# A REPORT ON:

# Power Optimization via Transition Probability-Based Stochastic Clock Gating

**BY**
Shaheen Ali
ID – 2021B4A33044H

**In fulfillment of**
**MATH F – Applied Stochastic Processes**
**Instructor:** Dr. Nirman Ganguly


**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI,HYDERABAD CAMPUS**

**SECOND SEMESTER 2024-2025**

# ACKNOWLEDGMENTS

I sincerely thank BITS Pilani Hyderabad campus for providing me with this wonderful opportunity to undertake an independent study project. The university's continued commitment to facilitating student experiential learning is deeply appreciated. I am extremely grateful to my project mentor, Dr. Nirman Ganguly, for his invaluable guidance and support throughout this project.

Nirman Sir went above and beyond to ensure our success, holding weekly review meetings to provide practical feedback and help refine our approach. His suggestions were instrumental in shaping our work. Thanks to his motivation, problem-solving assistance, and the university enabling remote project work, I could participate in this cherished learning experience.

This has been an enriching learning journey under Nirman Sir’s mentorship. I feel privileged to have had this opportunity, which was enabled by BITS Pilani Hyderabad.

# ABSTRACT

Dynamic power dissipation in digital circuits is a direct function of the circuit's switching activity. In this project, we develop a Markov chain model, which characterizes the state transition probabilities of a 2-bit PIPO register, and we use this model to derive its corresponding activity factor.
This computation enables us to implement stochastic clock gating, reducing unnecessary power consumption by probabilistically disabling the clock signal to the register when its activity is low.
We first implement this model with deterministic transition probabilities for the register state and then extending it to sampling these probability values from discrete and continuous probability distributions to reflect the real world variability in data flows.
We then tried to implement an adaptive feedback mechanism that dynamically adjusts the clock gating threshold based on observations made on switching patterns, improving power efficiency.
This approach is flexible and scalable for low-power design, offering significant energy savings while not protecting the performance of the circuit.

# INTRODUCTION

## Digital Electronics:

Digital circuits form the fundamental building blocks of modern computing systems. These circuits are composed of logic gates (eg. AND, NOT, OR, XOR, etc.) which are used to model Boolean logic based on binary information of 0s and 1s.

These circuits are responsible for performing computations, storing data, and controlling data flow in devices ranging from microcontrollers to AI accelerators and supercomputers.

These digital circuits are classified into two broad categories: Combinational circuits and sequential circuits.

Combinational circuits are digital circuits where the output depends only on the current input values. They are memoryless circuits where change in output corresponds directly to change in inputs, and cannot store past values and states. These circuits are built purely using logic gates (AND, OR, NOR, NAND, etc.) to perform Boolean functions. Output is a direct function of the input, and there is no feedback or state retention. These circuits are used for arithmetic operations, data routing, and signal processing.

Sequential circuits, on the other hand, are digital circuits where the output depends not only on the current input but also on the past sequence of inputs. They use memory elements to store/retain information. The output, therefore, is not a direct function of inputs but is influenced by both the current input values and the previous state. This state retention ability makes these circuits useful for counting, data storage, and control operations.

## Sequential Circuits:

1. The key components of sequential circuits are:
2. Memory elements – latches and flip-flops which store binary information. These elements retain their states until explicitly changed by input signals.
3. Feedback path: Output of memory elements is fed back to circuit to enable the system to use past states.

4. Clock Signal: Most sequential circuits are synchronized by a clock signal which controls when the memory elements update their states. This enables orderly and predictable operation.

Sequential circuits are of two types based on the reliance of their operation on the clock signal:

1. Synchronous Sequential Circuits – use the clock signal to synchronize state transitions.
2. Asynchronous Sequential Circuits – do not rely on the clock signal but other miscellaneous input signals (eg. load, reset, etc.) to change states. They are complex and susceptible to timing issues.

## Register:

In this project, we will be focusing on a simple synchronous sequential circuit, namely a register, which is a fast storage component used to store binary data temporarily. It is a sequential circuit because it relies on two input signals, the load signal and the clock signal.

The load signal loads input values into the register and the register values only get updated at either the rising or falling edges of the clock to maintain data stability. Once data is stored in a register, it remains unchanged until a new value is loaded even if the input changes.

In this project, we are specifically focusing on the parallel-in, parallel-out (PIPO) register. This is a collection of flip-flops arranged in series, with each flip-flop capable of storing one bit of data. It loads data and outputs it in parallel, and for this project, this computation will happen at every rising edge of the clock.

We will analyze the dynamic power dissipation that occurs in this circuit and optimize it using stochastic methods.

## Dynamic Power Dissipation:

Dynamic power dissipation is the power dissipated by a digital circuit when it actively switches between states (i.e. when a signal transitions from 0 to 1 or 1 to 0, which causes the charging and discharging of capacitors in transistors). It is one of the two main sources of power consumption in digital circuits, the other being static power dissipation, caused by leakage currents in transistors and wiring.

Dynamic power dissipation is given by:

$$P_{dynamic} = \alpha . C_L . V_{DD}^2 . f$$

Where,

- $\alpha$ is the activity factor, which is the probability that the signal with transition in a clock cycle,
- $C_L$ is the load capacitance, the total capacitance of the circuit nodes being switched,
- $V_{DD}$ is the supply voltage of the circuit,
- $f$ is the clock frequency.

Now let us view dynamic power dissipation in the context of the circuit that we have chosen, the PIPO register.

The PIPO register consists of multiple flip-flops which operate in synchronization with a clock signal. On each clock cycle, the register can either load new data (in parallel) or retain current data.

How is power dissipated in this circuit?

1) The clock signal triggers flip-flops in the register to update their states. Every time the clock signal transitions, the flip-flops switch their outputs based on the input data. This switching activity consumes dynamic power.

2) If the input data to the registers changes frequently, the flip-flops switch their values more often, increasing the activity factor ($\alpha$).

3) Higher clock frequencies increase the rate of switching, leading to higher dynamic power dissipation.

The main aim of studying digital electronics in this generation is to find ways to optimize two parameters of digital circuits: performance (speed) and power consumption.

Dynamic power dissipation of a N-bit PIPO register:

$$P_{dynamic} = \sum_{i=1}^{N} \alpha_i . C_{L,i} . V_{DD}^2 . f$$

Where $\alpha_i$ is the activity factor and $C_{L,i}$ is the load capacitance of the *i*th flip-flop of the N-bit register.

To reduce the dynamic power dissipated in this circuit, we can reduce any of the parameters in the above equation. In this project, we will focus on reducing the activity factor of the circuit.

Different methods to reduce activity factor in dynamic circuits are namely, clock gating, data encoding (gray coding), operand isolation, power aware scheduling, voltage scaling, logic optimization and state encoding.

The focus of this project will be to apply an advanced application of clock gating, stochastic clock gating.

## Clock Gating:

Clock gating is a power-saving technique that stops the clock signal from reaching certain parts of a digital circuit when those parts are idle or not actively performing useful work. By disabling the clock, the circuit avoids unnecessary toggling of flip-flops and logic gates which reduces dynamic power consumption.

Clock gating works by introducing a gating logic (typically AND gate) into the clock path of a module or register. The gating logic controls whether the clock signal propagates to the circuit based on a control signal.

Implementation:

Clock signal: *CLK*, Control Signal: *EN*, produce Gated Clock Signal: *GCLK*

$$GCLK = CLK.EN$$

When EN = 1, the clock signal passes through and the circuit operates normally.
When EN = 0, the clock signal is blocked, and the circuit remains idle (no transitions occur)

In the context of the PIPO register,
If EN = 1, the clock signal reaches the flip-flops and they update their values
If EN = 0, the clock signal is blocked and the flip-flops retain their current states.

In traditional, deterministic clock gating, the decision to gate the clock is based on fixed, predictable conditions (e.g. a control signal like “enable” or “idle”).

There are a few limitations with this technique of clock gating:

Coarse-Grained Control: A lot of the circuit is gated together when using traditional clock gating, which is frequently implemented at the module or register level. Because some circuit components could remain inactive even when the module is operational, this can result in lost possibilities for power savings.

Inflexibility: The choice to gate the clock is predicated on predetermined criteria, such as a control signal. This doesn't adjust to different data patterns or workloads. For instance, conventional clock gating might not gate the clock for the unused portion of a register if it is only half utilised.

Overhead of Control Logic: More control logic is needed to implement fine-grained clock gating, which can add complexity and area. In large designs, managing the control signals for many gating cells can become challenging.

Let us now define Stochastic Clock Gating.

## Stochastic Clock Gating:

Stochastic clock gating uses probabilistic models to gate the clock based on the statistical properties of the input data. Unlike traditional clock gating, which is deterministic, stochastic clock gating makes probabilistic decisions about when to gate the clock.

Mechanism of stochastic clock gating:

- Transition probabilities ($P_{trans}$) are estimated for each signal in the circuit.
- A probabilistic model is used to decide whether to gate the clock for a given signal or module.
- The clock gating decision is made based on the likelihood of a signal transitioning rather than a fixed condition.

Example, if a signal has a low transition probability say $P_{trans}$ = 0.1, then the clock is gated with a high probability say$P_g = 0.9$ and vice versa.

# RELEVANCE

There is a growing demand for energy efficient digital systems where power consumption is a critical concern. We have discussed in the introduction about traditional deterministic clock gating and while being effective in reducing dynamic power consumption, has limitations. Stochastic clock gating offers a more advanced and dynamic approach, by leveraging transition probabilities to make statistical gating decisions.

This fine-grained control enables a significant amount of power savings, especially in circuits with variable activity factors, as it can disable the clock even when traditional methods might not.

Some use cases are as follows:

1. Low-power IOT devices: IOT sensors and edge devices often operate on limited battery power and have highly variable workloads. Stochastic clock gating can dynamically adapt to these variations, thus significantly extending battery life.

2. AI Accelerators and Neural Networks: AI workloads often involve irregular and data-dependent calculations and computations, which leads to varying activity factors. Stochastic clock gating can optimize power dissipation by gating unused or low-activity components.

3. Real-Time Signal Processing: Digital signal processors and wireless communication often have bursty workloads. Stochastic clock gating can adapt to these bursts, reducing power during low-activity periods.

4. Data centers and High-Performance Computing: Data centers have servers with varying utilization levels, and stochastic clock gating can optimize power usage during low-demand periods. It can also be applied to GPUs and FPGAs which have highly variable workloads.

In this project, we implement a model for a 2-bit PIPO register. This model is highly scalable and be extended to larger systems, such as N-bit registers, multi-core processors, and complex digital designs. In an N-bit register, the transition probabilities and clock-gating logic can be applied to individual cores or modules or functional units, adapting to their specific workloads. In multi-core processors, stochastic clock gating can be applied to individual cores or functional units, adapting to specific workloads.

# OBJECTIVES/METHODOLOGY

1. Define transition probabilities and construct a Markov chain representation of the 2-bit PIPO register's state transitions.

2. Calculate the activity factor and power dissipation from the Markov model and determine an optimal threshold for clock gating.

3. Implement and compare deterministic and stochastic clock gating, using the calculated threshold.

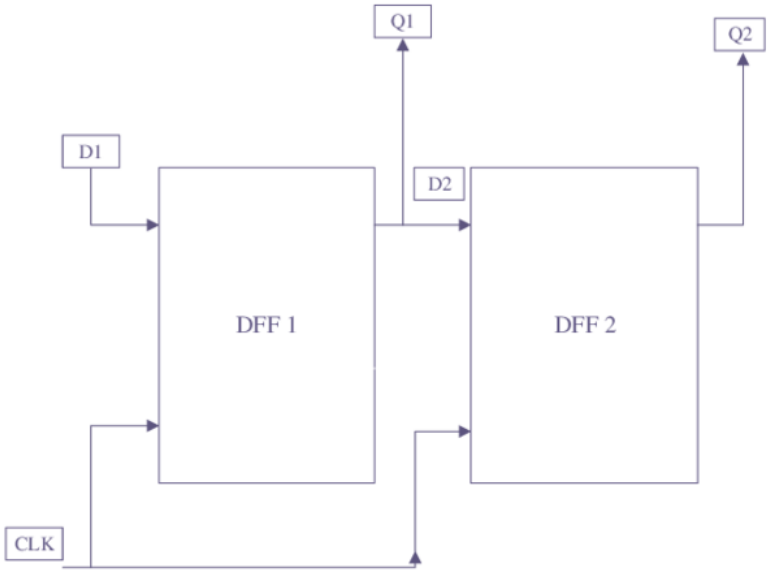
4. Develop and simulate adaptive feedback-based stochastic clock gating methods, namely, Window-Based Clock Gating and Event-Triggered Clock Gating.

5. Comparative analysis of the resultant activity factors of the developed stochastic clock gating methods with circuits without clock gating to analyse power savings and performance trade-offs.

6. Define future scope and methods to extend the model to real world scenarios and complex mechanisms.

# Defining Transition Probabilities

A 2-bit Parallel-In, Parallel-Out shift register:




Here DFF1 and DFF2 are 2 D-flipflops that store values input by inputs D1 and D2 which are then presented as outputs in the form Q1 and Q2. The other input to the two flip-flops is the system clock, which we are aiming to disable and enable based on our probabilistic model.

Each flip-flop can store a 1-bit value, either a 0 or a 1 and therefore a combination of 2 flip-flops can store $2^2 = 4$ different values.

These values are 00, 01, 10 and 11. We consider each value as a state in which this register can be in.

Therefore the 4 possible states of this system are:

$$S = \{00, 01, 10, 11\}$$

Assumptions:

We make the finite state space assumption:
The system cannot enter any other states beyond these four
There is no undefined behaviour – every transition must lead to a valid state.

We assume that all the states are recurring states and that none of the states are absorbent states.

For the first computation, we also assume that bit transitions are independent of

each other.

We define one-step transition probabilities for all the states:

$$P_{ij} = P(S_{n+1} = j \mid S_n = i) \;\; 0 \leq i, j \leq 3$$

Where $P_{ij}$ is the probability of the register to transition to state *j* when it is currently in state *i*.

There are the probabilities that output values of the registers will change from state i to state j at every rising clock edge, as input values can only be stored in registers at the instant when the clock signal transitions from high to low or vice versa.

Using these transition probabilities, we construct the state transition matrix:

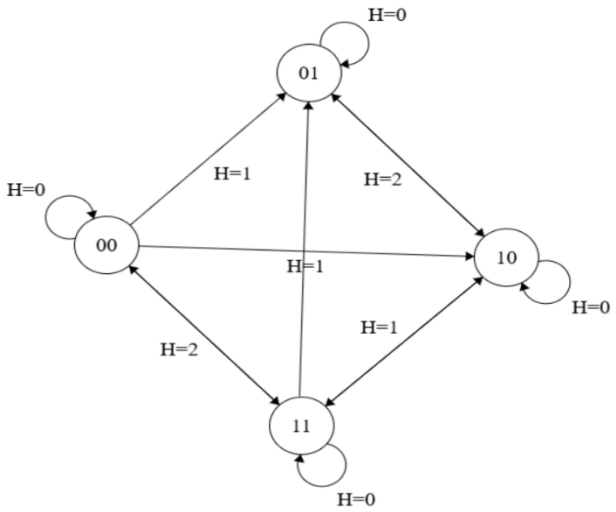
$$P = \begin{Vmatrix} P_{00\to00} & P_{00\to01} & P_{00\to10} & P_{00\to11} \\ P_{01\to00} & P_{01\to01} & P_{01\to10} & P_{01\to11} \\ P_{10\to00} & P_{10\to01} & P_{01\to10} & P_{10\to11} \\ P_{11\to00} & P_{11\to01} & P_{11\to10} & P_{11\to11} \end{Vmatrix}$$

# Determining Activity Fator α

We therefore define Hamming distance, which quantifies how many bits change during each state transition.

For 00 to 01 transition, 1 bit changes its value, for 10 to 01 two bits change their value and so on.

Here is a Finite State Machine representing the change of states and the resultant Hamming distance values of each transition:




We therefore define $H_{i,j}$ which is the number of bit transitions from state *i* to state *j*. We construct a matrix for our two bit system:

$$H = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{bmatrix}$$

We now compute the steady state values $\pi_j$ for each state, which satisfy:

$$\pi_j = \sum_i \pi_i . P_{ij}$$

Using the transition probabilities, hamming distance and steady state probabilities, we calculate the expected number of bit flips every clock cycle:

$$\alpha_{clk} = \sum_{i}\sum_{j} \pi_i . P_{i \to j} . H_{ij}$$

We normalize the expected bit flips every clock cycle to find the activity factor of the system:

$$\alpha = \frac{\alpha_{clk}}{N}$$

Where N = size of the register.

# Determining the threshold for clock gating

The main idea of this implementation is to calculate a parametric threshold value $\alpha_{th}$ which is used as the control parameter to determine whether the clock being supplied to the register should be gated or not.

One simple idea would be to set $\alpha_{th} = 1 - \alpha$, that is, set the threshold voltage as the direct complement of the activity factor, but this can be proven inefficient.

Consider extreme cases:
For extremely active circuits ($\alpha \approx 1$): $\alpha_{th} \approx 0$, which would mean the clock is almost never gated, therefore destroying the chance of saving some power.
For characteristically idle circuits ($\alpha \approx 0$): $\alpha_{th} \approx 1$, which would mean the clock is almost always gated, aggressively, therefore affecting the circuit's performance.

We therefore use a scaled version of $\alpha$, using a scaling factor k, which depends on our power saving requirements as follows:

$$\alpha_{th} = k.\alpha$$

Where $0 < k < 1$.

We can control the value of k to control the power savings and thus establish a tradeoff between performance and power saving, i.e, when power saving is prioritized, we use a higher value of k, and if we value performance, we lower the value of k.

# A Numerical Implementation:

Let us assume deterministic transition probabilities for the 2-bit PIPO register with the following states:

$$S = \{00, 01, 10, 11\}$$

The state transition matrix is as follows:

$$P = \begin{bmatrix} 0.5 & 0.2 & 0.2 & 0.1 \\ 0.3 & 0.4 & 0.2 & 0.1 \\ 0.3 & 0.2 & 0.4 & 0.1 \\ 0.1 & 0.3 & 0.3 & 0.3 \end{bmatrix}$$

To calculate steady-state probabilities $\pi = \{\pi_{00}, \pi_{01}, \pi_{10}, \pi_{11}\}$

$$\pi = P.\pi$$

$$[\pi_{00} \quad \pi_{01} \quad \pi_{10} \quad \pi_{11}].\begin{bmatrix} 0.5 & 0.2 & 0.2 & 0.1 \\ 0.3 & 0.4 & 0.2 & 0.1 \\ 0.3 & 0.2 & 0.4 & 0.1 \\ 0.1 & 0.3 & 0.3 & 0.3 \end{bmatrix} = [\pi_{00} \quad \pi_{01} \quad \pi_{10} \quad \pi_{11}].$$

This gives a system of linear equations:

$$\begin{aligned} \pi_{00} &= 0.5\pi_{00} + 0.3\pi_{01} + 0.3\pi_{10} + 0.1\pi_{11} \\ \pi_{01} &= 0.2\pi_{00} + 0.4\pi_{01} + 0.2\pi_{10} + 0.3\pi_{11} \\ \pi_{10} &= 0.2\pi_{00} + 0.2\pi_{01} + 0.4\pi_{10} + 0.3\pi_{11} \\ \pi_{11} &= 0.1\pi_{00} + 0.1\pi_{01} + 0.1\pi_{10} + 0.3\pi_{11} \end{aligned}$$

And the normalizing condition: $\pi_{00} + \pi_{01} + \pi_{10} + \pi_{11} = 1$
Bringing to one side:

$$\begin{aligned} -0.5\pi_{00} + 0.3\pi_{01} + 0.3\pi_{10} + 0.1\pi_{11} &= 0 \\ 0.2\pi_{00} - 0.6\pi_{01} + 0.2\pi_{10} + 0.3\pi_{11} &= 0 \\ 0.2\pi_{00} + 0.2\pi_{01} - 0.6\pi_{10} + 0.3\pi_{11} &= 0 \\ 0.1\pi_{00} + 0.1\pi_{01} + 0.1\pi_{10} - 0.7\pi_{11} &= 0 \end{aligned}$$

Which can be written in the matrix form $AX = b$

$$\cdot \begin{bmatrix} -0.5 & 0.3 & 0.3 & 0.1 \\ 0.2 & -0.6 & 0.2 & 0.3 \\ 0.2 & 0.2 & -0.6 & 0.3 \\ 0.1 & 0.1 & 0.1 & -0.7 \end{bmatrix} \begin{bmatrix} \pi_{00} \\ \pi_{01} \\ \pi_{10} \\ \pi_{11} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Replacing one row with normalizing condition:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.2 & -0.6 & 0.2 & 0.3 \\ 0.2 & 0.2 & -0.6 & 0.3 \\ 0.1 & 0.1 & 0.1 & -0.7 \end{bmatrix} \begin{bmatrix} \pi_{00} \\ \pi_{01} \\ \pi_{10} \\ \pi_{11} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

On numerical computation, $\pi = \{0.34375, 0.265625, 0.265625, 0.125\}$

We shall now compute $\alpha_{clk}$

$$\alpha_{clk} = \sum_i \sum_j \pi_i . P_{i \to j} . H_{ij}$$

$$P = \begin{bmatrix} 0.5 & 0.2 & 0.2 & 0.1 \\ 0.3 & 0.4 & 0.2 & 0.1 \\ 0.3 & 0.2 & 0.4 & 0.1 \\ 0.1 & 0.3 & 0.3 & 0.3 \end{bmatrix}, H = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{bmatrix}$$

$$\pi = \{0.34375, 0.265625, 0.265625, 0.125\}$$

Let us computer row-wise:

For state 00:

$$\alpha_{clk0} = 0.34375 \times (0.5 \times 0 + 0.2 \times 1 + 0.2 \times 1 + 0.1 \times 2) = 0.20625$$

For state 01:

$$\alpha_{clk1} = 0.265625 \times (0.3 \times 1 + 0.4 \times 0 + 0.2 \times 2 + 0.3 \times 1) = 0.2125$$

For state 10:

$$\alpha_{clk2} = 0.265625 \times (0.3 \times 1 + 0.2 \times 2 + 0.4 \times 0 + 0.1 \times 1) = 0.2125$$

Fore state 11:

$$\alpha_{clk3} = 0.125 \times (0.1 \times 2 + 0.3 \times 1 + 0.3 \times 1 + 0.3 \times 0) = 0.1$$

Summing every row computation:

$$\alpha_{clk} = 0.20625 + 0.2125 + 0.2125 + 0.1 = 0.73125$$

Let us assume k to have a standard value of 0.3

$$\alpha_{th} = k . \alpha = 0.3 \times 0.73125 = 0.219375$$

Let us round this value off to 0.2194 for ease of simulation.

$$\alpha_{th} = 0.2194$$

# Establishing the Mechanism for Performing Stochastic Clock Gating:

Now that we have established our Markov chain model to find the steady-state activity factor $\alpha$ of the 2-bit PIPO register, and constructed a relation to determine the gating threshold ($\alpha_{th}$) by making it dependent on our power-saving needs, let us now focus on how we use this calculated threshold to gate the clock to the register.

Now to implement this into hardware, we have developed two methods:

1. Window-Based Clock Gating
2. Event-Triggered Clock Gating

For both methods, we calculate average activity $\alpha_{observed}$ as follows:

$$\alpha_{observed} = \frac{Total\ no.of\ bit\ flips\ in\ the\ observation\ window}{Size\ of\ observation\ window\ \times Number\ of\ bits}$$

## 1) WINDOW-BASED CLOCK GATING

In this approach, we monitor the circuit activity for small windows of cycles, determine an average activity coefficient $\alpha_{observed}$ , compare it to the threshold calculated using the Markov chain model, and make the decision whether to gate the clock or not for the next window of cycles.

For this approach, we define different phases for the clock gating control:

**Observation phase:**

- The clock signal is enabled for a fixed window of cycles (called the observation window)
- During this phase, the system operates normally, and the bit changes are monitored by the control circuit.
- Activity calculation: Activity factor α_observed is computed in this observation window

**Decision phase:**

- $\alpha_{observed}$ is compared with $\alpha_{th}$

- If $\alpha_{observed} < \alpha_{th}$, the clock is gated (disabled) for the next gating window.
- If $\alpha_{observed} \geq \alpha_{th}$, we keep the clock enabled for the next gating window.

**Gating phase:**

- The gating window is kept larger than the observation window.
- If the clock is disabled, the system enters a low-power (idle) state for the gating window.
- If the clock remains enabled, the system continues its normal (active) mode of operation.

This approach can be further modified by making the sizes of the observation and gating window variable, dynamically computed based on the system's activity, or can be programmed by a higher level logic at regular intervals. However that adds even more hardware cost and complexity to the circuit, therefore we stick to fixed observation and gated windows.

## 2) EVENT-TRIGGERED CLOCK GATING

In this method, the system monitors the activity of the circuit (bit changes) continuously and when the activity falls below the threshold value for a fixed amount of clock cycles, the clock signal is gated until significant activity greater than the threshold is observed.

This method is also divided into different phases:

**Activity Monitoring:**

- Number of bit changes at the inputs are continuously monitored by the system.
- The average activity coefficient, $\alpha_{observed}$, is computed over a short window of cycles.

**Inactivity detection:**

- We compare $\alpha_{observed}$ with $\alpha_{th}$.

- If $\alpha_{observed} < \alpha_{th}$, an inactivity counter is activated.
- If the inactivity counter exceeds a predefined limit, the clock is disabled.

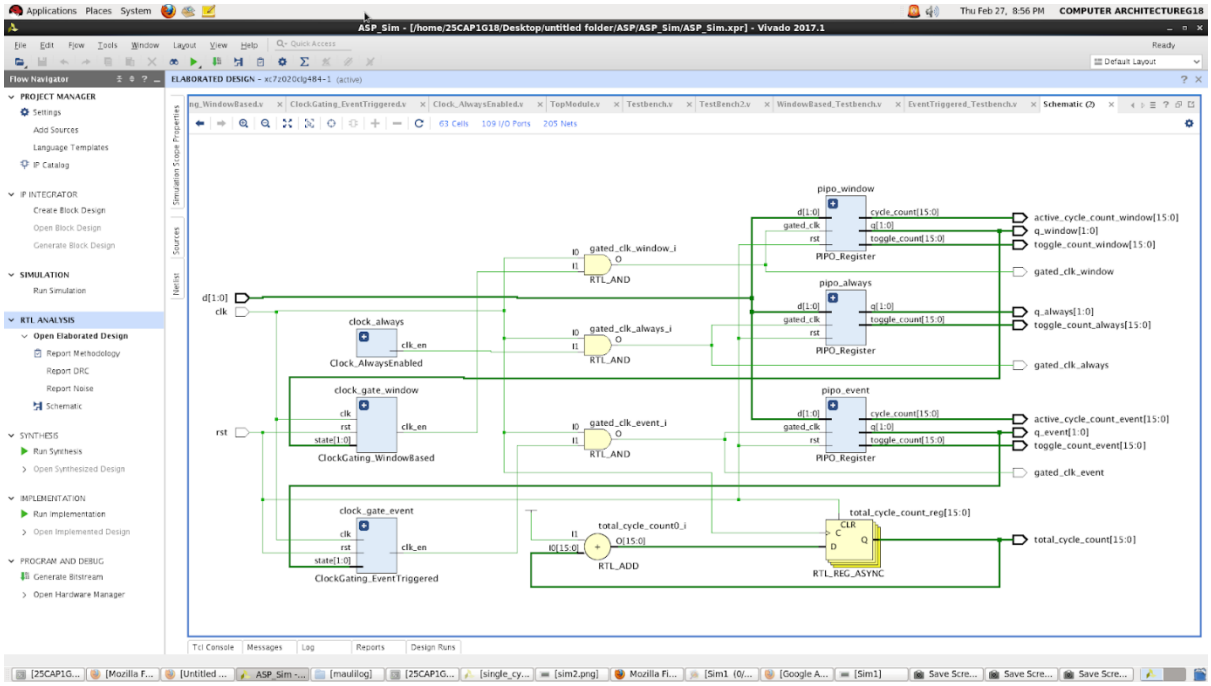
**Reactivation:**

- The activity of the input signals is still continuously monitored.
- If $\alpha_{observed} \geq \alpha_{th}$ we increment a reactivation counter.
- If the reactivation counter exceeds a predefined limit, we enable the clock.

# SIMULATIONS:

To test the model out, we carried out simulations in Verilog HDL using Xilinx Vivado platform to compile the code and generate simulation waveforms for the same.

Here is the RTL schematic for the modelled circuit:



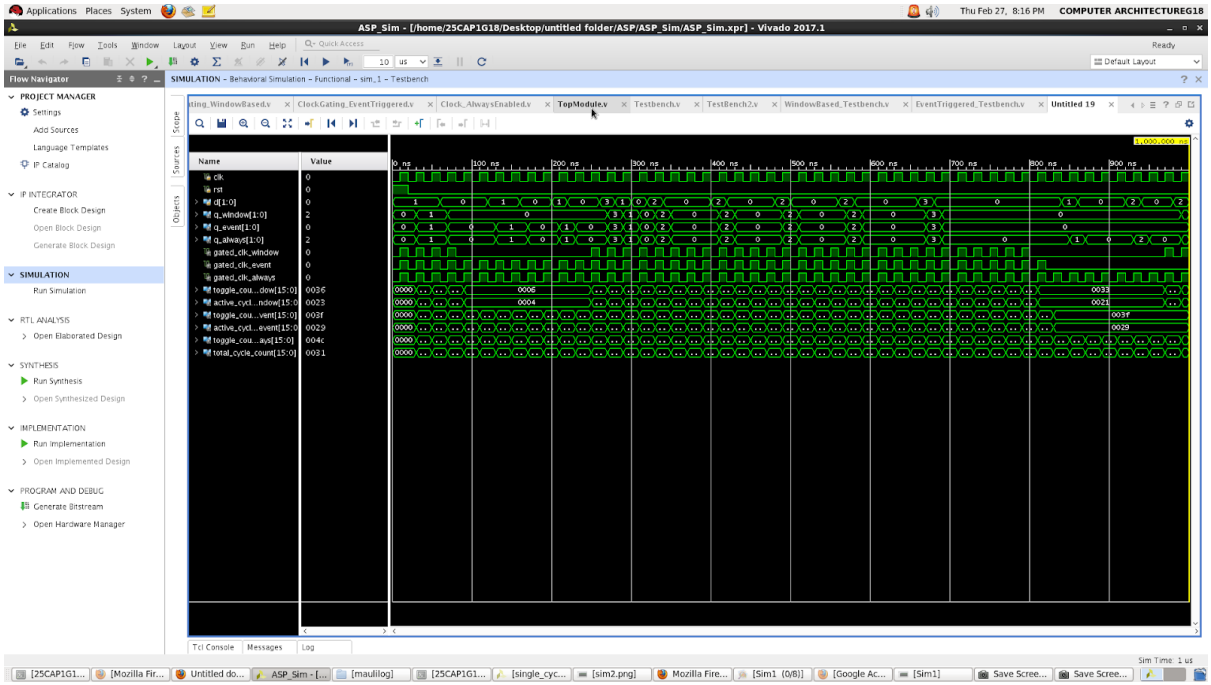

To test the model, 3 different types of testbenches where used:

1) First testbench whose inputs were generated based on the Markov chain model with the transition probabilities specified in the Numerical example which was calculated before.

2) A testbench with high activity followed by long idle periods to highlight the effectiveness of the window-based method for gating the clock and reducing the activity factor in the prolonged inactive periods.

3) A testbench with sporadic, low-activity changes and extended idle periods to show how the event-triggered method gates the clock efficiently during inactivity while responding quickly to bursts of events.

## Simulation #1 (Inputs generated using the Markov chain model):




Here the toggling activity of the output was monitored, when subjected to the window-based gated clock, the event-triggered gated clock as well as a non-gated clock, to compare the activity factors of each. The activity was stored in count registers namely “toggling_count_window”, “toggling_count_event” and “toggling_count_always” for the three methods respectively. Here activity factor is calculated as follows:

$$\alpha = \frac{Total\ number\ of\ output\ toggles\ during\ simulation\ period}{Total\ clock\ cycles\ \times 3\ (2\ bits + clock)}$$

Therefore:

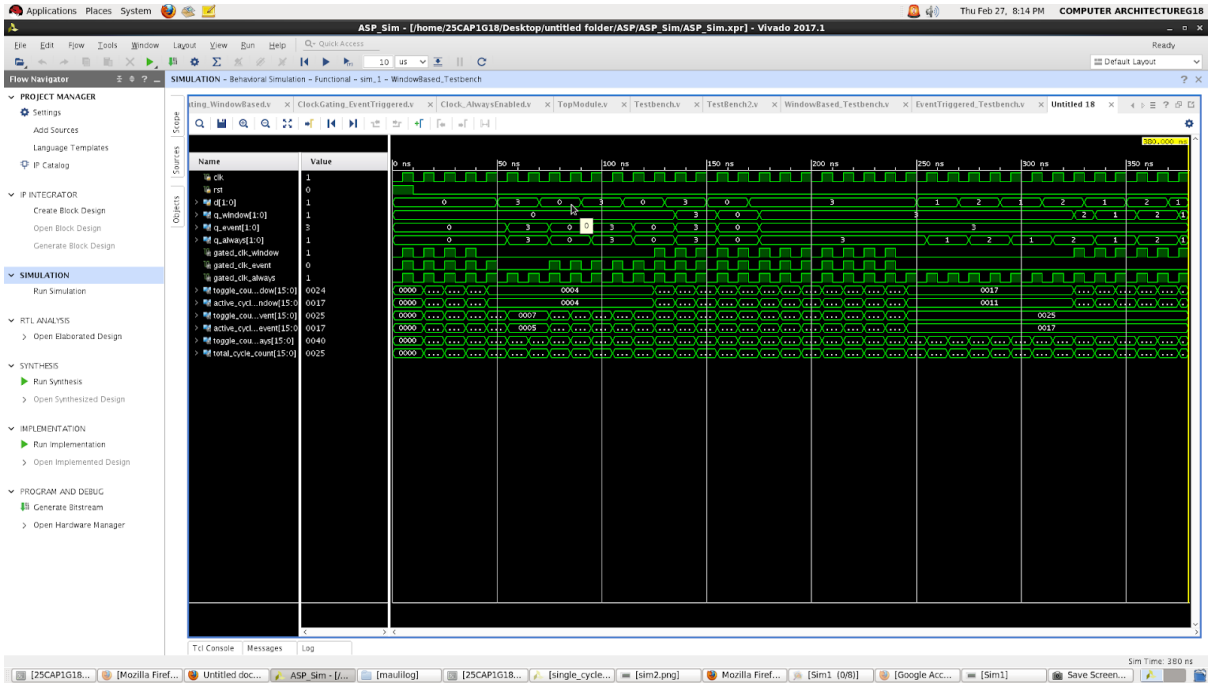
$$\alpha_{window} = \frac{54}{50 \times 3} = 0.36$$

$$\alpha_{event} = \frac{63}{50 \times 3} = 0.42$$

$$\alpha_{no\ gating} = \frac{76}{50 \times 3} = 0.5067$$

We can therefore see that we have successfully reduced the activity factor of the outputs using our clock-gating methods as compared to non-gated clock. The input-load in this case was highly active and therefore both the window and event based gating methods gated the clock for very short durations of time.

# Simulation #2 (Focus on effectiveness of window-based stochastic clock gating):




Here:

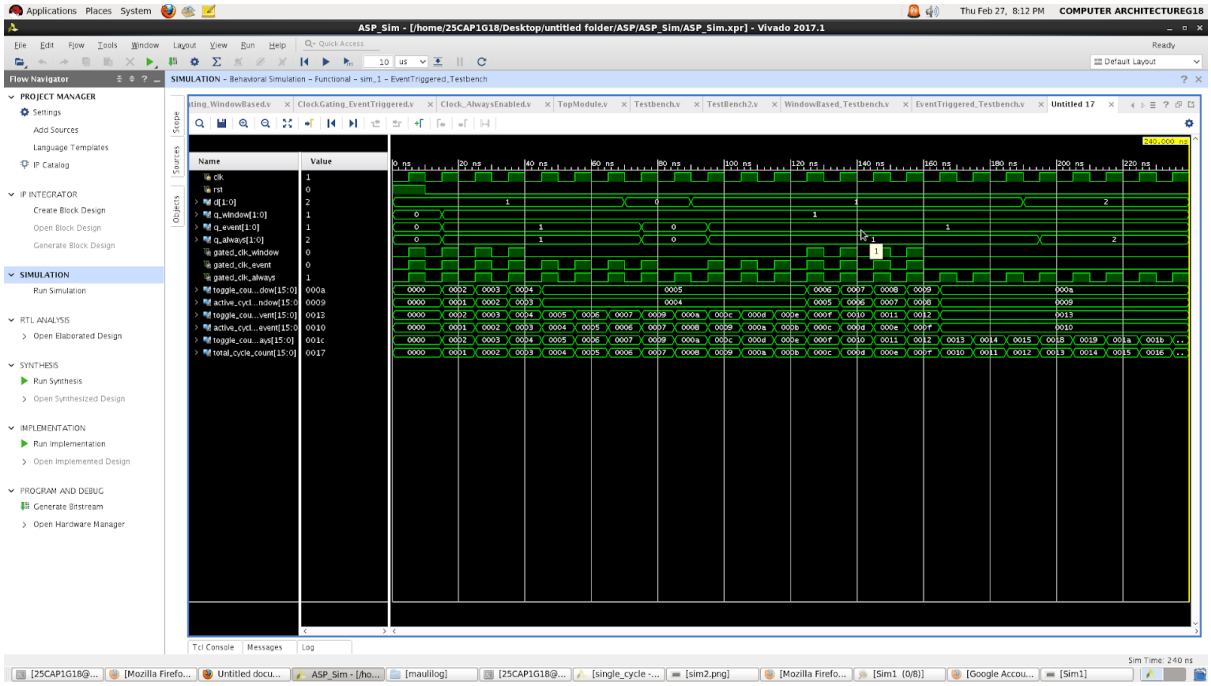
$$\alpha_{window} = \frac{23}{38 \times 3} = 0.20175$$

$$\alpha_{event} = \frac{37}{38 \times 3} = 0.324$$

$$\alpha_{no\ gating} = \frac{64}{38 \times 3} = 0.5614$$

Here the activity factor is drastically reduced by the window-based gating approach as compared to the non-gated clock, because of the low variation nature of the input load.

# Simulation #3 (Focus on effectiveness of event-triggered stochastic clock gating):




Here:

$$\alpha_{window} = \frac{10}{24 \times 3} = 0.1388$$

$$\alpha_{event} = \frac{19}{24 \times 3} = 0.2638$$

$$\alpha_{no\ gating} = \frac{28}{24 \times 3} = 0.3888$$

Here we provided generally non-variant load with occasional bursts of activity. The low activity of the window-based gating suggests that it was not able to capture the bursts of activity, whereas the event-triggered clock captured the bursts of activity and yet managed to reduce the activity factor compared to the non-gated clock thus enabling significant amount of power savings in an unpredictable circuit.

# CONCLUSION

This study offers a mathematical framework for employing stochastic clock gating approaches to analyse and optimise dynamic power dissipation in digital circuitry. We were able to determine the steady-state probabilities π and calculate the predicted activity factor α of a 2-bit PIPO register by expressing its state transitions as a Markov chain. By quantifying bit transitions using the Hamming distance metric $H(i, j)$, we were able to use a rigorous probabilistic model to estimate power dissipation.

We added a parametric gating threshold $\alpha_{th}$ proportional to the activity factor $\alpha$ in order to perform stochastic clock gating. Two distinct gating mechanisms—window-based clock gating and event-triggered clock gating—were simulated and examined using this threshold. We demonstrated the abilities of both gating techniques in power optimisation by using numerical computation to confirm that the activity factor was substantially lower in both when compared to an always-enabled clock.

According to the findings, event-triggered gating works better for workloads with irregular activity bursts, whereas window-based clock gating works well for organised workloads with predictable activity patterns.

I would like to add that the circuits designed, do bring with them additional overhead hardware costs and additional computational power, but we believe that the power saved due to the gating methods far outweighs the power consumed to power the control logic for the developed gating methods.

This work's mathematical underpinnings guarantee that the methodology is scalable to bigger systems, which makes it suitable for AI accelerators, low-power VLSI design, and IoT edge devices.

# FURTHER ADVANCEMENTS AND OPPORTUNITIES TO INCORPORATE RANDOMNESS INTO THE MODEL

This model was made using deterministic transition probabilities for the states of the 2-bit PIPO register to reduce the computational complexity of the hardware used in the model circuit. But to make it fit into real world scenarios, these transition probabilities could be sampled from discrete or continuous probability distributions (e.g. Poisson process, Binomial distribution).

We can also model the power saving factor k by making it a random variable instead of a fixed constant. k can vary dynamically based on higher level control circuits or can follow a probability distribution (e.g Exponential distribution) to model varying power saving requirements based on the state of the circuit as a whole or the demands of the user operating the circuit.

We can also implement hybrid models incorporating both techniques of clock gating if we have the information of the activity of workload (apply window-based gating on low-varying workloads and apply event-triggered gating on unpredictable loads with bursts of activity)

And as a cherry on top, machine learning techniques can be used to integrate different dependent and independent input workloads, varying power saving factor k, and multi-level gated circuits and other advancements can be made to these methods.

# REFERENCES

Github repository for the simulation code and modelled circuit:

https://www.sciencedirect.com/topics/computer-science/dynamic-power-dissipation

https://www.geeksforgeeks.org/difference-between-combinational-and-sequential-circuit/

https://www.geeksforgeeks.org/pipo-shift-register/

https://anysilicon.com/the-ultimate-guide-to-clock-gating/

https://madebyevan.com/fsm/

https://www.researchgate.net/publication/282429454_An_application_of_Markov_chains_in_digital_communication

https://www.ee.iitb.ac.in/~microel/download/markov.html

https://schaumont.dyn.wpi.edu/ece574f24/09power.html