Reinforcement Learning based Self-adaptive Voltage-swing Adjustment of 2.5D I/Os for Many-core Microprocessor and Memory Communication

Huang Hantao[1], Sai Manoj P. D.[1], Dongjun Xu[1,2], Hao Yu[1] and Zhigang Hao[3]
[1]School of Electrical and Electronic Engineering Nanyang Technological University, Singapore 639798
[2] Dept. of Electronic Engineering, Xi'an University of Technology, Xi'an, China 710048
[3] MediaTek Singapore Pte. Ltd.

ABSTRACT

A reinforcement learning based I/O management is developed for energy-efficient communication between many-core microprocessor and memory. Instead of transmitting data under a fixed large voltage-swing, an online reinforcement Q-learning algorithm is developed to perform a self-adaptive voltage-swing control of 2.5D through-silicon interposer (TSI) I/O circuits. Such a voltage-swing adjustment is formulated as a Markov decision process (MDP) problem solved by model-free reinforcement learning under constraints of both power budget and bit-error-rate (BER). Experimental results show that the adaptive 2.5D TSI I/Os designed in 65nm CMOS can achieve an average of $12.5mw$ I/O power, $4GHz$ bandwidth and $3.125pJ/bit$ energy efficiency for one channel under 10^{-6} BER, which has 18.89% power saving and 15.11% improvement of energy efficiency on average.

1. INTRODUCTION

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

However, all previous I/O circuit designs assume a constant and large output voltage-swing [11, 12]. For 2D wire-line communication, large output voltage-swing is required to compensate the channel loss and noise of the PCB trace. But a large output voltage-swing in communication with high data-rate can result in huge power consumption in communication. Meanwhile, bit-error-rate (BER) requirement of I/Os actually is not necessary to be low at all time as it depends on the workload specification. Therefore, the constant and large output voltage-swing of I/Os may be over designed with low utilization. Thereby, in order to have an energy-efficient I/O communication, one needs to develop a dynamic output voltage-swing scaling to adaptively adjust the output voltage-swing level under the dynamic BER constraint [13, 14].

This work is sponsored by Singapore MOE TIER-2 fund MOE2010-T2-2-037 (ARC 5/11) and A*STAR PSF funding 11201202015 from Singapore and JIP support from MediaTek Singapore.
ICCAD'14, November 03-06, 2014, San Jose, CA, USA.

On-line machine learning based power management has been recently practiced [15, 16, 17]. For example, Q-learning can be utilized to find an optimal action-selection policy from the set of states. However, Q-learning [18] is limited by the learning rate with slow convergence. A reinforcement Q-learning can improve the policy decision by using the prior knowledge of the system with a Markov decision process (MDP) [16]. In this paper, a reinforcement Q-learning algorithm is developed to adaptively adjust the output voltage-swing levels of 2.5D TSI I/Os. Instead of transmitting data under a fixed large voltage-swing, an on-line reinforcement Q-learning based management is applied to select the output voltage-swing at the transmitter. Based on the historical data, the voltage-swing adjustment is formulated as a MDP problem solved by model-free reinforcement learning under constraints of both power budget and BER. To accelerate the adjustment convergence, a prediction of BER and power as virtual experience is applied to reinforcement Q-learning algorithm. One corresponding 2.5D TSI I/O is designed in 65nm CMOS process for multi-level output voltage-swing with balanced power and BER. Experimental results show that the adaptive 2.5D TSI I/O circuit can achieve $12.5mW$ I/O power, $4GHz$ bandwidth and $3.125pJ/bit$ energy efficiency for one channel under 10^{-6} BER, which has 18.89% reduction of power and 15.11% improvement of energy efficiency on average when compared to the traditional I/O communication with constant output voltage-swing.

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

2. 2.5D TSI I/O COMMUNICATION

In this section, we will present logic-memory 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. Set of variables used in this paper are defined in Table 1.

2.1 Logic-memory Integration by 2.5D TSI

Printed circuit board (PCB) with backplane [2, 3, 4] containing sockets is the traditional 2D interconnection method between processors and memories as shown in Fig. 1(a)(i). It requires a long trace of T-line ($\geq 25cm$) 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 [2, 3]. The 2.5D TSI T-line [19, 20] 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. 1(a)(ii), the area overhead is also mitigated. A comparison of channel loss

Table 1: List of variables and their description

Notation	Definition
$S = \{s_1, ..., s_N\}$	Set of states
$A = \{a_1, ..., a_N\}$	Set of actions
$V = \{v_1, ..., v_N\}$	Set of voltage-swings
$P(s_i, a_i, s_{i+1})$	Transition probability from state s_i to s_{i+1} with action a_i
BER_i	Bit-error-rate at i^{th} output voltage-swing
Pw_i	Communication power at i^{th} output voltage-swing
$R(s_i, a_i, s_{i+1})$	Reward value for transition from state s_i to s_{i+1} with action a_i
γ	Discount rate
α, β	Weighted parameters for reward function
lr	Learning rate
$Q(s_i, a_i)$	Q-value from Q-learning
R_D	Resistance of driver
Z_{diff}	characteristic impedance of the T-line
$\mathbf{I}_t$	Tail current of CML driver
σ_v	Standard deviation of noise
N_{s_i}	Number of visits to state s_i

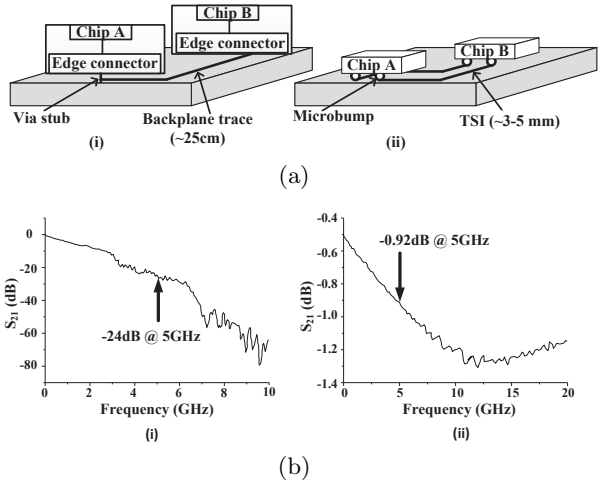

Figure 1: (a) Interconnect by: (i) Backplane trace (ii) TSI T-line; (b) Channel loss for: (i) Backplane trace; (ii) TSI T-line

between the 2D PCB trace and 2.5D TSI based T-lines is presented in Fig. 1(b). One can observe from Fig. 1(b)(i), at $5GHz$ clock frequency, the PCB trace with $25cm$ length has nearly $24dB$ channel loss; whereas at same frequency for 2.5D TSI with $10\mu m$ width, $3mm$ length has less than $1dB$ loss. Therefore, the 2.5D TSI T-line based integration is selected for the logic-memory integration. In addition, the 2.5D TSI based integration shows much better thermal dissipation capability when compared to the 3D TSV based integration.

Fig. 2(a) shows the system architecture for the logic-memory 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. 2(b) with tunable tail-current (based on output of I/O controller) to adaptively tune the output voltage-swing. By adjusting I/O output voltage swing, I/O communication power can be reduced with improved energy efficiency compared to the previous designs [11] that utilize 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 on-line management.

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

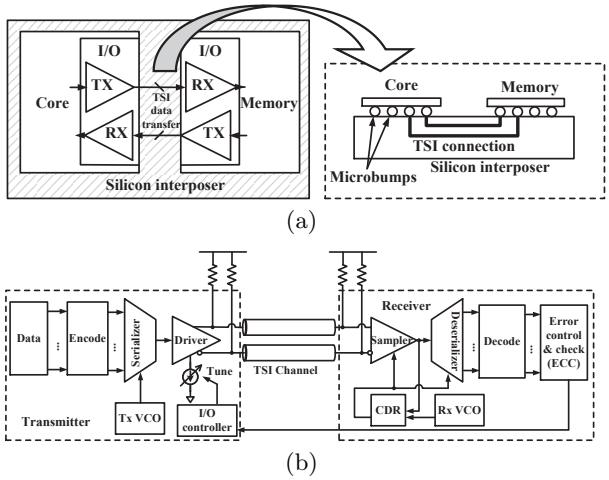

Figure 2: (a) Core-memory integration by 2.5D TSI I/O interconnect and its cross sectional view; (b) Adaptive tuning I/O based on error checking and correction

2.2 Problem Formulation

As previously discussed, the BER at the receiver increases with the decrease in the 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: the level of output voltage-swing at the transmitter can be tuned to achieve low power at the cost of BER based on the I/O communication channel characteristics.

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

where Pw_i and BER_i denotes the I/O communication power and BER under the i-th output voltage-swing level V_{s_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 of the output voltage-swing V_{s_i}, the I/O communication power Pw_i increases and BER BER_i decreases, vice-versa. As such, the output voltage-swing level V_{s_i} needs to be adaptively tuned for optimizing the I/O communication power and BER simultaneously. This problem can be modeled as a Markov decision process (MDP) and solved by the reinforcement learning based Q-learning discussed in next section.

3. Q-LEARNING BASED ADAPTIVE TUNING

In this section, we will first present the modeling of a Markov decision process (MDP) and the according reinforcement Q-learning algorithm for the adaptive tuning. System power model and BER model are then discussed as well.

3.1 Reinforcement Q-learning

The I/O management can be modeled by a Markov decision process (MDP) with variables defined as follows:

- State S: set of states indicating the value of system variable(s). We consider the voltage-swing level as the state.
- Action A: set of actions indicating the change of state. We consider the change of voltage-swing level as the action.
- State transition probability P (s_i,a_i,s_j): probability indicating the change from state s_i to state s_j due to the action a_i.

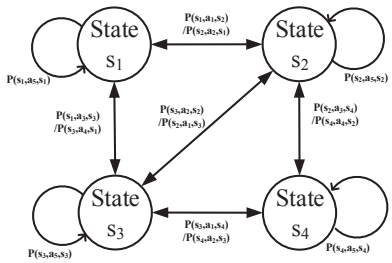

Figure 3: State diagram for reinforcement Q learning

- Reward $R(s_i, a_i, s_j)$: benefit of taking action a_i to change the state from s_i to s_j, which is dependent on historical BER and power Pw.
- Q-value $Q(s_i, a_i)$: set of values to measure the benefits of taking action a_i to state s_i. $Q'(s_i, a_i)$ is the updated value after taking action a_i.
- Policy: process of state change under sequence of action.

One example with 4-state is shown in Fig. 3. For state s_1, action a_1 can change its state to state s_2 with probability $P(s_1,\ a_1,\ s_2)$; For state s_2, action a_2 can change its state to state 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 $P(s_1,\ a_5,\ s_1)$. The state transition probability P is given by a decaying function. The probability under the decaying function is given by P = 1-1/(log(N_{s_i}+2) with N_{s_i} denoting the number of visits to state s_i. The probability based action will ensure the visit to all states at starting period. This will calculate Q-value to every available state accordingly. After this, Q-value based action will dominate and the optimal action with largest Q-value will be selected.

To find the optimal of MDP, reinforcement Q-learning algorithm can be utilized to evaluate the pair of state and action as the Q-value. The traditional Q-learning algorithm [18] converges to the optimal after unlimited iterations that may be too slow for convergence. The reinforcement Q-learning [15] can be utilized to find an the optimal with a faster convergence based on the predicted next state and the according transition probability with an initialized random action at first few states. In this paper, we utilize the reinforcement Q-learning to find the optimal for the modeled MDP to solve *Problem 1* formulated in Section 2.2, which is presented in Algorithm 1 as follows. The first phase is initialization to form a look-up-table with states (voltage-swing levels) and corresponding actions. In addition, the transition probability P for all the states is 1 and the reward is set to 0. This process of initialization is presented as $Init()$ of Algorithm 1.

Prediction of the next state (voltage-swing level) 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 Q-value is defined as the weighted sum of the reward and its past values by

$$Q'(s_i, a_i) = (1 - lr) * Q_(s_i, a_i) + lr * delta \tag{2a}$$

$$delta = (R(s_i, a_i, s_{i+1}) + \gamma * max(Q_(s_{i+1}, a_i) - Q(s_i, a_i))). \tag{2b}$$

where a_i and s_i belongs all available actions and states. $Q'(s_i, a_i)$ shows the updated Q-value after taking the action a_i to the next state s_{i+1}. The learning rate lr is initialized 0.5 and the higher means to more weight of estimate future Q value. γ is set as discount rate to discount the temporal difference and set to be 0.9 in the following experiments. In each iteration, the action is selected ether based on the transition probability or based on the maximum Q-value (or policy). If the transition probability is smaller than one threshold, the random action is selected; otherwise, the policy action with the maximum 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 that the action a_i with the maximum Q-value will be selected. The policy action with the maximum Q-value (2) can be described as below

$$a_i \leftarrow max(Q(s_{i+1}, a_i)). \tag{3}$$

Algorithm 1: Reinforcement Q learning algorithm

Input: Communication power Pw, BER feedback
Output: Output-voltage
function Init()
  $1 \rightarrow P(s_i, a_i, s_{i+1})$
  Reward $R(s_i, a_i, s_{i+1})= 0$
  $v_{predict} \rightarrow V_{s_i}$
  Selection()
end function

function Selection()
  for $k = 1 : n$
    $v_i \rightarrow s_i \in S$
    $Q'(s_i, a_i) \leftarrow (1 - lr) * Q(s_i, a_i) +$
    $lr * (R(s_i, a_i, s_{i+1}) + \gamma * \max(Q(s_{i+1}, a_i) - Q(s_i, a_i)))$
    If $P(s_i, a_i, s_{i+1}) > rand(0, 1)$
      $a_i \leftarrow rand(A)$
    else
      $a_i \leftarrow \max(Q(s_{i+1}, a_i))$
    end if
    Update()
  end for
end function

function Update()
  Reward: $R(s_i, a_i, s_{i+1}) = \alpha * \frac{max(Pw)}{Pw(k)} + \beta * \frac{max(BER)}{BER(k)}$
  Update Policy $(s_i,\ a_i)$, based on new Q
  $\forall\ s_i \in\ S$ {
    $a_i \leftarrow rand(A)$
    $Q'(s_i, a_i) = Q(s_i, a_i)$
    $P(s_i, a_i, s_{i+1}) = 1 - \frac{1}{log(N_{s_i}+2)}$
  }
end function

Action	*Reward*		*States*
	Power	*BER*	*Voltages*
a_1	$Power_1$	BER_1	V_1
⋮	⋮	⋮	⋮

The LUT is designed as above. Lastly, 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 $Power$ and will be updated as follows

$$R(s_i, a_i, s_{i+1}) = \alpha * \frac{max(Pw)}{Pw(k)} + \beta * \frac{max(BER)}{BER(k)} \tag{4}$$

where k denotes the iteration and max power and BER are predicted based on history input. At the end of $Update$, each state will be randomly visited and Q value (2) will be updated accordingly. The transition probability $P(s_i, a_i, s_{i+1})$ is also updated as N_{s_i} (the number of visits to state s_i) will increase after each iteration.

Note that with the prediction of states s_i as in function $Init()$ and $Update()$, the convergence to the optimal solution is accelerated [16]. This is done at the end of each round with the random action a_i to visit the state s_i.

3.2 System Models

The prediction of power and BER of I/O communication channel and their according models are discussed in this part. The first component of reward function is the I/O communication power. The system power model includes the I/O communication power of driver and the TSI T-line power. Both are the functions of the output voltage-swing V_{s_i}.

For the CML driver based TSI T-line [21] the I/O communication power is given by

$$Pw_i = V_{s_i} \cdot (\mathbf{I}_t + \frac{\eta * V_{dd} * s}{(R_D + Z_{diff})} * f). \quad (5)$$

where $\mathbf{I}_t$ 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 $\mathbf{I}_t$ at the current control-cycle is set by analog design and can be obtained from the measurement. Control-cycle is defined as the minimum time required for the state transition. Tail current for the next control-cycle can be predicted for the next control-cycle by auto-regression (AR)

$$\mathbf{I}_t(k+1) = \sum_{i=0}^{M-1} w_i I_t(k-i) + \xi. \quad (6)$$

Here $\mathbf{I}_t(k+1)$ is the predicted tail current at $k+1$-th control-cycle; w_i denotes the auto-regression coefficient obtained from autoregressive method; ξ is the prediction error and M represents the order of the AR prediction. Based on the predicted tail current, the output voltage-swing V_{s_i} can be calculated and the I/O communication power for the next control-cycle can be also calculated as $Pw_i(k+1)$.

The second component of the reward function is the BER of the I/O communication, which is feedback from the receiver. The BER depends on the output-voltage swing, external noise, channel noise etc, [22]. In a wire-line communication system, BER can be estimated with the dependence on the voltage-swing as

$$BER_i = \frac{1}{2} erfc(\frac{V_{s_i}}{\sqrt{2}\sigma_v}). \quad (7)$$

Here, the $erfc$ is complementary error function and σ_v is the standard deviation of the noise.

As such, the BER can be obtained from the ECC at the receiver by counting errors during a period. During the learning process, based on the BER obtained at the ECC under one output-voltage swing, σ_v is estimated from (7) for prediction.

4. ADAPTIVE 2.5D I/O CIRCUIT DESIGN

The overview of adaptive 2.5D I/O circuit design is described as follows. To achieve high bandwidth by 2.5D TSI single channel I/O, we employ 8:1 serializer in the transmitter (Tx) and 1:8 deserializer at the receiver (Rx). Eight digital D flip-flops followed by a 2:1 MUX is used to serialize data at Tx. Each of the Tx and Rx has a voltage-controlled-oscillator (VCO) to generate the required clock signal ($2GHz$) with 50Ω resistor. At the Rx, the serial bit stream is sampled and de-serialized. The received data is re-synchronized by the recovered clock from the clock-data recovery (CDR) block.

Compared to the traditional serial I/Os by the backplane PCB trace [2, 3, 4], the 2.5D TSI I/Os do not need the complex equalizer circuits at the receiver due to the small signal loss in the short TSI T-line channel. A sampler at the front-end of receiver is employed to convert the current-mode signals into digital levels. A delay-locked loop (DLL) based clock-data recovery (CDR) at receiver is implemented to de-skew the sampling clocks.

To protect transmitted data from channel loss and noise, data is encoded and transmitted which can be corrected at receiver. Here, we encode the data by Hamming code and corresponding parity bits are transmitted. As shown in Fig. 4, 32-bit parallel data ($D-32$) is initially stored in the output FIFO of the transmitter. For error detection and correcting purpose, 7 parity bits are generated by AND and XOR gates with MSB bit as 0. As such, the total encoded data to be transmitted will be 40-bit for every 32-bit data.

At the receiver, the first 32-bit of data is stored in the input FIFO ($D-32$ bits); and 7-bit of the last 8-bit (parity) is utilized for error checking and correction (ECC). The received data is checked through parity checker and left-shifter is used to correct bit error. As such, BER can be calculated by the number of error bits detected when compared to the total number of transmitted data bits over one period. The detected power and BER are provided as input to the I/O management controller guided by the reinforcement Q-learning algorithm. The resulting control bits are used to regulate the DAC currents at the tail of CML buffer.

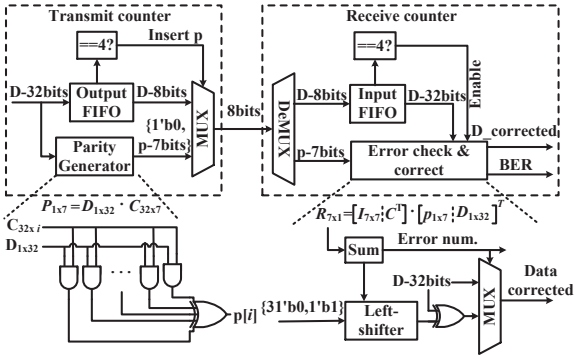

Figure 4: Encoding and decoding at transmitter and receiver

Table 2: System settings for logic-memory integration with TSI I/O

Item	Description	Value	Size
Core	Technology node	$65nm$	$0.3mm^2$
	Frequency	$500MHz$	
	Power	$15mW$	
I/O controller	Voltage swing	$0.1V$, $0.15V$, $0.2V$, $0.3V$	$0.03mm^2$
	Driving current	$2mA$, $3mA$, $4mA$, $5mA$	
	Action unit	10mV	
	Number of levels	4	
	Switching time	$0.4ns$	
TSI	Length	$3mm$	$3mm^2$
	Inductance	$300pH$	
	Resistance	5Ω	
	Capacitance	$60fF$	
Memory	SRAM	16 KB	$0.2mm^2$
	Power	$6mW$	

5. SIMULATION RESULTS

5.1 Experiment Setup

The reinforcement Q-learning algorithm for the adaptive 2.5D TSI I/O control is verified in Cadence Virtuoso (Ultrasim-Verilog) and Matlab. The flowchart in Fig. 5 reviews the overall implementation flow. The geometry and technology are presented in Table 2. An 8-core MIPS microprocessor with 8-bank of SRAM memory is designed with GF $65nm$ CMOS. The 2.5D TSI T-line is of length $3mm$ and $10\mu m$ width, driven by the CML buffer. The power traces are measured from Cadence Virtuoso and control cycle is set as $1ns$, larger than the switching time of I/O controller. The I/O management controller is based on the reinforcement Q-learning output to balance the I/O communication power and BER at receiver. The look-up-table (LUT) is designed with I/O communication power and BER are stored for adaptively tuning the voltage-swing. The LUT is set up as follows: ($100mV$, $6.27E-2mW$, $9.12E-2$), ($250mV$, $3.92E-1mW$, $4.28E-4$), ($350mV$, $7.68E-1mW$, $1.53E-6$), and ($450mV$, $1.27mW$, $9.81E-10$), for $filetransfer$ benchmark. The value is selected from the average of the output-voltage swing. The power of output signal will be tracked and power can be predicted from history power consumption. The average output-voltage swing therefore can be calculated and status can be set based on fixed output voltage step.

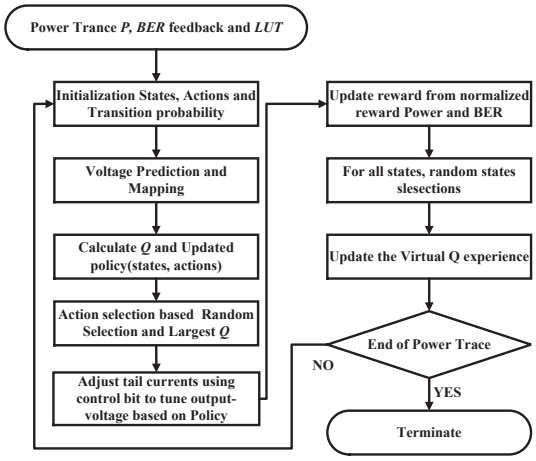

Figure 5: Flowchart of reinforcement Q-learning based self-adaptive voltage swing tuning

With the use of $65nm$ CMOS process, The overall I/O performance can provide $76mV - 190mV$ peak-to-peak signal swing with $4Gb/s$ bandwidth, and power consumption is only $12.5mW$ when implemented in 65nm CMOS process. Around $0.03mm^2$ area overhead incurs to realize the adaptive self-tuning of output voltage-swing and a latency of $100 - 200ps$ for the additional control circuits. For comparison, we have three implementations: 1) the I/O without management (normal); 2) the I/O control by the Q-learning without reinforcement (basic); and 3) the I/O control by the Q-learning with reinforcement (proposed). In the following, firstly, the adaptive tuning of the output voltage-swing with resulting eye-diagrams is presented; secondly, the reinforcement Q-learning results is presented with the prediction of the I/O communication power and the BER; and finally, the power saving and management time as well as energy efficiency are presented under different benchmarks of workloads.

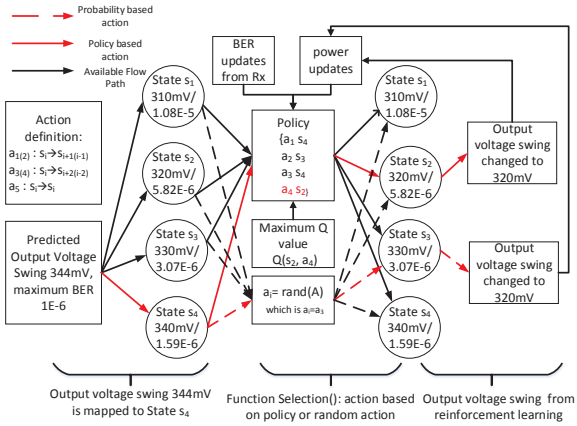

Figure 6: Example of one adaptive tuning procedure

5.2 Adaptive Tuning Results

The experiment of eye-diagram under the control of adaptive tuning is performed to verify the functionality of the adaptive I/O circuit tuning. Fig. 7 shows the current consumption under different levels of the output voltage-swing. The sources of error are introduced in three stages: stage 1 is to introduce 20% of clock jitter; stage 2 is to add additional 10% receiver offset; and stage 3 is to further add additional 10% power supply noise.

Moreover, an example of one adaptive tuning by the reinforcement Q-learning is shown in Fig. 6. Initially, the voltage-swing for the next control-cycle is predicted as $344mV$ and maximum BER is $1E-6$. The voltage-swing $344mV$ 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 1. As shown in Fig. 6 the 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 maximum Q-value will dominate and be considered as the optimal action. As Fig. 6 shows, action a_4 is eventually selected leading to the voltage level at state s_2, which is the smallest voltage-level with the BER requirement satisfied. Afterwards, the new voltage-swing is assigned based on the state s_2 as $320mV$. As such, the driver tail-current is tuned to have the output voltage-swing as $320mV$. Lastly, the Reward $R(s_4, a_4, s_2)$ will be updated based on the feedback of BER and power obtained at Rx.

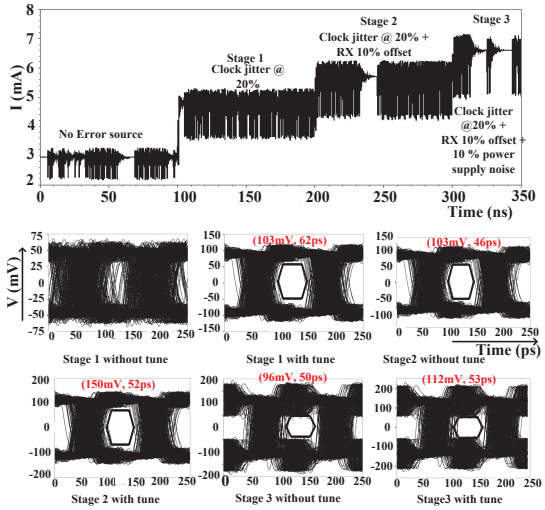

Figure 7: The eye-diagrams under adaptive current (or power) adjustment by output-voltage swing tuning

5.3 BER vs. Power Results

Fig. 8 shows the trade-off between BER (black circle) and the I/O communication power (blue rectangle). With the increase in output voltage-swing, BER decreases at the cost of the I/O communication power, which is validated through 4 benchmarks. 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 the I/O voltage-swing based on the reinforcement Q-learning can help to achieve an optimal trade-off of the communication power and BER for different benchmarks respectively as discussed below.

5.4 Benchmarking Results

The I/O communication power saving is verified for various SPEC benchmarks [23] in Table 3 by the self-adaptive tuning using: no Q-learning (normal); the Q-learning without reinforcement (basic); and the proposed Q-learning with reinforcement (proposed). It shows that the reinforcement Q-learning algorithm is more power efficient with faster convergence. On average, the measured power consumption is $19mW$ with an energy efficiency of $4.75pJ/bit$ for the I/O without the adaptive tuning, which is further reduced to $12.5mW$ by the reinforcement Q-learning. On average, the power saving of 18.89% and energy efficiency improvement of 15.11% are achieved by the adaptive tuning based on the reinforcement Q-learning; and only 12.95% of power and 14% of energy-efficiency improvement are achieved when using the basic Q-learning. What is more, on average, the reinforcement

Table 3: Power and run-time comparisons under various benchmarks

Benchmark	Communication power (mW)			Power saving		Run time(s)		
	Proposed	Basic	Normal	Proposed	Basic	Proposed	Basic	*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%

Q-learning takes $0.091s$ for convergence, whereas the basic Q-learning takes $0.115s$. The run time has also improved by an average of 20.83% as shown in Table 3

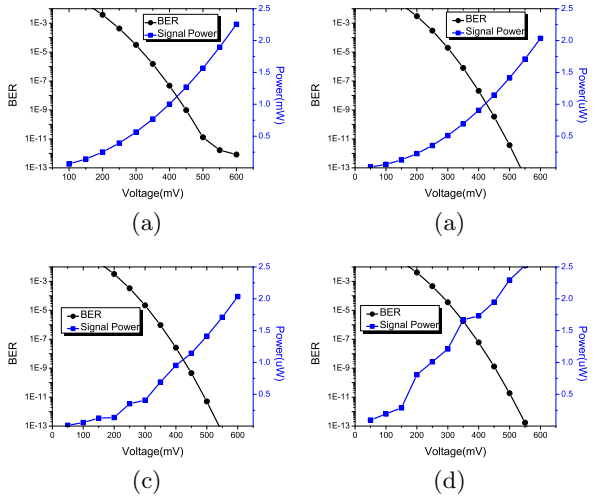

Figure 8: Trade-off between BER and power under various benchmarks: (a) *FFT* benchmark; (b) *Filetransfer* benchmark; (c) *bzip2* benchmark; (d) *gzip* benchmark;

6. CONCLUSION

In this paper, one reinforcement Q-learning based adaptive I/O management of multi-level output voltage-swings is developed for 2.5D TSI I/Os in logic-memory integration. Based on the predicted I/O communication power and BER from historical data, the voltage-swing adjustment is formulated as Markov decision process (MDP) problem and solved by model-free reinforcement learning under constraints of both power budget and bit-error-rate (BER). Experimental results have shown that the reinforcement Q-learning based adaptive I/O management for one 2.5D TSI I/Os designed in $65nm$ CMOS can achieve an average of $12.5mW$ I/O power, $4GHz$ bandwidth and $3.125pJ/bit$ energy efficiency for one channel under 10^{-6} BER, which has 18.89% reduction of power and 15.11% improvement of energy efficiency when compared to the I/Os with constant output-voltage swing.

7. REFERENCES

[1] S. Rusu and et.al., "Ivytown: A $22nm$ 15-core enterprise xeon® processor family," in *IEEE ISSCC*, 2014.

[2] J. F. Bulzacchelli and et.al., "A 10-Gb/s 5-tap DFE/4-tap FFE transceiver in 90-nm CMOS technology," *IEEE J. of Solid-State Circuits*, vol. 41, no. 12, pp. 2885–2900, Dec 2006.

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

[4] M. Pozzoni and et.al., "A multi-standard 1.5 to 10Gb/s latch-based 3-tap DFE receiver with a SSC tolerant CDR for serial backplane communication," *IEEE J. of Solid-State Circuits*, vol. 44, no. 4, pp. 1306–1315, Apr 2009.

[5] Y. Xie and et.al., "Design space exploration for 3D architectures," *ACM JETC*, vol. 2, no. 2, pp. 65–103, Apr 2006.

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

[7] J. Zhang and et.al., "Thermal stresses in 3D IC inter-wafer interconnects," *Elsevier J. of Microelectronic engineering*, vol. 82, no. 3, pp. 534–547, Dec 2005.

[8] J. R. Cubillo and et.al., "Interconnect design and analysis for through silicon interposers (TSIs)," in *IEEE 3DIC*, 2012.

[9] M.-J. Wang and et.al., "TSV technology for 2.5D IC solution," in *IEEE Electronic Components and Technology Conf.*, 2012.

[10] T. Ishii and et.al., "A 6.5-mW 5-Gbps on-chip differential transmission line interconnect with a low-latency asymmetric Tx in a 180nm CMOS technology," in *IEEE ASSCC*, 2006.

[11] J. Tschanz and N. Shanbhag, "A low-power, reconfigurable adaptive equalizer architecture," in *Asilomar Conf. on Signals, Systems, and Computers*, 1999.

[12] J.-S. Seo and et.al., "High-bandwidth and low-energy on-chip signaling with adaptive pre-emphasis in 90nm CMOS," in *IEEE ISSCC*, 2010.

[13] I. Foster, A. Roy, and V. Sander, "A quality of service architecture that combines resource reservation and application adaptation," in *IEEE Int. Workshop on Quality of Service*, 2000.

[14] V. Raghunathan, M. B. Srivastava, and R. K. Gupta, "A survey of techniques for energy efficient on-chip communication," in *ACM/IEEE DAC*, 2003.

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

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

[17] M. Triki and et.al., "Reinforcement learning-based dynamic power management of a battery-powered system supplying multiple active modes," in *European Modelling Symp.*, 2013.

[18] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning Journal*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[19] J. Wang and et.al., "High-speed and low-power 2.5D I/O circuits for memory-logic-integration by through-silicon interposer," in *IEEE. 3DIC*, 2013.

[20] S.-S. Wu and et.al., "A thermal resilient integration of many-core microprocessor and main memory by 2.5D TSI I/Os," in *ACM/IEEE. DATE*, 2014.

[21] I. Ndip and et.al., "High-frequency modeling of TSVs for 3-D chip integration and silicon interposers considering skin-effect, dielectric quasi-TEM and slow-wave modes," *IEEE Tran. on Components, Packaging and Manufacturing Technology*, vol. 1, no. 10, pp. 1627–1641, Oct 2011.

[22] M.-J. Lee and et.al., "Low-power area-efficient high-speed I/O circuit techniques," *IEEE J. of Solid-State Circuits*, vol. 35, no. 11, pp. 1591–1599, Nov 2000.

[23] "SPEC 2000 CPU benchmark suits," http://www.spec.org/cpu/.