Desarrollo de algoritmos para estimación de factores de actividad y consumo de potencia dinámica para circuitos digitales utilizando técnicas basadas en grafos

1st J. A. Franchi
Universidad de Costa Rica,
LIMA-EIE
email jose.franchichongkan@ucr.ac.cr

***Resumen*—En este trabajo, se presenta el desarrollo de algoritmos para la estimación de la potencia dinámica y los factores de actividad en circuitos digitales utilizando técnicas basadas en grafos. El enfoque implementado fue la construcción de un grafo dirigido acíclico (GDA) para representar la topología del circuito y mapearlo a una representación abstracta, aplicando un ordenamiento topológico al grafo se logra establecer la dirección de propagación dentro del circuito, lo cual permite calcular las probabilidades de conmutación y los factores de actividad de cada celda según se van propagando en el grafo. Se estimó el consumo de potencia dinámica para diferentes netlists, incluyendo sumadores de 1 bit y sumadores con pipeline de 8 bits, bajo distintas condiciones de frecuencia y tensión de alimentación.**

Los resultados obtenidos demuestran que el algoritmo es efectivo para estimar el consumo de potencia dinámica en topología simples. Sin embargo, se identificaron limitaciones al aplicar el algoritmo a circuitos más complejos, como la CPU PicoRV32, debido a la presencia de ciclos en el grafo que impiden el ordenamiento topológico necesario.

Este trabajo contribuye al desarrollo de herramientas para la estimación temprana de potencia en el diseño de circuitos digitales mediante un modelado teórico y generalizable. La extensión del algoritmo para manejar circuitos secuenciales más complejos y la integración de técnicas que permitan gestionar ciclos en el grafo son áreas prometedoras para continuar con la investigación en el área.

***Index Terms*—potencia dinámica, factor de actividad, teoría de grafos, Python, NetworkX**

I. Introducción

En las últimas décadas, la complejidad de los circuitos integrados ha sido de forma creciente, y la necesidad de reducir y optimizar el consumo de potencia en los dispositivos es una tarea de mucha relevancia.

Un concepto clave en este contexto es la es la potencia dinámica, que representa la energía disipada durante las transiciones de los estados lógicos de un circuito. Esta depende del factor de actividad, el cual mide la frecuencia de cambios en las señales digitales en relación con su capacidad de carga y voltaje de operación. La teoría de grafos establece herramientas matemáticas para analizar la estructura de las interconexiones de circuitos y realizar estimaciones sobre datos como la potencia dinámica.

A continuación se sintetiza información sobre trabajos dedicados al avance en tales estimaciones. Los cuales se deben tener en cuenta para avanzar con el objetivo de este proyecto: desarrollar algoritmos con técnicas basadas en grafos para la estimación de factores de actividad y consumo de potencia para circuitos digitales.

II. Trabajos relacionados

En el área de estimación de potencia en circuitos VLSI, Najm [1] presenta una revisión exhaustiva de técnicas que se han propuesto para resolver los problemas de dependencia en las entradas de circuitos. Su trabajo se centra en una comparativa entre las técnicas probabilistas y técnicas estadísticas, donde en los enfoques probabilísticos que utilizan modelos de probabilidad para estimar la densidad de transiciones, es decir, la frecuencia con la que las señales cambian de estado en un nodo del circuito. Estos métodos permiten una estimación más eficiente de la potencia, pero a menudo requieren suposiciones sobre las características espaciales y temporales de las señales, lo que puede reducir la precisión en ciertos casos. Además, Najm destaca que el consumo de potencia es un problema dependiente de la actividad de conmutación o switching, lo que implica que las técnicas de simulación tradicionales, aunque precisas, pueden ser computacionalmente costosas debido al alto número de patrones de entrada necesarios.

Pedram [2] describe como para la estimación de potencia se puede realizar a diferentes niveles como a nivel de software, de comportamiento, en donde en este se incluyen modelos basados teoria de información, modelos basados en complejidad y modelos basados en síntesis, además a nivel de RTL (Register-Transfer Level), a nivel de compuertas y a nivel de transistor.

El trabajo de Bryant [3] presenta un desarrollo para la manipulación de funciones booleanas con el estudio de algoritmos basados en grafos con la introducción de los Diagramas de Decisión Binaria Ordenada (OBDD), este método revoluciona la forma en que las funciones booleanas pueden ser representadas y manipuladas, y otorgando una estructura gráfica compacta que permite reducir la complejidad computacional.

Con el uso de OBDD se puede estimar con mayor precisión la densidad de conmutación o switching, un factor clave en el consumo de potencia.

III. Metodología

Este estudio consistió en dos etapas: primero el modelado de la potencia dinámica y posteriormente el desarrollo del algoritmo para la estimación del consumo de la misma. Ambos que se describen en las siguientes secciones.

III-A. Modelado de la potencia dinámica

La disipación de potencia en circuitos CMOS proviene de dos componentes de la potencia estática y la dinámica, en este trabajo se enfoca en el consumo de potencia dinámica, donde la potencia dinámica y su disipación se puede modelar con la potencia de conmutación (P_{sw}) debido a la carga y descarga de las capacitancias de carga cuando las compuertas conmutan, y la potencia de "corto circuito"(P_{sc}): [4].

$$P_d = P_{sw} + P_{sc} \tag{1}$$

En este trabajo, para la estimación del consumo de potencia dinámica se modela de forma exclusiva con la potencia de conmutación se puede modelar con la ecuación: [4]

$$P_{sw} = \alpha \cdot C \cdot V_{DD}^2 \cdot f \tag{2}$$

$$\implies P_d \approx P_{sw} \tag{3}$$

$$P_d = \alpha \cdot C \cdot V_{DD}^2 \cdot f \ . \tag{4}$$

Donde:

- f: frecuencia de conmutación de las compuertas en el Netlist.
- V_{DD}: Tensión de alimentación.
- C: Capacitancia de la carga de salida de las compuertas en el Netlist.
- α: es el factor de actividad, depende de la probabilidad de conmutación de cada compuerta.

El factor de actividad se modela mediante la ecuacion:

$$\alpha_Y = P_Y \cdot \overline{P_Y} \tag{5}$$

Donde:

- P_Y: la probabilidad de conmutación de una compuerta lógica
- $\overline{P_Y}$: la probabilidad de conmutación negada o complementaria de una compuerta lógica.
  Se debe cumplir que $\overline{P_Y} = 1 - P_Y$ y $P_Y + \overline{P_Y} = 1$
  La probabilidad de conmutación consiste en la probabilidad de que ocurra una transición de estado de $0 \rightarrow 1$ o $1 \rightarrow 0$ en una celda durante un ciclo de reloj, debe ser un valor que se encuentre entre 0 y 1. [4] Los valores de la probabilidad de conmutación dependen de la topología de cada tipo de celda. Se utiliza la biblioteca `sky130_fd_sc_hd - SKY130 High Density Digital Standard Cells` [5] [6], la cual posee 194 celdas, en este trabajo se trabajo y codifica las funciones para la probabilidad de conmutación de exclusivamente 73 celdas de la biblioteca, ya que eran las celdas necesarias para trabajar los 3 distintos tipos de netlists estudiados en este trabajo.
  La implementación de las probabilidades de conmutación de la lib `sky130_fd_sc_hd` se realizo en la archivo `get_activity_factor.py`, dentro de la clase `LogicGateActivity` y su método o función `calc_switching_probability` donde a partir del parámetro que identifica el tipo de celda o compuerta, y sus probabilidades de entrada, las cuales se propagan a traves del netlist, se calcula la probabilidad de conmutación de salida de la compuerta.
  Cuando las probabilidades de conmutación de entrada son de entradas primarias **(PI: Primary Inputs)**, se le asigna un valor de probabilidad de conmutación $P = 0{,}5$, dado que que poseer solamente dos estados posibles 0 o 1, para datos complementamente al azar, la probabilidad de transición ya sea del estado de $0 \rightarrow 1$ o del estado de $1 \rightarrow 0$, es un 50 % para cualquier caso. [4]
  A partir del valor de la probabilidad de conmutación de salida de la compuerta que se obtiene con el método `calc_switching_probability`, se calcula el factor de actividad con el uso del método `activity_factor`.

IV. El Algoritmo

El algoritmo principal para la estimación del consumo de potencia dinámica consiste en seis partes fundamentales, cada una implementada mediante sub-algoritmos especializados. Estos sub-algoritmos manejan: (1) la construcción del grafo que representa el circuito, (2) el ordenamiento topológico de dicho grafo, (3) el cálculo de probabilidades de conmutación y factores de actividad para cada celda del netlist considerando la propagación de señales a través del grafo, (4) la estimación de potencia dinámica individual por celda, (5) el cálculo de la potencia total del circuito, y (6) la generación de reportes detallados con los resultados obtenidos.

IV-A. Algoritmo principal

El algoritmo principal 1 implementa el siguiente enfoque para la estimación de potencia dinámica en circuitos digitales, está dividido en seis partes. Este algoritmo toma como entrada: un netlist en formato Verilog, información de fanout en formato CSV, y datos de capacitancias de la biblioteca de celdas estándar.
En la primera etapa, el algoritmo realiza la importación y procesamiento de datos. Esto con el fin de asegurar una representación precisa y fiel de las conexiones del circuito. En esta fase se crean de estructuras de datos optimizadas para el acceso eficiente a la información de requerida.
La segunda etapa (sub-algoritmo) implementa la elaboración de un grafo dirigido acíclico (GDA), que representa la topología del circuito digital. En el grafo se mapean las

relaciones de dependencia entre las diferentes celdas del circuito, donde los nodos representan las celdas lógicas y las aristas representan las conexiones entre ellas. La estructura del grafo es indispensable para el análisis posterior, sobre la propagación de las probabilidades de conmutación.

En la tercera fase (sub-algoritmo), se hace un ordenamiento topológico del grafo. Con el objetivo de garantizar que el cálculo de las probabilidades de conmutación se realice en el orden correcto, respetando las dependencias de las celdas entre sí, es que se lleva a cabo este paso. Ya que el ordenamiento asegura que todas las entradas de una celda sean procesadas antes de calcular su probabilidad de conmutación y no haya ciclos.

La cuarta etapa, o sub-algoritmo, constituye la parte central del algoritmo. En ella se calculan las probabilidades de conmutación y los factores de actividad. Esto sigue el ordenamiento topológico, comenzando desde las entradas primarias y propagando estos valores a través del grafo.

En la quinta etapa (sub-algoritmo), el algoritmo calcula la potencia dinámica para cada celda utilizada de manera individual, pero siguiendo la propagación dentro del grafo. Este cálculo utiliza la ecuación 4, basado en el modelado previamente explicado en la sección III-A.

Finalmente, el sexto sub-algoritmo suma los resultados obtenidos de manera individual para obtener la potencia dinámica total del circuito. En esta fase se logra así generar un reporte y archivos de salida en formatos CSV y TXT. Este reporte incluye información desglosada por celda del circuito.

Algorithm 1 Algoritmo principal para la estimación de Potencia Dinámica

Inputs:
netlist: Archivo Verilog con la descripción del circuito
fanout: Archivo CSV con información de conexiones
lib_pins: Archivo CSV con información de capacitancias

Outputs:
total_power: Potencia dinámica total del circuito
power_results: Resultados detallados por celda
reportes: Archivos CSV y TXT con resultados

function MAIN(*netlist*, *fanout*, *lib_pins*)
▷ Paso 1: Importar y preparar datos
$lib\_data \leftarrow$ LeerLibFile(lib_pins)
$cap\_dict \leftarrow$ ProcesarLibData($library\_data$)
$netlist\_data \leftarrow$ LeerFanoutFile($fanout$)
$cell\&pins\_mappings \leftarrow$ ParseNetlist($netlist$)

▷ Paso 2: Construir el grafo (DAG) - Sub-algoritmo
$G \leftarrow$ ConstruirGrafo()

▷ Paso 3: Realizar ordenamiento topológico - Sub-algoritmo
$nodos\_ordenados \leftarrow$ OrdenTopologico(G)

▷ Paso 4: Calcular probabilidades y factores de actividad - Sub-algoritmo
$G \leftarrow$ CalcularActividadNodo(G, *nodo*)

▷ Paso 5: Calcular potencia dinámica por celda - Sub-algoritmo
$power\_results \leftarrow$ CalcPotencia(G, cap_dict)

▷ Paso 6: Calcular potencia total y generar reporte - Sub-algoritmo
$total\_dynamic\_power \leftarrow$ Sum($power\_results$)
GenerarReporte($netlist$, $fanout$, $total\_dynamic\_power$, $power\_results$)
return $total\_dynamic\_power$,
end function

IV-B. Sub-algoritmo de construccion de grafo

El sub-algoritmo de construcción del grafo dirigido acíclico (DAG) 2 constituye una etapa fundamental en el proceso de estimación de potencia dinámica, para esto se utilizan los métodos de la biblioteca NetworkX en Python [7]. Este algoritmo transforma la descripción estructural del circuito en una representación en forma de grafo que captura las relaciones de dependencia entre las diferentes celdas lógicas y elementos secuenciales.

El algoritmo opera sobre tres tipos fundamentales de elementos: entradas primarias (PI), elementos secuenciales (flip-flops) y celdas combinacionales. El algoritmo establece una representación nodal específica que preserva la estructura del netlist del circuito original para cada elemento. Las entradas primarias se representan mediante nodos que mantienen el nombre del cable original.

Los flip-flops se modelan mediante pares de nodos que representan sus terminales D y Q, esto se hace para evitar ciclos en el grafo debido a los flip-flops. Por otra parte, las celdas combinacionales son representadas como nodos individuales, con sus correspondientes conexiones de entrada y salida.

La construcción del grafo procede de manera iterativa, procesando cada entrada del netlist secuencialmente. Para cada entrada, el algoritmo primero establece el nodo fuente (*source*) y luego procesa todos sus nodos destino (*sinks*). Esta estructura permite y garantiza que todas las conexiones del circuito sean capturadas fielmente en el grafo resultante.

Un aspecto primordial del algoritmo es el manejo diferenciado de los elementos secuenciales. Para los flip-flops, el algoritmo crea una representación especial que separa explícitamente las entradas (D) de las salidas (Q). Lo cual es necesario para el posterior análisis de la propagación de señales y el cálculo de factores de actividad. Hacer esto permite que se pueda realizar el ordenamiento topológico del circuito, lo implica que se podría realizar la propagación en el grafo.

La estructura de datos que resulta de lo anterior es un grafo dirigido acíclico, donde los nodos representan elementos funcionales del circuito (PI, FF, celdas lógicas). Sus aristas representan las conexiones físicas entre elementos, los atributos de los nodos almacenan información del tipo de celda y características eléctricas. Además, la dirección de las aristas indica el flujo de señal en el circuito.

Unido a ello, el algoritmo incluye una parte adicional y opcional de visualización. Con la cual es posible la inspección visual del grafo construido, facilitando la verificación de la correcta representación del circuito y la depuración del proceso de análisis de potencia.

Algorithm 2 Sub-algoritmo: Construir el Grafo Dirigido Acíclico (DAG)

ConstruirGrafo
Inicializar grafo dirigido vacío G
for all $entry$ en $netlist\_data$ **do**
    Obtener $source$ y $sinks$ de $entry$ ▷ Determinar el nodo fuente
    if $source$ es una entrada primaria (*PI*) **then**
        $source\_node \leftarrow$ Nombre del *wire*
    else if $source$ es un flip-flop **then**
        $source\_node \leftarrow$ Salida Q del flip-flop
    else
        $source\_node \leftarrow$ Instancia de la celda
    end if
    Agregar $source\_node$ al grafo G con su tipo de celda
    ▷ Procesar los nodos destino
    for all $sink$ en $sinks$ **do**
        if $sink$ es vacío **then**
            continuar
        end if ▷ Determinar el nodo sink
        if $sink$ es un flip-flop **then**
            $sink\_node \leftarrow$ Entrada D del flip-flop
        else
            $sink\_node \leftarrow$ Instancia de la celda
        end if
        Agregar $sink\_node$ al grafo G con su tipo de celda
        Agregar arista desde $source\_node$ a $sink\_node$ en G
    end for
end for
return $Grafo$
▷ Adicional y Opcional
VisulizarGrafo(G)

IV-C. Sub-algoritmo de ordenamiento topológico de grafo

El sub-algoritmo de ordenamiento topológico 3 es un componente crítico en el proceso de estimación de potencia dinámica. Pues es el que organiza los nodos del grafo en una secuencia lineal, que respeta todas las dependencias del circuito. Este ordenamiento es esencial para garantizar que el cálculo de las probabilidades de conmutación y los factores de actividad se realice en el orden correcto de propagación de señales.

El algoritmo implementa de nuevo los métodos de la biblioteca NetworkX en Python [7] mediante el cual se produce el ordenamiento deseado. Adicionalmente se detecta la presencia de ciclos en el grafo. Esta capacidad de detección es de gran importancia, ya que la presencia de ciclos en el grafo (excluyendo los ciclos naturales en elementos secuenciales) indicaría un error en la descripción del circuito o en su interpretación.

El ordenamiento resultante tiene varias propiedades importantes:

1. Garantiza que para cada arista (u, v) en el grafo, el nodo u aparece antes que v en el ordenamiento.

2. Asegura que los cálculos de probabilidad de conmutación pueden realizarse secuencialmente sin iteraciones.
3. Optimiza el proceso de propagación de señales al minimizar las dependencias hacia atrás.

En esta parte, el algoritmo también incluye una parte de visualización opcional. Con la cual es posible verificar visualmente el ordenamiento obtenido. Esto es particularmente útil durante la fase de desarrollo y verificación del análisis de potencia. En esta visualización se representan los nodos en una disposición lineal, reflejando explícitamente el orden de propagación de señales en el circuito.

Algorithm 3 Sub-algoritmo: Realizar Ordenamiento Topológico del Grafo

```
function OrdenTopologico(G)
    topo_order ← OrdenamientoTopológico(G) Exception GrafoCíclico
    Reportar error debido a ciclos
    ciclos ← DetectarCiclos(G)
    Mostrar ciclos
    detener
    return topo_order                          ▷ los nodos ordenados
                                               ▷ Adicional y Opcional
    VisulizarGrafoOrdenado(G)
```

IV-D. Sub-algoritmo: Cálculo de probabilidades de conmutacion y factores de actividad según su propagación en el grafo

El sub-algoritmo para el cálculo de probabilidades de conmutación y factores de actividad 4 corresponde a la parte central y fundamental en el proceso de estimación de potencia dinámica. En esta sección, el algoritmo implementa la propagación de las probabilidades de conmutación a través del circuito de manera sistemática y siguiendo el ordenamiento topológico que se estableció anteriormente.

El proceso se divide en dos partes: la inicialización de los valores de las entradas primarias y la propagación de probabilidades. En la primera parte, para todas las entradas primarias (PI) se inicializan con una probabilidad de conmutación de 0.5. Lo cual corresponde al caso de una señal completamente aleatoria, ergo resultando en un factor de actividad de 0.25.

La segunda parte implementa la propagación de probabilidades siguiendo el ordenamiento topológico del circuito. Con esta propagación secuencial se pretende garantizar que todas las entradas de una celda han sido procesadas antes de calcular su probabilidad de conmutación.

Para cada nodo del circuito, el algoritmo:

- Determina el tipo de elemento (combinacional o secuencial)
- Recopila las probabilidades de sus entradas.
- Calcula la probabilidad de conmutación según la función lógica específica para el tipo celda.
- Calcula el factor de actividad correspondiente.

El resultado final es un grafo donde cada nodo tiene asignada una probabilidad de conmutación y un factor de actividad. Esta información es indispensable para la subsiguiente estimación de potencia dinámica por celda.

Algorithm 4 Sub-algoritmo: Cálculo de Probabilidades y Factores de Actividad

```
AsignarProbabilidades        ▷ Inicializar entradas primarias (PI)
for all nodo en Grafo do
    if nodo es PI then
        Asignar probabilidad de conmutación = 0.5
        Asignar factor de actividad = 0.25
    end if
end for
        ▷ Propagar probabilidades según orden topológico
for all nodo en orden_topologico do
    if nodo ya tiene probabilidad asignada then
        continue
    end if
    ▷ Calcular probabilidades tipo de nodo uso métodos de get_activity_factor
    if nodo es flip-flop then
        Obtener probabilidad de entrada D
    else
        Obtener probabilidades de entradas
        Calcular probabilidad según función lógica
    end if
                ▷ Calcular y asignar factor de actividad
    Calcular factor = prob × (1 - prob)
    Asignar probabilidad y factor al nodo
end for
return Grafo actualizado probas y factores actividad =0
```

IV-E. Sub-algoritmo: Cálculo de la potencia dinámica para cada salida de celda según su propagación en el grafo

El sub-algoritmo de cálculo de potencia dinámica 5 implementa la estimación de consumo de potencia para cada celda del circuito. Para lograrlo se basa en los factores de actividad previamente calculados y los otros parámetros necesarios. Esta parte consiste en la fase final del proceso de estimación, donde se convierten los factores de actividad en valores concretos de potencia, basándose en el modelado de la sección III-A.

Este proceso inicia definiendo los parámetros tecnológicos fundamentales: un voltaje de alimentación de 1.8V, característico de la tecnología sky130 [5] y una frecuencia de operación de 100MHz. La última es estimada a partir de los valores de operación para otras libs de sky130 [5], que se considera un escenario aceptable de funcionamiento.

La estimación de potencia se realiza mediante la aplicación sistemática de la ecuación fundamental de potencia dinámica 4. El algoritmo procesa cada nodo del grafo de manera secuencial. En este procesamiento se excluyen explícitamente las entradas primarias (PI) que no contribuyen al consumo de potencia interno del circuito. Para cada celda, el proceso implica:

- La recuperación del factor de actividad previamente calculado y almacenado en el nodo.
- La obtención de la capacitancia específica de la celda según su tipo y configuración.
- El cálculo de la potencia dinámica aplicando la ecuación.
- El almacenamiento del resultado para análisis posteriores.

Un aspecto que no se debe dejar de lado del algoritmo es su manejo de las capacitancias asociadas a cada celda lógica. Estas son obtenidas directamente de la biblioteca de valores suministrados al programa. De esta manera se busca que las estimaciones reflejen las características físicas del proceso de fabricación, lo mejor posible.
Considerando lo anterior, la precisión de las estimaciones depende principalmente de la exactitud de los factores de actividad calculados de manera previa y de la fidelidad de los datos de capacitancia obtenidos de la biblioteca de celdas. El algoritmo asume una aproximación de relación lineal entre estos parámetros.

Algorithm 5 Sub-algoritmo: Cálculo de Potencia Dinámica por Celda

```
CalcPotenciaDinamica                          ▷ Parámetros tecnológicos
Definir voltaje de alimentación V_DD = 1,8V
Definir frecuencia de operación f = 100MHz
                                              ▷ Cálculo de potencia por celda
for all nodo en Grafo do
    if nodo es entrada primaria (PI) then
        continue
    end if
                                              ▷ Obtener parámetros de la celda
    Obtener factor de actividad del nodo
    Obtener capacitancia según tipo de celda
                                              ▷ Calcular potencia dinámica
    P_d = α × C × V_DD^2 × f
    Almacenar resultado para el nodo
end for
return resultados de potencia por celda
```

IV-F. Sub-algoritmo: Cálculo de la potencia dinámica total del grafo y generación de reporte

El sub-algoritmo de cálculo y reporte de potencia total 6, es la ultima parte del proceso de estimación de potencia dinámica.
El proceso comienza con la suma de las potencias dinámicas calculadas para cada celda en el sub-algoritmo pasado, generando como resultado la potencia dinámica total del circuito, donde este valor es el objectivo principal del algoritmo. Además que es un valor crítico para evaluar si el diseño cumple con las especificaciones de consumo energético y potencialmente para identificar posibles áreas de mejora en el diseño del netlist.
Un aspecto clave de este sub-algoritmo es la generación de reporte, que permite acceder a información detallada y organizada sobre el consumo de potencia, donde la eficacia de este algoritmo depende de la precisión de los cálculos previos y de la integridad de los datos manejados.

Algorithm 6 Sub-algoritmo: Cálculo y Reporte de Potencia Total

```
CalcPotenciaTotal
total_power ← Σ_i power_results[i]
SaveResults(power_results, 'resultados_potencia.csv')
GenerateReporte(total_power, power_results)
    return total_power
```

V. Resultados y Discusión

A partir del algoritmo desarrollado, se estimaron valores de consumo de potencia dinámica para los netlists del sumador de 1 bit y el sumador con pipeline de 8 bits, donde los valores utilizados por defecto en el código son una frecuencia de 100 MHz y una tensión de alimentación de 1.8 V. Los resultados se muestran en el cuadro I.
Para el netlist de la CPU PicoRV32, no se logró implementar de forma satisfactoria el algoritmo debido a la presencia de ciclos en el grafo construido, lo que impidió el ordenamiento topológico necesario para el cálculo de las probabilidades de conmutación. Este problema sugiere limitaciones en el sub-algoritmo de construcción del grafo (2) al manejar circuitos secuenciales complejos, donde las dependencias cíclicas son inherentes al diseño.

Cuadro I: Estimaciones de consumo de potencia dinámica

Netlist	**P Din. (W) - @1V**	**P Din. (W) - @1.8V**
Netlists con frecuencia de conmutación de 100 Mhz		
Sumador (1 bit)	$2{,}434 \times 10^{-5}$	$7{,}887 \times 10^{-5}$
Sumador con Pipeline (8 bits)	$3{,}381 \times 10^{-4}$	$1{,}096 \times 10^{-3}$
Netlists con frecuencia de conmutacion de 66 Mhz		
Sumador (1 bit)	$1{,}607 \times 10^{-5}$	$5{,}206 \times 10^{-5}$
Sumador con Pipeline (8 bits)	$2{,}232 \times 10^{-4}$	$7{,}231 \times 10^{-4}$
Netlists con frecuencia de conmutacion de 110 Mhz		
Sumador (1 bit)	$2{,}678 \times 10^{-5}$	$8{,}676 \times 10^{-5}$
Sumador con Pipeline (8 bits)	$3{,}719 \times 10^{-4}$	$1{,}205 \times 10^{-3}$

Los resultados presentes en el Cuadro I muestran como el consumo de potencia dinámica se incrementa significativamente al aumentar el voltaje de alimentación. Un ejemplo de ello se nota para el sumador de 1 bit a 100 MHz, la potencia se incrementa de $2{,}434 \times 10^{-5}$ W a $7{,}887 \times 10^{-5}$ W al aumentar el voltaje de 1V a 1.8V.

Esto representa un incremento de más del triple. Debido a la dependencia cuadrática de la potencia con respecto a la tensión, según la ecuación 4.

Por otra parte, el consumo de potencia es mayor para el sumador con pipeline de 8 bits comparado con el sumador de 1 bit. Como es lo esperado, este resultado se debe a que el circuito con pipeline es mucho más complejo y tiene un mayor número de celdas y nodos que contribuyen al consumo de potencia.

Para los valores 1.8 V y 100 MHz, el consumo de potencia obtenido es de $7{,}887 \times 10^{-5}$ W para el sumador de 1 bit y $1{,}096 \times 10^{-3}$ W para el sumador de 8 bits. Lo que equivale a un incremento de aproximadamente 14 veces en el consumo de potencia. Tendencia debida a la influencia directa de la complejidad y tamaño del circuito en el consumo energético, tal como se mencionó anteriormente.

Los resultados obtenidos para el el sumador de 1 bit fueron validados con el cálculo analítico para dicho netlist. Para este caso se valida la eficacia del algoritmo propuesto para estimar el consumo de potencia dinámica en circuitos simples.

Sin embargo, la dificultad para manejar circuitos más complejos destaca la necesidad de mejorar el algoritmo. Particularmente en la gestión de ciclos en el grafo y en el manejo de elementos secuenciales que introducen retroalimentación.

VI. Conclusiones

Los resultados obtenidos muestran que el algoritmo es capaz de estimar de manera efectiva el consumo de potencia dinámica en diferentes netlists simples. La consistencia y verificación entre los resultados obtenidos con cálculos teóricos y las estimaciones del algoritmo realizadas validan parcialmente la precisión del método propuesto.

Sin embargo, se lograron identificar limitaciones al aplicar el algoritmo a circuitos de mayor complejidad, como la CPU PicoRV32. La presencia de ciclos en el grafo impidió el ordenamiento topológico necesario para el cálculo de las probabilidades de conmutación, lo que sugiere que el algoritmo en su forma actual no es capaz de manejar adecuadamente diseños con retroalimentación inherente.

Se puede concluir que el método implementado constituye una herramienta útil para la estimación temprana del consumo de potencia dinámica en circuitos digitales de complejidad simple o moderada, si se desea ampliar su aplicabilidad, se recomienda explorar mejoras en la construcción y el manejo de ciclos en el grafo, particularmente con las celdas lógicas secuenciales.

Agradecimientos

Se desea dar un agradecimiento a la Escuela de Ingeniería Eléctrica de la Universidad de Costa Rica, por generar el marco en el cual se desarrolla el proyecto expuesto. Se brinda además un agradecimiento al docente e investigador E. Carvajal por su guía y seguimiento en el avance del mismo, así como al profesor José David Rojas.

Adicionalmente, se agradece por su apoyo incondicional a Meli y Maggie, sin ellas este proyecto no hubiera podido llevar a buen puerto. También a los amigos que no me conocen, pero aún así compartimos muchas noches en vela: Lex, Andrew, Joe, Jocko y Tim, sin su compañía y motivación este proyecto no hubiera sido posible.

Referencias

[1] F. N. Najm, "A survey of power estimation techniques in vlsi circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, 1994, dOI:10.1109/92.335013.

[2] M. Pedram, "Power simulation and estimation in vlsi circuits," 1999.

[3] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, 1986, dOI:.

[4] N. H. E. Weste and D. Harris, *CMOS VLSI design : A Circuits and Systems Perspective*, 4th ed. Massachusetts:: Pearson Education, 2011.

[5] Google and SkyWater Technology Foundry, "Skywater open source PDK," 2024, disponible en: https://skywater-pdk.readthedocs.io/. [Online]. Available: https://skywater-pdk.readthedocs.io/

[6] Google and SkyWater Technology Foundry, "SKY130 high density digital standard cells," GitHub repository, 2024, accedido: 20-10-2024. [Online]. Available: https://github.com/google/skywater-pdk-libs-sky130_fd_sc_hd

[7] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.