Journal of Engineering Sciences
Vol11,Issue2,FEB/2020
ISSN NO:0377-9254

IMPLEMENTATION MEMORY-EFFICIENT RECONFIGURABLE PROCESSOR FOR DEEP LEARNING ALGORITHM ON FPGA

T KRISHNAVENI*, M RANJITH**

PG SCHOLAR*, ASSISTANT PROFESSOR**

DEPARTMENT OF ECE, VAAGDEVI COLLEGE OF ENGINEERING, BOLLIKUNTA, WARANGAL

ABSTARCT:

Machine learning plays an important role in the field of artificial intelligence. Deep learning shows excellent ability in solving complex learning problems than that of machine learning. But, the size of the neural networks, and demand of practical applications possess significant challenge to construct a high-performance implementations of deep learning neural networks. Hence to improve the performance and maintain the low power cost, in this paper a scalable deep learning accelerator unit (DLAU), is designed for large-scale deep learning networks using field-programmable gate array (FPGA) as the hardware prototype. Dynamic power for streaming applications yielded by asynchronous dataflow designs by using clock gating techniques. Streaming applications constitute a very broad class of computing algorithms in areas such as signal processing, digital media coding, cryptography, video analytics, network routing and packet processing and many others. The paper introduces a set of techniques that, considering the dynamic streaming behavior of algorithms, can achieve power savings by selectively switching off parts of the circuits when they are temporarily inactive. The techniques being independent from the semantic of the application can be applied to any application and can be integrated into the synthesis stage of a high-level dataflow design flow The proposed DLAU uses carry save adder in the computation process and as further process of the project to increase the computation speed parallel self-timed adder is used in the design. Experimental results on the state-of-the-art Xilinx FPGA board demonstrate that the DLAU accelerator achieved more speed when compared to other devices like ASIC

INTRODUCTION:

Power dissipation is currently the major limitation of silicon computing devices. Reducing power has also other beneficial effects, it implies less stringent needs for cooling, improved longevity, longer autonomy in the case of battery operated devices and obviously, lower power costs. For all these reasons power also frequently affects the choice of the computing platform right at the outset. For example, Field-Programmable Gate Arrays (FPGAs) imply higher power dissipation per logic unit when compared to equivalent Application-Specific Integrated Circuit (ASIC), but often compare favorably to conventional processors used for the same functional tasks. For any silicon device, power dissipation can be partitioned into two components: a static and a dynamic component. Static power dissipation, also referred to as quiescent or standby power consumption, is the result of the leakage current of the transistors, also affected by the ambient temperature. By contrast, dynamic power dissipation is caused by transistors being switched and by losses of charges being moved along wires. Power dissipation increases linearly with frequency, due largely to the influence of parasitic capacitances. To counteract this effect, ASIC designers have employed clock gating (CG) techniques in the last twenty years [1], [2], [3]. Different strategies

for optimizing power consumption on ASICs and FPGAs are discussed in Section II. These papers describe the impact of a chosen technology for a given architecture, but do not describe how to reduce power at the design abstraction level. As a consequence, adding power controllers at the behavioral description design stage constitutes an additional task that has to be carried-out with care to avoid introducing undesired application behaviors and might reduce the portability of the code (i.e platform is changed during the development process). In addition, it is extremely difficult for HLSs approaches that are based on Imperative Model of Computations (MoCs) [4] to apply power optimization solutions that can be yielded by automatic tools starting from the (imperative) behavioral description. Conversely, dynamic dataflow [5], [6], [7] designs such as for instance the ones expressible using the formal RVC-CAL language possess interesting properties that can be exploited for reducing the power consumption without affecting, by construction, the behavioral characteristics of the application. In RVC-CAL, every actor can concurrently execute processing tasks, executions might be disabled by input blocking reads, and every communications among actors can occur only by means of order preserving lossless queues. As a consequence, an actor may be stopped for a certain period if its processing tasks are idle or its outputs queues (buffers) are full without impacting the overall throughput and semantical behavior of the design. In addition, to higher levels of dynamic behaviors that might be present in a given dataflow design, correspond higher levels of power reduction opportunities. This is not the case for synchronous dataflow designs that always consume and produce a fixed amount of data tokens. Thus, synchronous dataflow design always dissipate a constant amount of power compared to asynchronous dataflow. In this perspective the techniques that transform intrinsically dynamic algorithm into static versions such as the ones that are implemented by static dataflow MoC for deriving analytical guaranteed bounds or other analyzability purposes. In general this transformations are done by introducing dummy tokens guaranteeing constant rates. Thus, in terms of power optimization such approaches are inefficient.

RELATED WORK

Globally Asynchronous Locally Synchronous (GALS) based systems consist of several locally synchronous components which communicate with each other asynchronously. Works on GALS can be separated into three categories: partitioning, communication devices, and dedicated architectures. Dataflow design modeling, exploration, and optimization for GALS-based designs has been studied previously by several authors. Shen et al. [8] proposed a design and evaluation framework for modeling application-specific GALS-based dataflow architectures for cyclo-static applications, where system performance, e.g. throughput, is taken into account during optimization. Similarly, Wuu et al. [9] and Ghavami et al. [10] proposed a method for automatic synthesis of asynchronous digital systems. These two approaches were developed for fine-grained dataflow graphs, where actors are primitives or combinational functions. Related to our work, authors in [11] proposed a multiple clock, domain-design methodology for reducing the power consumption of dataflow programs. Their design objective was to optimize the mapping of an application while still meeting design performance requirements. This optimization was achieved by assigning each clock domain an optimized clock frequency to reduce power consumption.

CLOCK-GATING STRATEGY

Current FPGA families support different clock gating strategies and each manufacturer provides its own IP for managing these different approaches. The methodology described here is based on using primitives specific to Xilinx

FPGA architectures. However, this methodology can be modified to support other FPGA vendors primitives. In the remainder of this section, it is briefly described how clock gating techniques are implemented on Xilinx FPGAs and how an automatic clock gating strategy within Xronos HLS is realized.

A. Profile guided buffer size

The execution of a dataflow program consists of a sequence of action firings. These firings can be correlated to each other in a graph-based representation using an approach called Execution Trace Graphing (ETG). The graph is an acyclic directed graph where each node represents an action firing, and a directed arc represents either a data or a control dependency between two different action firings. The effectiveness of analyzing a dataflow program using an ETG is demonstrated in [12]. Xronos provides profiling for each firing execution in clock cycles. This is achieved by retrieving the difference of DONE and GO signals for each action firing during RTL
simulation [13]. Timing information is added to the ETG for each firing and each dependency arises according to a corresponding time value, thus transforming the ETG into a weighted graph. A close-to-optimal buffer size configuration, in terms of execution throughput and buffer memory utilization, can be obtained through an iterative analysis of the algorithmic critical path evaluated using the weighted ETG. For a detailed description the interested reader can refer
to [14].

B. Coarse-grained clock gating strategy

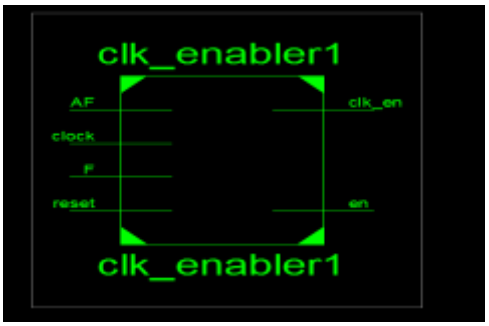
When the output buffer of any actor is full, the clock of this actor should be turned off as the actor is idle. This is because switching off its clock will not have an impact on design throughput. Even though RVC-CAL dataflow designs are used for the behavioral description, such clock gating strategy is more general and can be applied to systems that represent the execution of a process that communicates with asynchronous FIFO buffers. The queues should be asynchronous for lossless communication when an actor is clock gated and a design has differing input clock domains.

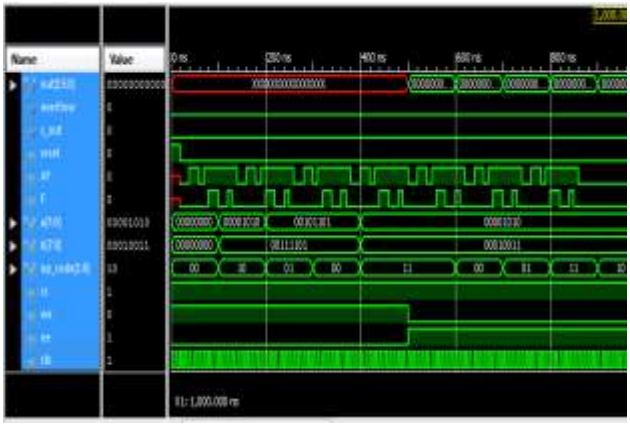
IMPLEMENTATION:

In this paper we boom the DLAU primarily based simply implementation of Memory Efficient Re-Configurable Deep Learning Accelerator Unit. Presently, we have were given prolonged this venture, with the resource of imposing the clock gating-based totally absolutely surely operation of DLAU. Which also can be called as Memory Efficient Re-Configurable Deep Learning Accelerator Unit the usage of FPGA.

Clock gating works with the useful resource of considering the permit situations joined with the registers and makes use of them to gate the clocks. A layout want to do not forget those allow situations to be able to employ and benefit from clock gating. This clock gating method additionally allows in appreciably reducing the die place and power utilization, as it expels brilliant numbers of multiplexers and replaces them with clock gating commonplace feel. This clock gating not unusual experience is substantially talking within the shape of "protected clock gating" (ICG) cells. In any case, the clock gating unique judgment will trade the clock tree form, because of the fact the clock gating not unusual experience will take a seat within the clock tree

SIMULATION RESULTS:





APPLICATIONS:

1. Multimedia

2. Digital signal processing

ADVANTAGES:

1. Area and power reduced

CONCLUSION

This paper presents a clock-gating methodology applied to dataflow designs that can be automatically included in the synthesis stage of a HLS design flow. The application of the power saving technique is independent from the sematic of application and does not need any additional step or effort during the ”design” of the application at the dataflow program level. The clock gating logic is generated during the synthesis stage together with the synthesis of the computational kernels connected via FIFO queues constituting the dataflow network. Conceivably, these techniques could be extended to other dataflow Methods of Computation. Experimental results are very encouraging: savings in power dissipation achieved with a slight increase in control logic without any reduction in throughput have been achieved. Unsurprisingly, clock gating is attractive in situations where the design is not used to its full capacity. In these circumstances clock gating is a simple, automatic, and effective way to recover power otherwise lost in ”idle” cycles. As a result, this technique is particularly interesting in applications with dynamically varying performance requirements, when designing to a particular performance point is impossible, and when power consumption is deemed costly. Further investigations into clock gating should consider more aggressive control logic, whereby control is given to each individual actor, allowing greater flexibility to actor inactivity. Furthermore, it will be necessary to develop tools that partition complex applications onto the limited number of clock domains for more efficient implementations. Lastly, additional considerations could be given to controlling clock speed and, possibly, voltage transitions.

REFERENCES

[1] Massoud Pedram, “Power minimization in ic design: Principles and applications,” ACM Trans. Des. Autom. Electron. Syst., vol. 1, no. 1, pp. 3–56, Jan. 1996.

[2] QingWu, M. Pedram, and XunweiWu, “Clock-gating and its application to low power design of sequential circuits,” Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, vol. 47, no. 3, pp. 415–420, Mar 2000.

[3] G.E. Tellez, A. Farrahi, and M. Sarrafzadeh, “Activity-driven clock design for low power circuits,” in Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on, Nov 1995, pp. 62–65.

[4] E. Lee and A. Sangiovanni-Vincentelli, “Comparing models of computation,” in

Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design. IEEE Computer Society, 1997, pp. 234–241.

[5] Gilles Kahn, "The Semantics of Simple Language for Parallel Programming," in IFIP Congress, 1974, pp. 471–475.

[6] Edward A. Lee and David G. Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," IEEE Trans. Comput., vol. 36, no. 1, pp. 24–35, 1987.

[7] E.A. Lee and T.M. Parks, "Dataflow process networks," Proceedings of the IEEE, vol. 83, no. 5, pp. 773 –801, may 1995.

[8] Syed Suhaib, Deepak Mathaikutty, and Sandeep Shukla, "Dataflow architectures for GALS," Electronic Notes in Theoretical Computer Science, vol. 200, no. 1, pp. 33–50, 2008.

[9] Tzyh-Yung Wuu and Sarma B. K. Vrudhula, "Synthesis of asynchronous systems from data flow specification," Research Report ISI/RR-93-366, University of Southern California, Information Sciences Institute, Dec 1993.

[10] Behnam Ghavami and Hossein Pedram, "High performance asynchronous design flow using a novel static performance analysis method," Comput. Electr. Eng., vol. 35, no. 6, pp. 920–941, Nov. 2009.

[11] S.C. Brunet, E. Bezati, C. Alberti, M. Mattavelli, E. Amaldi, and J.W. Janneck, "Partitioning and optimization of high level stream applications for multi clock domain architectures," in Signal Processing Systems (SiPS), 2013 IEEE Workshop on, Oct 2013, pp. 177–182.

[12] Simone Casale-Brunet, Analysis and optimization of dynamic dataflow programs, Ph.D. thesis, STI, Lausanne, 2015.

[13] Endri Bezati, High-level synthesis of dataflow programs for heterogeneous platforms, Ph.D. thesis, STI, Lausanne, 2015.

[14] S. Casale-Brunet, M. Mattavelli, and J.W. Janneck, "Buffer optimization based on critical path analysis of a dataflow program design," in Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, May 2013, pp. 1384–1387.

[15] Xilinx, Analysis of Power Savings from Intelligent Clock Gating, August 2012, XAPP790.

[16] "Open RVC-CAL Applications," 2014, http://github.com/orcc/orc-apps, accessed 25-February-2014].

[17] M. Canale, S. Casale-Brunet, E. Bezati, M. Mattavelli, and J. Janneck, "Dataflow programs analysis and optimization using model predictive control techniques," Journal of Signal Processing Systems, pp. 1–11, 2015.