Clock Gating Scheme of Streaming Applications Using De-Blocking Filter

[1]Chintha VenuVishwanath chinthavenu20@gmail.com[1]
[2]Mr.V.V.V.S Prasad vvvsp9@gmail.com[2]
[1]M.Tech (VLSI-SD), J.B Institute of Engineering and Technology
[2]Associate Professor J.B Institute of Engineering and Technology

Abstract: **This paper presents the reduction of 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, packet processing, etc. Clock gating is a power-saving feature in semiconductor microelectronics that enables switching off circuits. This paper introduces clock gating 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. Experimental results show that power reduction is achieved with no loss in data throughput.**

Index Terms—Clock-gating, dataflow, high-level synthesis.

I. 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:

1) A static and

2) 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, largely due to the influence of parasitic capacitances. To counteract this effect, ASIC designers have employed clock gating (CG) techniques in the last 20 years.

Different strategies for optimizing power consumption on ASICs and FPGAs.Clock gating is a power-saving feature in semiconductor microelectronics that enables switching off circuits. Many electronic devices use clock gating to turn off buses, controllers, bridges and parts of processors, to reduce dynamic power consumption.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 andmight reduce the portability of the code (i.e., platform is changed during the development process).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:

1) Partitioning;

2) Communication devices; and

3) Dedicated architectures.

Dataflow design modeling, exploration, and optimization for GALS-based designs has been studied previously by several authors.Dynamic dataflow 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.

II. LITERATURE REVIEW

High-level synthesis of dynamic dataflow programs on heterogeneous MPSoC platforms.

The growing complexity of digital signal processing applications make a compelling case the use of high-level design and synthesis methodologies for the implementation on programmable logic devices and embedded processors. Past research has shown that, for complex systems, raising the level of abstraction of design stages does not necessarily come at a penalty in terms of performance or resource requirements. Dataflow programs provide behavioral descriptions capable of expressing both sequential and parallel components of application algorithms and enable natural design abstractions, modularity, and portability. In this paper, an open source tool, implementing dataflow programs onto embedded heterogeneous platforms by means of high-level synthesis, software synthesis and interface synthesis is presented Experimental design results demonstrate the capability and the effectiveness of the tool for implementing a wide range of applications when combined with Vivado HLS.

Comparing models of computation

We give a denotational framework (a "meta model") within which certain properties of models of computation can be compared. It describes concurrent processes in general terms as sets of possible behaviors. A process is determinate if, given the constraints imposed by the inputs, there are exactly one or exactly zero behaviors. Compositions of processes are processes with behaviors in the intersection of the behaviors of the component processes. The interaction between processes is through signals, which are collections of events. Each event is a value-tag pair, where the tags can come from a partially ordered or totally ordered set. Timed models are where the set of tags is totally ordered. Synchronous events share the same tag, and synchronous signals contain events with the same set of tags. Synchronous processes have only synchronous signals as behaviors. Strict causality (in timed tag systems) and continuity (in untimed tag systems) ensure determinacy under certain technical conditions. The framework is used to compare certain essential features of various models of computation, including Kahn process networks, dataflow, sequential processes, concurrent sequential processes with rendezvous, Petri nets, and discrete-event systems.

Clock-gating and its application to low power design of sequential circuits

This paper models the clock behavior in a sequential circuit by a quaternary variable and uses this representation to propose and analyze two clock-gating techniques. It then uses the covering relationship between the triggering transition of the clock and the active cycles of various flip flops to generate a derived clock for each flip flop in the circuit. A technique for clock gating is also presented, which generates a derived clock synchronous with the master clock. Design examples using gated clocks are provided next. Experimental results show that these designs have ideal logic functionality with lower power dissipation compared to traditional designs.

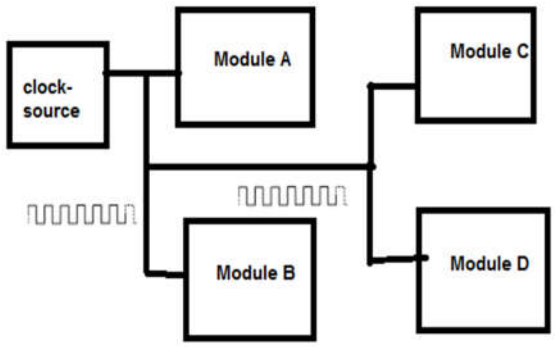
III. EXISTING SYSTEM

The system which process on continuous stream of bits is known as streaming application.Example Digital media coding, cryptography, video analytics, network routing, packet processing, etc.

Here, we are taking video analytics(MPEG-De-blocking Filter) as an example for streaming application.

Problem in Streaming applications

1) Due to continuous-direct clock input, dynamic power consumption of streaming application is more.
2) More heat dissipation due to power consumption.



streaming application with direct clocking(with out clock gating)

REMEDY:

We can achieve power savings by selectively switching off parts of the circuits when they are temporarily inactive.

Steps: 1) Initially we are giving clock to all modules(A,B,C,D and E) in the system.

2)Identify the module which is temporarily not in use(by the feed back signals F,AF).

3) switch of the circuit by disabling clock input to it.

4) It can be achieved by clock gating.

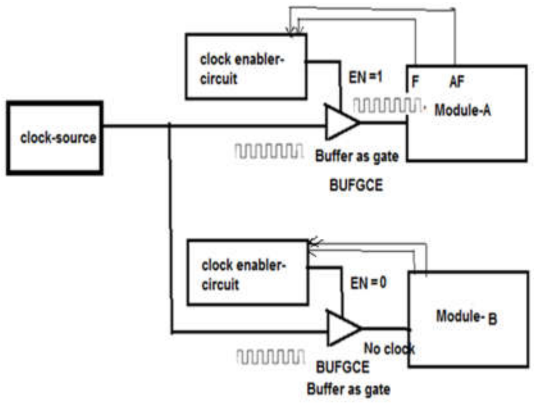
IV. PROPOSED SYSTEM

Here, we should not give input clock signal directly to the modules that are inside the application system. If we give clock through a gate or buffer it is known as clock gating.

We have to give it through the gates that are controlled by enable signal(EN).

- ➢ If EN=1 the gate allows the clock signal to a particular module.
- ➢ Else (EN=0) the gate stop the clock input for that particular module.

This enable signal (EN) has to be driven from clock enabler circuit.



streaming application with clock gating scheme

We can check the active state of the each module by its feedback signal(A,AF).

If we find that any module is in Inactive state(get into stop mode) we can stop clock input to the module by setting EN=0. Then we can switch off the module which is in temporary inactive state.

- ➢ Clock-gating avoids un-necessary clocking to the modules.
- ➢ Whenever, a module needs the clock then only it supplies otherwise it stops the clock.
- ➢ In this way clock-gating reduces the dynamic power consumption of the streaming application.

Implementation :

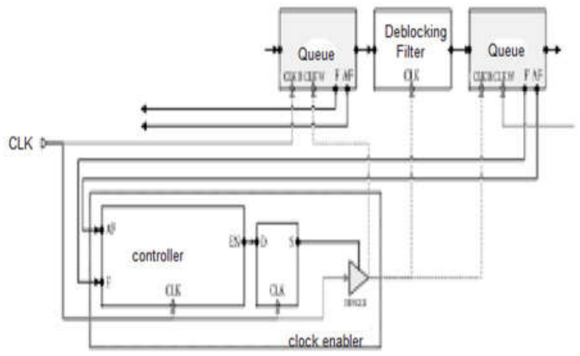


Fig1 . CG methodology applied for actor A.

Here in place of actor-A(module-A) we can take any kind of circuit. For example Let us take a De-blocking filter as actor-A. It is a part of MPEG encoder and filters image frames of video signal.

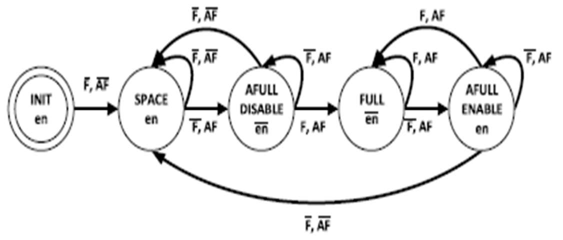
Operation of clock-gating technique:

- ➢ From figure, first queue stores the pixel data of the image and De-blocking filter takes it as input filter the data of the image.
- ➢ Then filtered output immediately stored onto the second queue. Later on data will be

stored on to the main memory or send for the broadcasting.

- ➢ When second queue is filled with FULL of data or ALMOST-FULL there is no need of filter (ACTOR-A) working, it should be off until second queue has some empty space to store data.
- ➢ To avoid overwriting of data in second queue or to reduce the dynamic power consumption we need to stop the clocking to Actor-A , second queue and first queue.
- ➢ To stop clock signal we need to give EN=0 to the clock buffer. EN has determined by clock enabler circuit which senses the F,AF signals from second queue.
- ➢ There are so many modules or ACTORS in MPEG encoder, we can switch off the modules which are temporarily in not-working mode by stopping the clock to them.
- ➢ Using clock gating technique(clock enabler circuit + clock buffer) we can reduce the dynamic power consumption of streaming application which has lot of modules.

State diagram of clock enabler circuit:



1) clock enabler circuit takes F,AF signals as inputs from second queue and gives EN as output.
2) when F=0, AF=0 for second queue that means it is not FULL($\overline{F}, \overline{AF}$) and it is having some space to store data. In this case we have to provide the clock to the circuit by enabling gate or buffer (en means en=1).

3) when F=1, AF=1 (F, AF) It means second queue is FULL and it has no space to store data. Clock enabler circuit gives output EN=0.($\overline{en}$ means en=0).
4)Depends on inputs(F,AF) clock enabler circuit gives output(en) .

F, AF
F, AF
FULL
en
F, AF

Internal operation of De-blocking Filter

It takes horizontal pixel values of video frame to H1,H2 filters(filter unit1,unit2) and vertical pixel values of video frame to V1,V2 filters(filter unit3,unit4).

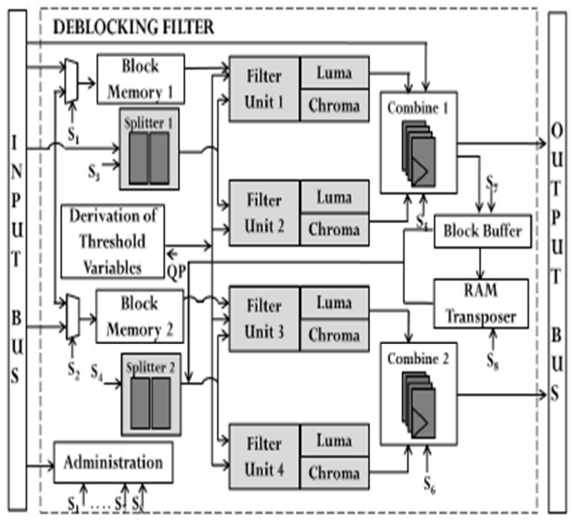


Fig.2 Parallel De-blocking Filter as an Actor unit

Fig. 2: The block diagram of the proposed parallel de-blocking filter architecture.

There are five main components in this architecture. The first component is the two block memories to store the data received from an external memory. Each memory is a 32*32 bit dual-port SRAM. Block Memory 1 is used to store the left neighbor blocks (E1-E8) for Filter Unit 1. Block Memory 2 is used to store the top neighbor blocks (F1-F8) for Filter Unit 3. The second component is Filter Units including two horizontal filters and two vertical filters. Filter Unit 1 and Filter Unit 2 are two horizontal filter modules, HF1 and HF2. Filter Unit 3 and Filter Unit 4 are two vertical filter modules, VF3 and VF4. In addition, the luminance and chrominance de-blocking filter architectures are separated and executed at the same time. The third component is Derivation of Threshold Variables generating the threshold variables and t_C from the input quantization parameter (QP). In addition, a RAM Transposer, which is used to transpose the data flow of 8x8 blocks from row to column in order to help reuse the data of the horizontal filter for the vertical filter without storing back to the memory.

Extension method

Purpose of DET FF

- It allows high data rates.
- We can increase the speed of data transfers
- in registers of microprocessor which has multi-bit FFs.
- SET FF transfer 3 bits per 3 clock pulses.
- But DET FF transfer 6 bits for the same 3 clock pulses.
- If we maintain same data rate we can use less number of DET-FFs in place of more number of SET-FFs.

DET FF :Information in processor is processed only at single clock edge, for example, positive edge while negative edge is not used to perform useful data processing. We propose to use the following technique (first time in the data center literature): double edge clock triggering where both the positive and negative clock edges are used to process data. The clock frequency, f, can drop to half which will save processor energy significantly due to equation $P=\Sigma\alpha*C*V2f$.

The key to implement the above methodology is to employ double edge triggered flip flop (DET FF) that can work on both edges of clock instead of using the conventional single edge triggered flip-flop (SET FF). DET flip-flop have not been reported to be used in any processors in servers in data center so far.

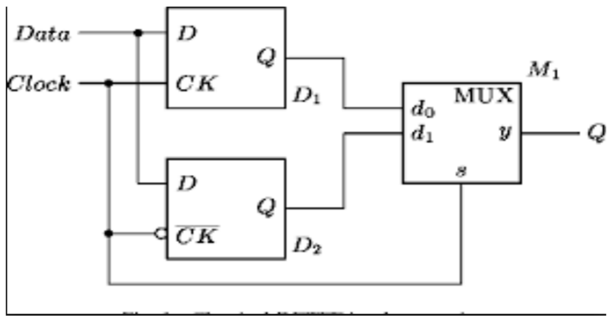


Fig.3:Double-Edge triggered Flip-Flop

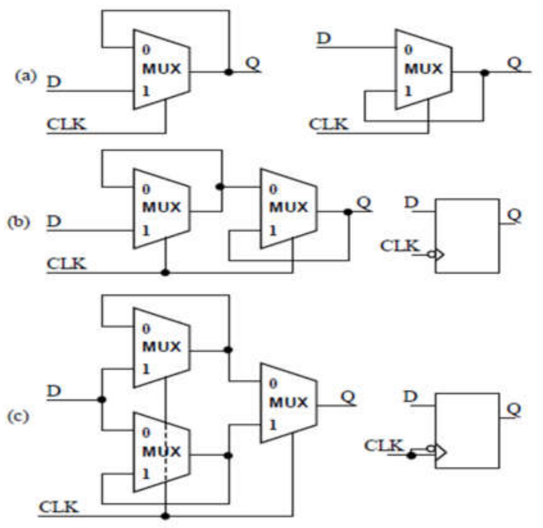


FIGURE 1 (a) Positive and negative level-sensitive latches; (b) SET flip-flop; (c) DET flip-flop

Because the flip-flop's state can change at both falling and rising edges of the clock, it is named "Double-Edge-Triggered Flip-Flop".

LOGIC STRUCTURE OF A DET FLIP-FLOP

Latch is the basic unit for composing a flip-flop. The levels of a clock, CLK, are used to drive the latch to either the storage state or the input state. If we use D, Q and Q' to express the input signal, present state and next state of a latch, the state equations for positive and negative level-sensitive latch can be expressed as: Q' = D·CLK +Q ·CLK (1) Q' = D·CLK +Q ·CLK (2) Equation (1) describes a latch which passes the input data when CLK = 1 and stores it when CLK = 0. Inversely, equation (2) describes a complementary latch, which receives input data at CLK = 0 and stores it at CLK = 1. The corresponding logic structures can be realized with a MUX.

The above discussion shows that the master latch does not receive the input data when CLK = 0. Obviously, if the input data has to be received at both clock levels, these two complementary latches should be connected in parallel rather than in series. Then we can obtain a "side-by-side flip-flop" as shown in fig 3. Since the flip-flop is required to be non transparent from input to output, the output terminal should always be connected to the latch which is in storage state. Therefore, the MUX framed by the dotted line is needed. Because the flip-flop's state can change at both falling and rising edges of the clock, it is named “Double-Edge-Triggered Flip-Flop”.

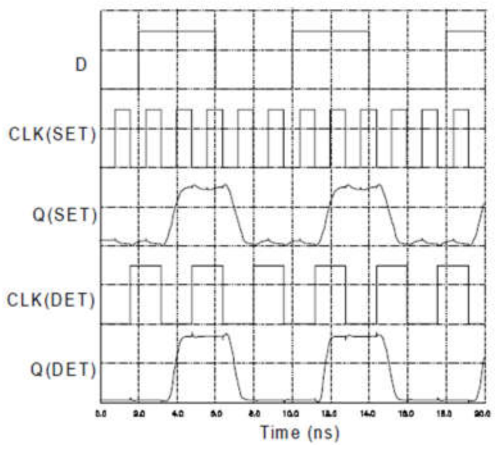


Fig. 5: Transient analysis waveforms

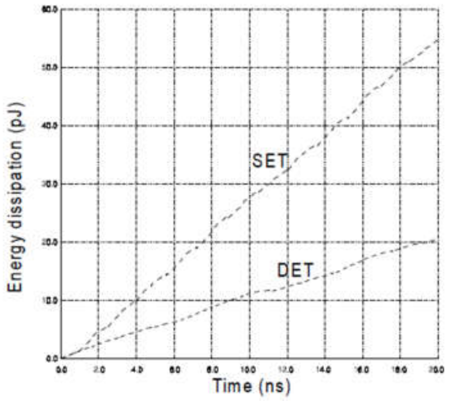


Fig. 6: Energy dissipation diagram

The proposed new circuit design of DET flip-flop has ideal logic function. It can receive, sample, and store the input data at the same data rate but at half the clock frequency in comparison with the traditional SET flip-flop.

The circuit power is reduced when using the DET flip-flops.

(2) By analyzing the simulation results of designed CMOS DET flip-flop and the traditional CMOS SET flip-flop the former has not only lower delay time, but also higher maximum data rate since its latch in storage state is closer to the output.

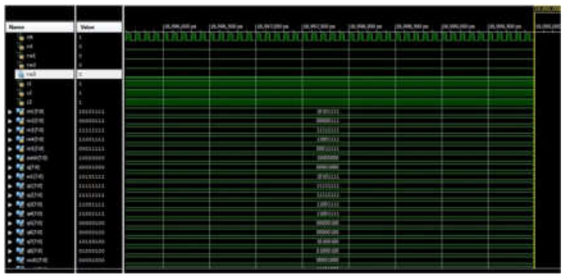
(3) By comparing the simulation results of the designed CMOS DET flip-flop with SET, it was found that our design has a simpler structure, less delay time and the highest maximum date rate.

In addition, we simulated an improved DET flip-flop and found that this DET flip-flop has the most prefer low power quality.

V. RESULTS

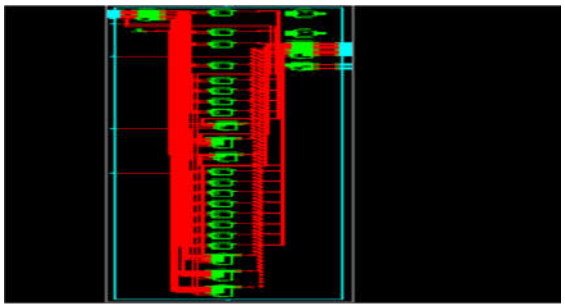
PROPOSED RESULTS:

Simulation

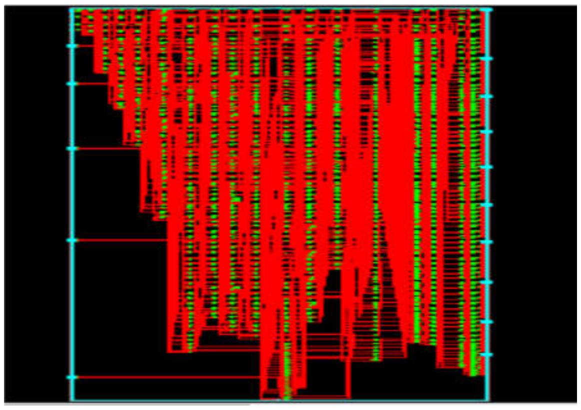


SYNTHESIS

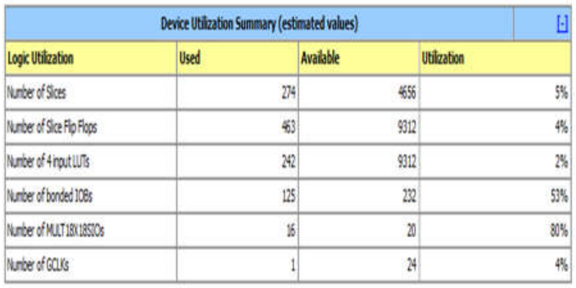
RTL schematic:



Technological schematic:

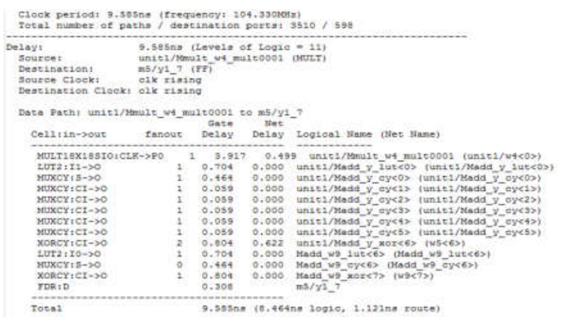


Design and summary



Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	274	4656	5%
Number of Slice Flip Flops	463	9312	4%
Number of 4 input LUTs	242	9312	2%
Number of bonded IOBs	125	232	53%
Number of MULT18X18SIOs	16	20	80%
Number of GCLKs	1	24	4%

Timing report

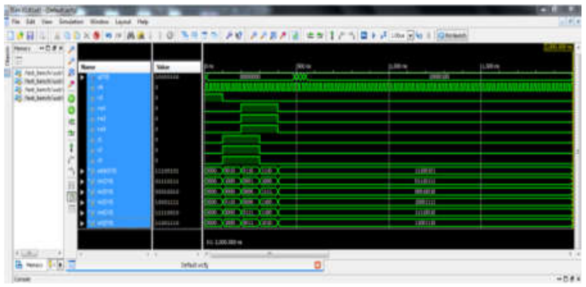


```
  Clock period: 9.585ns (frequency: 104.330MHz)
  Total number of paths / destination ports: 3510 / 598
-------------------------------------------------------------------------
Delay:               9.585ns (Levels of Logic = 11)
  Source:            unit1/Mmult_w4_mult0001 (MULT)
  Destination:       m5/y1_7 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: unit1/Mmult_w4_mult0001 to m5/y1_7
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     MULT18X18SIO:CLK->P0    1   3.917   0.499  unit1/Mmult_w4_mult0001 (unit1/w4<0>)
     LUT2:I1->O            1   0.704   0.000  unit1/Madd_y_lut<0> (unit1/Madd_y_lut<0>)
     MUXCY:S->O            1   0.464   0.000  unit1/Madd_y_cy<0> (unit1/Madd_y_cy<0>)
     MUXCY:CI->O           1   0.059   0.000  unit1/Madd_y_cy<1> (unit1/Madd_y_cy<1>)
     MUXCY:CI->O           1   0.059   0.000  unit1/Madd_y_cy<2> (unit1/Madd_y_cy<2>)
     MUXCY:CI->O           1   0.059   0.000  unit1/Madd_y_cy<3> (unit1/Madd_y_cy<3>)
     MUXCY:CI->O           1   0.059   0.000  unit1/Madd_y_cy<4> (unit1/Madd_y_cy<4>)
     MUXCY:CI->O           1   0.059   0.000  unit1/Madd_y_cy<5> (unit1/Madd_y_cy<5>)
     XORCY:CI->O           2   0.804   0.622  unit1/Madd_y_xor<6> (w5<6>)
     LUT2:I0->O            1   0.704   0.000  Madd_w9_lut<6> (Madd_w9_lut<6>)
     MUXCY:S->O            0   0.464   0.000  Madd_w9_cy<6> (Madd_w9_cy<6>)
     XORCY:CI->O           1   0.804   0.000  Madd_w9_xor<7> (w9<7>)
     FDR:D                     0.308          m5/y1_7
    ----------------------------------------
    Total                      9.585ns (8.464ns logic, 1.121ns route)
```

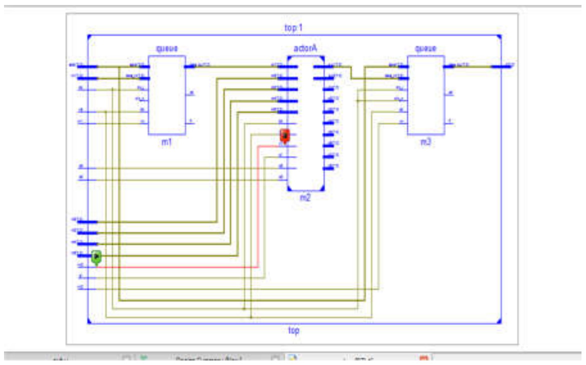
EXTENSION RESULTS:

Simulation:

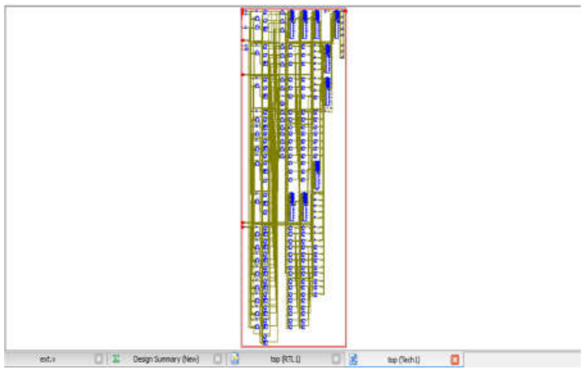


SYNTHESIS:

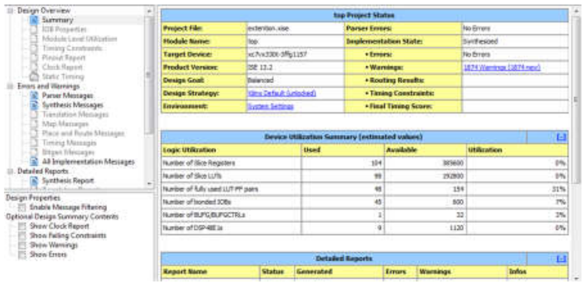
RTL schematic:



Technological schematic:



Design and summary:

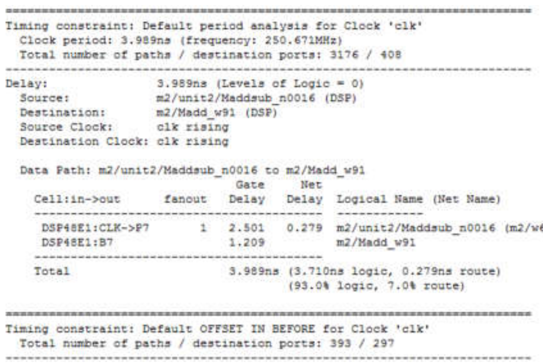


top Project Status			
Project File:	extention.xise	Parser Errors:	No Errors
Module Name:	top	Implementation State:	Synthesized
Target Device:	xc7vx330t-3ffg1157	• Errors:	No Errors
Product Version:	ISE 13.2	• Warnings:	1874 Warnings (1874 new)
Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	104	385600	0%
Number of Slice LUTs	99	192800	0%
Number of fully used LUT-FF pairs	48	159	31%
Number of bonded IOBs	45	600	7%
Number of BUFG/BUFGCTRLs	1	32	3%
Number of DSP48E1s	9	1120	0%

Report Name	Status	Generated	Errors	Warnings	Infos

Timing report



```
=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 3.989ns (frequency: 250.671MHz)
  Total number of paths / destination ports: 3176 / 408
-------------------------------------------------------------------------
Delay:               3.989ns (Levels of Logic = 0)
  Source:            m2/unit2/Maddsub_n0016 (DSP)
  Destination:       m2/Madd_w91 (DSP)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: m2/unit2/Maddsub_n0016 to m2/Madd_w91
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     DSP48E1:CLK->P7       1   2.501   0.279  m2/unit2/Maddsub_n0016 (m2/w6<7>)
     DSP48E1:B7                1.209          m2/Madd_w91
    ----------------------------------------
    Total                      3.989ns (3.710ns logic, 0.279ns route)
                                       (93.0% logic, 7.0% route)

=========================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'clk'
  Total number of paths / destination ports: 393 / 297
-------------------------------------------------------------------------
```

VI. CONCLUSION

This paper presents a CG methodology applied to dataflow designs that can be automatically included in the synthesis stage of an HLS design flow. The CG logic is generated during the synthesis stage together with the synthesis of the computational kernels connected via FIFO queues constituting the dataflow network. Experimental results show that savings in power dissipation achieved with a slight increase in control logic without any reduction in throughput have been achieved. Unsurprisingly, CG is attractive in situations where the design is not used to its full capacity. 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 CG should consider more aggressive control logic, whereby control is given to each individualactor, allowing greater flexibility to actor inactivity.

REFERENCES

[1] M. Pedram, "Power minimization in IC design: Principles and applications," ACM Trans. Design Autom. Electron. Syst., vol. 1, no. 1, pp. 3–56, Jan. 1996. [Online]. Available: http://doi.acm.org/10.1145/225871.225877

[2] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," IEEE Trans. Circuits Syst.I, Fundam. Theory Appl., 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 IEEE/ACM Int. Conf. Comput.-Aided Design Dig. Tech. Papers (ICCAD), San Jose, CA, USA, Nov. 1995, pp. 62–65.

[4] E. A. Lee and A. Sangiovanni-Vince

, USA, Nov. 1995, pp. 62–65.

[4] E. A. Lee and A. Sangiovanni-Vincentelli, "Comparing models of computation," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design, Austin, TX, USA, 1997, pp. 234–241.

[5] G. Kahn, "The semantics of simple language for parallel programming," in Proc. IFIP Congr., Stockholm, Sweden, 1974, pp. 471–475.

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

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

[8] S. Suhaib, D. Mathaikutty, and S. Shukla, "Dataflow architectures for GALS," Electron. Notes Theor. Comput.Sci., vol. 200, no. 1, pp. 33–50, 2008.

[9] T.-Y. Wuu and S. B. K. Vrudhula, "Synthesis of asynchronous systems from data flow specification," Inf. Sci. Inst., Univ. Southern California, Los Angeles, CA, USA, Tech. Rep. ISI/RR-93-366, Dec. 1993.

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

[11] S. C. Brunet et al., "Partitioning and optimization of high level stream applications for multi clock domain architectures," in Proc. IEEE Workshop Signal Process. Syst. (SiPS), Taipei, Taiwan, Oct. 2013, pp. 177–182.

[12] S. Casale-Brunet, "Analysis and optimization of dynamic dataflow programs," Ph.D. dissertation, STI Elect.Eng., STI, Lausanne, Switzerland, 2015.

[13] E. Bezati, "High-level synthesis of dataflow programs for heterogeneous platforms," Ph.D. dissertation, STI, Lausanne, Switzerland, 2015.

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

[15] Analysis of Power Savings from Intelligent Clock Gating, Xilinx, San Jose, CA, USA, Aug. 2012.

[16] (2014). Open RVC-CAL Applications. Accessed on Feb. 25, 2014.[Online]. Available: http://github.com/orcc/orc-apps

[17] M. Canale, S. Casale-Brunet, E. Bezati, M. Mattavelli, and J. Janneck, "Dataflow programs analysis and optimization using model predictive control techniques," J. Signal Process.Syst., vol. 84, no. 3, pp. 371–381, 2015.[Online]. Available: http://dx.doi.org/10.1007/s11265-015-1083-4

BIOGRAPHIES:

GUIDE DETAILS:
Mr. V.V.V.S PRASAD is working as an Associate professor in the Department of Electronics and communications in J.B Institute of Engineering and Technology.

STUDENT DETAILS:
ChinthaVenuVishwanath is pursuing his M.Tech VLSI-SD in J.B Institute of Engineering and Technology.

.

.