Activity Trend Guided Efficient Approach to Peak Power Estimation Using Emulation

Gaurav Saharawat
Mentor Graphics
Gaurav_Saharawat@mentor.com

Saurabh Jain
Mentor Graphics
Saurabh_Jain@mentor.com

Madhur Bhatia
Mentor Graphics
Madhur_Bhatia@mentor.com

***Abstract*-With the advent of increasingly complex systems, the verification of SoC designs is becoming more challenging. Simulation and emulation are common technologies used to verify the functionality of a design-under-test (DUT) before tape-out, but with the expanding use of battery operated electronic devices; power has become a critical area for design verification. There are issues like excessive heating and supply voltage drops due to very high design activity at a particular point in time under special circumstances that can lead to incorrect hardware functionality or chip burnout. These types of power related issues are difficult to identify with traditional verification approaches creating the need to have a dedicated Power Verification and Estimation flow using the best technology available. Emulation is the most appropriate technology for uncovering real power bugs for long testbench run times and real SoC use modes. In a typical emulation based solution, the designer needs to dump waveform trace data for the complete duration of a testbench, and then the waveform data is fed as an input to power analysis tools to perform power computation of the SoC over the complete testbench time range and target functionality. This gives rise to the issue that it takes a very long time to generate such long waveforms, and the disk space requirements becomes gigantic for any meaningful test run. In addition, the power tool has to process a huge amount of waveform data, which adds more processing time before a designer can find out peak power consumption of the design. This practically renders the flow unusable over long, real testbenches, and designers tend to skip some tests and extrapolate the results obtained from smaller test runs. This leads to inaccuracies and can result in post-silicon respin issues. This situation is not unprecedented and requires an innovative solution, which this paper attempts to provide.**

Keywords-Emulation, Power Estimation, Peak Power

I. INTRODUCTION

A precise understanding of power peaks is necessary to ensure correct functioning of the design's physical blocks and memories. A sudden rise in power dissipation from the electronic circuit can cause a chip to stop functioning accurately and at times cause voltage drops and chip burnout. Hence it is important to track the power usage patterns and spot any unexpected high-activity regions to prevent such problems apart from average power analysis of the chip. It is possible, at times, that some zones where activity is low, but still beyond the expected threshholds for those zones, are revealed by this method. This allows the user to observe the activity trends for the entire run and spot areas for focused analysis. For example, in the low power mode portion of design, if any design activity is seen, it can be a scope for functional bug as well as for power optimization. In general, this paper is not limited to just hot spot analysis at high-activity regions, but allows detection and analysis of any unexpected activity regions. In that sense it also helps to analyze the overall power usage of the chip and allows engineers to find the areas of unexpected power zones and unexpected high activity zones and lets them zoom in to those areas to look for scopes for power optimizations.

Typically, the power consumed by a design consists of two types: Static Power and Dynamic Power. Static power is power consumed by design when there is no circuit activity and is caused by leakage current. Static energy leakage depends on the state of the design and the portion of the design that is in power ON mode. Powering down inactive portions of the design helps reduce static power dissipation. Further, with FINFET technology, static power has been

optimized up to a large extent. On the other hand, dynamic power is power consumed by the design when the circuit is switching and high-switching activity leads to unexpected high power consumption. Dynamic Power dominates the chip power dissipation and especially after FINFET technology adoption, its relative portion is higher. Several anecdotes have observed this split of dynamic versus static power components and invariably dynamic power comes out to be the primary focus. This paper focuses on detection, analysis and debug of such high dynamic power consumption regions in both time and design space.

Existing power estimation tools provides two main modes of operations: vector-less and vector-based. The vector-less mode is based on propagating static probabilities and toggle rates through the circuit. In this mode, the runtime is roughly independent of trace length and is usually small. The vector-less mode may produce significant estimation errors because the correlation between signals is not captured. It is based on a probabilistic analysis and is not fully accurate. Hence it is not a very effective way of estimating the power of digital circuits.

On the other hand, a vector-based power estimation flow offers accurate power estimation results but is very demanding in terms of run-time and disk resources, and only allows running a limited number of cycles (generally the performance is a few 100's cycles per CPU hour). For a processor running at about a Gigahertz speed, it takes only one second to produce one billion clock cycles worth of trace data for the full chip. Running simulation for such a long duration on a relatively large design or full SoC design is impractical. Due to these issues and time to market pressure, designers have had to rely on either power estimates generated through vector-less method or on extrapolating the power results obtained by running vector-based power estimation solution for a small testbench case. This leads to inaccuracies in power estimation results. These inaccuracies can have a major impact on the robustness of the way a specific block is powered through the power/ground grids, and can lead to chips being unusable due to power issues.

This paper presents a novel approach based on the emulation technology to address the issue. It uses the traditional strengths of hardware emulation: speed, capacity and access to all the signals of the designs, and the power of parallel processing to provide a flow that can be used to detect and analyze peak power issues at the early stages of SoC design without instrumenting any additional logic. There is other prior work [7] that tries to solve this problem by instrumenting logic in the design and hence paying higher capacity overhead in an emulator. However, this paper presents an approach that uses emulation trace data for all the switching states of the designs without any capacity overhead, and projects the high and low power zones. Emulators are known to provide faster verification over long and real testbenches, and they have the additional advantage of enabling the in-circuit verification on real targets in addition to allowing it to run on deep and long HDL/C/System C testbenches that could never be possible in other methods of producing simulation/waveform data.

A key part of this paper is to provide a user with a view of a specific coarseness (configurable) activity captured in emulation. This is to capture and compute the cumulative switching activity on all design registers, latches and memory ports at each time point. In essence, it captures all the design state changes. The rate a design changes its state is the key to its power requirements. We have observed that design state changes lead to accurate power trend or activity trend capture for the whole design. The plot of state change activity can help verification and power engineers see the trend of the design where it gets into high activity or high power modes and vice versa. Tracking the design state changes with pre-configured coarseness is orders of magnitude faster than the plotting of design power consumption at each and every clock cycles of interest. This approach brings the overall power estimation time down from weeks to hours without losing accuracy.

This approach consists of a two pass methodology. In the first pass, the designer runs emulation and captures the switching activity data of only the design registers at pre-determined, regular intervals. This is orders of magnitude faster and more disk efficient than normal waveform capture methods, and it also provides accurate power trends. After the first pass, the designer is provided with a graphical view of such register activity of the full design or the relevant design blocks. From this, a designer can spot and select time zones of high or unexpected activity.

In the second pass, the designer runs emulation to capture full coarse trace data and activity for the selected time zones, and the same is fed to the power tool in parallel to get exact power numbers for these highest or unexpected activity zones. The generation and processing of different time zone data is done in parallel and in overlap mode

with the capture of the rest of the data for best performance by using on the power of parallel programming. Switching information is extracted and supplied to power analysis tools live as the emulation run progresses eliminating the need for the large disks to store waveform data.

The rest of the paper is organized as follows: In Section II there is a description of the two-pass methodology used to determine peak power. There is a description of the first pass where an activity plot is generated and used to determine high activity regions of the testbench and how the activity plot can be used to perform RTL exploration by generating activity plots of sub-hierarchies. This is followed by a description of the second pass that is used for more detailed peak power analysis for selected time zones, and how data generation and computation has been parallelized using a dynamic read API. In Section III, the design used in the experiment is described and the results of the experiment are examined. The last section, Conclusion, describes what has been achieved and how it can be used to perform peak power estimation in a way that was not feasible before.

II. PEAK POWER DETECTION METHOD

The primary idea of this work is to detect high activity areas in a first quick pass over a long testbench sequence such as depicting real activity like OS boot or video playback etc. that otherwise is not feasible for a user to run and then run detailed and time consuming per cycle power estimation only in those areas. What is described is a push button approach where a user doesn't need to worry about making any instrumentation in the design to run the complete flow.

The use of the emulator is important because it is the only platform capable of running testbenches similar to the final real-life applications; in particular OS boot sequence and software drivers used on such a platform. This allows the production of realistic scenarios suitable for power analysis that cannot be achieved by RTL and/or gate-level simulators. A precise estimate of the maximum current drawn by the logic and the memories is mandatory to assess the robustness of the power supplies and avoid any chip burnouts.

A. First pass: Activity plot generation

In the first pass, a real life testbench is run on the DUT modeled on an emulator platform. In this run, trace data is captured from the emulator at pre-determined regular intervals. To achieve this, as the emulation is run, an internal trace system in the emulation box is turned on for a fixed small number of cycles (For e.g. 1ns time interval for a 1 Ghz Processor system) and then turned off for X number of cycles (configurable). This way the trace buffer inside the emulator box that has finite depth keeps on filling. As soon as it is filled, the testbench execution is stopped and the trace data is extracted from the trace buffers of the emulator and they are cleaned up. Once trace data is extracted, the testbench resumes running and software processing is kicked off in parallel on trace data extracted to generate toggle data for an already captured duration. A typical interval to pick the sample is 100, i.e. 1 cycle is captured after every 100 cycles. Hence this runs approximately 100X faster when compared to an emulation run where a trace for every cycle is captured, thus making it practical to run long testbench sequences depicting real user scenarios. The user can configure this interval based on design and testbench knowledge to both decrease and increase values.

As per observation, toggle activity of the state elements depicts the toggle activity of whole design so toggles for only the state/flop signals are computed instead of the whole design for generating an activity plot. Users can view the activity plot generated in a GUI window (Figure 1). Figure 1 shows an activity plot generated for a customer design with a real life testbench running on it. In this plot, three highest peaks are selected (enclosed in vertical red lines) and then a time zone file(tzf) is dumped for those three peaks selected as indicated in the blue box in figure 1.

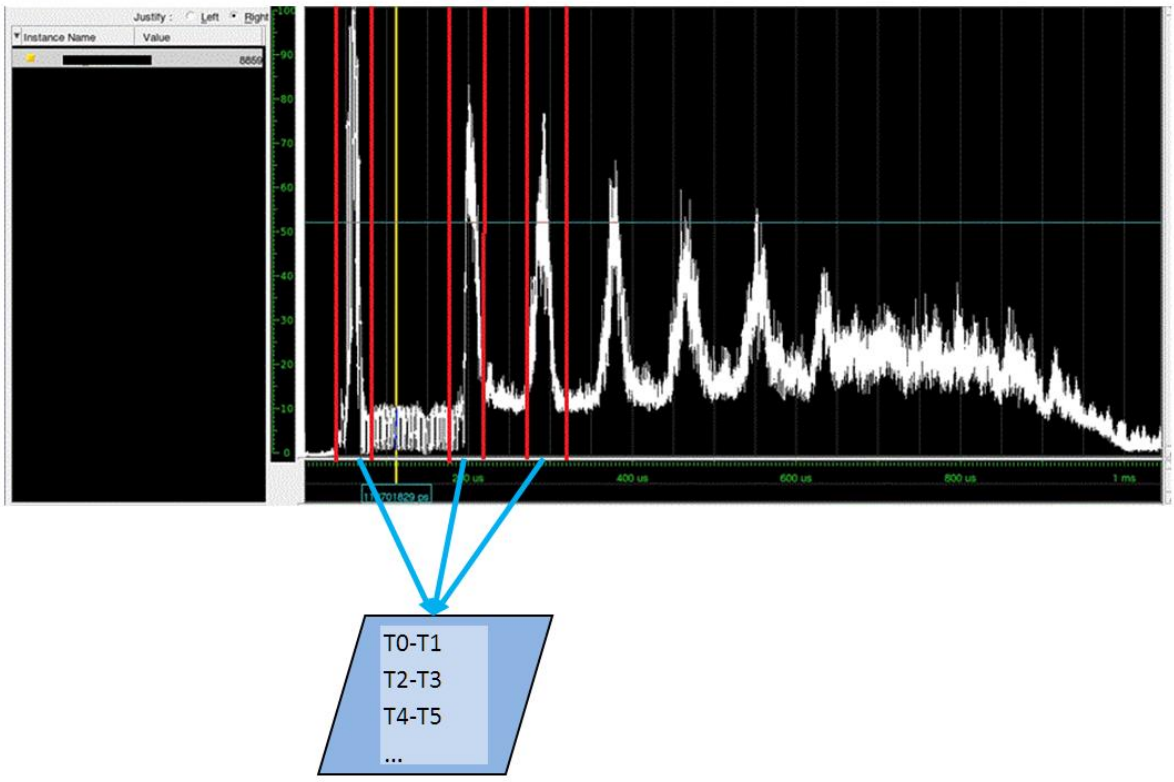

Figure 1. Time zone selection from activity plot.

Format of time zone file is as follows:
T0-T1
T2-T3
T4-T5
…
Each line in the time zone file represents a time zone where the user intends to capture the trace data. The first line (T0-T1) means that trace data needs to be captured from time point T1 to time point T2 and so on. This file can be edited by the user and can also be automated for capture with a graphical tool as illustrated above in the figure 1.

Exploration in design space using activity plot: The basic idea here is that the distribution of activity across various sub-blocks, IPs or recursive instances can be populated and shown on demand as selected by the user. This allows designers to know the distribution of activity in a design space, and hence provides them a very good method to spot unexpected space zones of high or low activity and to analyze the hot spot areas of the design. This paper primarily proposes a way to detect zones of high activity and fix them to have a more uniform or more expected power profile. However nothing in this approach prevents users from seeing the low or unexpected low activity zones. At times, low activity zones can also expose a bug around expected power down modules, which otherwise could be erroneously high during some real life scenario and hence drive a user to either try a different test where such zones also become active and add to overall design power and allow him to see and imagine the real peaks and lead him to the discovery of such scenarios. It is also possible to provide some ‘what-if’ computations by the tool on the fly by merging activity zones of virtually non-existing or non-overlapping zones of high activity.

In this mode a user can use the activity plot to find the instance(s)/hierarchy/IP contributing to the peak activity at any given point in time. To achieve the same, we have developed a way where a user can specify the list of instances/IPs/Hierarchies that he wants to see on separate activity plots, and based on user input, toggle the activity of state elements present in that particular hierarchy/IP/instance is computed and the activity plot of those instances can be viewed in GUI (Figure 2).

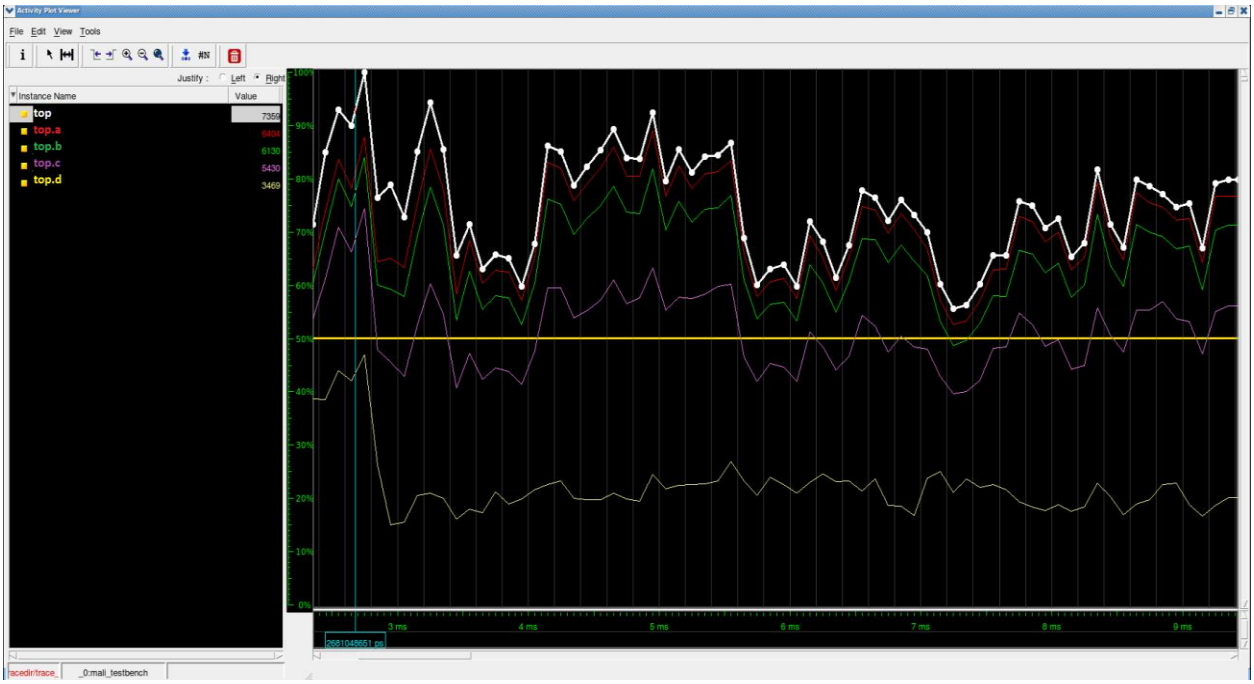

Figure 2: Activity plot of multiple sub-hierarchies.

From these individual activity plots a user can detect which particular instance is contributing to peak activity at a particular time point and can perform detailed analysis either through a 2nd pass or by further breaking down the activity plot of high activity instances into its sub-hierarchies to understand which sub-hierarchy is causing the high activity contribution by that instance.

B. Second pass

In the second pass run, the same testbench and DUT setup is used as was used in the first pass. In this run, a time zone file generated from the first pass is fed to the emulator and the tracing system is turned on/off during emulation as per the time zones specified in time zone file and trace data for every clock is generated only for the time zones specified. The trace data captured can either be converted to a vector file, such as a Fast Signal Database (FSDB) or a Value Change Dump (VCD) or a power tool's native format file and fed to the power tool for peak power analysis. Typically the "fsdb" or "vcd" dumping approach is both runtime and disk intensive since these files need to be dumped by the emulation tool and then the power tool needs to consume the file. Any typical waveform format developed for design debug is arranged so that its signal-wise access is optimized and due to that its time-wise access it not very optimal while power tool require data to be supplied in a time based manner. This mismatch in data orientation causes performance loss during both the reading and writing operation for any typical waveform formats. On the other hand, an emulation box inherently processes and generates trace data in time-wise manner which is optimal for power tool consumption, and hence improves the scope of performance and efficiency, and also allows for a more direct, tighter integration with power tools.

So to address the inefficiencies of the typical waveform formats, we have created a dynamic read API. In this approach, trace data is directly streamed to a power tool using the waveform API interface. In this approach, both the emulation system and the power tool run in parallel such that as soon as a time range trace data is produced by the emulator it is fed to the power tool for power analysis. The emulation system then starts working on producing the next time range trace data. A high-level architecture diagram is presented in Figure3.

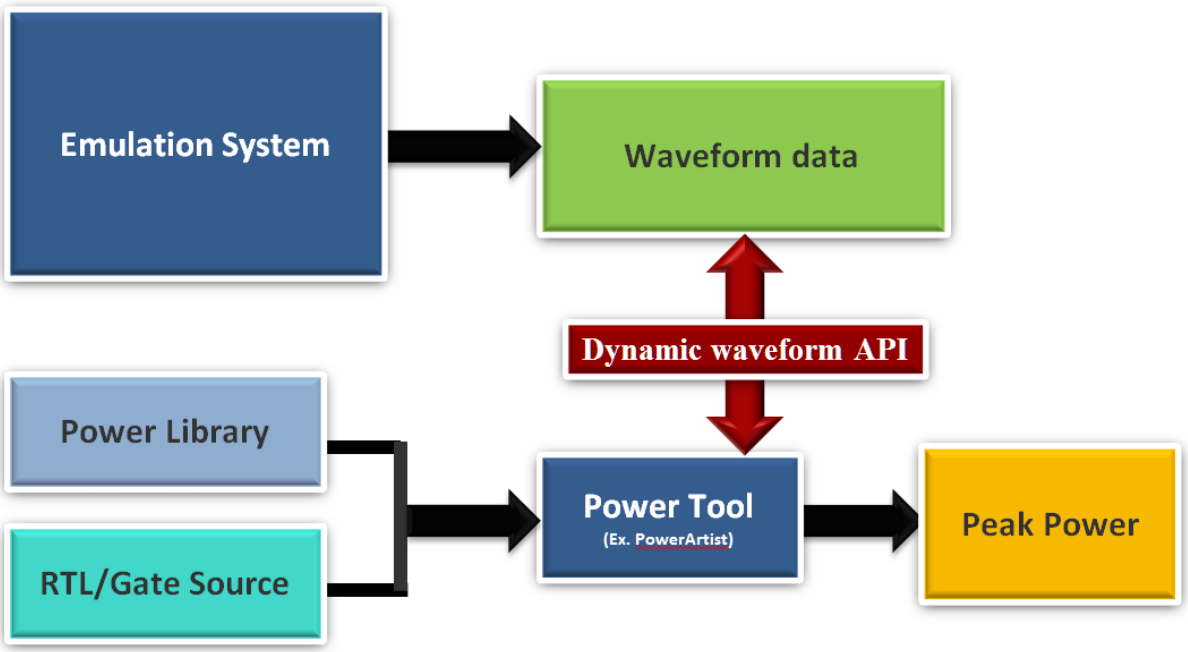

Figure3: High level framework diagram.

It was observed that the data production rate of the emulation system is much faster than the data consumption rate of the power tool. So to further optimize the overall runtime, multiple distributed power tool instances are invoked simultaneously and connected to same emulation session through dynamic waveform APIs as shown in Figure 4.

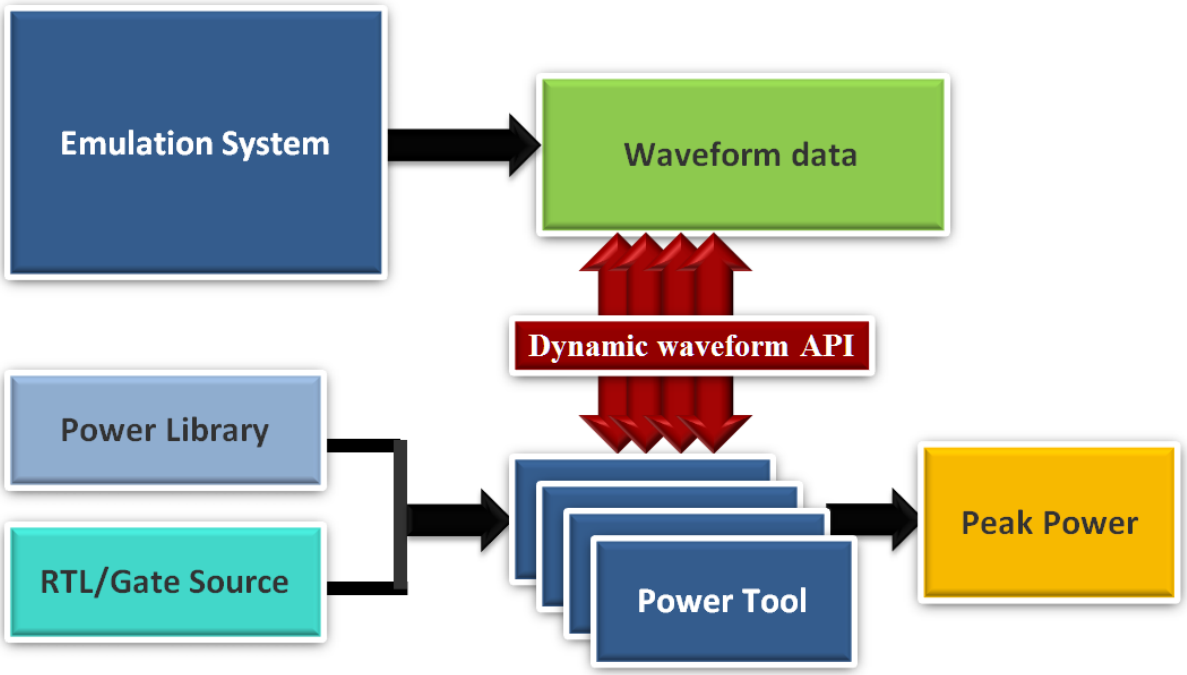

Figure 4: Multiple power tool instances connected to emulation session.

III. EXPERIMENTAL RESULTS

This scheme has been prototyped using an industry standard emulator and a commercially available power estimation tool. Experiments have been performed on the following two large VLSI designs –

1) A graphics processing unit (GPU)

2) A central processing unit (CPU)

The first design, the GPU, had approximately 3.7M nets and the test ran approximately 12M cycles. On this first design, the traditional flow was used to generate an industry standard waveform file, which was fed to the power tool. This approach took close to 109 hours. The waveform file size generated was close to 90GB. With this approach, the activity plot was generated as shown in Figure 5 in 12 minutes. Two time zones, containing top three peaks, were selected for pass 2 for this approach.

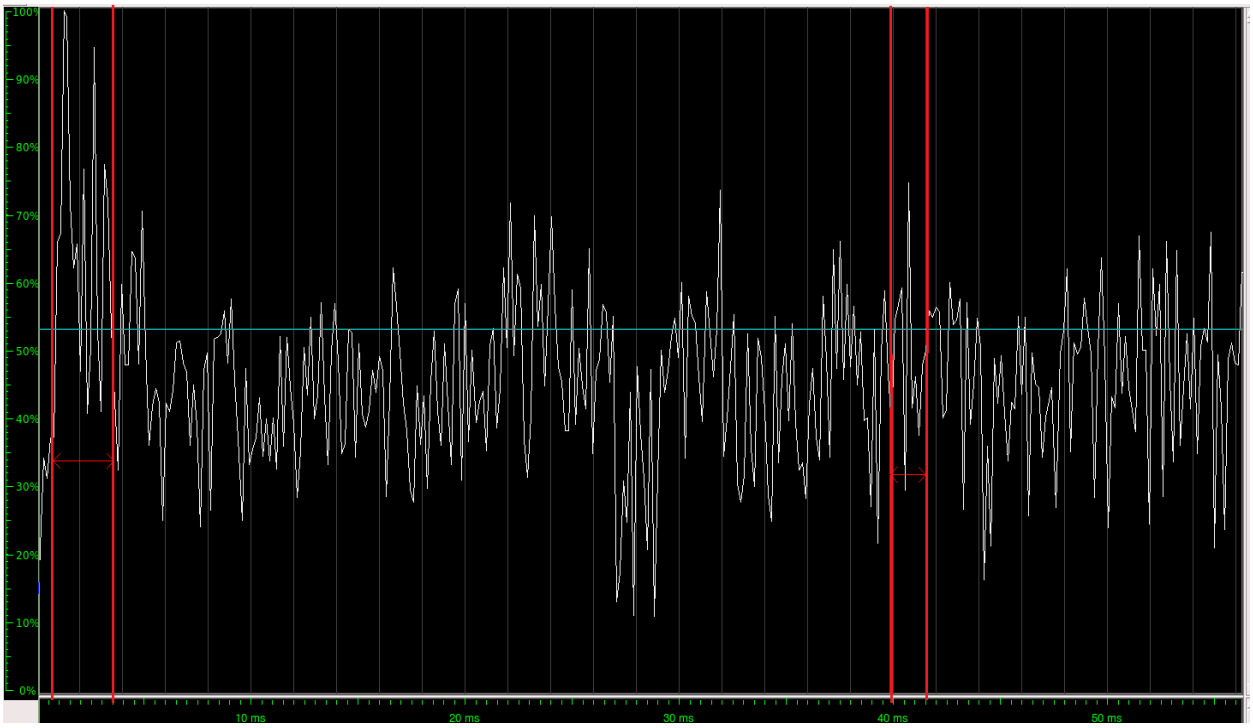

Figure 5: Activity plot of GPU design

Pass two was run on selected time zones for detailed power analysis taking approximately 1.5 hours to generate the peak power numbers for those time zones.

The second design (CPU) had approximately 40M nets and the emulation run on this design captured approximately 10M cycles. With this approach it took close to 30minutes to generate the activity plot as shown in Figure 6. Here one peak was selected for further analysis (marked by red lines in Figure 6), and a detailed peak power analysis was performed in that time region. With this approach the complete analysis was finished in less than **3 hours** while the normal waveform file-based approach took close to **131 hours**.

The results generated with regard to performance gain in time and disk space are listed in TABLE1 and TABLE2 respectively.

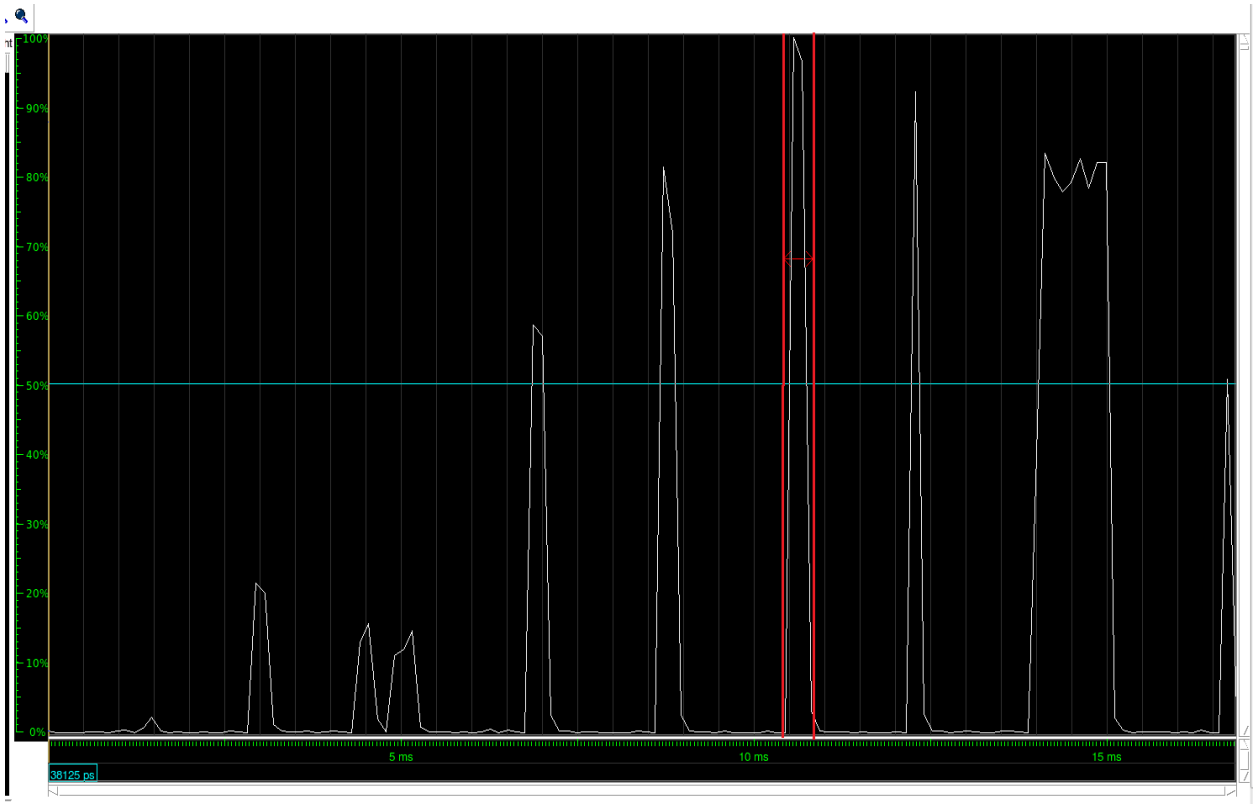

Figure 6: Activity plot of CPU design

TABLE 1
TRADITIONAL APPROACH VS OUR APPROACH

	Traditional waveform file based Single Flow solution (minutes)	Pass1 time (Our approach) (A) (minutes)	Pass2 time (Our approach) (B) (minutes)	Total time (our approach) (C= A+B) (minutes)	Performance Gains over traditional flow
GPU design	6540	12	97	109	**60x**
CPU design	7860	30	146	176	**45x**

TABLE 2

	Traditional waveform file based Single Flow solution (GBs)	Our approach(GBs)	Disk space Gains
GPU design	90	8	**11x**
CPU design	122	12	**10x**

IV. CONCLUSION

With the rapid increase in battery operated devices, it has become more imperative that SoC designs stick to their power budgets and that the peak power consumed by these SoC designs remain in an acceptable range. To achieve this, accurate analysis of peak power consumed by a design under real life circumstances, over long excitation sequences and testbenches has become necessary. In this paper, a novel approach for peak power analysis on real life stimulus and its performance was presented. A two pass approach was suggested where the first pass is used to achieve the objective of reducing the overall runtime and disk space requirements so that a user can run real life scenarios on the DUT and identify critical time zones and culprit instances. In the second pass, detailed analysis is performed only for selected time zones. The second pass fully utilizes the opportunity to perform parallel computation and analysis using a dynamic read API methodology. This paper further presented the novel idea to align the emulation data in a manner to suit the power tool consumption pattern to achieve efficiency. In addition, the paper outlines an interesting proposal on how to bridge the gap between emulation data generation speed and power tools consumption by invoking multiple power tool invocations on the same emulation model of the design. This paper calls for further research in the area of providing the fastest possible synergy between emulation speed and power tool capability. Having multiple power tool instances connected to the same emulation mode is an effective approach for making it a scalable method for power estimation. In this way, this approach has made it possible to locate and measure peak power estimation of the DUTs on real life stimulus and over very long duration tests that was not feasible earlier due to runtime and disk space limitations.

V. FUTURE SCOPE

The merging of results across testbenches, across various design blocks and creating the largest possible activity zones can stimulate the further thoughts about design power optimizations. The use of activity plots in various other interesting activities regarding design functionality checks can also be explored. The use of activity plot generation over long regression testbenches also provides a good alternative for keeping a check on expected activity trends for specified time windows, and can act as a desirable functional verification solution in addition to power estimation.

There are many possibilities in the areas of design debug, power analysis and functionality verification for specific design blocks and SoCs in light of available fast mode switching activity data. This has the potential to attract several interesting experiments based on design activity charts including power analysis of the switching elements, as explored in this paper.

VI. ACKNOWLEDGMENT

We would like to express our gratitude to Vijay Chobisa and Carole Dunn from Mentor Graphics for their insightful comments and technical reviews that helped us make improvements to the quality of this paper.

VII. REFERENCES

[1] Michael Butts, Jon Batcheller, and Joseph Varghese. 1992. An Efficient Logic Emulation System. In Proceedings of the 1991 IEEE International Conference on Computer Design on VLSI in Computer & Processors (ICCD '92). IEEE Computer Society, Washington, DC, USA, 138-141.

[2] Jonathan Babb, Russell Tessier, Matthew Dahl, Silvina Zimi Hanono, David M. Hoki, and Anant Agarwal. 2001. Logic emulation with virtual wires. In Readings in hardware/software co-design, Giovanni De Micheli, Rolf Ernst, and Wayne Wolf (Eds.). Kluwer Academic Publishers, Norwell, MA, USA 625- 642.

[3] Klaus Harbich, Joern Stohmann, Erich Barke, and Ludwig Schwoerer. 1999. A Case Study: Logic Emulation - Pitfalls and Solutions. In Proceedings of the Tenth IEEE International Workshop on Rapid System Prototyping (RSP '99). IEEE Computer Society, Washington, DC, USA, 160-.

[4] Cindy Schiess. 2001. Emulation: debug it in the lab --- not on the floor. In Proceedings of the 33nd conference on Winter simulation (WSC '01). IEEE Computer Society, Washington, DC, USA, 1463-1465.

[5] Abhishek Bhattacharjee, Gilberto Contreras and Margaret Martonosi. 2008 .Full-System Chip Multiprocessor Power Evaluations Using FPGA-Based Emulation. ISLPED'08.

[6] Che-Hua Shih; Chia-Chih Yen; Shen-Tien Lin; Lin, H.; Jing-Yang Jou. 2011. Accelerating Dynamic Peak Power Analysis Using an Essential-Signal-Based Methodology. International symposium on VLSI Design, Automation and Test (VLSI-DAT), 25-28 April 2011

[7] Berthet, C.; Georgelin, P.; Ntyame, J.; Raffin, M. 2012. Peak power estimation using activity measured on emulator. Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on