

# Efficient implementation of FPGA based central pattern generator using distributed arithmetic

Xiaojun Li<sup>a)</sup> and Lin Li

*Robotics Lab, Faculty of Mechanical Engineering and Automotives,*

*South China University of Technology*

*510640 Wushan, Tianhe, Guangzhou, China*

*a) [tony.lee2009@hotmail.com](mailto:tony.lee2009@hotmail.com)*

**Abstract:** A scheme for efficient hardware implementation of central pattern generators (CPGs) on Field Programmable Gate Arrays (FPGAs) is proposed. A revised distributed-arithmetic (DA) algorithm is applied to the implementation to maximize the utilization of look up tables (LUTs) in FPGAs. The proposed scheme performances satisfactory experiment results which have correlation coefficients of 0.99 with simulation ones. In the mean time, it demonstrates 74% reduction in LUTs consumption, 75% in registers and 100% in embedded multipliers.

**Keywords:** central pattern generators (CPGs), Field Programmable Gate Arrays (FPGAs), look up tables (LUTs), distributed-arithmetic (DA)

**Classification:** Integrated circuits

## References

- [1] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, pp. 642–653, May 2008.
- [2] K. Nakada, T. Asai, and Y. Amemiya, "An analog CMOS central pattern generator for interlimb coordination in quadruped locomotion," *IEEE Trans. Neural Netw.*, vol. 14, pp. 1356–65, Sept. 2003.
- [3] R. J. Vogelstein, F. V. G. Tenore, L. Guevremont, R. Etienne-Cummings, and V. K. Mushahwar, "A Silicon Central Pattern Generator Controls Locomotion in Vivo," *IEEE Trans. Biomed Circuits Syst.*, vol. 2, pp. 212–22, Sept. 2008.
- [4] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, pp. 239–55, Dec. 2010.
- [5] E. Monmasson, "FPGAs in Industrial Control Applications," *IEEE Trans. Industrial Informatics*, 2011.
- [6] C. Torres-Huitzil and B. Girau, "Implementation of Central Pattern Generator in an FPGA-based embedded system," *Artificial Neural Networks-ICANN 2010*, 6353, pp. 276–285.
- [7] M. T. Tommiska, "Efficient digital implementation of the sigmoid function

for reprogrammable logic,” *IEEE P-Comput Dig T*, pp. 403–411, Nov. 2003.

## 1 Introduction

Central pattern generators (CPGs) are neural circuits that are capable of generating rhythmic outputs with coordinated patterns in the absence of sensory feedback or higher control commands. They have been extensively applied to robotics in developing robust control system, or for biology research [1]. To achieve a “mimic” approach, hardware implementation of CPGs is drawing attentions. Most works use analog devices [2, 3]. Analog circuits could perform nonlinear equations inherently and have reduced power consumption. Nevertheless, works based on Very large Scale Integrated Circuit (VLSI) need a relatively longer design circle and they are lack of flexibility and compatibility comparing to those on Field Programmable Logic Arrays (FPGAs). Although FPGAs have been intensively used in hardware implementation of Artificial Neuron Networks (ANNs) [4] and intelligent control systems [5], very few works of FPGA-based CPG implementations are reported [6]. On the other hand, the previous works use straight-forward implementation methods which does not make the best of FPGA resources.

In this paper, we firstly exam the important issues of implementing CPGs on FPGA. We consider the conflict between CPG’s need for frequent multiplication processing and the limited embedded multiplier resources on FPGA as a major hindrance for the implementation. Therefore, this work introduces the distributed-arithmetic algorithm to maximize the use of look up table (LUT) and to eliminate the reliance on embedded multipliers. The results of this new approach are compared with conventional ones.

The remainder of this paper is organized as follows. In section II, the mathematical description of the CPG is presented and its implementation issues are discussed. In section III, a DA-based implementation scheme of CPG on FPGA is proposed. In addition, the DA algorithm is revised for direct manipulation of fixed-point data. Section IV presents the experiment results of the FPGA based CPG, as well as the comparison of hardware resource consumptions. Conclusions are made in Section V.

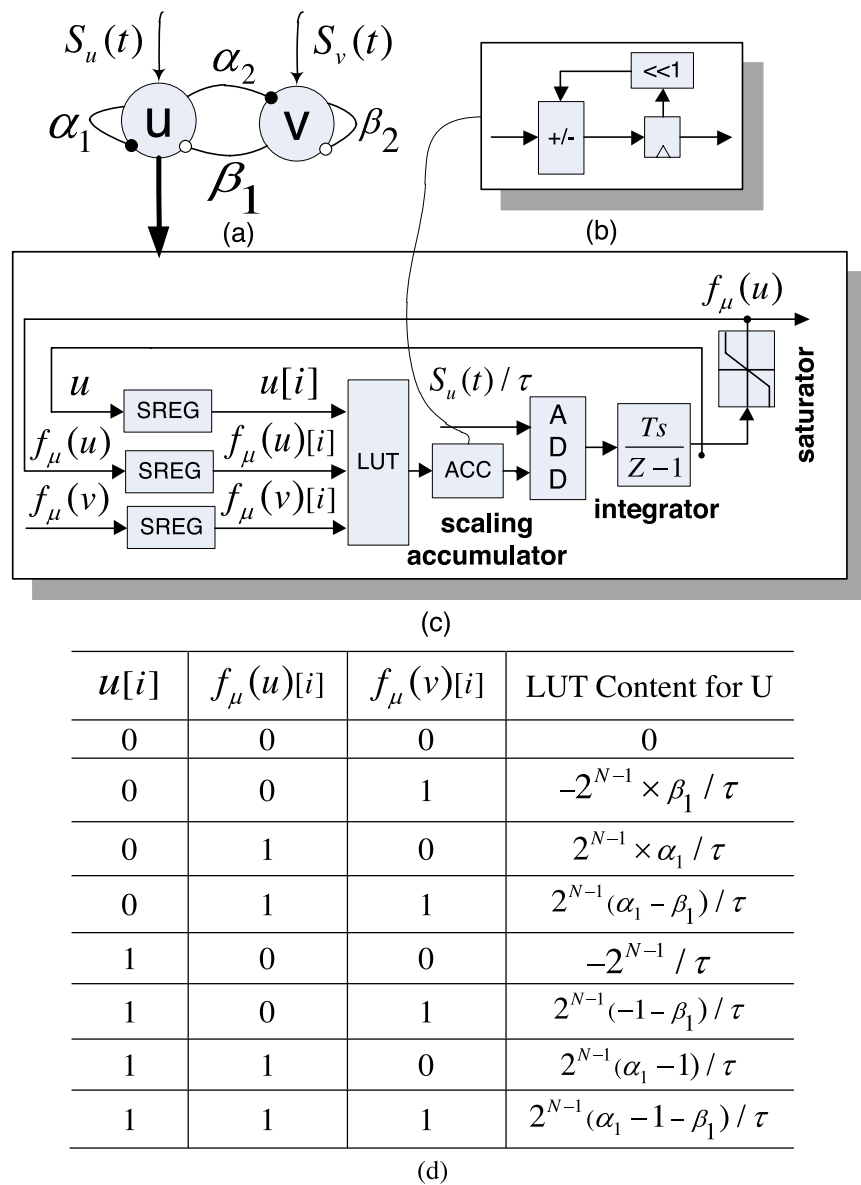
## 2 Arithmetic models and implementation issues of CPG

CPGs could be designed under several levels as biophysical, connectionist models and coupled oscillators [1]. The coupled nonlinear oscillators with reciprocal inhibitions are most frequently used. Well-known models of coupled oscillators such as Amari-Hopfield, Van De Pol and Matsuoka types share a similar principle. As an example, the dynamics of a Amari-Hopfield oscillator [2] is given by following equations:

$$\begin{aligned}\tau \dot{u} &= -u + \alpha_1 \times f_\mu(u) - \beta_1 \times f_\mu(v) + S_u(t) \\ \tau \dot{v} &= -v + \alpha_2 \times f_\mu(u) - \beta_2 \times f_\mu(v) + S_v(t)\end{aligned}\quad (1)$$

$$f_{\mu}(x) = \frac{1 + \tanh(\mu x)}{2}$$

Where  $u$  and  $v$  denote the neuron outputs,  $S_u(t)$  and  $S_v(t)$  stand for the external inputs.  $\alpha_1, \alpha_2, \beta_1, \beta_2, \mu$  are the control parameters and  $f_{\mu}(x)$  is the transfer function. See Fig. 1 (a) [2] for the architecture of an Amari-Hopfield type coupled oscillators. As the equations shows, implementation of the oscillator requires several basic operations: addition/subtraction, multiplication, integration/deviation and the transfer function (optionally). A single oscillator would require only several multipliers. However, a larger network would need more reciprocal inhibition paths thus makes the limited embedded multiplier resources a major hindrance for efficient implementation of CPGs. On the other hand, most FPGAs have large resources in Look-up tables (LUTs).



**Fig. 1.** DA based CPG (a) The neural oscillator schematic (b) Details of the scaling accumulator (c) Implementation based on distributed arithmetic (d) Contents of the LUT for neuron  $u$

With the hope of improving multiplication performance, we introduce the Distributed Arithmetic (DA) to CPG implementation which utilizes LUTs instead of multipliers. Another important issue about the implementation of transfer function has been discussed by Tommiska [7].

### 3 DA based CPG

Original DA algorithm demands the input to be fractional or integral. In FPGA systems, normally the input signal is transformed into fractional number through a binary point cast module and vice versa for the output. In this paper, the DA algorithm is revised so that no transform process is needed, consider a sum of products

$$Y = \sum_{k=0}^{K-1} A_k x_k \quad (2)$$

For a certain signed fixed-point  $X_k$ , it could be presented using a  $(N + M = Q)$ ,  $N$  for integer part and  $M$  for fractional part) bit binary form as  $b_{N-1}, \dots, b_1, b_0, b_{-1}, \dots, b_{-M}$

$$x_k = -b_{k,N-1} \times 2^{N-1} + \sum_{n=0}^{N-2} b_{k,n} 2^n + \sum_{m=-M}^{-1} b_{k,m} 2^m \quad (3)$$

By introducing  $C_k = A_k \times 2^{N-1}$ ,  $X_k$  could be shifted into a pure fractional form. Accordingly, the value of  $Y$  is given by:

$$\begin{aligned} Y &= \sum_{k=0}^{K-1} A_k \left( -b_{k,N-1} \times 2^{N-1} + \sum_{n=0}^{N-2} b_{k,n} 2^n + \sum_{m=-M}^{-1} b_{k,m} 2^m \right) \\ Y &= \sum_{k=0}^{K-1} C_k \times 2^{1-N} \left( -b_{k,N-1} \times 2^{N-1} + \sum_{n=0}^{N-2} b_{k,n} 2^n + \sum_{m=-M}^{-1} b_{k,m} 2^m \right) \\ &= - \sum_{k=0}^{K-1} C_k b_{k,0} + \sum_{i=1}^{Q-1} \left( \sum_{k=0}^{K-1} C_k b_{k,i} \right) 2^{-i} \\ &= \sum_{i=0}^{Q-1} L_i 2^{-i} \end{aligned} \quad (4)$$

$L_i$  is defined as the sum of products between  $C_k$  and every binary bit of  $X_k$ ,

$$L_i = \left\{ \sum_{k=0}^{K-1} C_k b_{k,i} \ (i \neq 0), - \sum_{k=0}^{K-1} C_k b_{k,0} \ (i = 0) \right\} \quad (5)$$

$L_i$  can be pre-computed and stored in a LUT, the input signal for which is  $X_k$ . The output from the LUT should be shifted and summed by a scaling accumulator.

Now we apply the proposed DA arithmetic to the implementation of Amari-Hopfield neuron oscillator. Assuming that all variants of the oscillator are in a  $(Q = N + M)$ -bit binary form, we will have the DA-based

neuron oscillator algorithm as follows:

$$\begin{aligned} u &= \int \left[ S_u(t) + \sum_{i=0}^{Q-1} 2^{N-1}(-u[i] + \alpha_1 f_\mu(u)[i] - \beta_1 f_\mu(v)[i]) \cdot 2^{-i} \right] / \tau dt \\ v &= \int \left[ S_v(t) + \sum_{i=0}^{Q-1} 2^{N-1}(-v[i] + \alpha_2 f_\mu(v)[i] - \beta_2 f_\mu(u)[i]) \cdot 2^{-i} \right] / \tau dt \quad (6) \\ f_\mu(x) &= \frac{|x+1| - |x-1|}{2} \end{aligned}$$

The content of the LUT for neuron  $u$  is presented in Fig. 1 (d). Another LUT for neuron  $v$  takes a similar form. Notice that the transfer function is replaced by a saturation function without significantly reducing the quality and generality of the rhythmic patterns [6]. The DA-based Amari-Hopfield neuron architecture is showed in Fig. 1 (c). It consists of a three-input LUT, a scaling accumulator (detail of which is given in Fig. 1 (b)), a parallel adder, a stature block and an integrator (uses forward Euler method). The entire system is designed using DSP Builder offered by Altera (System Generator from Xilinx is an alternative option) under Matlab/Simulink with a modular approach. It can be tested by Matlab/Simulink simulation and incorporated into a soft core possessor as custom IP cores.

#### 4 Experiment

The proposed circuit is implemented on an Altera Cyclone II EP2C8Q208 chip with a maximum frequency of 50 MHz. We adopt a 20-bit fixed point data presentation form (with 8 bits for integer part). The neuron parameters are given by following:

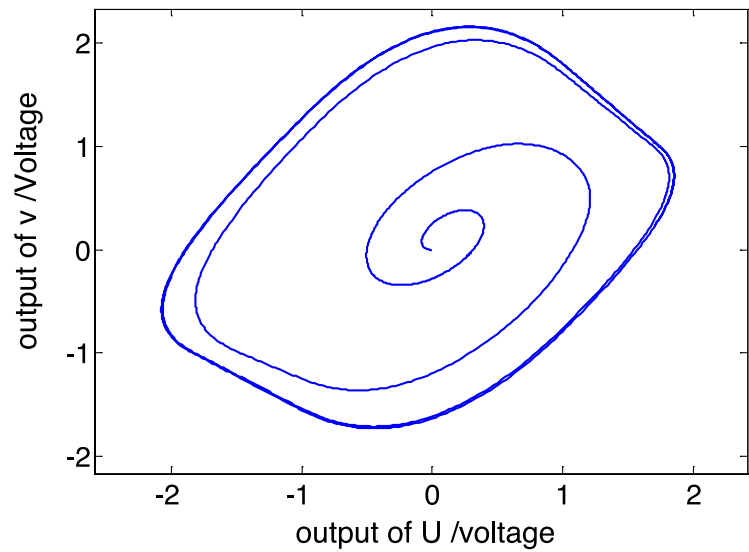
$$\tau = 1, \alpha_1 = 3, \alpha_2 = 3, \beta_1 = 3, \beta_2 = 0, \mu = 1, S_u = 0, S_v = -0.3 \quad (7)$$

The experiment data are obtained by Signal Tap II logical analyzer and analyzed in Matlab. See Fig. 2 (a) for the limit circle attractor. The experimental neuron outputs are compared with simulation ones, details of their slight differences are given in Fig. 2 (b) Based on our calculation, the correlation coefficients between experimental and simulation results is 0.99 for both  $u$  and  $v$ , indicating a high accuracy of the implementation.

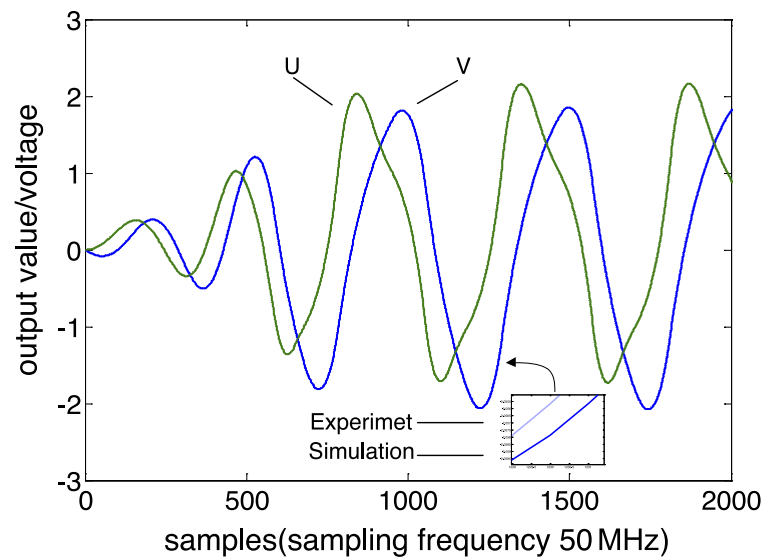
Table I presents the resource consumption of conventional and the DA-based implementation. The proposed method saves up to 74% of look up tables (LUTs), 75% of logic registers and 100% percent of embedded multipliers.

#### 5 Conclusion

The Distributed Arithmetic algorithm is introduced to the implementation of Central Pattern Generators on FPGAs. Compared with conventional approach based on multipliers, the proposed scheme achieves significantly savings in hardware resources. Furthermore, this method could be applied to the implementation of Artificial Neuron Network or other control systems.



(a)



(b)

**Fig. 2.** Experiment results: (a) phase plot of the limit circle attractor (b) output waves of the neural oscillator

**Table I.** Compare of resource consumptions

Resource Consumption	Multiplier-based CPG	DA-based CPG
LUTs	1549	401
Logic Registers	160	40
9-bit Embedded Multipliers	36	0

## Acknowledgments

---

Thanks to the 863 State High-Tech Development Plan (no. 2009AA043901-3) for its continuous financial support.