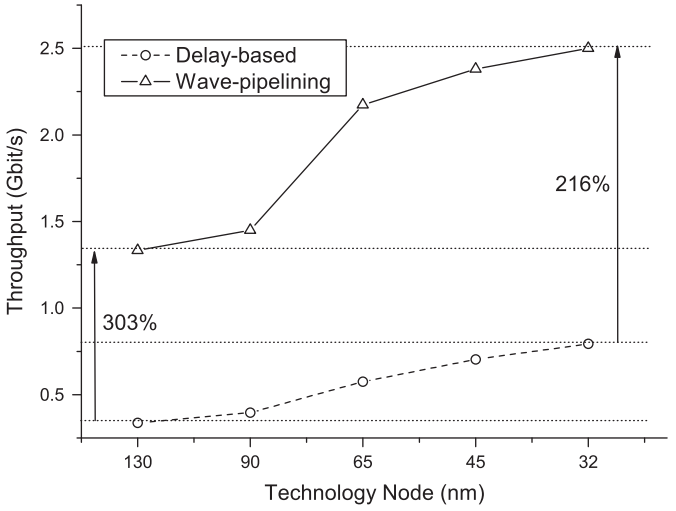

Fig. 16. Throughput comparison between delay-based and wave-pipelined signaling for different technology nodes.

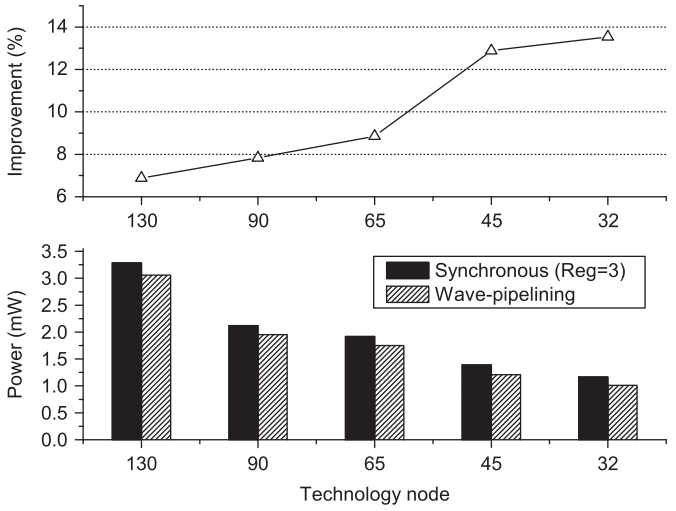

Fig. 17. Power improvement of wave-pipelined signaling over register pipelining.

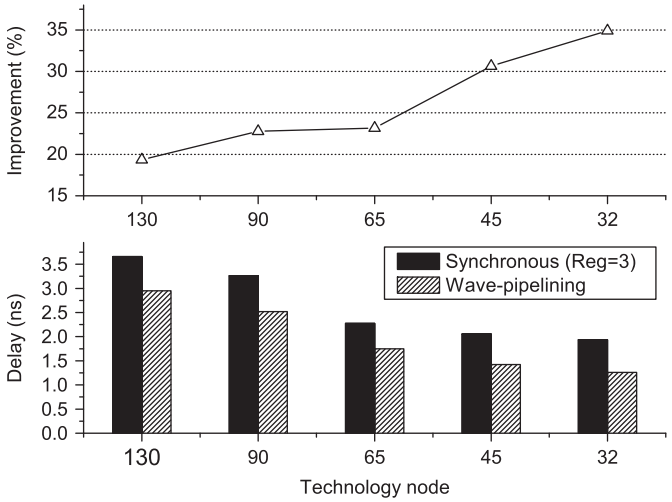

Fig. 18. Delay improvement of wave-pipelined signaling over register pipelining.

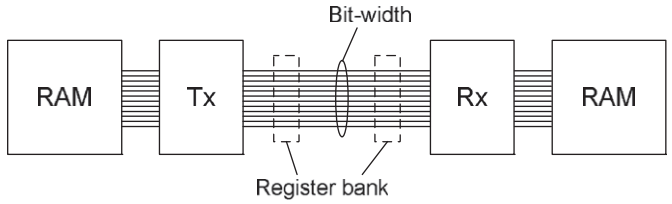

Fig. 19. An typical communication link for data transfer between two RAMs in FPGAs.

Table 4
Implementation of communication links (64-bit) in FPGAs.

	Async. (4-phase)	Async. (2-phase)	Sync.	Phase	Oversampling
Length (tile)	100	100	100	100	100
Freq. (MHz)	35.1	57.6	125	185	250
Area (Slice)	67	204	80	99	221
Power (mW)	4.25	10.53	28.3	40.6	62
Γ (Gb/s)	2.25	3.69	8	11.8	16

schemes. In this section, we present the results studying the performance of such communication link with different design strategies. We used a Xilinx Virtex-4 XC4V-LX200 FPGA, in which we constrained the placement of transmitter and receiver at the two corner in order to obtain a long signal path. The circuits were designed in VHDL and synthesize using Xilinx ISE tools. The designs have also been verified with real chip testing on a Xilinx ML400 board with a Virtex-4 FPGA.

The implementation results for a link with five different designs are presented at Table 4. When comparing to a simple synchronous link design, it is not surprising to find out that asynchronous link is relatively slow with 2.25 and 3.69 Gbit/s versus 8 Gbit/s in a synchronous link. Although the handshaking protocol is relatively simple, the signal path requires multiple register stages and interconnects, which are not efficient for FPGA implementation. In contrast, the phase adaptation and oversampling approaches with wave-pipelined signaling can achieve a larger throughput (48% and 100%, respectively) comparing to the synchronous approach. Moreover, the throughput obtained from actual FPGA implementation is far less than the theoretical prediction (see Table 1). This is due to the clocking frequency limitation of transmitter and receiver logics, despite of the interconnects can achieve a significantly high throughput. Further, there are area and power overheads for the wave-pipelining schemes when comparing to the synchronous approach. In the following, we will present the results of the trade-offs.

Fig. 20 compares the throughput achievement for four different implementation strategies of a communication link with different bit-width. In general, throughput increases monotonically for all approaches. The register-pipelined link with three registers inserted provides the highest throughput and closely followed by the wave-pipelined link with oversampling receiver, which doubles the throughput of a simple synchronous link. The wave-pipelined link with phase adaptation also outperforms the simple synchronous link. Note that for the synchronous link, the throughput against bit-width is not linear but curved as the bit-width increases. This is because long wires at each channel are limited. As the bit-width of a communication link increases, interconnections are required to traverse further away for long wire which are available. This

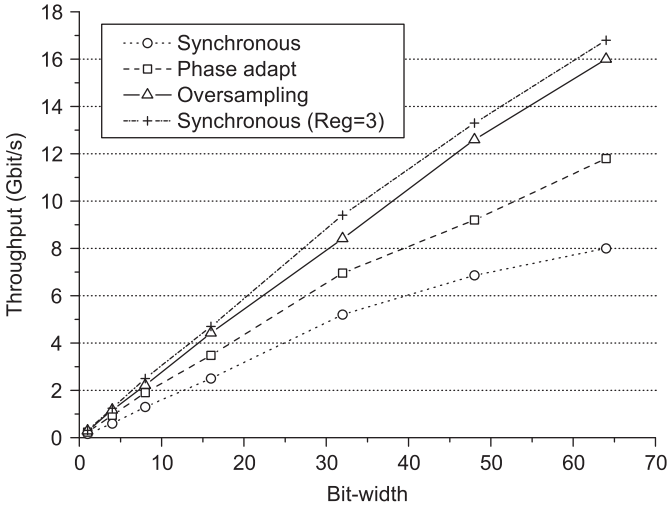

Fig. 20. Throughput for the communication links.

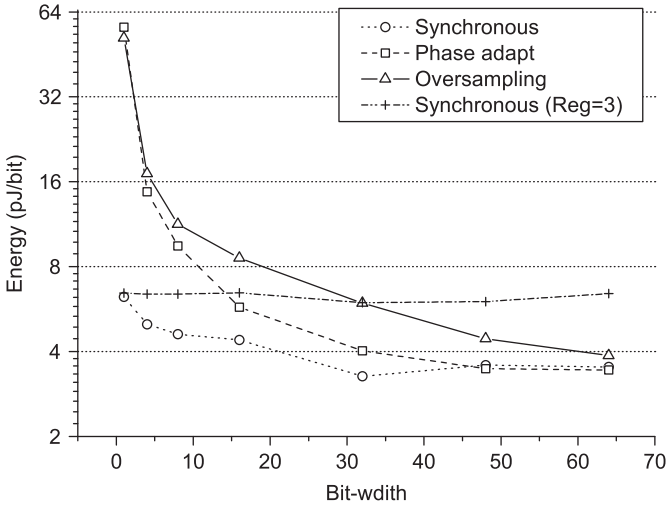

Fig. 21. Energy consumption (energy per bit transfer) for the communication links.

Table 5
Standard deviation of interconnection lengths for different bit-width in a link.

Bit-width	Standard deviation
1	0
4	0.05
8	0.5
16	0.55
32	0.58
48	0.77
64	1.73

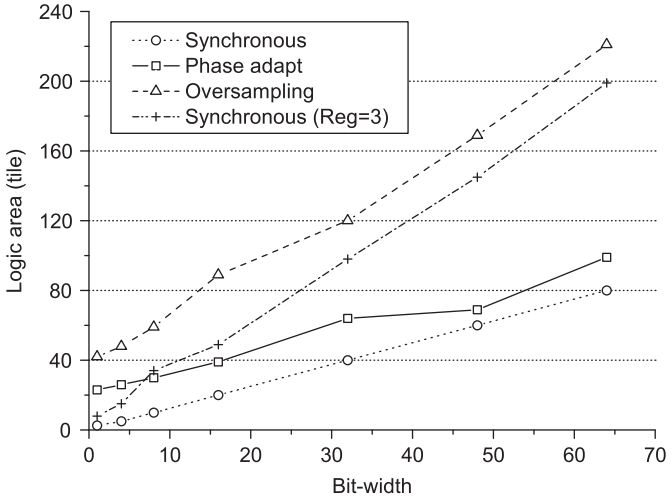

Fig. 22. Area for the communication links.

increases the average interconnection length [25]. The wave-pipelining approaches are not affected by the average interconnection length, but their variance. Interconnects length variance also increases as the bit-width increases moderately (see Table 5). We can therefore observe from the figure that the curve for phase adaptation is also not linear. Oversampling approach is less affected by either the interconnection length and variance.

We compare the energy efficiency of the four approaches using the bit-energy metric, which quantifies energy required to transmit one bit information. Bit energy can be computed based on the power and throughput as follows:

$$\text{Bit-Energy} = \frac{\text{Energy/Time}}{\text{Bit/Time}} = \frac{\text{Power}}{\text{Throughput}} \tag{29}$$

The energy results for the four approaches are shown in Fig. 21. The smaller bit-energy implies a better energy efficiency of the link. The simple synchronous and register-pipelined link have a relatively constant bit-energy whereas bit-energy in wave-pipelining approaches decreases over the bit-width. The oversampling approach outperforms the register-pipelined approach when bit-width is larger than 32-bit while the phase adaptation approach outperforms the register-pipelined with bit-width larger than 16-bit. Since there is a constant overhead for the receiver circuitry for the wave-pipelining design, the overhead will be significant for small bit-width. For a large bit-width, the throughput performance of the wave-pipelining link outweighs the overhead and, thus, a better energy efficiency can be obtained (Fig. 22).

Synchronous approach consumes the least area among the four approaches. For the phase adaptation approach, it consumes more area than the register-pipelined approach initially, and it outperforms the register-pipelined approach with larger bit-width. For a link with bit-width larger than 32-bit, oversampling approach consumes an average 16.6% more in area than the register-pipelining. However, for the phase adaptation approach, it consumes an average 39.4% less in area when comparing to the register-pipelining link.

The bit error rate (BER) can be estimated based on the operating frequency of the communication link and using the Eq. (28). The standard deviations of noise sources, including both static and dynamic noise, are inputted to the expression to obtain the BER. In [29], a range of noise sources obtained based on HSPICE simulations are published. These values can be used in Eq. (28) to obtain an approximation of BER and provide a reliability guideline. Furthermore, one important source of noise is the static skew which caused by the discrepancy of interconnections in the communication link. The standard deviation of the static skew can be obtained from the direct measurement of interconnection length of the circuits. Figures in Table 5 are used in my analysis. The other noise sources figures are shown in Table 6. The Vdd of the circuit is assumed 1.2 V and the noise margin V_M is 0.6 V. The BER results are shown in Fig. 23. The communication link is considered reliable if one has a $\log_{10}(\text{BER})$ equal to -25. As the link bit-width increases, the reliability of the

Table 6
Parameters used in the bit-error-rate analysis.

Noise source	σ
Crosstalk (with shielding)	1.74 ps
Crosstalk (without shielding)	12 ps
DC noise	15 mV
Jitter & dynamic skew	50 ps
Skew (Static)	16.9 ps (8-bit) 19.6 ps (32-bit) 26.0 ps (48-bit) 58.0 ps (64-bit)
V_{dd}	1.2 V
V_M	0.6 V

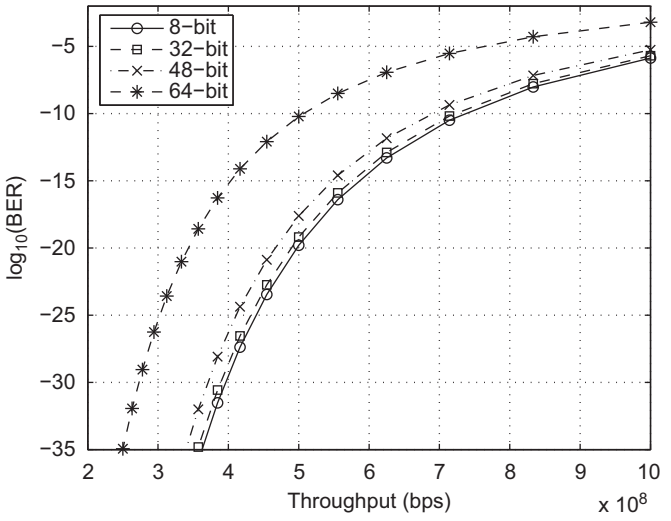

Fig. 23. Bit-error-rate (BER) estimates for wave-pipelined link with different throughput and bit-width.

link decreases. The link with 64-bit bit-width can achieve a maximum 300 MHz operating frequency and, thus, can provide 19.2 Gbps communication throughput. Links with smaller bit-width can achieve a higher throughput. For example, a 8-bit link can achieve 400 MHz and, thus, can provide 3.2 Gbps data rate.

7. Conclusion

The ever-increasing interconnection delay presents a difficulty for high bandwidth communication in FPGA. Wave-pipelined signaling allows multiple bits traversing along the line simultaneously and thus can substantially increase the interconnection throughput. A novel FPGA interconnect model for throughput analysis has been presented. The model captures the important electrical characteristics of the interconnects and is able to accurately predict wave-pipelining throughput of the link. Besides, the new signaling scheme requires novel designs of transmitter and receiver circuits in order to sample the high-speed analog signals correctly. Two new on-chip communication circuits are presented and can significant improve communication throughput and energy efficiency. Especially, for communication link with large bit-width, the oversampling and phase adaptation approaches outperform the register-pipelining in terms of energy efficiency. It is also interesting to observe that the throughput performance of a wave-pipelined link depends on the skew of the parallel lines instead of the delays. The new FPGA-based signaling scheme poses a throughput-centric paradigm and an interesting problem open to further investigation for throughput-centric FPGA architectures and CAD tools to mitigate the interconnect challenge in future technology processes.

References

[1] International Roadmap Committee, International technology roadmap for semiconductors 〈http://www.itrs.net/〉.
[2] T. Mak, C. D'Alessandro, P. Sedcole, P. Cheung, A. Yakovlev, W. Luk, Global interconnections in FPGAs: modeling and performance analysis, in: Proceedings of ACM International Workshop on System Level Interconnect Prediction, 2008.
[3] L. Cotton, Maximum rate pipelined systems, in: Proceedings of AFIPS Spring Joint Computer Conference, 1969.
[4] W. Burleson, M. Ciesielski, F. Klass, W. Liu, Wave-pipelining: a tutorial and research survey, IEEE Trans. VLSI Systems 6 (1998) 464–474.
[5] E.I. Boemo, S. Lopez-Buedo, J.M. Meneses, The wave pipeline effect on LUT-based FPGA architectures, in: Proceedings of the Fourth International ACM Symposium on Field-Programmable Gate Arrays, 1996.
[6] G. Lakshminarayanan, B. Venkataramani, Optimization techniques for FPGA-based wave-pipelined DSP blocks, IEEE Trans. VLSI Systems 13 (7) (2005) 783–793.
[7] V.V. Deodhar, J.A. Davis, Optimization of throughput performance for low-power vlsi interconnects, IEEE Trans. VLSI System 13 (2005) 308–318.
[8] A. Joshi, G. Lopez, J. Davis, Design and optimization of on-chip interconnects using wave-pipelined multiplexed routing, IEEE Trans. VLSI Systems 15 (9) (2007) 990–1002.
[9] S.-J. Lee, K. Kim, H. Kim, N. Cho, H.-J. Yoo, Adaptive network-on-chip with wave-front train serialization scheme, in: 2005 Symposium on VLSI Circuits Digest of Technical Papers, 2005, pp. 104–107.
[10] R.R. Dobkin, Y. Perelman, T. Liran, R. Ginosar, A. Kolodny, High rate wave-pipelined asynchronous on-chip bit-serial data link, in: Proceedings of the 13th IEEE International Symposium on Asynchronous Circuits and Systems, 2007.
[11] M. Khellah, S. Brown, Z. Vranesic, Modelling routing delays in SRAM-based FPGAs, in: Proceedings of Canadian Conference on VLSI, 1993.
[12] V. Betz, J. Rose, A. Marquardt, Architecture and CAD for Deep-Submicron FPGAs, Kluwer Academic Publishers, Dordrecht, 1999.
[13] D. Chen, J. Cong, P. Pan, FPGA design automation: a survey, Found. Trends Electron. Des. Automat. (2006) 139–169.
[14] G. Lemieux, E. Lee, M. Tom, A. Yu, Directional and single-driver wires in FPGA interconnect, in: Proceedings of the International Conference on Field-Programmable Technology, 2004.
[15] E. Lee, G. Lemieux, S. Mirabbasi, Interconnect driver design for long wires in field-programmable gate arrays, in: Proceedings of the International Conference on Field-Programmable Technology, 2006.
[16] Xilinx, Virtex-4 Data Sheets.
[17] H. Bakoglu, Circuits, Interconnections, and Packaging for VLSI, Addison-Wesley, Reading, MA, 1990.
[18] L.-Y. Lin, C.-Y. Wang, P.-J. Huang, C.-C. Chou, J.-Y. Jou, Communication-driven task binding for multiprocessor with latency insensitive network-on-chip, in: ASP-DAC, 2005.
[19] T. Sakurai, Closed-form expressions for interconnection delay, coupling, and crosstalk in VLSI's, IEEE Trans. Electron. Devices 40 (1) (1993) 118–124.
[20] J. Rabaey, A. Chandrakasan, B. Nikolic, Digital Integrated Circuits: A Design Perspective, Prentice-Hall, Englewood Cliffs, NJ, 2003.
[21] W. Elmore, The transient analysis of damped linear networks with particular regard to wideband amplifiers, J. Appl. Phys. 19 (1) (1948) 55–63.
[22] R. Ho, K. Mai, M. Horowitz, The future of wires, Proc. IEEE 89 (4) (2001) 490–504.
[23] V. Deodhar, J. Davis, Optimization of throughput performance for low-power vlsi interconnects, IEEE Trans. VLSI Systems 13 (3) (2005) 308–318.
[24] J. Cong, C. Wu, FPGA synthesis with retiming and pipelining for clock period minimization of sequential circuits, in: Proceedings of the 34th Annual Conference on Design Automation, 1997.
[25] T. Mak, P. Sedcole, P. Cheung, W. Luk, Interconnection lengths and delays estimation for communication links in FPGAs, in: ACM International Workshop on System Level Interconnect Prediction, 2008.
[26] P. Borjesson, C. Sundberg, Simple approximations of the error function $Q(x)$ for communications applications, IEEE Trans. Commun. (1979) 639–643.
[27] J. Sparso, S. Furber, Principles of Asynchronous Circuit Design: A System Perspective, Kluwer Academic Publishers, Dordrecht, 2001.
[28] I. Sutherland, Micropipelines, Commun. ACM 32 (6) (1989) 720–738.
[29] P. Teehan, G. Lemieux, M. Greenstreet, Towards reliable 5 Gbps wave-pipelined and 3 Gbps surfing interconnect in 65 nm FPGAs, in: ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2009.

Terrence Mak (S'05–M'09) received both his B.Eng. and M.Phil. degrees in Systems Engineering from the Chinese University of Hong Kong in 2003 and 2005, respectively, and his Ph.D. degree from Imperial College London in 2009. He joined the School of Electrical, Electronic and Computer Engineering at Newcastle University as a lecturer in 2010. During his Ph.D., he worked as a Research Engineer Intern in the VLSI group at Sun Microsystems Laboratories in Menlo Park, California. He also worked as a Visiting Research Scientist in the Poon's Neuroengineering Laboratory at MIT. He was the recipient of both the Croucher Foundation Scholarship and the US Navel Research Excellence in Neuroengineering in 2005. In 2008, he served as the Co-Chair of the UK Asynchronous Forum, and in March 2008 he was the Local Arrangement Chair of the Fourth International Workshop on Applied Reconfigurable Computing. His research interests include FPGA architecture design, Network-on-Chip, reconfigurable computing and VLSI design for biomedical applications.

Pete Sedcole (S'96–M'99) received his B.E. degree in Electrical and Electronic Engineering from the University of Canterbury, Christchurch, New Zealand, and his Ph.D. degree from Imperial College London, London, UK, in 1999 and 2006, respectively. He is currently a Research Associate with the Department of Electrical and Electronic Engineering, Imperial College London, where he works on variation-aware design in reconfigurable systems. In-between degrees, he was an Electronics Engineer with Trimble Navigation Ltd., Christchurch, New Zealand, where he designed embedded electronics for navigational guidance systems and handheld GPS products. Dr. Sedcole is a member of the IET.

Peter Y.K. Cheung (M'85–SM'04) received his B.S. degree with first class honors from Imperial College of Science and Technology, University of London, London, UK, in 1973. Since 1980, he has been with the Department of Electrical Electronic Engineering, Imperial College, where he is currently a Professor of digital systems and deputy head of the department. He runs an active research group in digital design, attracting support from many industrial partners. Before joining Imperial College he was with Hewlett Packard, Scotland. His research interests include VLSI architectures for signal processing, asynchronous systems, reconfigurable computing using FPGAs, and architectural synthesis. Prof. Cheung was elected as one of the first Imperial College Teaching Fellows in 1994 in recognition of his innovation in teaching.

Wayne Luk (S'85–M'89) received his M.A., M.Sc., and D.Phil. degrees in Engineering and Computer Science from the University of Oxford, Oxford, UK. He is Professor of Computer Engineering with the Department of Computing, Imperial College London, and a Visiting Professor with Stanford University, Stanford, CA, and Queen's University Belfast, Belfast, Northern Ireland. His research interests include theory and practice of customizing hardware and software for specific application domains, such as graphics and image processing, multimedia, and communications. Much of his current work involves high-level compilation techniques and tools for parallel computers and embedded systems, particularly those containing reconfigurable devices such as field-programmable gate arrays.