International Journal of Engineering Research and Technology.
ISSN 0974-3154 Volume 6, Number 3 (2013), pp. 285-291
© International Research Publication House
http://www.irphouse.com

Reconfigurable PLL for Digital System

R.N. Patil[1] and Mrs. S. Subbaraman[2]

[1]Asst. Professor in Electronics Engineering,
DKTES Textile & Engineering Institute, Ichalkaranji. INDIA
[2]Professor and Head, Department of Electronics & Telecommunication Engineering,
Annasaheb Dange College of Engineering and Technology, Sangli. INDIA
rnpatil@yahoo.com[1], shailasubbaraman@yahoo.co.in[2]

Abstract

Low power and high throughput digital systems are required in the number of applications such as Computer Graphics, Virtual Reality, System Control, Image Processing, and Digital Signal Processing etc. The various approaches suggested by researchers for power efficient high throughput architectures include pipelining, parallelism, retiming, folding-unfolding, using multiprocessor environment etc. An approach to dynamically derive different on chip clock signals to drive various sub blocks depending upon their time complexity is proposed in this paper. This approach is expected to reduce power dissipation besides increasing the throughput of the digital system. A phase locked loop circuit which is generally used to generate the frequencies which are integral multiple of input signal frequency is proposed to be implemented in a reconfigurable aspect to derive the required clock signals. In this paper we have proposed reconfigurable PLL, which generates another clock signal in run time, as per the requirement of the processing blocks in the system.

Keyword: power efficiency, high throughput, reconfigurable PLL

I. INTRODUCTION

In number of real-time applications, such as Computer Graphics, Virtual Reality, System Control, Image Processing, Digital Signal Processing etc. a sequence of data sets have to be processed by multiple functional units. If these functional units demand sequential processing then they can be implemented in pipeline architecture. On the other hand if these units are functionally mutually exclusive, then a parallel architecture can be used. Sometimes it may be necessary to implement a mixed architecture consisting of both pipeline and parallel architecture.

In the applications, involving complex computations or data intensive operations, the application can be decomposed into a series of subtasks. Generally in this approach, the subtasks are processed in pipeline architecture. Each subtask block is driven by the same clock input. Inputs are continuously fed into the first stage of the pipeline while the results emerge out of the last stage. A typical task consisting of subtasks is shown in Fig.1 (a).

If a task can be decomposed into subtasks, which are mutually exclusive, then they can be processed parallelly, like T_{11}, T_{12}, and T_{13} as shown in Fig.1 (b). The output of these subtasks may control the output of the next stage. The processing of the second level task can be carried only after completion of the subtask at level 1.

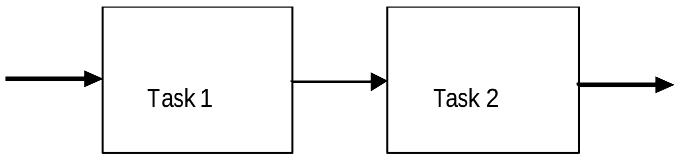


Fig 1 (a): The task is linearly partitioned into subtasks Task1, and Task2, which are pipelined

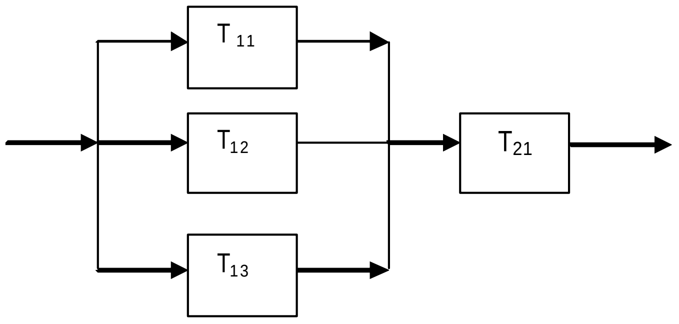


Fig 1 (b): The task is partitioned into subtasks Task $_{11,}$ Task $_{12,}$ and Task $_{13,}$ which are mutually exclusive and hence processed parallelly.

The complexity of each task could be different as well as the time taken to accomplish different tasks would be different. This would be due to implementation of slower algorithm or need to handle voluminous data, in one block while due to implementation of faster algorithm or need to handle small data, in another task block. The clock frequency in such case is generally determined by the time taken by the slowest task. If all data blocks in a system are driven by the same clock, then the different subtasks will get completed at different times depending upon their complexity. A fast block will have to wait even after completion of its task till a slowest block finishes its task. This is generally achieved through handshake signals. However this causes unnecessary clock transitions at the inputs of the already processed data blocks, resulting into unnecessary power dissipation. Secondly since the internal latch period has to be higher than the time taken by the slowest task, overall throughput gets hampered.

To overcome above problem, it is proposed through this paper, to generate system clock dynamically using reconfigurable PLL, to synchronize the clocks of all data processing blocks both in pipeline and parallel approach in such a way that all data blocks handling different tasks, whether slower or faster, will produce valid output almost simultaneously.

Generally Digital Systems designer prefers only one external clock. The on-chip clock signals of frequencies, which are integral multiple of system clock frequency, can be generated using phase lock loops. For a specific application, since the on-chips frequencies are fixed, the PLLs are hard wired to generate these frequencies. In a generic way, PLLs need to be programmed appropriately to generate clocks with frequencies, which are application specific. It is proposed to implement field reconfigurable PLL and follow a specific design implementation process to achieve this. The details of these concepts are presented in this paper.

II Background

Researchers have proposed, a Chip Multi-Processor (CMP) with smaller processor cores as a means to achieve high aggregate throughput and improved energy efficiency [1]. In the review it is also noticed that, it is possible to design a new digital phase-locked loop (PLL), utilizing the intrinsic synchronisability of electrical oscillators, on a field programmable gate array and providing dynamically reconfigurable clock networks, which were not realized by conventional PLL techniques [2]. Design of a clock generator circuit, to dynamically reconfigure the clock frequency of a synchronous digital system according to the changing needs of the application for a reconfigurable processor architectures to enhance the overall power efficiency of a systems is observed [3].

Use of multiple clocks for synchronizing purpose both to reduce power dissipation and increase throughput of a digital system is a novel approach. PLL is the obvious choice for achieving multiple clocks. The research work carried out on reconfigurable PLL and availability of reconfigurable FPGA by both Xilinx and Altera dictates the feasibility of implementing such architectures for generic applications.

III. PROPOSED SYSTEM

The block diagram of the proposed system to generate application specific on-chip clock signal is as shown in Fig. 2. Here CLK_1 is the system clock. The main task is partitioned into two tasks (T1) and (T2). The time taken by one of the tasks would be lower than the other one. CLK_2 is an on-chip signal generated by PLL which is N times faster than CLK_1. Where N is the count in the counter ensured in the feedback loop of PLL. Reconfigurable PLL demands that, this count is dynamically programmable, depending upon the time complexity of T1 and T2. A logic block capable of detecting the time difference between the completion of the tasks of blocks T1 and T2, initially fed with system clock CLK_1, would generate an appropriate count. To make the design more general in nature so as to offer ease of implementing complex digital designs, it is proposed to implement above concept and to divide the execution process in two steps as below.

- In the first step, the two task blocks are fed with system clock, as mentioned above, to generate the count *N*, which will be loaded in the PLL. An additional signal Clksel, will be generated to indicate which task is slower, using clock section logic.
- In the second step, T1 and T2 blocks are fed with appropriate clocks CLK_1 and CLK_2, through *Clock Selection Circuit.* A stream of input bits is fed to T1 block while stream of output is collected from T2 block.

This approach is expressed to

- Reduce the energy consumption by saving wastage of clock cycles, by dynamically controlling clock signal of a processor, which is handling time-consuming computations.
- Increase throughput of a digital system by speed matching of the processing units in multiprocessing environment, with the help of reconfigurable PLL.

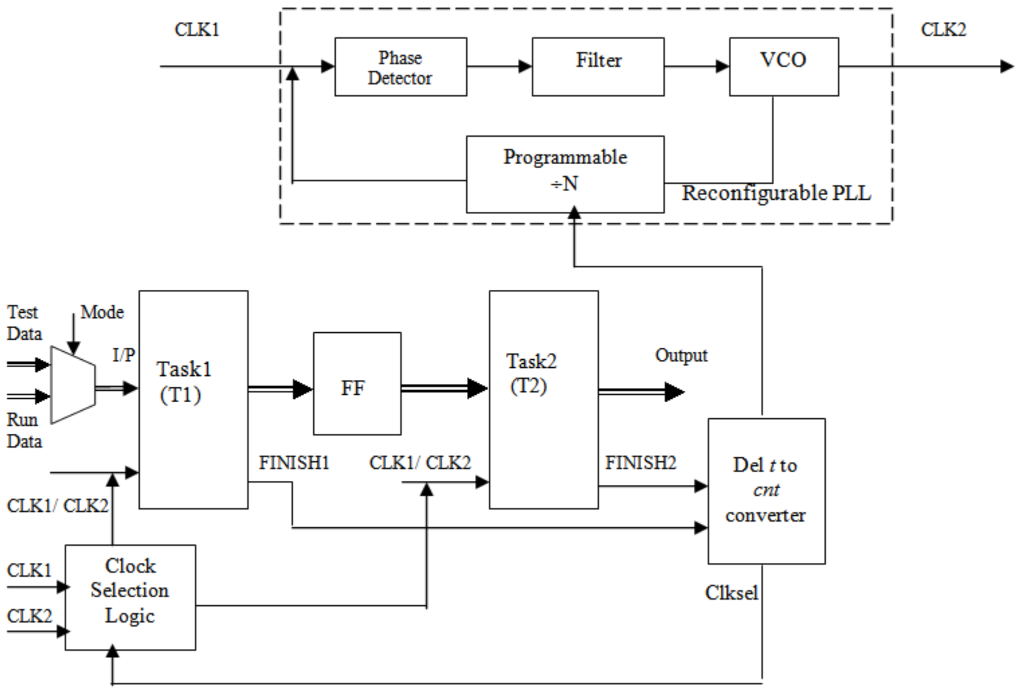


Fig. 2: Bock diagram of proposed system

The heart of the concept is reconfigurable PLL. The PLL does not utilize any passive components, and it uses digital design approaches, making it easily be integrated into digital systems.

The proposed reconfigurable PLL consists of five major functional blocks: a phase/frequency detector (PFD) a pulse forming circuit, an up-down counter, digital controlled oscillator (DCO), and a frequency divider in the feedback path of PLL.

- *Phase Frequency Detector*: PFD detects the phase and frequency difference between reference clock (CLK_1) and divided clock (F_{feed}) from the frequency divider in the feedback path. If the F_{ref} falls first, the signal UP is high level and the signal DN is low level. It indicates the CLK_1 leads the F_{feed} (or vice versa). If the signals Up and DN are both high levels, the two signals are both reset to low levels by a feedback reset signal. The sensed phase error by the PFD is the XOR-operation result of the signals UP and DN [6].
- *Pulse Forming Circuit*: By XOR operation of UP and DN signals from PFD, a clock is generated, to feed to the up-down counter.
- *Up-Down Counter*: Up-Down counter makes up counting or down counting, depending upon, UP and DN signals generated by PFD, and synchronized with clock generated by pulse forming circuit.
- *Digitally Controlled Oscillator (DCO)*: Voltage Controlled Oscillator (VCO) from conventional PLL is replaced by DCO in digital PLL. The DCO is designed with the help of, chain of delay cells and a multiplexer. The count from Up-Down counter is feed to the mux, which is going to select appropriate path from a chain of delay cell. The DCO generates clock frequencies in the range of 1.348 MHz to 21.74 MHz (while implemented on Spartan II FPGA).
- *Programmable Frequency Divider*: It divides the clock signal generated by DCO (CLK_2) by count N. As mentioned above Reconfigurable PLL demands that, this count is dynamically programmable, depending upon the time complexity of block T1 and T2. The frequency of the CLK_2, is governed by the standard equation for PLL i.e.

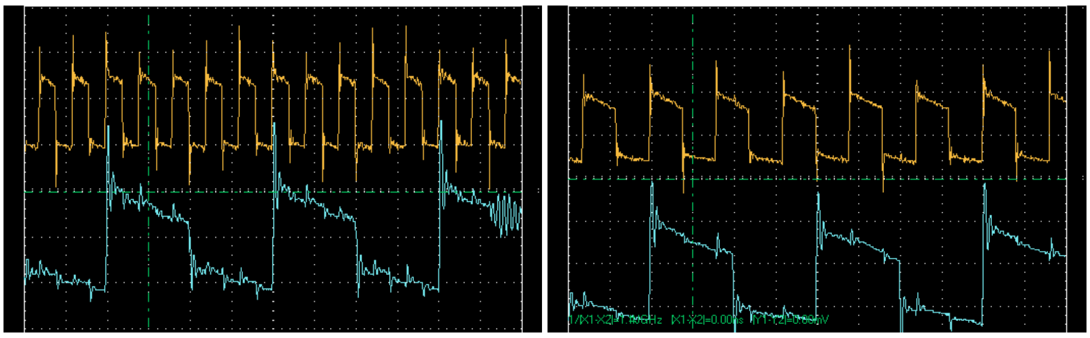
$$CLK_2 = CLK_1 * N \text{ ---------- } (1)$$

The measure characteristic of the frequency divider is that, it can divide the clock in the range of 0.5 also, as a result w.r.t. equation (1), we can get output clock CLK_2 in the multiple range of 0.5.

IV. RESULT

The proposed reconfigurable PLL has been implemented on Spartan II FPGA and observed the following results.

The observed frequency range of the reconfigurable PLL is 1.348 MHz to 21.74 MHz. The locking time is directly proportional to the initial frequency difference between the reference clock and the feedback signal, and inversely proportional to the loop bandwidth of the PLL.



(a) (b)

Fig 3 (a): Measured frequency when the Reconfigurable PLL locks for divide count *N* equals to (a) 5 and (b) 2.5

Advantage of the proposed reconfigurable PLL over on-chip FPGA, DCM, that the count to be provided to the DCM will be generic value and cannot be changed dynamically, where as in proposed reconfigurable PLL the count can be given dynamically as per the requirement of the application.

Table 1: Resources used for Reconfigurable PLL on SPARTAN II FPGA

Logic Utilization	Used
Total Number Slice Registers	291
Number used as Flip Flops	21
Number used as Latches	270
Total Number of 4 input LUTs	309
Number of bonded IOBs	21
IOB Flip Flops	2
IOB Latches	1
Number of GCLKs	2
Number of GCLKIOBs	1
Total equivalent gate count for design	3,522

V CONCLUSION

A reconfigurable PLL is proposed in this paper. The PLL can be used in the real-time digital systems, to reduce the energy consumption by saving wastage of clock cycles at multiple functional units, which are handling time consuming computations, by dynamically controlling clock signal of a processor. Due to the reconfigurable PLL it is also possible to increase the throughput of the system by speed matching of the processing units in multiprocessing environment.

VI. REFERENCES

[1] Hisashige Ando, Akira Asato, Motoyuki Kawaba, Hideki Okawara and William Walker: "A Case Study: Energy Efficient High Throughput Chip Multi-Processor Using Reduced-complexity Cores for

[2] Transaction Processing Workload", *IPSJ Digital Courier*, Vol. 1, pp.204-215. (2005).

[3] Tanaka, H.; Hasegawa, A.; Haruyama, S "Reconfigurable phase-locked loops on FPGA utilizing intrinsic synchronisability." Electronics Letters Volume 37, Issue 2, 18 Jan 2001 Page(s): 77 – 78

[4] Secareanu, R.M.; Albonesi, D.; Friedman, E.G. "A dynamic reconfigurable clock generator" ASIC/SOC Conference, 2001. Proceedings. 14th Annual IEEE International Volume, Issue, 2001 Page(s): 330 – 333

[5] David Katz, Rick Gentile "Dynamic Power Management Optimizes Performance vs. Power in Embedded Applications of Blackfin^{TM} DSPs". Analog Dialogue (2002), Page(s): 36-04

[6] Xin Chen, Jun Yang, "A Fast Locking All-Digital Phase-Locked Loop via Feed-Forward Compensation Technique" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS 1.

[7] Duo Sheng, Ching-Che Chung, and Chen-Yi Lee "A Low-Power and Portable Spread Spectrum Clock Generator for SoC Applications" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS 1.

[8] Application Note - Implementing PLL Reconfiguration in Stratix II Devices http://www.altera.com

[9] http://www.xilinx.com.