JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Verification of digital PLL using UVM Methodology

[1]B S Sudha, [2]Mounika Reddy R

[1]Assistant Professor, [2]PG Student

[1, 2] Dept. of Electronics and Communication Engineering,

[1, 2] Dr. Ambedkar Institute of Technology, Bengaluru, India

Abstract : The use of both analog and digital signal is widespread in semiconductor area where most of the researchers are intended to create fast and accurate designs, which include both analog and digital components. Hence, verification of these design is very obvious and it has gained more interest in research domain. Conventional verification methods have a long verification time and a poor level of resilience. This paper presents a UVM verification architecture by using System Verilog for a digital phase-locked loop (DPLL). The characteristics of proposed UVM architecture helps to create a reusable, robust and faster verification environment making faster product release in market. Cadence Incisive Enterprise Simulator was used to create and simulate the testbench. To achieve substantial verification efficiency benefits, the proposed verification architecture employs constrained-random stimulus generation, analog assertions, and coverage metrics. In comparison to standard mixed-signal or analog verification methods, the suggested testbench architecture demonstrated superior verification effectiveness, while concurrently lowering simulation time by an exponential factor. The proposed verification architecture employs limited random real-valued stimulus generation, analogue assertions, coverage metrics, and various test scenarios to achieve significant verification process improvements**.**

IndexTerms **- Digital signals, Phase Locked Loop (PLL), UVM, Power, Timing, Area.**

I. INTRODUCTION

Modern System-on-Chip (SoC) architectures include both digital and analogue components. These mixed-signal designs might show a variety of difficulties all the way through the SoC manufacturing process. Acquiring high accuracy when modelling mixed-signal architectures is important, but it is not the only one. Another critical point to consider is the challenge of integrating analogue and digital parts into a single system with excellent performance. Hence, proper modelling and simulation of designs are critical aspects in accelerating the accurate development of such systems. For verification, the majority of mixed-signal, analog, or digital designs rely heavily on simulation runs. The verification goal is met by incorporating simulation data and data accuracy based on the design's complexity. However, in many situations, testing a mixed-signal circuit might take a lengthy time, resulting in a manufacturing process delay. A significant solution to this challenge is mixed-signal verification using UVM utilising SystemVerilog are suggested in [1-4]. The most well-known verification standard in contemporary digital circuits is UVM. Nonetheless, it is now widely utilised in mixed-signal applications. UVM features such as constrained-random impulse generation, verification planning, assumptions, and coverage metrics development are used in the digital scenario. Traditionally, directed testing, Monte Carlo simulations, and corner analysis have been used to verify the analog component of mixed-signal systems. Processes and methods like as coverage metrics and testbench automation, on the other hand, are essentially non-existent in current analogue solvers [1]. The combination of Real Number Modeling (RNM) with UVM is a significant approach for providing a rapid and repeatable verification environment for mixed-signal systems [5-10]. RNM simulates the voltage behavior of analogue components by using real number values. The RNM method solves the system using a digital solver, which allows for faster simulation than traditional analog approaches [1], [11-14]. This paper presents a UVM-based verification architecture for a DPLL real number model using SystemVerilog. A PLL is a negative feedback loop control system that creates an output signal with the same phase as an input reference signal.

A Digital PLL contains a Phase Frequency detector, low pass filter, charge pump, oscillator and a frequency divider. Phase frequency detects the error between the 2 signals i.e, reference clock signal and a feedback signal. These output errors up and down are fed to the low pass filter. The output of the low pas filter is then fed to the oscillator to generate the voltage output. The proposed UVM verification offer efficient, reusable and higher coverage than FPGA verification. The paper is categorized into subsections like section-II (review of literature), section-III (Problem Statement), section-IV (Research Methodology), section-V (Results and discussion) and section-VI (Conclusion).

II. REVIEW OF LITERATURE

The existing works in the area verification of SoC by using UVM are described in this section. A work of Hussien et al., 2019, have introduced a reusable and generic UVM model to enhance the speed of verification. The UVM approach is adapted over

three different SoC buses and performed its performance analysis. The outcomes suggests that the proposed methodology offers highly reliable, reusable and faster verification [15]. A significant UVM approach with input-output interface is illustrated in Yashas et al., 2019. This approach generates the random test bench to get 100% code and functional coverage by running multiple tests [16]. The increased complexity in silicon chips has witnessed many of the verification challenges. Hence, Xu et al., 2019 have explained UVM based AX14 stream verification and outcomes with improved verification [17]. A work of Georgoulopoulos and Hatzopoulos, 2019 have mentioned a system Verilog based UVM verification for DPLL [18]. The proposed verification model is reusable, fast and reliable in digital market as it offers high gain in verification. A serial peripheral interface (SPI) module for UVM based verification is introduced in Guo et al., 2020 that helps in realizing the communication between the SoC components. The SPI module offers 100% coverage rate and the verification model is significantly migrated over the SoC verification environment [19].

In order to ease the verification of complex SoC, a complex test bench is need to be established and it can be a difficult task. Hence, Goel et al., 2020 have presented a controller area network based on UVM and outcomes with efficient verification [20]. A generic verification environment for NoC platform is introduced in El-Ashry, 2020 that evaluates DUT and offers reusable and error injection solution. The approach decides compatibility as well as feasibility of few error injection techniques of NoC for verification [21]. A UM based single wire protocol (SWP) interface module for verification is introduced for SIM card chips considered in Hao et al., 2019. The study has conserved the random incentives that brings improvement in verification performance of interface module [22]. A Verilog based design of UART and verification of it by using UVM is described in Priyanka et al., 2021. The design has elaborated the complete functioning of UART and the verification using UVM reduces the energy consumption and reusability [23].

A UVM approach for verification of an RTL code interconnect is proposed in Kwon et al., 2021 for memory centric computing. The packets generated in the proposed UVM approach are transmitted into the switch ports via virtual interfaces. The approach brings effective verification for memory centric computing [24]. A work of Jain and Cooperman, 2020 introduced a checkpoint restart architecture for fault tolerant CUDA applications using UCM and streams. The checkpoint reduces intercommunication overhead than existing approaches [25]. A system level verification based on UVM is presented for SRAM in Yan et al., 2020. The verification module generates random excitation signals which reduces development time, verification cycle and improved reusability [26]. In Wang et al., 2020, a verification platform of UVM is introduced for RISC-V SoC. The verification platform at module level generates the random stimulus for testing of module functions [27].

III. Problem Statement

Wireless communication is powered by analogue circuitry, but because analogue devices are more difficult to miniaturize, digital components have gradually gained hold. Radio-frequency circuits are particularly susceptible to design modifications, because the qualities of analogue components like as inductors do not increase as devices shrink. As a result, analogue chips lag behind their all-digital counterparts by a couple of manufacturing-process generations, resulting in substantially coarser details. Digital circuits have gradually taken over more of the analogue domain over time. Phase locked loop (PLL), a basic communication block constructed utilizing digital circuits, is the poster child for this approach. The fundamental benefit of any digital implementation is that with a modern manufacturing process, circuits become quicker and take up less space.

- A lot of study has been done recently on the design of PLL circuits, and more research is continuously being done on this area.
- The majority of studies have been undertaken in order to achieve a larger lock range PLL with reduced lock time and tolerable phase noise.
- Clock production and recovery in microprocessors, networking, communication systems, and frequency synthesizers are the most versatile applications of the PLL.
- In high-performance digital systems, phase locked loops (PLLs) are often utilized to provide well-timed on-chip clocks.

Hence, verification of these design is very obvious and it has gained more interest in research domain. Conventional verification methods have a long verification time and a poor level of resilience.

IV. Methodology

Some of the existing works [19-26] have described UVM as a standard methodology to verify the SoC designs. The UVM uses features from System Verilog which helps to create components for verification and build communication among the components. Also, UVM uses library to develop universal verification components (UVC) and test bench modules to boost the verification portability and reusability. Also, it forms a component that can be used to drive the stimulus generated from test bench for the Design Under Test (DUT). A basic architecture of UVM test bench is given in **Figure 1** and is formed by test bench module (top level) connecting the DUT and verification components.

- A test represents the test cases for test bench where a test class composed of environment, configurability of environment etc. The sequences can be build and initiated in test to execute the test scenario.
- The container class of UVM is environment and it links agents and scoreboard. Also, it composed of top level monitor, predictor and a checker.
- An agent can be deployed in active and passive mode where it generates necessary stimulus (active mode) and drives the same stimulus via interfacing with DUT. A sequencer, monitor and driver are the units of an active agent along with the configurable object. An agent in passive mode performs the sampling of received signals from DUT and it composes single monitor. A monitor samples signals and converts them to transaction level activities and forwards to the scoreboard.
- A sequencer is a stimulus generator and it performs the controlling of transaction flows among the driver and sequences. A driver is an element which continuously receives sequences from sequences and drives those sequences to DUT.

- The scoreboard forms an element to DUT functionality verification by comparing DUT responses against expected values (originated from predictor). The scoreboard obtains transactions from analysis of monitors.
- A sequence item composed of information or data needed for verification process during stimulus generation. These sequences items are generated in sequential manner.

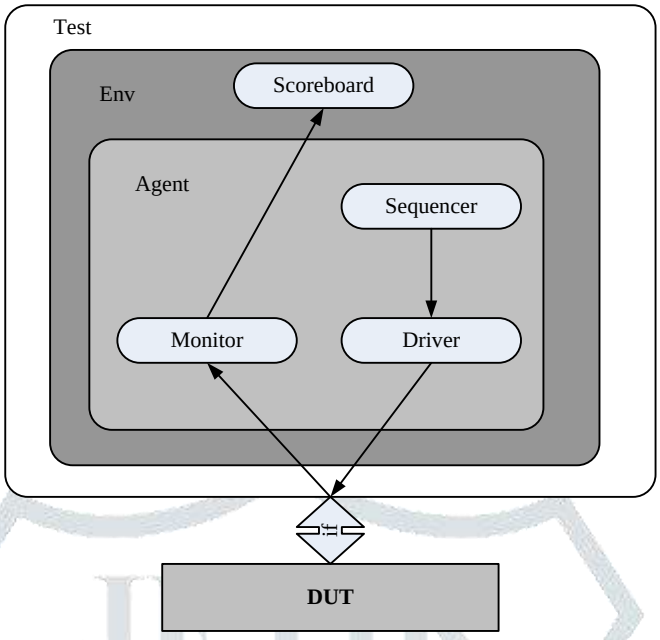


Figure 1. Architecture of UVM testbench

3.1 Design Implementation

The UVM testbench's proposed architecture, a System Verilog-based DPLL, is utilised to validate the digital-signal DUT. Charge pump, phase frequency detector (PFD), 1st order derivative, voltage crystal oscillator (VCO), and six frequency dividers make up the proposed DPLL model. A reference crystal oscillator (RCO) is assumed to create a particular reference clock signal (RCS). One of the local clock signals released from the frequency divider is compared to the phase of RCS. VCO is used to ensure that phases are synced. Figure 2 shows the DPLL construction, which includes a digital-frequency divider, phase detector, charge pump, low pass filter, and VCO. The phase deviations between the input reference frequency and the reference crystal, as well as the divided output local clock frequency and the frequency received from the frequency divider, are first detected by the phase detector. To charge the pump, the resulting phase error signals (leading or trailing) are analysed. The leading signal becomes high during the rising edge of the reference clock if the reference clock is ahead of the local clock created by 6 dividers. The leading signal remains in the high position (1) until the local clock transitions from low to high activity. The lagging signal in this point gets high, and both flip-flops are reset to 0 via the asynchronous reset signal.The charge pump is a current source (bipolar switch) and it yields pulses (positive or negative) into the loop filter. Further, the loop filter performs the correction of control voltage (Vcontrol) subjected VCO based on the phase error signals received from the phase detector via charge pump. Then, the divider performs the division of the output frequency generated from VCO before subjecting to phase detector to compare with input reference frequency.

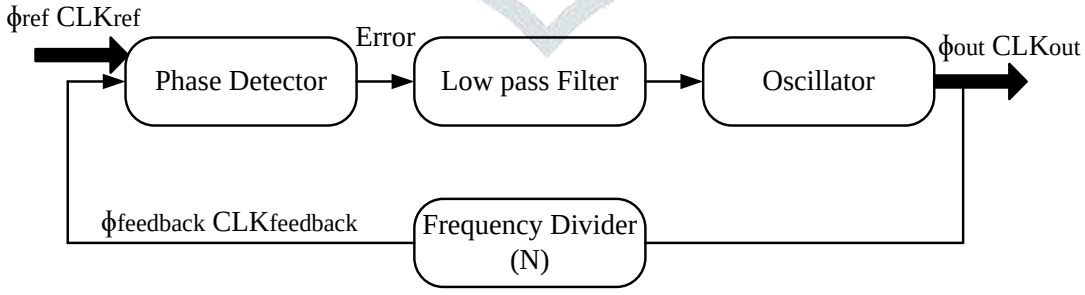


Figure 2. Proposed DPLL (DUT)

3.2 Verification Model

The proposed verification model composed of top module of DPLL where testbench declares the instances as well as interfaces. The testbench runs different test cases generated to verify the DUT. Every test performs the instantiates main of test bench architecture and verification environment. The environment exhibits single agent, predictors, scoreboard, monitor and essential agent configuration for agent instantiation. The agent holds the sequencer, driver, monitor and configuration object. The sequencer of the agent acts as controller among driver and test sequences. Every sequence generates the set of sequence to forward to driver. The logic variables leading, lagging (phase error) signals, local clock, system clock, charge pump current, Vcontrol and output frequency are exist in sequence item.

V. RESULTS AND ANALYSIS

The proposed architecture of a system Verilog-based DPLL, is utilized to validate the digital-signal DUT. Charge pump, phase frequency detector (PFD), 1st order derivative, voltage crystal oscillator (VCO), and six frequency dividers make up the proposed DPLL model. The obtained outcomes with different nm technology are discussed in this section. **Figure 3** represents the schematic diagram of DPLL designed in 45nm technology and is extracted from Cadence Innovus tool.

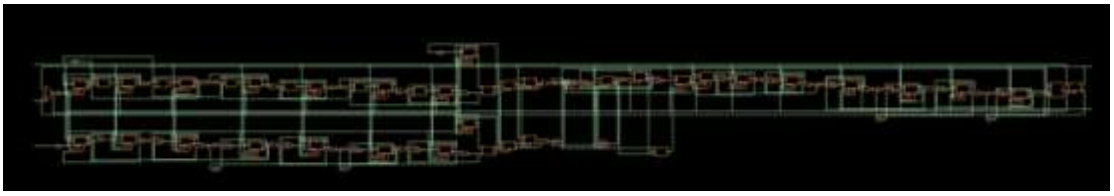


Figure 3. 45nm DPLL Schematic Diagram

Similarly, **Figure 4** indicates the schematic diagram of DPLL designed in 180nm technology and is extracted from Cadence Innovus tool.

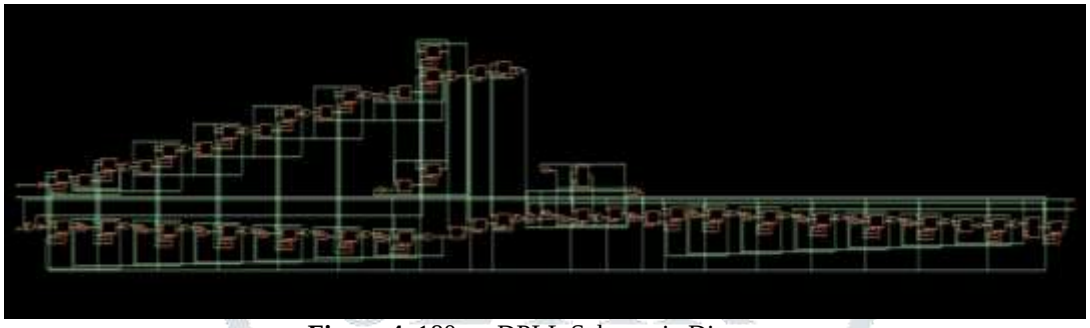


Figure 4. 180nm DPLL Schematic Diagram

Figure 5 shows the output waveform of the DPLL Design. The output signal clk_out is generated with respect to the input signals rst, clk_in and clk_ref. the another output signal clK_out_8x is generated w.r.t the output signal clk_out. The clk_out_8x signal is the ouput of the clk_out signal which is multiplied 8 times. The clk_out_8x signal has 8times each the output of clk_out in both up and down signals.

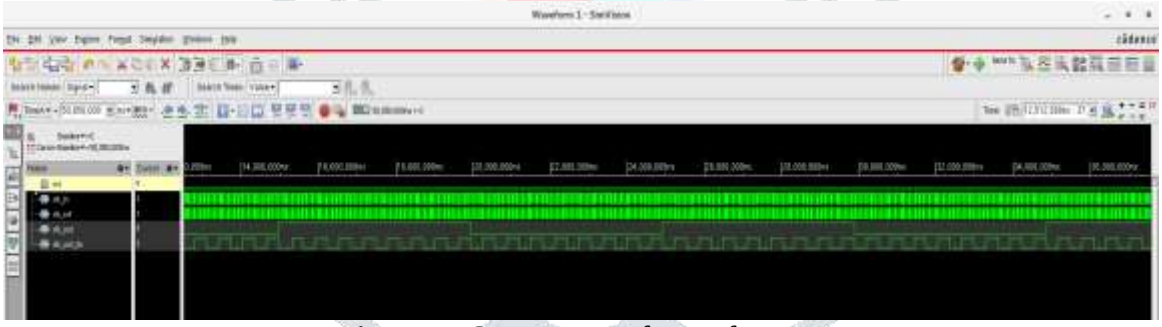


Figure 5. Output Waveform of DPLL

Figure 6 is the layout of the DPLL design. The layout is extracted for the 180nm DPLL design. Based on the this layout, the timing constraints are obtained.

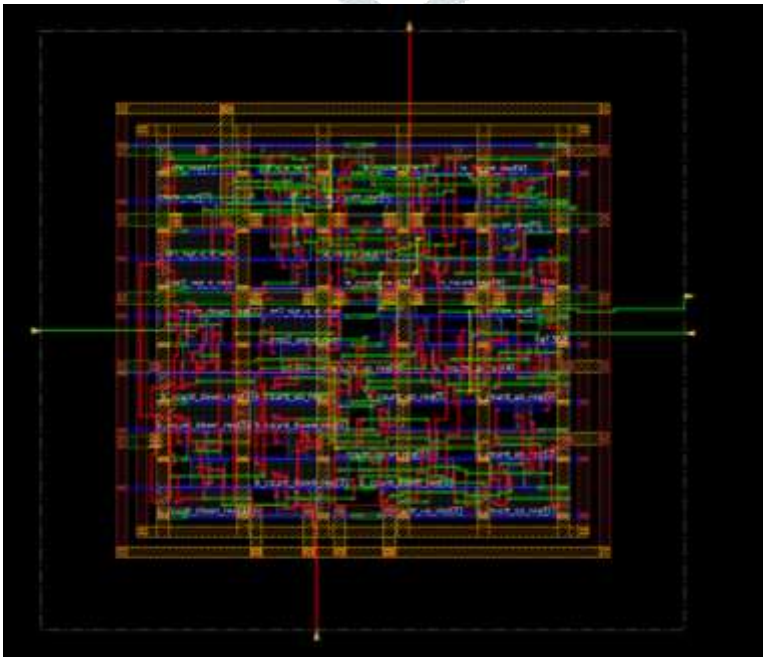


Figure 6 Layout of DPLL

The following **Figure 7** represents the simulating steps for the UVM verification. At first the system Verilog model of DPLL design and DPLL testbench are complied. And then UVM packages are extracted and imported to the design for further verification.

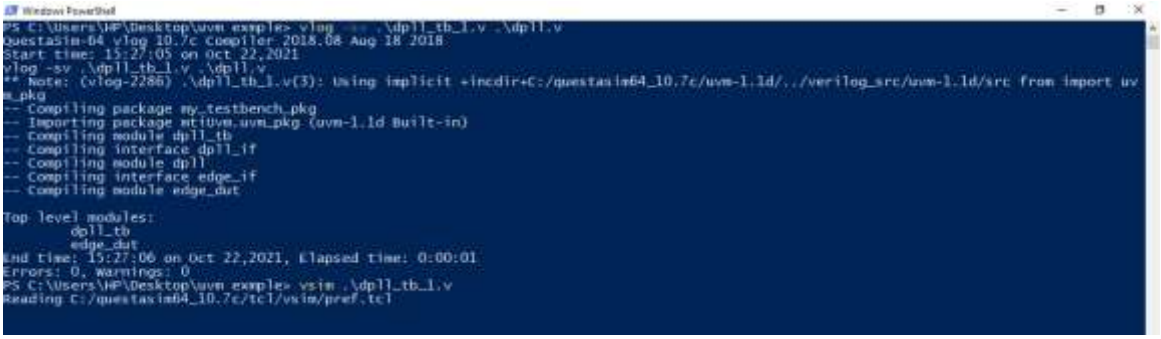


Figure 7 Simulating the design

The final verified UVM based DPLL design waveform is presented in **Figure 8**. UVM based architecture is designed for the verification of the Digital PLL design. The DPLL working results are verified with the obtained UVM results.

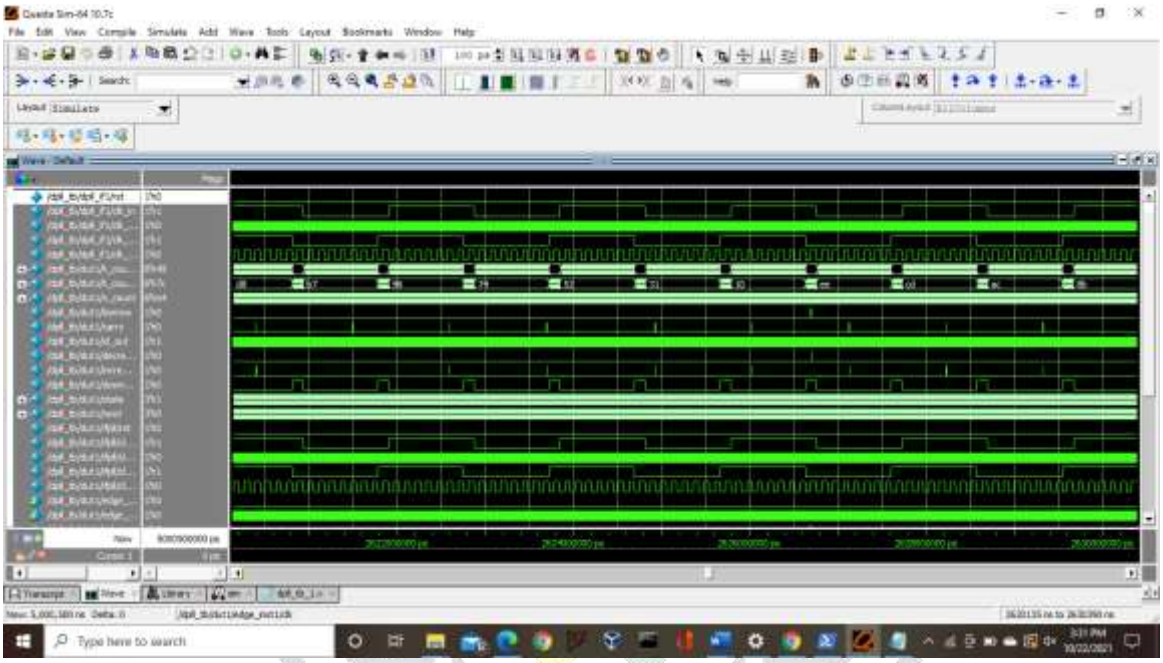
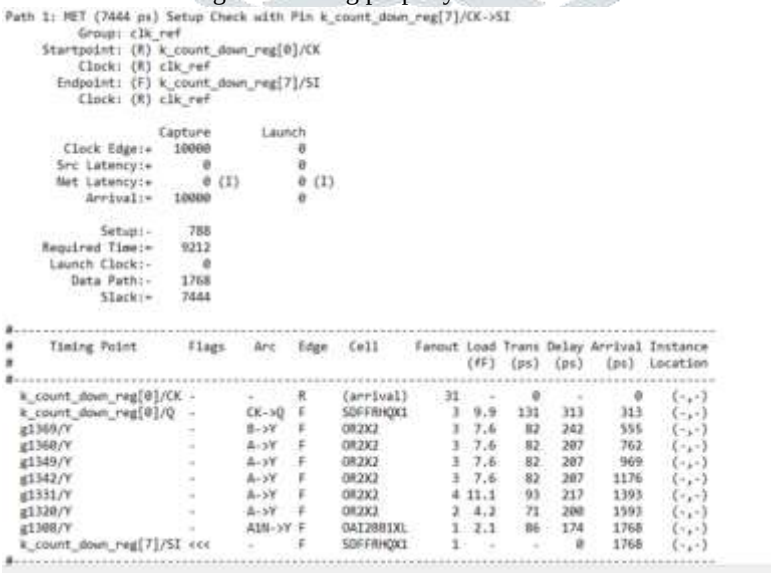


Figure 8. UVM Verified DPLL Results Waveform

The following **Figure 9** gives the timing report of the DPLL design. The slack time is defined by the time difference between the achieved time and desired time for the design timing path. The slack time given in the timing report is positive, which indicates that the design is working properly.



```
Path 1: MET (7444 ps) Setup Check with Pin k_count_down_reg[7]/CK->SI
          Group: clk_ref
     Startpoint: (R) k_count_down_reg[0]/CK
          Clock: (R) clk_ref
       Endpoint: (F) k_count_down_reg[7]/SI
          Clock: (R) clk_ref

                    Capture       Launch
      Clock Edge:+    10000            0
      Src Latency:+       0            0
      Net Latency:+       0 (I)        0 (I)
          Arrival:+   10000            0

            Setup:-     788
    Required Time:=    9212
     Launch Clock:-       0
        Data Path:-    1768
            Slack:=    7444

#-------------------------------------------------------------------------------------------------------------
#     Timing Point       Flags    Arc   Edge   Cell      Fanout Load Trans Delay Arrival Instance
#                                                          (fF)  (ps)  (ps)   (ps)  Location
#-------------------------------------------------------------------------------------------------------------
  k_count_down_reg[0]/CK -        -     R      (arrival)     31    -     0     -       0    (-,-)
  k_count_down_reg[0]/Q  -        CK->Q F      SDFFRHQX1      3  9.9   131   313     313    (-,-)
  g1369/Y                -        B->Y  F      OR2X2          3  7.6    82   242     555    (-,-)
  g1360/Y                -        A->Y  F      OR2X2          3  7.6    82   207     762    (-,-)
  g1349/Y                -        A->Y  F      OR2X2          3  7.6    82   207     969    (-,-)
  g1342/Y                -        A->Y  F      OR2X2          3  7.6    82   207    1176    (-,-)
  g1331/Y                -        A->Y  F      OR2X2          4 11.1    93   217    1393    (-,-)
  g1320/Y                -        A->Y  F      OR2X1          2  4.2    71   200    1593    (-,-)
  g1308/Y                -        A1N->Y F     OAI2BB1XL      1  2.1    86   174    1768    (-,-)
  k_count_down_reg[7]/SI <<<      -     F      SDFFRHQX1      1    -     -     0    1768    (-,-)
#-------------------------------------------------------------------------------------------------------------
```

Figure 9. Timing Report for the DPLL design

Figure 10, represents the power report the design. The power report is of leakage power in the DPLL design, and the dynamic power used by the DPLL design and total power used by the design. For 86 cells, the leakage power, dynamic power is 106.006nW, 433751.915nW and 433857.922nW respectively.


dll_power - Notepad

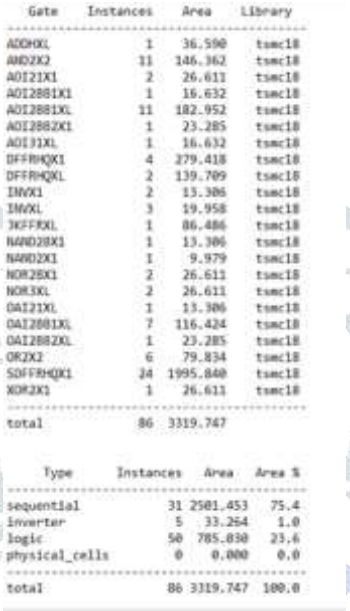
File Edit Format View Help

```
Generated by:          Genus(TM) Synthesis Solution 17.22-s017_1
Generated on:          Oct 08 2021  11:15:21 am
Module:                dpll
Operating conditions:  slow (balanced_tree)
Wireload mode:         enclosed
Area mode:             timing library
```

Instance	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
dpll	86	106.006	433751.915	433857.922

Figure 10. Power Report for the DPLL design

Figure 11 gives the report of the gates used by the digital PLL. The number of the gates and types of the gates used by the digital PLL is reported. The area utilization of sequential, inverter, logic and physical cell is 75.4%, 1%, 23.6% and 0% respectively. This indicates that sequential cell uses more area.

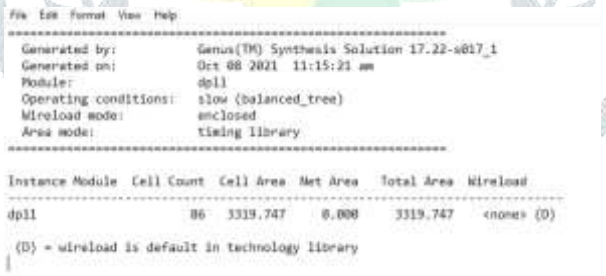


Gate	Instances	Area	Library
ADDHXL	1	36.590	tsmc18
AND2X2	11	146.362	tsmc18
AOI21X1	2	26.611	tsmc18
AOI2B1X1	1	16.632	tsmc18
AOI2B1XL	11	182.952	tsmc18
AOI2B2X1	1	23.285	tsmc18
AOI31XL	1	16.632	tsmc18
DFFRHQX1	4	279.418	tsmc18
DFFRHQXL	2	139.709	tsmc18
INVX1	2	13.306	tsmc18
INVXL	3	19.958	tsmc18
JKFFRXL	1	86.486	tsmc18
NAND2BX1	1	13.306	tsmc18
NAND2X1	1	9.979	tsmc18
NOR2BX1	2	26.611	tsmc18
NOR3XL	2	26.611	tsmc18
OAI21XL	1	13.306	tsmc18
OAI2B1XL	7	116.424	tsmc18
OAI2B2XL	1	23.285	tsmc18
OR2X2	6	79.834	tsmc18
SDFFRHQX1	24	1995.840	tsmc18
XOR2X1	1	26.611	tsmc18
total	86	3319.747	

Type	Instances	Area	Area %
sequential	31	2501.453	75.4
inverter	5	33.264	1.0
logic	50	785.030	23.6
physical_cells	0	0.000	0.0
total	86	3319.747	100.0

Figure 11. Gates report for the DPLL design

The total number of area consumed by the design is shown in the Figure 12. The cell area and total area is 3319.747.



File Edit Format View Help

```
Generated by:          Genus(TM) Synthesis Solution 17.22-s017_1
Generated on:          Oct 08 2021  11:15:21 am
Module:                dpll
Operating conditions:  slow (balanced_tree)
Wireload mode:         enclosed
Area mode:             timing library
```

Instance Module	Cell Count	Cell Area	Net Area	Total Area	Wireload
dpll	86	3319.747	0.000	3319.747	<none> (D)

(D) = wireload is default in technology library

Figure 12. Area report for the DPLL design

VI. CONCLUSION

The Digital-PLL design is implemented using Cadence Innovus tool and results are extracted according. Further for the verification, Questa-Sim is used for the Verification of the design. System-Verilog with UVM methodology is used for the design verification. The simulation waveform of the proposed verification testbench with lock reference is displayed in the figure. For a digital PLL, a UVM verification architecture was presented. It takes advantage of UVM's excellent verification efficiency, as well as SystemVerilog – RNM's mixed-signal design capabilities. In comparison to standard mixed-signal or analog verification methods, the suggested testbench architecture demonstrated superior verification effectiveness, while concurrently lowering simulation time by an exponential factor. The outcomes power analysis suggest that for 86 cells, the leakage power, dynamic power is 106.006nW, 433751.915nW and 433857.922nW respectively. The area utilization of sequential, inverter, logic and physical cell is 75.4%, 1%, 23.6% and 0% respectively. This indicates that sequential cell uses more area. The cell area and total area is 3319.747.

The proposed verification architecture employs limited random real-valued stimulus generation, analogue assertions, coverage metrics, and various test scenarios to achieve significant verification process improvements. Further, the study can be extended with different nm technology.

VII. ACKNOWLEDGMENT

We would like to thanks Dr. Ramesh S, Head of Department of Electronics and Communication Engineering, Dr. Ambedkar

Institute of Technology, Bengaluru, India; Also, all faculty members of Department of Electronics and Communication Engineering, for excellent support to carried out this work and giving us inspiration at each and every time. Finally, we would like to thanks our family members who always support, help and guide us during our work.

REFERENCES

[1] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," in Proc. *International Conf. Compilers, Architecture, Synthesis Embedded Syst*., pp. 238–246, 2001.
[2] Wang, Jiacun (2019). Formal Methods in Computer Science. CRC Press. p. 34. ISBN 978-1-4987-7532-8.
[3] "Finite State Machines – Brilliant Math & Science Wiki". brilliant.org. Retrieved 2018-04-14.
[4] Belzer, Jack; Holzman, Albert George; Kent, Allen (1975). Encyclopedia of Computer Science and Technology. 25. USA: CRC Press. p. 73. ISBN 978-0-8247-2275-3.
[5] Koshy, Thomas (2004). Discrete Mathematics With Applications. Academic Press. p. 762. ISBN 978-0-12-421180-3.
[6] Wright, David R. (2005). "Finite State Machines" (PDF). CSC215 Class Notes. David R. Wright website, N. Carolina State Univ. Archived from the original (PDF) on 2014-03-27. Retrieved 2012-07-14.
[7] Keller, Robert M. (2001). "Classifiers, Acceptors, Transducers, and Sequencers" (PDF). Computer Science: Abstraction to Implementation (PDF). Harvey Mudd College. p. 480.
[8] John E. Hopcroft and Jeffrey D. Ullman (1979). Introduction to Automata Theory, Languages, and Computation. Reading/MA: Addison-Wesley. ISBN 978-0-201-02988-8.
[9] Pouly, Marc; Kohlas, Jürg (2011). Generic Inference: A Unifying Theory for Automated Reasoning. John Wiley & Sons. Chapter 6. Valuation Algebras for Path Problems, p. 223 in particular. ISBN 978-1-118-01086-0.
[10] Jacek Jonczy (Jun 2008). "Algebraic path problems" (PDF). Archived from the original (PDF) on 2014-08-21. Retrieved 2014-08-20., p. 34
[11] Brutscheck, M., Berger, S., Franke, M., Schwarzbacher, A., Becker, S.: Structural Division Procedure for Efficient IC Analysis. IET Irish Signals and Systems Conference, (ISSC 2008), pp.18–23. Galway, Ireland, 18–19 June 2008.
[12] Hamon, G. (2005). A Denotational Semantics for Stateflow. International Conference on Embedded Software. Jersey City, NJ: ACM. pp. 164–172. CiteSeerX 10.1.1.89.8817.
[13] "Harel, D. (1987). A Visual Formalism for Complex Systems. Science of Computer Programming, 231–274" (PDF). Archived from the original (PDF) on 2011-07-15. Retrieved 2011-06-07.
[14] Alur, R., Kanade, A., Ramesh, S., & Shashidhar, K. C. (2008). Symbolic analysis for improving simulation coverage of Simulink/Stateflow models. International Conference on Embedded Software (pp. 89–98). Atlanta, GA: ACM.
[15] Hussien *et al*., "Development of a Generic and a Reconfigurable UVM-Based Verification Environment for SoC Buses," *2019 31st International Conference on Microelectronics (ICM)*, Cairo, Egypt, 2019, pp. 195-198.
[16] D. Yashas, P. S. Hari Babu and N. Shylashree, "UVM-based Logic Verification of Input Output Interface," *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, Bangalore, India, 2019, pp. 420-423.
[17] C. Xu, W. Ni and Y. Song, "UVM-based Functional Coverage Driven AXI4-Stream Verification," *2019 IEEE 13th International Conference on ASIC (ASICON)*, Chongqing, China, 2019, pp. 1-4.
[18] N. Georgoulopoulos and A. Hatzopoulos, "UVM-based Verification of a Digital PLL Using SystemVerilog," *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Rhodes, Greece, 2019, pp. 23-28.
[19] Y. Guo *et al*., "A SPI Interface Module Verification Method Based on UVM," *2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA)*, Chongqing, China, 2020, pp. 1219-1223.
[20] A. Goel, B. B. T. Sundari and S. Mathew, "UVM based Controller Area Network Verification IP (VIP)," *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, 2020, pp. 645-652.
[21] S. El-Ashry, M. Khamis, H. Ibrahim, A. Shalaby, M. Abdelsalam and M. W. El-Kharashi, "On Error Injection for NoC Platforms: A UVM-Based Generic Verification Environment," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1137-1150, May 2020.
[22] Y. Hao, W. Feng, X. Feng, Y. Hu and X. Tang, "A SWP Interface Module Verification Method Based on UVM," *2019 IEEE 2nd International Conference on Electronics and Communication Engineering (ICECE)*, Xi'an, China, 2019, pp. 5-9.
[23] Priyanka, M. Gokul., A. Nigitha. and J. T. Poomica., "Design of UART Using Verilog And Verifying Using UVM," *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2021, pp. 1270-1273.
[24] H. J. Kwon, M. -H. Oh and W. -o. Kwon, "Verification of Interconnect RTL Code for Memory-Centric Computing using UVM," *2021 International Conference on Electronics, Information, and Communication (ICEIC)*, Jeju, Korea (South), 2021, pp. 1-4.
[25]T. Jain and G. Cooperman, "CRAC: Checkpoint-Restart Architecture for CUDA with Streams and UVM," *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, Atlanta, GA, USA, 2020, pp. 1-15.
[26]S. Yan, S. Geng, X. Peng, H. Tang, Y. Zhang and Z. Wang, "Design of Verification Platform for CAN-FD IP Customized SRAM Controller Based on UVM," *2020 IEEE 5th International Conference on Integrated Circuits and Microsystems (ICICM)*, Nanjing, China, 2020, pp. 218-222.
[27] J. Wang, N. Tan, Y. Zhou, T. Li and J. Xia, "A UVM Verification Platform for RISC-V SoC from Module to System Level," *2020 IEEE 5th International Conference on Integrated Circuits and Microsystems (ICICM)*, Nanjing, China, 2020, pp. 242-246.