A Q-Learning Based Self-Adaptive I/O Communication for 2.5D Integrated Many-Core Microprocessor and Memory

Sai Manoj P. D., *Student Member, IEEE*, Hao Yu, *Senior Member, IEEE*, Hantao Huang, *Student Member, IEEE*, and Dongjun Xu, *Student Member, IEEE*

Abstract—A self-adaptive output-voltage swing adjustment is introduced in the design of energy-efficient I/O communication for 2.5D integrated many-core microprocessor and memory. Instead of transmitting signal with large voltage swing, a Q-learning based I/O management is deployed to adaptively adjust the I/O output-voltage swing under constraints of both communication power and bit error rate (BER). Simulation results show that the proposed adaptive 2.5D I/Os (in 65 nm CMOS) can achieve an average of 12.5 mW I/O power, 4 GHz bandwidth and 3.125 pJ/bit energy efficiency for one channel under 10^{-6} BER. With the use of conventional Q-learning and further accelerated Q-learning, we can achieve 12.95 and 18.89 percent power reduction and 14 and 15.11 percent energy efficiency improvement when compared to the use of uniform output-voltage swing based I/O communication.

Index Terms—Through-silicon interposer (TSI), 2.5D integration, many-core microprocessor, voltage-swing tuning, Q-learning, low power I/O, memory-logic integration

1 Introduction

As the number of I/Os grow dramatically when integrating multi-core microprocessor and main memory for data oriented computing, there is an emerging need to develop high data-rate and low power I/O communication circuits [1], [2]. Previous 2D wire-line communication by PCB trace of backplane [3], [4], [5] has a large latency and poor signal-to-noise ratio (SNR) in channels. Though 3D integration by stacking layers of dies vertically using through-silicon via (TSV) I/O [6], [7], [8], [9], [10], [11], [12] can provide a scalable memory-logic integration, it has high thermal density with poor heat dissipation and hence can result in serious reliability concern [13], [14], [15]. Recent 2.5D integration by through-silicon interposer (TSI) on common substrate can provide better thermal dissipation [16], [17]. With the design of TSI-based transmission line (T-line), one can further achieve high data-rate and low power [18], [19] 2.5D I/O communication without area overhead, since the TSI T-line is fabricated underneath the common substrate. As such, the TSI T-line based 2.5D I/Os have become the recent interest towards energy-efficient integration of multi-core microprocessor and main memory.

However, previous I/O circuit designs assume a constant and large output-voltage swing [20], [21], [22], [23]. For 2D wire-line communication, a large output-voltage swing is required to compensate the channel loss and noise of PCB trace. But, a large output-voltage swing with high data-rate can result in large power consumption in communication. In [23], a single-ended low-swing voltage transmission scheme has been proposed with self-resetting logic repeaters (SRLRs) embedded inside the routers. Due to the use of single-ended signalling, the communication is prone to the noise. Meanwhile, the bit error rate (BER) requirement of I/Os is not necessary to be low at all times, since it depends on the workload specification. Therefore, the constant and large output-voltage swing of I/Os may be over designed with low utilization. As such, in order to have an energy-efficient I/O communication, one needs to develop an I/O management with a dynamic output-voltage swing scaling to adaptively adjust the output-voltage swing level under the dynamic BER constraint [24], [25].

On-line machine learning based power management has been recently practiced [26], [27], [28], [29], [30], [31]. For example, Q-learning [32], [33] can be utilized to find an optimal action-selection policy from set of states. However, the conventional Q-learning is limited by the learning rate with slow convergence [34]. In [35], a large range of learning rates are dynamically applied to improve the convergence rate, but with the increase in complexity. A reinforcement Q-learning approach can improve the policy decision by using the prior knowledge of the system with a Markov decision process (MDP) [27]. In [27], reinforcement Q-learning with accelerated convergence is applied to a multimedia system to achieve a trade-off between distortion rate and complexity.

In this paper, one conventional Q-learning based I/O management is applied to adjust the level of output-voltage swing at transmitter of 2.5D TSI I/Os such that one can achieve a reduced power under specified BER requirement. To overcome the slow convergence issue in conventional Q-learning algorithm, an online reinforcement Q-learning [27], termed

• *The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore. E-mail: haoyu@ntu.edu.sg.*

Manuscript received 21 Sept. 2014; revised 23 Apr. 2015; accepted 13 May 2015. Date of publication 31 May 2015; date of current version 17 Mar. 2016. Recommended for acceptance by K. Chakrabarty.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2015.2439255

TABLE 1
List of Variables and Their Description

Notation	Definition
$S = \{s_1, \ldots, s_N\}$	Set of states
$A = \{a_1, \ldots, a_L\}$	Set of actions
$V = \{v_1, \ldots, v_N\}$	Set of voltage-swings
$P(s_i, a_k, s_j)$	Transition probability to select action a_k in state s_i
BER_i	Bit-error-rate at i^{th} output-voltage swing
Pw_i	Communication power at i^{th} output-voltage swing
$R(s_i, a_k, s_j)$	Reward value for state change from s_i to s_j with action a_k
γ	Discount rate
b_1, b_2	Weighted parameters for reward function
α	Learning rate
$Q(s_i, a_k)$	Q-value
R_D	Resistance of driver
Z_{diff}	Characteristic impedance of the T-line
I_t	Tail current of CML driver
σ_v	Standard deviation of noise
N_{s_i}	Number of visits to state s_i
M	Number of possible actions
N	Order of AR prediction

as accelerated Q-learning based management approach is further developed in this paper to adaptively adjust the output-voltage swing levels of 2.5D TSI I/Os. Based on the historical data, the voltage-swing adjustment is formulated as a MDP problem solved by model-free reinforcement learning under the constraints of both power budget and BER. To accelerate the adjustment convergence, a prediction of BER and power as virtual experience is applied to the reinforcement Q-learning algorithm. One corresponding 2.5D TSI I/O is designed in 65 nm CMOS process for multi-level output-voltage swing with balanced power and BER. The proposed algorithm is carried out in Matlab with system power obtained from Cadence simulator. Simulation results show that the adaptive 2.5D TSI I/O circuit can achieve 12.5 mW I/O power, 4 GHz bandwidth and 3.125 pJ/bit energy efficiency for one channel under 10^{-6} BER. By using conventional Q-learning and further accelerated Q-learning, we can further achieve 12.95 and 18.89 percent communication power reduction and 14 and 15.11 percent energy efficiency improvement compared to the traditional I/O communication with constant output-voltage swing.

The remainder of this paper is organized as follows. First, we describe the memory-logic integration architecture by 2.5D integration with an adaptive I/O management and the according problem formulation in Section 2. In Section 3, the adaptive I/O management by the conventional Q-learning and the accelerated Q-learning are presented. Section 4 presents the circuit blocks of 2.5D TSI I/Os including: receiver/transmitter, error-correcting code and adaptive tuning. The experimental results are shown in Section 5 with conclusion in Section 6.

2 2.5D TSI I/O Communication

In this section, we present memory-logic integration architecture by 2.5D TSI I/Os. Furthermore, the problem of self-adaptive voltage-swing adjustment for low-power I/O communication is formulated. The set of notations used in this paper are defined in Table 1.

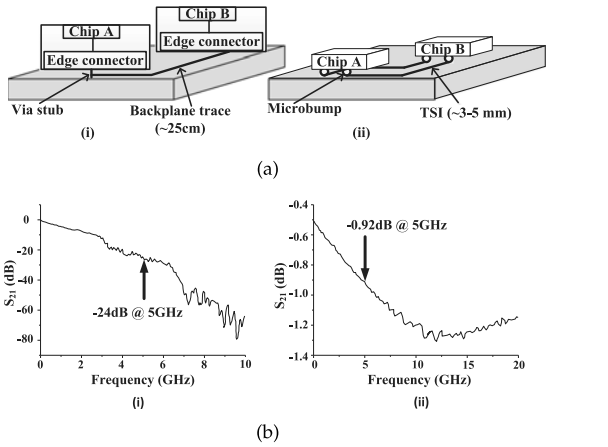


Fig. 1. (a) Interconnect by: (i) Backplane trace (ii) TSI T-line; (b) Channel loss for: (i) Backplane trace; (ii) TSI T-line.

2.1 Memory-Logic Integration by 2.5D TSI I/O

A printed circuit board (PCB) [3], [4], [5] containing sockets is the traditional 2D interconnection method between processors and memories, as shown in Fig. 1a(i). It requires a long trace ($\geq$25 cm) with non-ideal vias, suffering from channel loss and noise. In order to compensate this channel loss with high data-rate, current-starved circuits and equalizers need to be employed [3]. The 2.5D TSI T-line [16] does not need a long trace to interconnect processors and memories. What is more, since TSIs are deployed underneath the common substrate, as shown in Fig. 1a(ii), the area overhead is also mitigated. Unlike traditional backplane based interconnects, 2.5D TSIs are much shorter with a few mm in length and less routing overhead. A comparison of channel loss between 2D PCB trace and 2.5D TSI based T-lines is presented in Fig. 1b. One can observe from Fig. 1b(i), at 5 GHz clock frequency, the PCB backplane with long trace has nearly 24 dB channel loss; whereas at same frequency and to perform similar interconnection, for 2.5D TSI with 10 μm width, 3 mm length has less than 1 dB loss. Therefore, the 2.5D TSI T-line based integration is selected for the memory-logic integration. In addition, the 2.5D TSI based integration shows much better thermal dissipation capability compared to 3D integration [17].

Fig. 2a shows the system architecture for the memory-logic integration by the 2.5D TSI I/O. Each of the core and memory blocks comprises of transmitter as well as receiver to enable a full duplex communication. To further reduce the I/O communication power between logic and memory blocks, a self-adaptive voltage-swing adjustment is required. The current-mode logic (CML) buffer is shown in Fig. 2b with tunable tail-current (based on the output of I/O controller) to adaptively tune the output-voltage swing. By adjusting the I/O output-voltage swing, I/O communication power can be reduced with improved energy efficiency compared to the previous designs [20], [21], [22] that utilizes the fixed full output-voltage swing. However, the BER increases when the output-voltage swing decreases. Hence, a trade-off needs to be maintained between the I/O communication power and BER, which requires an optimized online management.

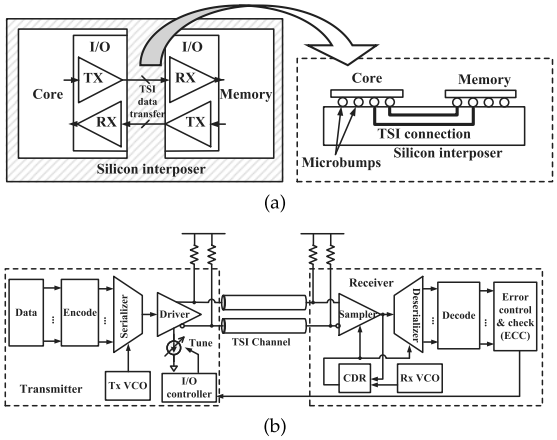


Fig. 2. (a) Core-memory integration by 2.5D TSI I/O interconnect and its cross sectional view; (b) Adaptive tuning I/O based on error checking and correction.

Detailed description of the transmitter, receiver, coding and the adaptive tuning circuits are presented in Section 4. In the following, we formulate an adaptive I/O output-voltage swing tuning problem for the proposed architecture.

2.2 Problem Formulation

As previously discussed, BER at the receiver increases with the decrease of I/O communication power due to channel loss and noise. As such, one needs to find an optimal output-voltage swing at the transmitter such that balanced power reduction is obtained with the maintained BER, which can be defined as the following problem.

Problem: Tune the output-voltage swing at the transmitter to achieve low power at the cost of BER based on the I/O communication channel characteristics.

$$\begin{aligned} &Opt.: Pw_i,\ BER_i \\ &S.T.(i)\ Pw_i \ \le\ Pw_T \\ &\quad (ii)\ BER_i \ \le\ BER_T, \end{aligned} \tag{1}$$

where Pw_i and BER_i denotes the I/O communication power and BER under the ith output-voltage swing level V_i. Note that the BER and power are both functions of the output-voltage swing. Pw_T and BER_T represents the targeted I/O communication power and BER of one TSI I/O channel under the normal operation. With the increase in output-voltage swing, I/O communication power increases with reduced BER and vice-versa. As such, the output-voltage swing level V_i needs to be adaptively tuned for optimizing the I/O communication power and BER simultaneously. In this paper, a self-adaptive tuning of the output-voltage swing at transmitter is performed based on the conventional Q-learning and further by the accelerated Q-learning, discussed in next section.

3 Q-Learning Based Adaptive Tuning

In this section, we will first present the basics of Q-learning theory, followed by the conventional Q-learning, modeling of Markov decision process and the according accelerated Q-learning algorithm for adaptive tuning. System power and BER models are then discussed as well.

3.1 Q-Learning Theory

Machine learning algorithms such as Q-learning theory [36] are generally practised to find an optimal action-selection policy from the set of states S. Both these algorithms evaluate *state* and *action* pairs from the previous inputs. To solve (1) using conventional Q-learning and further by accelerated Q-learning algorithms, we consider the I/O communication power Pw and BER BER as the state vector; and uses the output-voltage swing level V_i as the action. State vector S can be given as

$$S =< Pw,\ BER > .$$

Before describing the conventional Q-learning and accelerated Q-learning algorithms, we present a few terminologies used in these algorithms:

- State S: set of states indicating the value of system variable(s). We consider the communication power and BER as the state vectors.
- Action A: set of actions indicating the change of state. We consider the change of output-voltage swing level as the action.
- State transition probability P (s_i, a_k, s_j): probability indicating whether to take an optimal action a_k based on Q-learning or perform a random action.
- Reward R(s_i, a_k, s_j): evaluation value of action a_k to change the state from s_i to s_j, which is dependent on historical BER and power Pw.
- Q-value $Q(s_i,\ a_k)$: set of accumulated Q-values to measure the benefits of taking action a_k at state s_i.
- Expected Q-value $\hat{\mathbf{Q}}(s_i, a_k)$: set of values to measure the expected benefits of taking action a_k at state s_i.
- Policy: process of state change under sequence of action.

In order to obtain the state-action pairs and form a look-up-table (LUT), input samples (voltage-levels) are trained and the corresponding communication power and BER are denoted as outputs. A sample LUT will be as follows:

Action (*Voltage swing*)	*State*	
	Power	BER
$a_1(V_1)$	Pw_1	BER_1
⋮	⋮	⋮

The input samples are collected at regular time intervals, called control cycle, at ns scale. Control cycle is defined as the minimum time required for the state transition. Duration of control cycle is based on the speed of I/O controller circuit. The next state variable needs to be predicted with an action for the input sample. This can be done by calculating a reward function to achieve an optimally estimated value based on state vectors, given by

$$R = f(Pw, BER). \tag{2}$$

Here, reward R is a function of communication power Pw and BER value BER as the state vectors. The relation between state variables and reward value is presented later. The next state and the current state can be the same depending on the workload characteristic. The reward R forms a

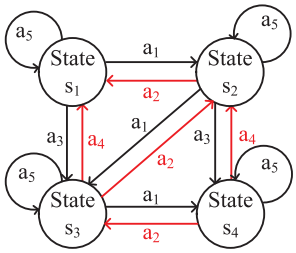


Fig. 3. State transition based on conventional Q-learning.

part of the expected Q-value, which decides the direction of state transition. The optimal estimation is chosen among the set of states to satisfy the required criteria by taking the corresponding action selected from the formed LUT. The expected Q-value is calculated as

$$\hat{\mathbf{Q}}(s_i,\ a_k) = Q(s_i,\ a_k)(1-\alpha) + \alpha(R + \gamma E). \tag{3}$$

Here α, γ denotes the learning rate and discount factor respectively. The optimal estimation E of state s_i can be calculated as follows

$$E = min\{\hat{\mathbf{Q}}(s_i,\ a_k)\},\ k = 0, \ldots, M. \tag{4}$$

Here, $\hat{\mathbf{Q}}(s_i, a_k)$ represents the expected Q-value after taking action a_k; M denotes the number of possible actions available at state s_i. The optimal estimate can be min or max depending on the reward function.

3.2 Conventional Q-Learning Control Flow

The self-adaptive tuning of the output-voltage swing at the CML buffer can be performed with the help of conventional Q-learning. The I/O communication power Pw_i and BER BER_i corresponding to output-voltage swing level V_i are considered as the components of state vectors S and output-voltage swing level as the action variable.

One example of a state diagram depicting the change of states by the conventional Q-learning algorithm is shown in Fig. 3. We consider four states and five actions. When the system is in state s_1 (s_2) and action a_1 (a_2) is selected, then the system changes to state s_2 (s_1); when the action a_3 is chosen, the state changes to s_3 (s_4) and remains in same state when action a_5 is chosen. Similarly, other state transitions also happens. One needs to note that the state transition depends on the action chosen at the current state.

Algorithm 1. Conventional Q-learning based adaptive tuning of output-voltage swing

Input: Communication power trace P_i, BER feedback from receiver and look-up-table (LUT)
Output: Adaptive tuning of output-voltage swing V_i
1. Predict tail current: $I_t(k+1) = \sum_{i=0}^{N-1} w_i I_t(k-i) + \xi$
2. Calculate corresponding communication power and BER
3. Reward: $R_w(s_i, a_k, s_{i+1}) = b_1\Delta(Pw_i) + b_2\Delta(BER_i)$
4. $\hat{\mathbf{Q}}(s_i, a_k) \leftarrow Q(s_i, a_k)(1-\alpha) + \alpha(R_w + \gamma E)$
5. Optimal value estimate: $E = min\{\hat{\mathbf{Q}}(s_i, a_k)\}, j = 0, \ldots, M$
6. By adjusting tail current using control bits, tune corresponding V_i

The proposed self-adaptive output-voltage swing tuning by the conventional Q-learning is presented in Algorithm 1.

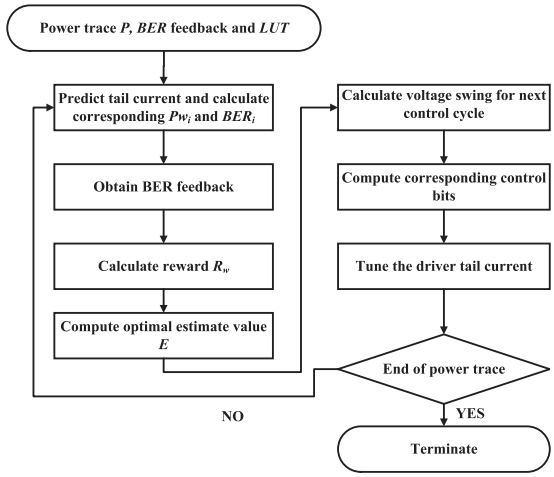


Fig. 4. Flowchart showing Q-learning based self-adaptive voltage swing tuning.

LUT is formed with output-voltage swing as action and the I/O communication power and BER as the state vectors. The tail current of the CML buffer is predicted by auto-regression (AR), as given in Line 1 of Algorithm 1. Based on the predicted tail current, the corresponding I/O communication power and the BER will be calculated. Furthermore, using the present I/O communication power and BER values, reward R_w is also decided. Since we have two factors, we consider the weighted sum of I/O communication power and BER. The reward function is given as follows

$$R_w(s_i,\ a_k,\ s_{i+1}) = b_1\Delta(Pw_i) + b_2\Delta(BER_i) \tag{5}$$

here b_1 and b_2 denote the weighted coefficients for normalized rewards of the communication power $\Delta(Pw_i)$ and BER $\Delta(BER_i)$. This calculation of reward is given in Line 3 of Algorithm 1. After calculating reward, the expected Q-value is calculated, Line 4 of Algorithm 1, and the optimal action is selected based on Q-values, as in Line 5 of Algorithm 1. This is how the adaptive tuning is performed by the conventional Q-learn algorithm. As a summary, the whole flow of adaptive tuning by the conventional Q-learning algorithm is shown in Fig. 4.

3.3 Accelerated Q-Learning

The conventional Q-learning algorithm [33] converges to the optimal after unlimited or large number of iterations, that may be too slow for convergence [34]. To overcome this convergence issue, we further use an accelerated Q-learning for adaptive tuning of output-voltage swing to achieve low power and faster convergence. Here, we will first present an example to illustrate the difference between conventional Q-learning and the accelerated Q-learning, followed by the modeling of a MDP and the according accelerated Q-learning algorithm for the adaptive tuning.

One example state diagram with four-states is shown in Fig. 5. For state s_1, action a_1 can change its state to s_2 with probability $P(s_1,\ a_1,\ s_2)$; For state s_2, action a_2 can change its state to s_1 with probability $P(s_2,\ a_2,\ s_1)$. Whereas action a_5 causes no change in state, whose probability is given as $P(s_1,\ a_5,\ s_2)$. The state transition probability P is given by a decaying function, $P = 1 - 1/(log(N_{s_i} + 2))$ with N_{s_i}

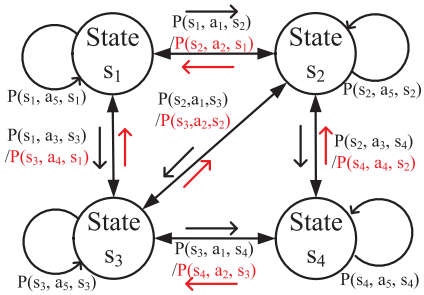


Fig. 5. State transition based on accelerated Q-learning.

denoting the number of visits to state s_i. The transition probability based state change ensures visit to all states at starting period. This will calculate Q-value to every available state. After this, Q-value based action will dominate and the optimal action with smallest Q-value will be selected.

Algorithm 2. Accelerated Q learning algorithm

Input: Communication power Pw, BER feedback
Output: Output-voltage
function Init()
  $1 \rightarrow P(s_i, a_k, s_{i+1})$
  Reward $R(s_i, a_k, s_{i+1}) = L$
  $v_{predict} \rightarrow V_{s_i}$
  Selection()
end function

function Selection()
  for $k = 1 : n$
    $V_{s_i}, BER_i \rightarrow s_i \in S$
    $\hat{\mathbf{Q}}(s_i, a_k) \leftarrow (1-\alpha) * Q(s_i, a_k) +$
    $\alpha * (R(s_i, a_k, s_{i+1}) + \gamma * \min(Q(s_i, a_k))$
    If $P(s_i, a_k, s_{i+1}) > rand(0, 1)$
      $a_k \leftarrow rand(A)$
    else
      $a_k \leftarrow \min(\hat{\mathbf{Q}}(s_i, a_k))$
    end if
    Update()
  end for
end function

function Update()
  Reward: $R(s_i, a_k, s_{i+1}) = b_1\Delta(P_i) + b_2\Delta(BER_i)$
  Update Policy (s_i, a_i), based on new Q
  $\forall\ s_i \in S$ {
    $a_k \leftarrow rand(A)$
    $Q(s_i, a_k) = \hat{\mathbf{Q}}(s_i, a_k)$
    $P(s_i, a_k, s_{i+1}) = 1 - \frac{1}{log(N_{s_i}+2)}$
}
end function

To find optimal value of MDP, probability based action selection, accelerated Q-learning algorithm can be utilized to evaluate the action-state pair as the Q-value. Conventional Q-learning algorithm converges to the optimal value after unlimited number of iterations [34]. The accelerated Q-learning [26] can be utilized to find an optimal solution with a faster convergence based on the predicted next state and the according transition probability with an initialized random action at first few states. Initial random actions helps the system explore environment faster and more easily find optimal states. To achieve a faster convergence, transition probability is utilized to select the action instead of directly selecting the next state. In this paper, we further utilize the accelerated Q-learning to find optimal point for the modeled MDP to solve *Problem 1* in Section 2.2, presented in Algorithm 2.

The first phase is initialization to form a LUT with states and corresponding actions. In addition, the transition probability P for all the states is set as 1 and the reward is set to a maximum value L. This process of initialization is presented as $Init()$ of Algorithm 2.

Prediction of the next state is performed to obtain the corresponding action. In the action selection phase, given by $Selection()$, the Q-value for the state and action pair is found iteratively, where the expected Q-value is given by

$$\hat{\mathbf{Q}}(s_i,\ a_k) = (1-\alpha) * Q(s_i,\ a_k) + \alpha * delta \tag{6a}$$

$$delta = R(s_i, a_k,\ s_{i+1}) + \gamma * \min_{a \in A}(Q(s_i,\ a_k)). \tag{6b}$$

Where $Q(s_i, a_k)$ represents accumulated Q-value and $\hat{\mathbf{Q}}(s_i, a_k)$ represents expected Q-value after taking action.

In each iteration, the action is selected either based on the transition probability or based on the minimum Q-value (or policy). If the transition probability is larger than the threshold, a random action is selected; otherwise, the policy action with the minimum Q-value is selected. The random action will happen at the first few rounds to explore the design space. As the learning process continues, the policy action with the calculated Q-value will dominate and become more accurate to use. As such, a higher probability exists to select the action a_k with minimum Q-value. The policy action with the minimum Q-value can be described as below

$$a_k \leftarrow min(\hat{\mathbf{Q}}(s_i,\ a_k)). \tag{7}$$

Last, the phase of $Update()$ is activated at the end of each iteration of $Selection()$ function. The reward is defined as the weighted value of BER and Pw and updated as given in (5).

At the end of $Update()$, each state will be randomly visited and Q-value will be updated accordingly. The transition probability $P(s_i,\ a_k,\ s_{i+1})$ is also updated as N_{s_i} (increases with the number of visits to state s_i) after each iteration.

Note that with the prediction of states s_i as in function $Init()$ and $Update()$ and the transition probability, the convergence to the optimal solution is accelerated [27], [37]. This is done at the end of each round with the random action a_k to visit state s_i. In brief, whole flow of adaptive tuning by accelerated Q-learning is shown in Fig. 6.

LUT can be implemented online with the corresponding control bits calculated and fed back to CML buffer to tune the DAC current of CML buffer. Note that LUT can be implemented in the hardware with multiple AND/OR partial matching logic circuit instead of read only memory (ROM). This LUT implementation has higher speed and low power consumption compared to ROM.

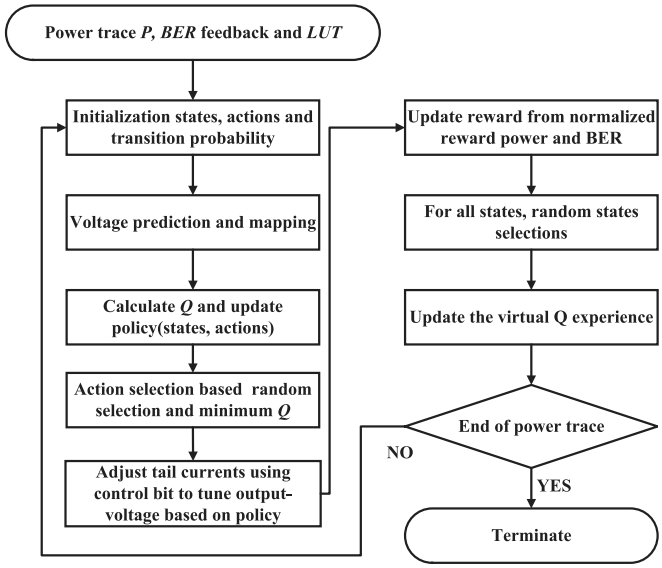


Fig. 6. Flowchart of accelerated Q-learning based self-adaptive voltage swing tuning.

3.4 System Models

The prediction of power and BER of I/O communication channel and their models are discussed in this section. The first component of the state vector is the I/O communication power. The system power model includes the I/O communication power of driver and the TSI T-line. Both are the functions of output-voltage swing V_i. For the CML based driver with TSI T-line [38], the I/O communication power is given by

$$Pw_i = V_i \cdot \left(I_t + \frac{\eta * V_{dd} * s}{R_D + Z_{diff}} * f \right). \tag{8}$$

Here I_t is driver tail current; s is duration of signal pulse; η is activity factor; R_D is the resistance of driver; and Z_{diff} is the characteristic impedance of the TSI T-line.

The tail current I_t at the current control cycle is set by analog design and can be obtained from measurement. Tail current for the next control cycle can be predicted by auto-regression as

$$I_t(k+1) = \sum_{i=0}^{N-1} w_i I_t(k-i) + \xi. \tag{9}$$

Here $I_t(k+1)$ denotes the predicted tail current at $k+1$th control cycle; w_i represents AR coefficient; ξ is the prediction error and N represents order of AR prediction. Based on the predicted tail current, I/O communication power for next control cycle can be calculated.

The second component of the state vector is the BER of I/O communication, which is feedback from the receiver. BER depends on the output-voltage swing, external noise, channel noise etc., [39]. In a wire-line communication system [40], the BER can be estimated with the dependence on the output-voltage swing as

$$BER_i = \frac{1}{2} erfc\left(\frac{V_i}{\sqrt{2}\sigma_v} \right). \tag{10}$$

Here, the $erfc$ is complementary error function; V_i refers to the ith output-voltage swing level and σ_v is the standard deviation of the noise.

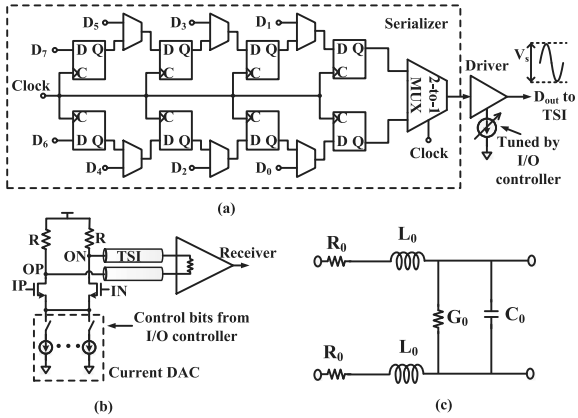


Fig. 7. (a) Transmitter with 8:1 serializer; (b) Adaptive tuning of driver tail current;(c) TSI realized by a T-line.

As such, the BER can be obtained from the ECC at the receiver by counting errors. During the learning process, based on the BER obtained at the ECC under one output-voltage swing, the standard deviation of noise, σ_v is estimated from (10).

4 2.5D TSI I/O Circuit Design

In this section, we discuss about each component of the 2.5D TSI I/O link such as transmitter (Tx) and receiver (Rx). To operate with high bandwidth by single channel of 2.5D TSI I/O, we employ 8:1 serializer in the Tx and 1:8 deserializer at the Rx. Each of the Tx and Rx has a voltage-controlled oscillator (VCO) to generate the required clock signal (2 GHz). Both the Tx and Rx are terminated for the 2.5D TSI based T-line with matched 50Ω resistance. At the Rx, the serial bit stream is sampled and deserialized; and is re-synchronized by the recovery clock from the clock data recovery (CDR) block.

4.1 Transmitter and Receiver

Transmitter Tx employs a 8:1 serializer to convert 8-bit parallel data into serial data, as shown in Fig. 7a. Four digital D flip-flops are implemented as a shift-register chain for each of the odd (D_1, D_3, D_5, D_7) and even (D_0, D_2, D_4, D_6) bits of data. This is followed by a 2:1 MUX to combine them altogether. A current-mode logic output driver is used to drive the TSI T-line from Tx to Rx on the common substrate. The CML output stage is powered by the fixed supply ($1.2\,V$). The I/O communication power Pw depends on the output-voltage swing and the tail current of the driver. For example, one can generate control bits to tune the tail current of the CML driver and alter the output-voltage swing, as shown in Fig. 7b.

What is more, compared to the traditional serial I/Os based on backplane PCB trace [4], [5], 2.5D TSI I/Os does not need complex equalizer circuits at the receiver due to small signal loss in the TSI T-line channel. A sampler at receiver front-end is employed to convert the current-mode signals into digital levels. After data decision, this data is processed in the digital domain, saves more power compared to analog de-multiplexer. A delay-locked loop (DLL) based clock-data recovery at receiver is implemented to de-skew the sampling clocks, shown in Fig. 8a.

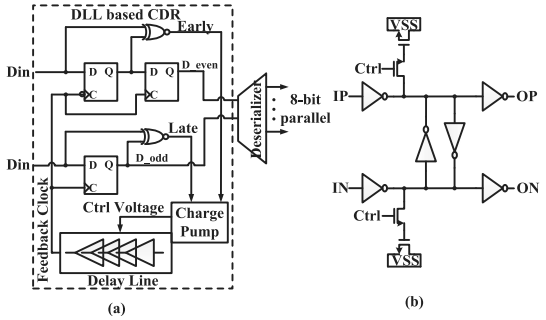


Fig. 8. (a) The architecture of DLL based CDR; (b) The voltage controlled delay cell in the DLL.

In this CDR design, a half-rate clock architecture is employed to decrease the digital circuit working frequency and save power. Two exclusive-or (XOR) gates in Fig. 8a form a phase detector to judge the sampling clock position compared to input data. It compares the input data edge with rising edge sampled signal to obtain the "$early$" and "$late$" pulse. And then a charge-pump block converts these pulses into variable voltage to control the DLL delay line, which can tune the delay phase of the clocks and also provide feedback to the sampler. The schematic of voltage-controlled delay cell (in Fig. 8a) is illustrated in Fig. 8b, which is based on inverter chain for reducing the constant current consumption. This implementation of DLL in the CDR circuit makes inherently stable and avoids jitter accumulation.

4.2 Error Correcting Code

To determine the historical BER for future control, data is encoded using the hamming code [41] and transmitted along with the parity check bits. As shown in Fig. 9, 32-bit parallel data (D-32) is initially stored in the output FIFO of the transmitter. For the 32-data bits, seven parity bits are generated by the parity generator and an additional MSB of parity check vector is set as 0. Parity generator uses the code generator matrix C to generate parity bit vector P as shown in Fig. 10a, where the parity generator consists of a set of AND and XOR gates. As such, the total encoded data to be transmitted will be 40-bit for every 32-bit of data. One MUX is implemented for serial transmission. The data format of the transmitted data is presented in Fig. 10b.

At the receiver, the 32-bit data is stored in the input FIFO (D-32 bits) and the last 7-bits of the 8-bit (parity) are utilized for error checking and correction. The checking result vector (R) is generated from the parity code P. By summing the result vector, one can detect if any bit is wrong and a left-shifter is used to correct bit error. The current implementation of ECC has capability to correct 1-bit error but can detect multiple bit errors. It can be used to obtain the historical BER at the receiver and is further fed back to the transmitter. BER is calculated as the ratio of total number of error bits received and the total number of bits transmitted.

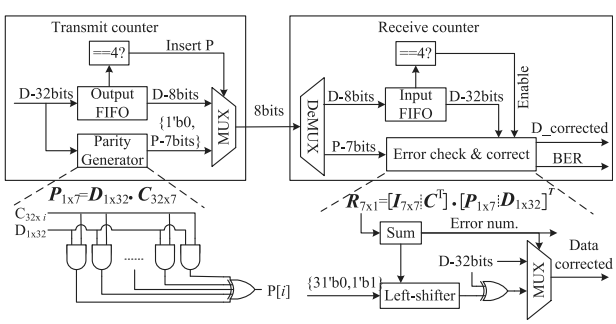


Fig. 9. Encoding and decoding at transmitter and receiver.

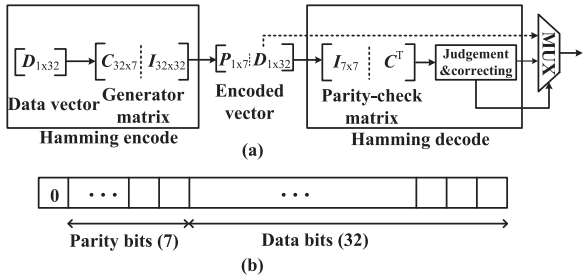


Fig. 10. (a) Data encoding and decoding; (b) Transmitted data format.

4.3 Adaptive Circuit Design

Based on the calculated BER from the ECC, a feedback signal is sent to the I/O controller at the transmitter. This signal is considered as one of the component for I/O control block that forms a look-up-table. The I/O controller generates the corresponding control bits. The control bits can control the DAC current at the tail of CML buffer driving the TSI T-line. Thus, the output-voltage swing is tuned by varying the tail current of CML buffer. As shown in Fig. 7b, the CML driver with variable current source is set by the DAC current and load resistor. The DAC tail current source is composed of a group of current sources in parallel with switches controlled by the control bits generated from the I/O controller. When the driver tail current is varied, the output-voltage swing will change. Generally, the load resistor is set 50Ω for the TSI T-line impedance matching. In this paper, tail current source is varied from 2 to 5 mA.

5 Simulation Results

The 2.5D adaptive TSI I/O circuit verification is performed in Cadence Virtuoso (Ultrasim-Verilog) and Matlab 2013a. The geometry and technology details are presented in Table 2. An eight-core MIPS microprocessor with eight-bank of SRAM memory is designed with GF 65 nm CMOS. The 2.5D TSI T-line is of length 3 mm and $10\ \mu$m width,

TABLE 2
System Settings for Memory-Logic Integration with TSI I/O

Item	Description	Value	Size
Microprocessor	Technology node	65 nm	$0.3mm^2$
	Frequency	500 MHz	
	Dissipation power	15 mW	
I/O controller	Output-voltage swing	0.1V, 0.15V, 0.2V, 0.3V	$0.03mm^2$
	Driving current	2 mA, 3 mA, 4 mA, 5 mA	
	Number of levels	4	
	Switching time	0.4 ns	
TSI	Length	3 mm	$3mm^2$
	Inductance	300 pH	
	Resistance	5Ω	
	Capacitance	60 fF	
Memory	SRAM	16 KB	$0.2mm^2$
	Power dissipation	6 mW	

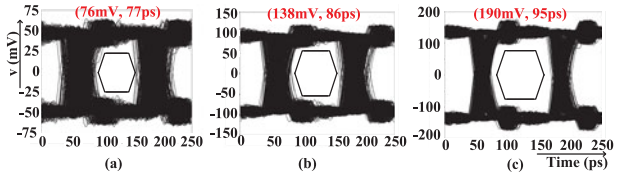


Fig. 11. Eye diagram of output data with different driver current (or output-voltage swing) levels: (a) $2\,mA$; (b) $3\,mA$; (c) $4\,mA$.

driven by the CML buffer. The power traces are measured from Cadence Virtuoso and control cycle is set as 1 ns, larger than switching time of I/O controller. The I/O management controller is based on the conventional Q-learning and accelerated Q-learning output to balance the I/O communication power and BER at receiver. The look-up-table is designed with I/O communication power and BER, utilized for adaptively tuning the output-voltage swing. The LUT with voltage-swing level, communication power and BER values is set up as follows: (100 mV, $6.27E-2$ mW, $7.03E-2$BER), (150 mV, $1.41E-1$ mW, $1.35E-2$BER), (200 mV, $2.51E-1$ mW, $1.61E-3$BER), and (300 mV, $5.64mE-1$ mW, $4.93E-6$BER). This LUT is more dependent on the characteristics of the circuit rather than the application. The learning rate α and discount factor γ are set as 0.5 and 0.9 respectively. Auto-regression of order 8 is used for load current (or I/O communication power) prediction. The error between the predicted and actual values is less than 0.3 percent on average. The adaptive voltage-swing tuning algorithm is carried out in Matlab 2013a with offline training of samples. The simulations are carried out on a PC with Intel i7 core processor running at 3.2 GHz with 8 GB RAM. Please note that the power of training processor is not included in the below reported results. The overall I/O performance can provide a minimum of 76 mV peak-to-peak signal swing with 4 Gb/s bandwidth, and the power consumption is only 12.5 mW. The adaptive self-tuning of output-voltage swing may come with a little area overhead of $0.03\,\mathrm{mm}^2$ for additional control circuits and a latency of 100-200ps.

For comparison, we have three implementations: 1) the I/O without management (normal); 2) I/O control by the conventional Q-learning (conv); and 3) I/O control by the accelerated Q-learning (acc). In the following, first, the tuning of output-voltage swing and the resulting eye-diagrams is presented; second, we present the I/O management results by conventional Q-learning and further by accelerated Q-learning based adaptive tuning; and finally, the power saving and management time as well as energy efficiency are presented under different benchmarks of workloads.

5.1 Adaptive Tuning Results

The characteristics of eye-diagrams under different driver currents are presented in Fig. 11 by introducing 10 percent clock cycle-to-cycle jitter (noise) at the TSI I/O channel. Note that different driving currents can make different eye openings under the noise in channel. A larger eye opening is associated with a higher current driving ability (or a larger output-voltage swing), which has a minimum opening of 76 mV amplitude and 77 ps timing margin with 2 mA driver current and further increases with the

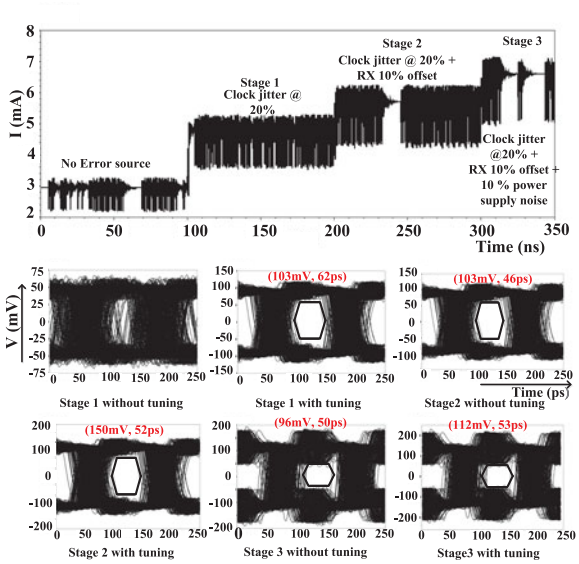


Fig. 12. The eye-diagrams under adaptive current (or power) adjustment by output-voltage swing tuning.

driving current. One needs to observe that with the increase in driver current BER decreases, but at the cost of power. As BER does not need to be low at all times, one can leverage the trade-off between the power reduction and the necessary BER.

We further study the eye-diagram under the control of voltage-swing tuning to verify the functionality of adaptive I/O circuit tuning. Fig. 12 shows the current consumption under different levels of output-voltage swing. The sources of error are introduced in three stages: stage 1 is to introduce 20 percent of clock jitter; stage 2 is to add additional 10 percent receiver offset; and stage 3 is to further add additional 10 percent power supply noise. As discussed previously, with the increase in noise, tail current at CML buffer is varied to tune the output-voltage swing. For example, stage 1, which has only clock jitter, current is increased to 5 mA to improve eye-opening as (103 mV, 62 ps). With the increase in noise i.e., stage 3, the current is increased adaptively. The difference in eye-diagrams with

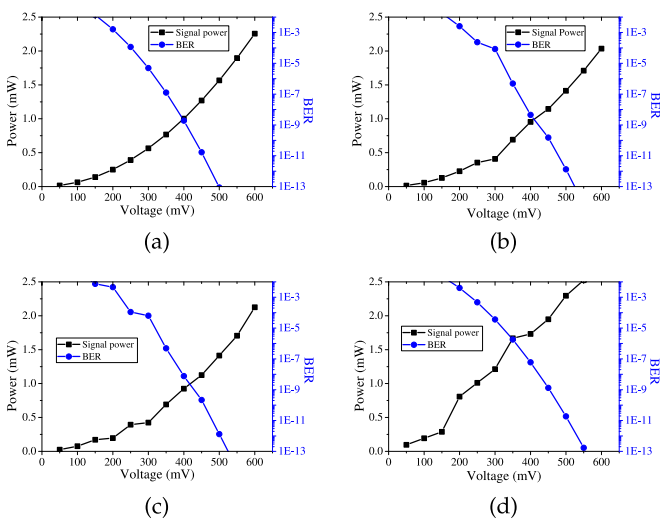


Fig. 13. Trade-off between BER and power for: (a) *FFT* benchmark; (b) *File transfer* benchmark; (c) *bzip2* benchmark; (d) *gzip* benchmark.

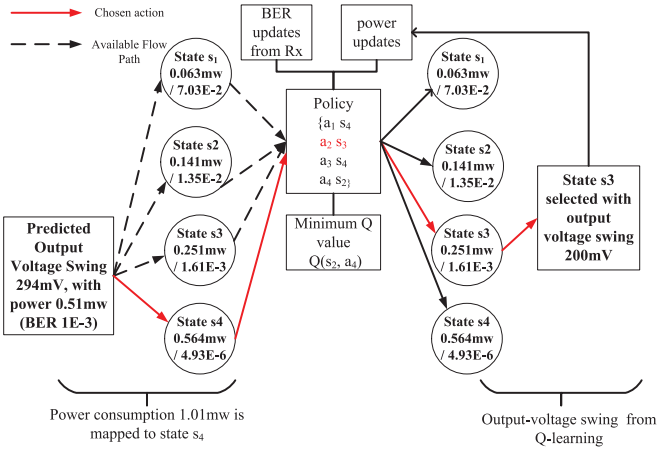


Fig. 14. Example of one adaptive I/O control by conventional Q-learning.

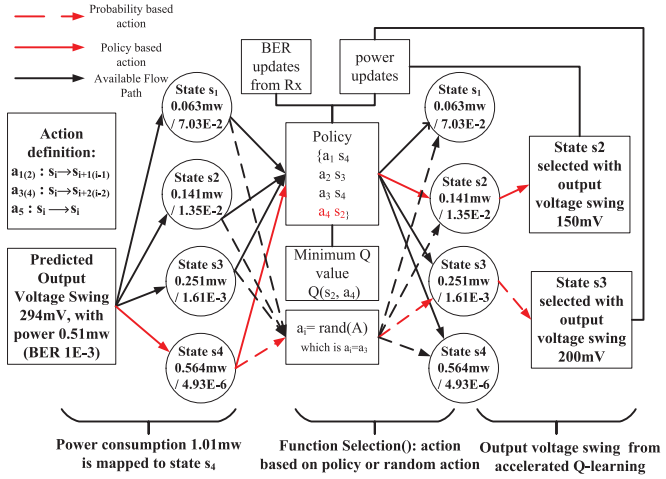


Fig. 15. Example of one adaptive I/O control by accelerated Q-learning.

and without tuning the output-voltage swing is shown in Fig. 12. One can observe that for stage 3, without tuning the tail current, the eye opening is 96 mV, but the eye opening increases to 112 mV by adaptively tuning the tail current. Similar improvement in eye openings is shown for other stages as well.

5.2 BER versus Power Results

Fig. 13 shows the trade-off between BER (blue circle) and the I/O communication power (black rectangle). With the increase in output-voltage swing, BER decreases at the cost of the I/O communication power, which is validated through four benchmarks. For example, for *file transfer* benchmark, at an output voltage-swing of 350 mV, the I/O communication power is 0.693 mW with a BER of $4.85E-7$; whereas at an output-voltage swing of 400 mV, BER goes down to $4.47E-9$ at the cost of increased communication power by 0.953 mW. This observation shows that there is a balance point where we can use less power with the BER guaranteed. Furthermore, as the sensitivity of power and BER with voltage-swing level are different for each benchmark, adaptive tuning of I/O voltage swing based on conventional Q-learning and further accelerated Q-learning can help to achieve an optimal trade-off between communication power and BER for different benchmarks respectively.

5.3 Adaptive I/O Control by Conventional Q-Learning

An example of adaptive output-voltage swing tuning by conventional Q-learning is shown in Fig. 14. Initially, the voltage-swing for next control-cycle is predicted as 294 mV. As it is impractical to have an LUT with continuous values, we map the incoming value to the closest value in LUT. Thus, the voltage-swing 294 mV is fit to the state s_4 because of the nearest voltage level. Based on the predicted value and present value, reward is calculated as in (5). Further, corresponding voltage-swing level is obtained based on the calculated reward and optimal estimate function. This results in setting the voltage-swing level (action) to 200 with 0.25 mW communication power i.e., state s_3. It needs to be noted that the reduction in voltage swing level is due to the tolerance to BER. Further, the BER and I/O communication power are updated.

5.4 Adaptive I/O Control by Accelerated Q-Learning

An example of one adaptive tuning by the accelerated Q-learning is shown in Fig. 15. Initially, the voltage-swing for the next control-cycle is predicted as 294 mV. The voltage-swing 294 mV with power 0.51 mW fits to the state s_4 because of the nearest voltage level. The action will be selected based on the probability check as in $Selection()$ of Algorithm 2. As shown in Fig. 15, the solid red line and red dotted line indicate the policy-based action selection and probability-based action selection, respectively. For the probability-based action selection (red dotted line), a random action is selected to visit any of the available states. This takes place with high probability at the starting period to ensure the visit to all the available states, where its Q-value will be updated accordingly. Afterwards, the action with minimum Q-value will dominate and be considered as the optimal action. As Fig. 15 shows, action a_4 is eventually selected leading to the voltage level associated with state s_2. Afterwards, the new voltage-swing is assigned based on the state s_2 as 0.141 mW. As such, the driver tail-current is tuned to have the output voltage-swing as 150 mV. Last, the reward $R(s_4,\ a_4,\ s_2)$ will be updated based on the feedback of BER and power obtained at Rx. One needs to note that as shown in Fig. 14, conventional Q-learning selects state s_3 due to action based state change, however it may converge to state s_2 after few iterations.

5.5 Benchmarking Results

The I/O communication power saving is verified for various SPEC benchmarks [42] in Table 3 by the self-adaptive tuning using: no Q-learning (normal); the conventional Q-learning (conv); and the accelerated Q-learning (acc). It shows that the accelerated Q-learning algorithm is more power efficient with faster convergence. For example, for *bzip2* benchmark, the I/O communication power without the adaptive tuning is 0.267 mW, which is reduced to 0.224 mW when adaptively tuned by the conventional Q-learning; and is further reduced to 0.210 mW with the accelerated Q-learning. On average, the power consumption

TABLE 3
Power and Run-Time Comparisons for the Adaptive Tuning with Q-Learning under Various Benchmarks

Benchmark	Communication power (mW)			Power saving			Run time(s)	
	Acc	Conv	Normal	Acc	Conv	Acc	Conv	*Improvement*
ammp	0.231	0.250	0.296	22.11%	15.54%	0.0804	0.1049	23.28%
applus	0.474	0.494	0.555	14.64%	10.99%	0.0817	0.1051	22.25%
apsi	0.652	0.674	0.744	12.34%	9.41%	0.0817	0.1051	22.25%
art	0.199	0.216	0.255	22.01%	15.29%	0.0829	0.1057	21.55%
bzip2	0.210	0.224	0.267	21.19%	16.10%	0.0829	0.1056	21.50%
crafty	0.381	0.419	0.461	17.39%	9.11%	0.0822	0.1061	22.53%
eon	0.234	0.253	0.298	21.32%	15.10%	0.0807	0.1051	23.24%
equake	0.232	0.249	0.295	21.07%	15.59%	0.0821	0.1047	21.60%
facerec	0.333	0.364	0.409	18.53%	11.00%	0.0833	0.1060	21.42%
fft	0.153	0.179	0.199	23.12%	10.05%	0.2002	0.2309	13.30%
file transfer	0.512	0.554	0.598	14.58%	7.36%	0.2057	0.2485	17.22%
fma3d	0.539	0.561	0.623	13.35%	9.95%	0.0850	0.1083	21.51%
galgel	0.241	0.261	0.307	21.56%	14.98%	0.0820	0.1069	23.29%
gap	0.284	0.310	0.355	19.86%	12.68%	0.0808	0.1036	22.01%
gcc	0.507	0.536	0.587	13.60%	8.69%	0.0838	0.1064	21.24%
gzip	0.238	0.257	0.302	21.08%	14.90%	0.0817	0.1051	22.26%
lucas	0.518	0.548	0.602	13.90%	8.97%	0.0818	0.1053	22.32%
mcf	0.248	0.264	0.307	19.41%	14.01%	0.0817	0.1056	22.63%
mesa	0.228	0.243	0.287	20.45%	15.33%	0.0846	0.1065	20.56%
mgrid	0.274	0.297	0.346	20.83%	14.16%	0.0818	0.1051	22.17%
parser	0.429	0.458	0.508	15.43%	9.84%	0.0825	0.1084	23.89%
perlbmk	0.207	0.226	0.266	22.21%	15.04%	0.0822	0.1052	21.86%
sixtrack	0.341	0.357	0.412	17.39%	13.35%	0.0819	0.1065	23.10%
swim	0.242	0.262	0.310	21.82%	15.48%	0.0812	0.1052	22.81%
twolf	0.216	0.235	0.279	22.38%	15.77%	0.0839	0.1037	19.09%
vortex	0.225	0.239	0.288	21.94%	17.01%	0.0832	0.1056	21.21%
vprs	0.240	0.255	0.300	19.96%	15.00%	0.0826	0.1059	22.00 %
wupwise	0.424	0.442	0.502	15.39%	11.95%	0.0836	0.1058	20.98%
Average	-	-	-	18.89 %	12.95%	0.0912	0.1152	20.83%

of whole system (transmitter, receiver and the I/O) is 19 mW with an energy efficiency of 4.75 pJ/bit without the adaptive tuning, which is further reduced to 12.5 mW by the adaptive tuning. The system power is directly obtained from Cadence simulation result. On average, the I/O communication power saving of 18.89 percent and energy efficiency improvement of 15.11 percent is achieved by the adaptive tuning based on the accelerated Q-learning; and 12.95 percent of power saving and 14 percent of energy-efficiency improvement are achieved when using the conventional Q-learning. Please note that the power values reported in Table 3 is the I/O communication power, which does not include the transmitter and receiver power consumption. What is more, on average, the accelerated Q-learning takes 0.091 s for convergence, whereas the conventional Q-learning takes 0.115 s. The reported time includes the training of samples, which is done off-line. The run time has also improved with accelerated Q-learning by an average of 20.83 percent compared to conventional Q-learning based I/O management as shown in Table 3.

6 Conclusion

In this paper, towards the energy-efficient 2.5D memory-logic integration, we have investigated the I/O management by self-adaptive adjustment of I/O output-voltage swing. With the use of predicted I/O communication BER and power, the conventional and accelerated Q-learning based I/O managements have been developed upon the workload characteristics. Experimental results have shown that the developed adaptive 2.5D I/Os designed in 65 nm CMOS can achieve an average of 12.5 mW I/O power, 4 GHz bandwidth and 3.125 pJ/bit energy efficiency for one channel under 10^{-6} BER. When compared to the uniform output-voltage swing based I/O, the I/O managements by conventional Q-learning and the accelerated Q-learning can achieve 12.95 and 18.89 percent communication power reduction and 14 and 15.11 percent energy efficiency improvement, respectively.

Acknowledgments

The authors would like to acknowledge the funding support from Intel Research Lab (USA), MOE Tier-2 (Singapore), and A*STAR PSF Fund (Singapore). The preliminary result was published in ISLPED'14.

References

[1] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in Multi-core architectures: understanding mechanisms, overheads and scaling," in *Proc. IEEE Int. Symp. Comput. Arch.*, 2005, pp. 408–419.

[2] S. Rusu, H. Muljono, D. Ayers, S. Tam, W. Chen, A. Martin, S. Li, S. Vora, R. Varada, and E. Wang, "Ivytown: A 22*nm* 15-core enterprise xeon processor family," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2014, pp. 102–103.

[3] J. Bulzacchelli, M. Meghelli, S. Rylov, W. Rhee, A. Rylyakov, H. Ainspan, B. Parker, M. Beakes, A. Chung, T. Beukema, P. Pepeljugoski, L. Shan, Y. Kwark, S. Gowda, and D. Friedman, "A 10-Gb/s 5-tap DFE/4-tap FFE transceiver in 90-nm CMOS technology," *IEEE J. Solid-State Circuits*, vol. 41, no. 12, pp. 2885–2900, Dec. 2006.

[4] S. Gondi and B. Razavi, "Equalization and clock and data recovery techniques for 10-Gb/s CMOS Serial-link receivers," *IEEE J. Solid-State Circuits*, vol. 42, no. 9, pp. 1999–2011, Sep. 2007.

[5] M. Pozzoni, S. Erba, P. Viola, M. Pisati, E. Depaoli, D. Sanzogni, R. Brama, D. Baldi, M. Repossi, and F. Svelto, "A multi standard 1.5 to 10Gb/s Latch-based 3-tap DFE receiver with a SSC tolerant CDR for serial backplane communication," in *Proc. IEEE Symp. VLSI Circuits*, 2008, pp. 172–173.

[6] W. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. Sule, M. Steer, and P. Franzon, "Demystifying 3D ICs: The pros and cons of going vertical," *IEEE Design Test Comput.*, vol. 22, no. 6, pp. 498–510, Nov./Dec. 2005.

[7] Y. Xie, G. H. Loh, B. Black, and K. Bernstein, "Design space exploration for 3D architectures," *ACM J. Emerging Technol. Comput. Syst.*, vol. 2, no. 2, pp. 65–103, Apr. 2006.

[8] M. Motoyoshi, "Through-silicon via (TSV)," *IEEE Proc.*, vol. 97, no. 1, pp. 43–48, Jan. 2009.

[9] I. Loi, P. Marchal, A. Pullini, and L. Benini, "3D NoCs-unifying inter & intra chip communication," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2010, pp. 3337–3340.

[10] M. Sai, H. Yu, Y. Shang, C. S. Tan, and S. K. Lim, "Reliable 3-D clock-tree synthesis considering nonlinear capacitive TSV model with Electrical-thermal-mechanical coupling," *IEEE Trans. Comput.-Aided Design Integrated Circuits Syst.*, vol. 32, no. 11, pp. 1734–1747, Nov. 2013.

[11] D. H. Kim, K. Athikulwongse, M. Healy, M. Hossain, M. Jung, I. Khorosh, G. Kumar, Y.-J. Lee, D. Lewis, T.-W. Lin, C. Liu, S. Panth, M. Pathak, M. Ren, G. Shen, T. Song, D. H. Woo, X. Zhao, J. Kim, H. Choi, G. Loh, H.-H. Lee, and S. K. Lim, "Design and analysis of 3D-MAPS (3D massively parallel processor with stacked memory)," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 112–125, Jan. 2015.

[12] Y.-J. Lee and S. K. Lim, "Ultra high density logic designs using monolithic 3D integration," *IEEE Trans. Comput.-Aided Design Integrated Circuits*, vol. 32, no. 12, pp. 1892–1905, Dec. 2013.

[13] J. Zhang, M. O. Bloomfield, J.-Q. Lu, R. J. Gutmann, and T. Cale, "Thermal stresses in 3D IC inter-wafer interconnects," in *Elsevier J. Microelectron. Eng.*, vol. 82, no. 3, pp. 534–547, Dec. 2005.

[14] H. Yu, Y. Shi, L. He, and T. Karnik, "Thermal via allocation for 3-D ICs considering temporally and spatially variant thermal power," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 12, pp. 1609–1619, Dec. 2008.

[15] H. Yu, J. Ho, and L. He,"Allocating power ground vias in 3D ICs for simultaneous power and thermal integrity," *ACM Trans. Design Autom. Electron. Syst.*, vol. 14, no. 3, p. 41, 2009.

[16] J. Cubillo, R. Weerasekera, Z. Z. Oo, E.-X. Liu, B. Conn, S. Bhattacharya, and R. Patti, "Interconnect design and analysis for through silicon interposers (TSIs)," in *Proc. IEEE Int. 3D Syst. Integration Conf.*, 2012, pp. 1–6.

[17] S.-S. Wu, K. Wang, M. Sai, T.-Y. Ho, M. Yu, and H. Yu, "A thermal resilient integration of many-core microprocessors and main memory by 2.5D TSI I/Os," in *Proc. ACM/IEEE Conf. Design, Autom. Test Eur.*, 2014, pp. 1–4.

[18] T. Ishii, H. Ito, M. Kimura, K. Okada, and K. Masu, "A 6.5-mW 5-Gbps on-chip differential transmission line interconnect with a low-latency asymmetric Tx in a $180nm$ CMOS technology," in *Proc. IEEE Asian Solid-State Circuits Conf.*, 2006, pp. 131–134.

[19] J. Wang, S. Ma, M. Sai, M. Yu, R. Weerasekera, and H. Yu, "High-speed and low-power 2.5D I/O circuits for memory-logic-integration by through-silicon interposer," in *Proc. IEEE Int. 3D Syst. Integration Conf.*, 2013, pp. 1–4.

[20] J. Tschanz and N. Shanbhag, "A low-power, reconfigurable adaptive equalizer architecture," in *Proc. Asilomar Conf. Signals, Syst. Comput.*, 1999, pp. 1391–1395.

[21] N. Tzartzanis and W. Walker, "Differential current-mode sensing for efficient on-chip global signaling," *IEEE J. Solid-State Circuits*, vol. 40, no. 11, pp. 2141–2147, Nov. 2005.

[22] J. sun Seo, R. Ho, J. Lexau, M. Dayringer, D. Sylvester, and D. Blaauw, "High-bandwidth and low-energy on-chip signaling with adaptive pre-emphasis in 90nm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest Tech. Papers*, 2010, pp. 182–183.

[23] S. Park, M. Qazi, L.-S. Peh, and A. P. Chandrakasan, "40.4fJ/bit/mm low-swing on-chip signaling with self-resetting logic repeaters embedded within a mesh NoC in 45nm SOI CMOS," in *Proc. ACM/IEEE Design, Autom. Test Eur. Conf. Exhib.*, 2013, pp. 1637–1642.

[24] I. Foster, A. Roy, and V. Sander, "A quality of service architecture that combines resource reservation and application adaptation," in *Proc. IEEE Int. Workshop Quality Serv.*, 2000, pp. 181–188.

[25] V. Raghunathan, M. B. Srivastava, and R. K. Gupta, "A survey of techniques for energy efficient on-chip communication," in *Proc. ACM/IEEE Design Autom. Conf.*, 2003, pp. 900–905.

[26] A. Gosavi, "A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis," *Mach. Learn. J.*, vol. 55, no. 1, pp. 5–29, Apr. 2004.

[27] N. Mastronarde and M. van der Schaar, "Online reinforcement learning for dynamic multimedia systems," *IEEE Trans. Image Process.*, vol. 19, no. 2, pp. 290–305, Feb. 2010.

[28] M. Triki, A. Ammari, Y. Wang, and M. Pedram, "Reinforcement learning-based dynamic power management of a battery-powered system supplying multiple active modes," in *Proc. IEEE Eur. Modelling Symp.*, 2013, pp. 437–442.

[29] H. Shen, Y. Tan, J. Lu, Q. Wu, and Q. Qiu, "Achieving autonomous power management using reinforcement learning," *ACM Trans. Design Autom. Electron. Syst.*, vol. 18, no. 2, pp. 24:1–24:32, Apr. 2013.

[30] N. Mastronarde, K. Kanoun, D. Atienza, P. Frossard, and M. van der Schaar, "Markov decision process based energy-efficient on-line scheduling for slice-parallel video decoders on multicore systems," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 268–278, Feb. 2013.

[31] D. Xu, M. Sai, H. Huang, N. Yu, and H. Yu, "An energy-efficient 2.5D Through-silicon interposer I/O with self-adaptive adjustment of output-voltage swing," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, 2014, pp. 93–98.

[32] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *ACM J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, May 1996.

[33] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn. J.*, vol. 8, nos. 3/4, pp. 279–292, May. 1992.

[34] M. L. Littman, "Value-function reinforcement learning in Markov games," *ACM Cognitive Syst. Res.*, vol. 2, no. 1, pp. 55–66, Apr. 2001.

[35] M. Ghavamzadeh, H. J. Kappen, M. G. Azar, and R. Munos, "Speedy Q-learning," in *Proc. Adv. Neural Inform. Process. Syst.*, 2011, pp. 2411–2419.

[36] E. E-Dar and Y. Mansour, "Learning rates for Q-learning," *J. Mach. Learn.*, vol. 5, pp. 1–25, 2003.

[37] N. Mastronarde and M. van der Schaar, "Online reinforcement learning for dynamic multimedia systems," *IEEE Trans. Image Process.*, vol. 19, no. 2, pp. 290–305, Feb. 2010.

[38] I. Ndip, B. Curran, K. Lobbicke, S. Guttowski, H. Reichl, K. Lang, and H. Henke, "High-frequency modeling of TSVs for 3-D chip integration and silicon interposers considering skin-effect, dielectric quasi-TEM and slow-wave modes," *IEEE Trans. Components, Packaging Manufacturing Technol.*, vol. 1, no. 10, pp. 1627–1641, Oct. 2011.

[39] S. K. Das, S. K. Sen, and R. Jayaram, "Call admission and control for quality-of-service provisioning in cellular networks," in *Proc. IEEE Int. Conf. Universal Personal Commun. Rec.*, 1997, pp. 109–113.

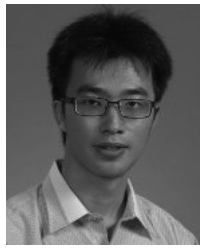
[40] R. Shafik, S. Rahman, and R. Islam, "On the extended relationships among EVM, BER and SNR as performance metrics," in *Proc. IEEE Int. Conf. Elect. Comput. Eng.*, 2006, pp. 408–411.

[41] D. K. Bhattacharryya and S. Nandi, "An efficient class of SEC-DED-AUED codes," in *Proc. Int. Symp. Parallel Archit., Algorithms, Netw.*, 1997, pp. 410–416.

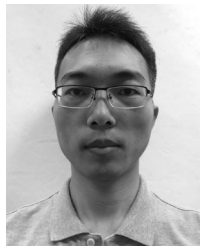
[42] (2000). SPEC 2000 CPU benchmark suits. [Online]. Available: http://www.spec.org/cpu/

Sai Manoj P. D. (S'13) received the BTech and MTech degrees from JNTU Anantapur, India in 2010 and IIIT, Bangalore, India in 2012, respectively, and he is currently working towards the PhD degree with the School of Electrical and Electronics Engineering. He joined Nanyang Technological University, Singapore in 2012. His research interests includes 3D TSV and 2.5D I/O modeling, system-level power management, machine learning, and Network-on-Chips. He also received A. Richard Newton Young Research Fellow Award in DAC 2013. He is a student member of the IEEE.

Hao Yu (M'06-SM'14) received the BS degree from Fudan University, China and the PhD degree from the Electrical Engineering Department, University of California. He was a senior research staff at Berkeley Design Automation. Since October 2009, he has been an assistant professor at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His primary research interests are 3D-IC and RF-IC at nano-tera scale. He has 156 peer-reviewed IEEE/ACM publications, including Best Paper Award from ACM TODAES'10, Best Paper Award nominations in DAC'06, ICCAD'06, ASP-DAC'12, Best Student Paper (advisor) Finalist in SiRF'13, RFIC'13, IMS'15 and Inventor Award from semiconductor research cooperation. He is an associate editor and a technical program committee member for a number of journals and conferences such as DAC, DATE, ICCAD, ISLPED, ASPDAC, ASSCC etc. He is a senior member of the IEEE.



Hantao Huang (S'14) received the BS degree from Nanyang Technological University (NTU), Singapore in 2013. Since 2014, he is working towards the PhD degree from the School of Electrical and Electronic Engineering in Nanyang Technological University. His PhD is sponsored by MediaTek. His research interests are 3D ICs, Network-on-Chips, and low-power system design. He is a student member of the IEEE.



Dongjun Xu (S'14) received the BS degree from the Xi'an University of Technology, China in 2010. Later, he started working towards the PhD degree in Xi'an University of Technology, China. As a part of the PhD work, he started working as a research assistant in Nanyang Technological University, Singapore. His research interests are 3D ICs, multi-core and low-power designs. He is a student member of the IEEE.