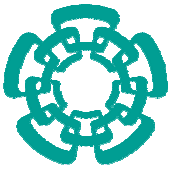


CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL IPN

UNIDAD ZACATENCO

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SECCIÓN DE ELECTRÓNICA DEL ESTADO SÓLIDO

Sistema Analógico Paralelo en Tecnología CMOS para la Solución del Problema de Asignación

Tesis que presenta

Ing. Lizeth González Carabarín

Para obtener el grado de

Maestra en Ciencias

en la especialidad de

Ingeniería Eléctrica

Directores de Tesis

Dr. Felipe Gómez Castañeda
Dr. José Antonio Moreno Cadenas

México D.F.,
Noviembre 2009

INDICE

RESUMEN

Han sido amplias las aplicaciones del transistor MOS de compuerta flotante dentro del diseño analógico de circuitos integrados; en algunos casos, el FGMOSFET simplifica diseños que originalmente tenían un alto grado de complejidad debido a sus características inherentes. Por otro lado, trabajar con el FGMOSFET en la región de inversión débil ha permitido la implementación de funciones no lineales, como es el caso de la exponenciación, el logaritmo natural, potencias, entre otras, además de que es una de las técnicas más utilizadas hoy en día para el diseño de circuitos que operen a baja potencia. Estas ventajas han sido aprovechadas en este trabajo para la implementación de una red analógica que resuelve el *Problema de Asignación*, cuya solución esta relacionada con la implementación de funciones no lineales y entre sus aplicaciones se encuentra la optimización de la arquitectura de Conmutación de *Paquetes de Datos.*

ABSTRACT

Floating Gate Transistor has several applications on the analog circuit design area; in some cases FGMOS Transistor simplifies high complexity circuits because of its inherent properties. On the other side, working with FGMOS transistor on weak inversion regime has allowed non-linear functions implementation such as exponential function, natural logarithm function and so on. Moreover, working on weak inversion regime is one of the most useful techniques for low-power designs.

Theses advantages have been used in this work to implement an analog network that solves the *Assignment Problem*, whose solution is related with non-linear functions implementation, where the *Assignment Problem* can be applied to optimize the *Packet Switching Architecture.*

INTRODUCCIÓN GENERAL

Hasta hace algunos años se había predicho que el diseño de circuitos digitales sustituiría gradualmente al analógico, tareas que antes se resolvían con ayuda de circuitos analógicos ahora se podían resolver dentro del campo digital. Sin embargo, las necesidades actuales exigen circuitos cada vez mas complejos, más rápidos y más pequeños, es en ese punto donde para algunas aplicaciones el diseño puramente digital deja de ser útil. Actualmente son muchas las aplicaciones del diseño analógico (que ya forman parte de mas del 20% del mercado actual de diseño de circuitos integrados) entre las mas importantes se encuentran el diseño de sistemas neuromórficos y aplicaciones en comunicaciones que requieren velocidades de procesamiento mucho mayores, menor potencia consumida, así como un menor espacio en el chip que los actuales, lo cual muchas veces solo se logra a través de una solución analógica.

De esta forma, el objetivo de esta tesis es el diseño analógico de una red que resuelva el *Problema de Asignación*. Este circuito tiene aplicaciones en la arquitectura de redes de comunicaciones conocida como “Packet Switching” o paquetes conmutados utilizada para transmitir bloques de información, llamados paquetes, de un destino a otro. Sin embargo, a veces sucede que dos o más paquetes son asignados al mismo destino y es necesario realizar la asignación óptima de acuerdo a los requerimientos. Ya se han propuesto varias soluciones para este problema, que no resultan factibles pues el tiempo de procesamiento es grande además de utilizar sistemas muy complejos.

Como se verá en capítulos posteriores, el diseño analógico es bastante bueno para resolver algunas tareas que en el dominio digital resultarían complejas, sin embargo existen ciertas desventajas; uno de los problemas que se tiene que considerar seriamente es la variación de parámetros durante el proceso de fabricación y que pueden afectar drásticamente el desempeño general del circuito. Para evitar estas variaciones se abordarán algunos métodos propuestos para eliminar los problemas de desacoplamiento en los transistores, a través de una red de transistores encargados de programar los voltajes de umbral utilizando los métodos de inyección de electrones calientes y tuneleo de electrones.

En el Capítulo 1 se introduce a la teoría del transistor MOS de compuerta flotante, así como su uso en la implementación de funciones no lineales; en el Capítulo 2 se presenta el

planteamiento matemático del *Problema de Asignación*, así como la solución para implementarlo en circuitos electrónicos. El Capítulo 3 presenta la implementación CMOS de la solución matemática para el *Problema de Asignación*, así como los resultados de la simulación para matrices de dimensiones de 2x2, 4x4 y 8x8. En el Capítulo 4 se abordan distintos métodos para eliminar el desacoplamiento entre transistores; se da un enfoque principal al método de programación indirecta que permite la programación del voltaje de umbral sin la necesidad de usar una topología compleja. Finalmente, el Capítulo 5 presenta el diseño geométrico de una fuente de corriente con circuitos extras para la implementación del método de programación indirecta.

1.1 Introducción

Desde su surgimiento a finales de la década de los 60's [1], el FGMOSFET (Floating Gate MOSFET o MOSFET de compuerta flotante) ha jugado un papel muy importante en el desarrollo de diversas aplicaciones, tanto en el campo del diseño de circuitos digitales como en el de circuitos analógicos.

En un inicio, el FGMOSFET fue utilizado como elemento de almacenamiento [2,3] debido a su capacidad de retener carga en su compuerta flotante por largos periodos de tiempo, así, fue el elemento principal de memorias FLASH, EPROM y EEPROM´s a partir de la década de los 70´s. Mas adelante, se explotaría la estructura de compuerta flotante para resolver tareas que hasta ese momento requerían de una topología compleja, disminuyendo así la complejidad del circuito y por tanto la potencia consumida; debido a su analogía con la neurona biológica ha sido de interés usar al FGMOSFET como elemento de almacenamiento sináptico (memoria analógica) [4]; por la propiedad de sumar los voltajes de sus múltiples entradas ha tenido aplicaciones en el área de procesamiento de señales como bloque de suma, también en el diseño de Convertidores Digital-Analógico[5], en donde permite ponderar el valor de cada bit de entrada eliminando la necesidad de circuitos extras para realizar esta tarea; ha sido utilizado además como elemento de ajuste de offset en amplificadores operacionales [6], multiplicador de cuatro cuadrantes, en fuentes de corriente, entre otras aplicaciones[7]. Simultáneamente, se propondría el uso del FGMOSFET dentro del régimen de inversión débil para la implementación electrónica de funciones no lineales. [8].

Una de las características, que hasta hace algunos años hacía poco confiable realizar diseños analógicos usando FGMOSFET, era la incertidumbre de la carga inicial almacenada en su compuerta flotante; dicha carga es almacenada durante todo el proceso de fabricación y es un valor desconocido; con el fin de superar este inconveniente ya se han propuesto diversas técnicas para eliminar la carga inicial, las cuales serán abordados brevemente en las siguientes secciones.

Otra ventaja que presenta el FGMOSFET, es la capacidad de variar su voltaje de umbral a través de sus múltiples entradas, lo que permite trabajar en la región de operación óptima

del transistor con un amplio rango de niveles de entrada, sin la necesidad de utilizar circuitos para el ajuste de niveles. Los métodos de inyección de electrones calientes y tuneleo de electrones también permiten la modificación del voltaje de umbral a través de la modificación de la carga almacenada en la compuerta flotante, lo que permite tener valores muy exactos del voltaje de umbral para un conjunto de transistores FGMOSFET, permitiendo el diseño de circuitos mas precisos; además éstos dos últimos métodos permiten la implementación de sistemas adaptativos [9]. Los siguientes temas profundizan más en la teoría del transistor de compuerta flotante. Se describe su estructura así como las expresiones que rigen su funcionamiento.

1.2 Estructura del transistor MOS de compuerta flotante

La estructura del transistor MOS de compuerta flotante es una estructura de una o mas entradas que consiste en un MOS convencional con su compuerta aislada eléctricamente o "flotando"; por un lado se encuentra una capa de óxido delgada que la separa del canal y por el otro se encuentra una capa de SiO_2 que la separa de la segunda capa de polisilicio (Figura 1). Si hablamos de un FGMOSFET de múltiples entradas, los voltajes de entrada se encuentran acoplados capacitivamente a la compuerta flotante a través de las capacitancias de entrada, cuyo valor dependerá del área ocupada por éstas.

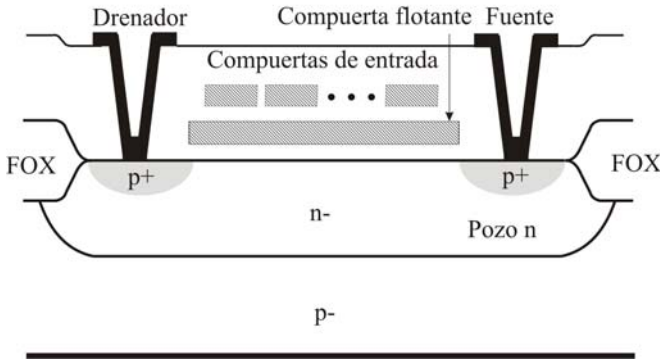


Figura 1.1 Corte transversal del transistor MOS de compuerta flotante de múltiples entradas

La figura 1.2 (a) muestra el diseño topológico para un transistor PMOS de compuerta flotante y la figura 1.2 (b) muestra su símbolo electrónico.

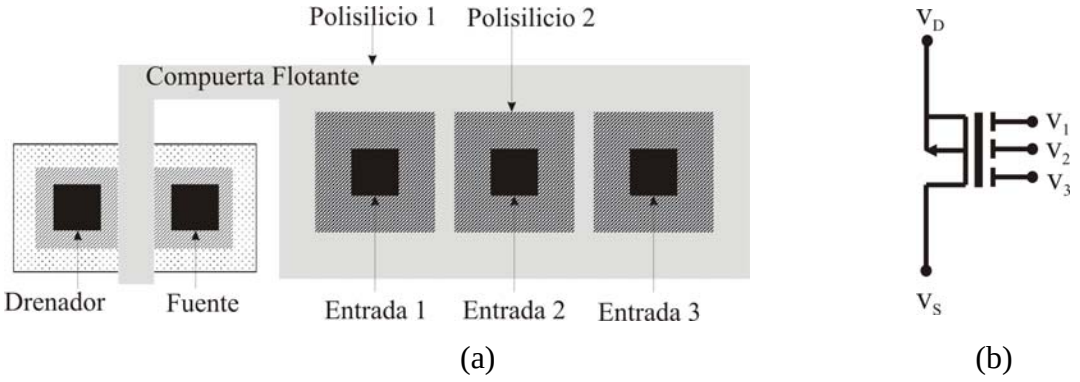


Figura 1.2 (a) Diseño topológico de un FGMOSFET de 3 entradas (b) Símbolo utilizado para representar un FGMOSFET canal p de 3 entradas.

1.3 Capacitancias asociadas al FGMOSFET

La figura 1.3 muestra todas las capacitancias asociadas al transistor MOS de compuerta flotante.

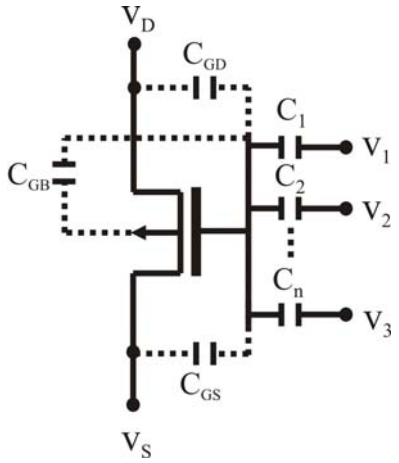


Figura 1.3 Capacitancias asociadas al FGMOSFET

La siguiente tabla presenta un listado de todas las capacitancias que intervienen en el modelo del FGMOSFET

Nomenclatura	**Significado**
C_n	Capacitancia de entrada.
C_{GD}	Capacitancia compuerta – drenador
C_{GS}	Capacitancia compuerta – fuente
C_{GB}	Capacitancia compuerta – substrato
W	Ancho de canal del transistor
L	Largo de canal del transistor
CGBO	Capacitancia de traslape de compuerta a substrato
CGSO	Capacitancia de traslape de compuerta a fuente
CGDO	Capacitancia de traslape de compuerta a drenador
C'_{ox}	Capacitancia de óxido por unidad de área

Tabla 1.1 Capacitancias asociadas al FGMOSFET

Para el cálculo de las capacitancias mencionadas en la tabla 1.1, las ecuaciones (1.1) a (1.6) expresan las capacitancias asociadas al FGMOSFET trabajando en las regiones de inversión fuerte e inversión débil.

- **Inversión débil**

$$C_{GB} = C'_{ox}(W)(L) + CGBO(L) + \frac{\varepsilon_{SiO_2}}{t^F_{SiO_2}}(A_E) \tag{1.1}$$

$$C_{GS} = CGSO(W) \tag{1.2}$$

$$C_{GD} = CGDO(W) \tag{1.3}$$

- **Inversión fuerte**

$$C_{GB} = CGBO(L) + \frac{\varepsilon_{SiO_2}}{t^F_{SiO_2}}(A_E) \tag{1.4}$$

$$C_{GS} = CGSO(W) + \frac{2}{3}C'_{ox}(W)(L) \tag{1.5}$$

$$C_{GD} = CGDO(W) \tag{1.6}$$

Donde ε_{SiO_2} es la permitividad del dióxido de Silicio, $t^F_{SiO_2}$ es el espesor de la capa de óxido de campo y A_E es el área ocupada por las nuevas entradas efectivas, que corresponde al área que ocupa la compuerta flotante fuera de la región activa y esta dada por

$$A_E = \sum_{i=1}^{n} A_i + SES1(n-1)L_{SE} + 2(SES)\sum_{i=1}^{n} W_{SEi} + 2(SES1)(SES)(n-1) + 4(SEL)(SES) \tag{1.7}$$

Donde A_i es el área efectiva de las compuertas de entrada y los parámetros SES, SES1 y SEL dependen de las reglas de diseño de la tecnología usada y se muestran en la tabla 1.2.

Parámetro	
SES	Separación entre el borde de la capa de polisilicio 2 al borde (superior o inferior) de la capa de polisilicio 1
SESn	Separación entre capas de polisilicio 2
SEL	Separación entre el borde de la capa de polisilicio 2 al borde (lateral) de la capa de polisilicio 1
L_{SEi}	Largo de la capa de polisilicio 2 que forma el capacitor i
W_{SEi}	Ancho de la capa de polisilicio 2 que forma el capacitor i

Tabla 1.2 Parámetros utilizados en (1.7)

Los parámetros de la tabla 1.2 se pueden visualizar mejor en la figura 1.4.

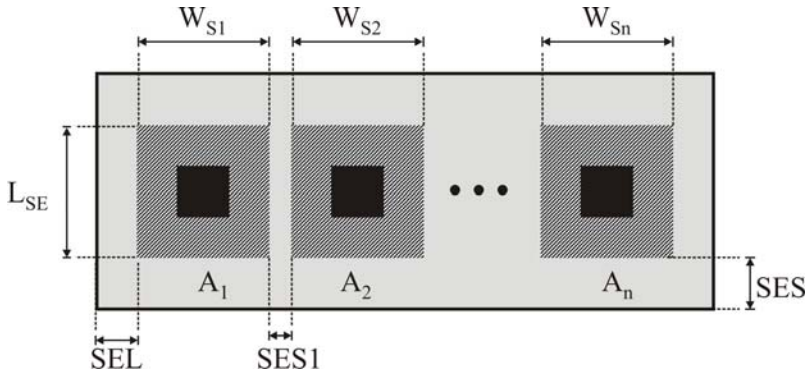


Figura 1.4. Ilustración de las variables de (1.2)

Para el cálculo de las capacitancias de entradas, se utiliza la siguiente expresión:

$$C_i = \frac{\varepsilon_{SiO_2}}{t_{ox}} A_i \qquad (1.8)$$

Donde t_{ox} es el espesor entre la capa de polisilicio 1 y polisilicio 2.

1.4 Potencial en la compuerta flotante

La Figura 1.5 muestra el modelo eléctrico del FGMOSFET, que incluye las capacitancias de entrada así como las capacitancias parásitas consideradas en este trabajo.

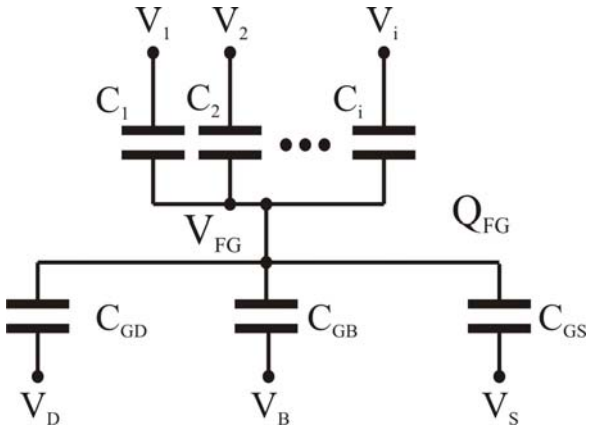


Figura 1.5. Modelo eléctrico del FGMOS

El voltaje en la compuerta flotante puede ser obtenido por balance de cargas. La carga debida a los capacitores de entrada en términos de voltaje y capacitancia está dada por:

$$\sum_i C_i \left(V_i - V_{FG} \right) = Q_i \tag{1.9}$$

Donde Q_i es a su vez igual a la carga total producida por las capacitancias parásitas y los voltajes involucrados. La ecuación (1.10) representa esta carga.

$$Q_i = C_{GS} \left(V_{FG} - V_S \right) + C_{GB} \left(V_{FG} - V_B \right) + C_{GD} \left(V_{FG} - V_D \right) \tag{1.10}$$

Igualando (1.9) con (1.10) tenemos:

$$\sum C_i V_i + C_{GS} V_S + C_{GB} V_B + C_{GD} V_D = V_{FG} \left(\sum_i C_i + C_{GS} + C_{GB} + C_{GD} \right) \tag{1.11}$$

Si $C_{TOT} = \sum_i C_i + C_{GS} + C_{GB} + C_{GD}$ y sustituyéndolo en la ecuación (1.11) obtenemos la expresión para el potencial en la compuerta flotante:

$$V_{FG} = \sum_i \frac{C_i}{C_{TOT}} V_i + \frac{C_{GS}}{C_{TOT}} V_S + \frac{C_{GB}}{C_{TOT}} V_B + \frac{C_{GD}}{C_{TOT}} V_D \qquad (1.12)$$

Si consideramos la carga inicial almacenada en la compuerta flotante, que se originó durante todo el proceso de fabricación del transistor, la expresión (1.12) queda como:

$$V_{FG} = \sum_i \frac{C_i}{C_{TOT}} V_i + \frac{C_{GS}}{C_{TOT}} V_S + \frac{C_{GB}}{C_{tot}} V_B + \frac{C_{GD}}{C_{TOT}} V_D + \frac{Q_{FG}}{C_{TOT}} \qquad (1.13)$$

Donde Q_{FG} es la carga almacenada en la compuerta flotante y es un valor desconocido.

1.5 Análisis en subumbral del FGMOSFET

La potencia consumida por los circuitos integrados comerciales tiende a decrecer exponencialmente conforme avanza la tecnología. Demandas actuales de dispositivos portátiles y con aplicaciones en comunicaciones, han originado la necesidad de diseñar circuitos integrados que trabajen a baja potencia. Muchas son las técnicas propuestas para lograr este fin; una de las más utilizadas hoy en día es el diseño de circuitos CMOS que trabajen en la región de subumbral. Se dice que un MOSFET entra en la región de subumbral o inversión débil cuando el voltaje en la compuerta es menor que el voltaje de umbral, esto es $V_{GS} < V_{TH}$. Así, la corriente de drenador resultante será una corriente debida a un mecanismo de difusión similar a la de los transistores bipolares (BJT) y estará dentro del orden de los nanoamperios. Considerando que $V_{BS} = 0$, la corriente del drenador de un FGMOS trabajando en la región de inversión débil se puede aproximar a:

$$I_D = I_0 \exp\left(\sum_{i=1}^{n} \frac{C_i}{C_{TOT}} \frac{V_{iS}}{nv_t} \right) \exp\left(\frac{C_{GD}}{C_{TOT}} \frac{V_{DS}}{nv_t} \right) \exp\left(\frac{Q_{FG}}{C_{TOT} nv_t} \right) \qquad (1.14)$$

Donde I_0 es el factor de corriente de subumbral y está dado por [11]:

$$I_0 = \frac{W}{L}\mu v_t^2 C_{DEP} \exp\left(-\frac{V_{FG}^* + 0.5\phi_f}{v_t}\right)\exp\left(\frac{Q_{FG} + C_{fsub}\left(1.5\phi_f - V_{FG}^*\right)}{C_{TOT} v_t}\right) \tag{1.15}$$

Y v_t es el voltaje térmico que está dado por:

$$v_t = \frac{k_b T}{q} \tag{1.16}$$

Donde k_b es la constante de Boltzmann y para una temperatura nominal de 27°C, el valor del voltaje térmico es de 26mV. C_{DEP} es la capacitancia de la región de empobrecimiento que está dada por la siguiente expresión.

$$C_{DEP} = \sqrt{\frac{q\varepsilon_{Si} N_B}{2\phi_s}} \tag{1.17}$$

V_{FG}^* es el potencial en la compuerta flotante para el cual el potencias de superficie $\phi_s = 1.5\phi_f$.

Por otro lado, para mantener a los transistores en saturación en la región de inversión débil se debe de cumplir con la condición $V_{DS} > 4v_t$.

Esto es una ventaja, pues el voltaje mínimo para mantener a los transistores en saturación trabajando en subumbral es alrededor de 100mV. Si realizamos una gráfica de $\log(I_D)$ vs (V_{GS}), la parte lineal representa la zona en la cual el transistor opera en la región de subumbral (Figura 1.6). Entre la región de inversión débil y fuerte existe una región intermedia llamada de inversión moderada y en ésta participan tanto corrientes de difusión como de arrastre y su modelo resulta bastante complejo.

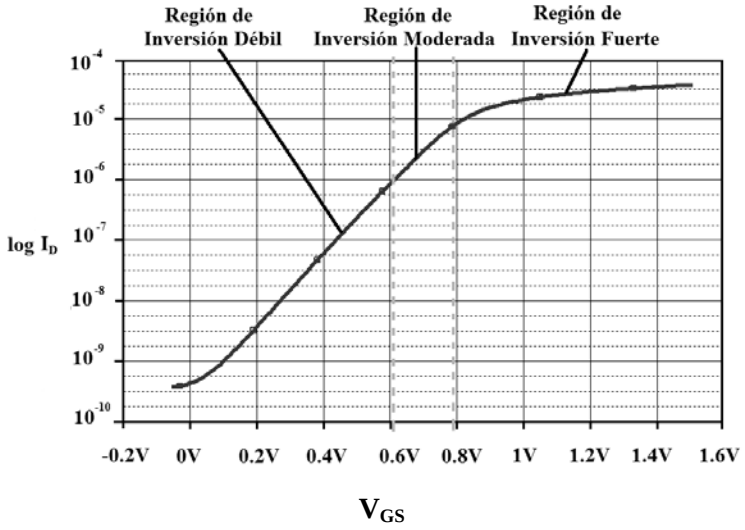


Figura 1.6 Gráfica logarítmica de las regiones de trabajo para un transistor MOSFET canal n

Más adelante se verá que el término exponencial se utilizará para implementar una función no lineal que interviene en la solución del *Problema de Asignación*.

Aunque esta técnica resulta muy útil en el diseño de circuitos de baja potencia, el término exponencial acentúa el problema de desacople, pues con una pequeña variación en los parámetros del transistor, ésta se verá reflejada exponencialmente en la corriente de salida.

1.6 El FGMOSFET como bloque sumador

El FGMOSFET puede encontrar diversas aplicaciones en el campo del diseño analógico como se ha visto anteriormente; una de las más comunes es utilizarlo como bloque sumador; esto se puede ver en la ecuación (1.19). Si despreciamos las capacitancias parásitas (esto es, haciendo C_i mucho mayor que las capacitancias parásitas), el potencial en la compuerta flotante será aproximadamente la suma de cada uno de los potenciales de entrada multiplicados por el factor de acoplamiento capacitivo.

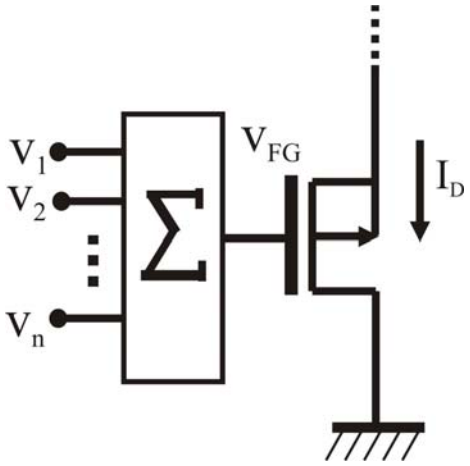


Figura 1.7 El transistor FGMOS como bloque sumador

La ventaja del transistor MOS de compuerta flotante para realizar la suma ponderada de los voltajes de entrada elimina la necesidad de utilizar circuitos analógicos extras que realicen esta función. Esta propiedad ha sido utilizada para simplificar ciertas topologías, un ejemplo es en el diseño de convertidores Digital – Analógico para generar voltajes ponderados de cada bit de entrada. Si consideramos despreciables las capacitancias parásitas, el potencial en la compuerta flotante estará dado por:

$$V_{FG} = \frac{C_1}{C_{TOT}} V_1 + \frac{C_2}{C_{TOT}} V_2 + ... + \frac{C_n}{C_{TOT}} V_n \qquad (1.19)$$

Otra forma de expresar la ecuación (1.19) está dada por la ecuación (1.20).

$$V_{FG} = \sum_{i=1}^{n} \frac{C_i}{C_{TOT}} V_i \qquad (1.20)$$

En capítulos posteriores se utilizará esta propiedad del FGMOSFET para implementar una función que requiere de la suma de varios potenciales.

1.7 Carga almacenada en la compuerta flotante

Uno de los inconvenientes que impedía el uso del FGMOSFET para diversas aplicaciones en el campo de diseño analógico, era la incertidumbre de la carga almacenada en la compuerta flotante. Idealmente, esta carga debería de ser cero, pero durante todo el proceso de fabricación se almacena una cantidad desconocida de carga en la compuerta flotante diferente para cada proceso, lo cual genera una modificación del voltaje de umbral distinta para cada FGMOSFET, afectando drásticamente el desempeño del circuito. Ya se han propuesto varios métodos para eliminar la carga en la compuerta flotante [10], algunos de ellos se describen a continuación. En un inicio se propuso eliminar la carga por radiación UV; cuando la compuerta flotante es iluminada, los electrones que están atrapados en la compuerta son capaces de atravesar la barrera de la interfase oxido/silicio. Una de las desventajas de este método es su incompatibilidad con tecnologías CMOS, en las cuales sus capas de pasivación no son transparentes a la luz UV. En tales casos es necesario eliminar la pasivación, sin embargo esto afecta la vida útil del dispositivo. Otro método es utilizar el tuneleo Fowler-Nordheim, que consiste en la generación de un campo eléctrico alto para que los electrones atrapados en la compuerta atraviesen la barrera de potencial de SiO_2 hacia la terminal polarizada positivamente. Otra alternativa es conectar la compuerta flotante a un interruptor conectado a 0V, cuando el interruptor esté cerrado, la compuerta se descargará, sin embargo, en la realidad este interruptor generará una resistencia, aunque grande, finalmente hará que la compuerta no esté flotando, influyendo en el desempeño del circuito. Otra solución propuesta para la eliminación de la carga, es por medio de contactos desde la capa de polisilicio 1 (compuerta flotante), hasta la capa superior metálica. Durante el proceso de fabricación (antes de la etapa de grabado), todas las partes del chip en contacto con dicha capa metálica estarán conectadas, lo cual hace que la carga atrapada fluya hacia otra parte de la oblea, como por ejemplo el substrato. Cuando el proceso termina, la capa metálica ya no estará conectada a ninguna otra parte del circuito, manteniendo la naturaleza flotante de la compuerta.

1.8 Macromodelo del FGMOSFET para simulación en PSPICE

PSPICE es una poderosa herramienta utilizada en la simulación de circuitos electrónicos que nos permite obtener resultados muy cercanos a la realidad a través de modelos propuestos de distintos dispositivos. Sin embargo, no es posible realizar la simulación de las características en DC del FGMOSFET, pues las capacitancias de entradas serán vistas por PSPICE como nodos flotantes en el análisis en DC, generando un problema de convergencia; Rodríguez Villegas [10] menciona un método de simulación del FGMOSFET por medio de resistores en serie con las capacitancias de entrada, con el fin de eliminar nodos flotantes, en donde los valores de los resistores son muy grandes para no afectar el desempeño del circuito simulado; otra solución más precisa fue dada por Ochiai y Hatano[12] quienes propusieron un macromodelo para simulación en PSPICE que consiste en un arreglo de resistores y fuentes dependientes de voltaje y corriente. De esta forma se puede generar el potencial de la compuerta flotante; la figura 1.8 ilustra el macromodelo utilizado.

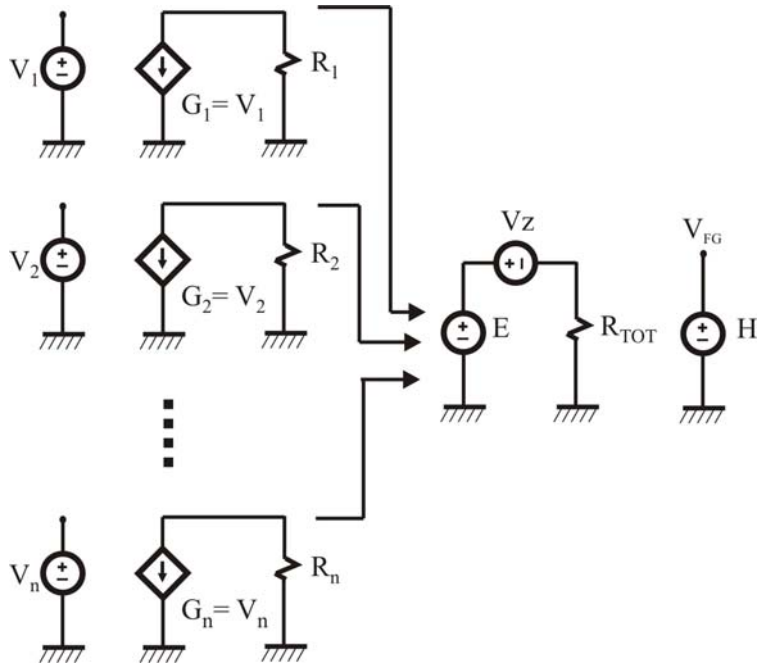


Figura 1.8 Macromodelo utilizado para la simulación de las características eléctricas del FGMOS

La fuente dependiente de voltaje E está dada por:

$$E = R_1 V_1 + R_2 V_2 + \ldots + R_n V_n \tag{1.21}$$

Este potencial generará una corriente por el resistor R_{tot} dada por:

$$I_z = \frac{R_1 V_1 + R_2 V_2 + \ldots + R_n V_n}{R_{tot}} \tag{1.22}$$

Y a su vez la fuente de voltaje dependiente de voltaje H, generará un voltaje proporcional a la corriente I_z, como lo muestra la ecuación (1.23)

$$V_{FG} = H = \frac{R_1}{R_{TOT}} V_1 + \frac{R_2}{R_{TOT}} V_2 + \ldots + \frac{R_n}{R_{TOT}} V_n \tag{1.23}$$

Del desarrollo anterior, el macromodelo generará una suma de voltajes ponderados, sólo hay que mantener la relación $\frac{R_i}{R_{TOT}}$ igual a la relación $\frac{C_i}{C_{TOT}}$.

1.9 Implementación de funciones no lineales utilizando el FGMOSFET trabajando en la región de subumbral

Una de las aplicaciones del FGMOSFET trabajando bajo el régimen de inversión débil es la de poder implementar funciones no lineales. Como se vió en los temas anteriores, la expresión que predice el comportamiento de la corriente de un MOSFET en subumbral incluye un término exponencial, el cual ha sido aprovechado para la implementación de diversas funciones de características no lineales. A continuación se muestran los circuitos para obtener la raíz cuadrada y segunda potencia de una variable [13]. Ambos trabajan en modo de corriente. Para el primer caso, se parte de la Figura 1.9.

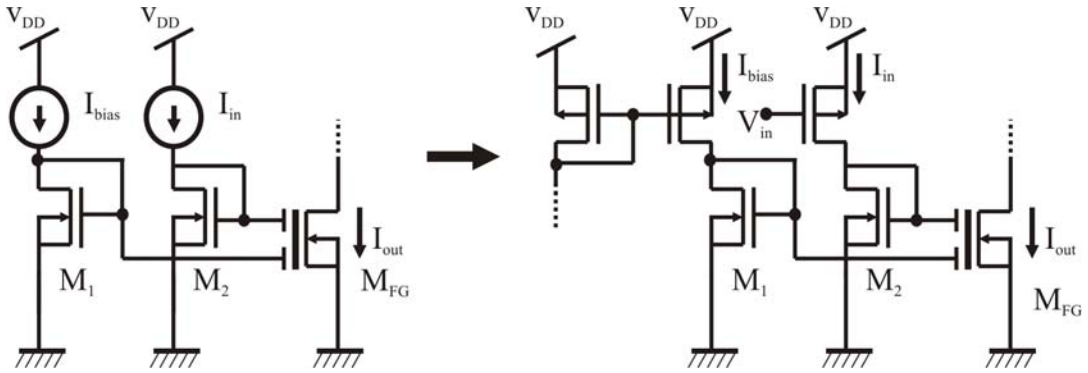


Figura 1.9 Circuito que obtiene la función de raíz cuadrada

El voltaje en la compuerta de M_1 y M_2 están dados por:

$$V_{G1} = v_t \ln\left(\frac{I_{in}}{I_0}\right) \tag{1.24}$$

$$V_{G2} = v_t \ln\left(\frac{I_{bias}}{I_0}\right) \tag{1.25}$$

Donde I_0 está dada por la ecuación (1.15). Si las capacitancias de entrada de M_{FG} son iguales, esto es $C_1 = C_2 = C$, el voltaje en su compuerta flotante estará dado por:

$$V_{FG} = \frac{1}{2} v_t \ln\left(\frac{I_{bias}}{I_0}\right) + \frac{1}{2} v_t \ln\left(\frac{I_{in}}{I_0}\right) \tag{1.26}$$

Donde V_{G1} y V_{G2} son multiplicadas por el factor de acoplamiento capacitivo, que para el caso supuesto es de ½. También V_{FG} está dada por:

$$V_{FG} = v_t \ln\left(\frac{I_{out}}{I_0}\right) \tag{1.27}$$

Igualando la ecuación (1.26) con la (1.27), resulta:

$$\ln\left(\frac{I_{out}}{I_0}\right) = \ln\left(\frac{I_{bias}}{I_0}\right)^{1/2} + \ln\left(\frac{I_{in}}{I_0}\right)^{1/2} \tag{1.28}$$

De la expresión anterior, se puede obtener la corriente de salida I_{out}

$$I_{out} = (I_0)\left(\frac{I_{bias}}{I_0}\frac{I_{in}}{I_0}\right)^{1/2} = \sqrt{I_{bias}I_{in}} \tag{1.29}$$

El circuito anterior puede generalizarse [14] para obtener la raíz n-ésima de una variable de la siguiente manera; a partir de la Figura 1.10, inicia nuestro análisis.

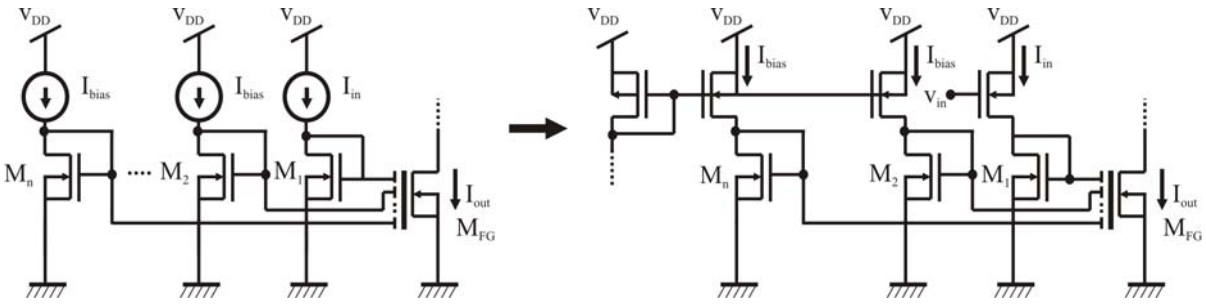


Figura 1.10 Generalización del circuito de la Fig.1.9 para obtener la raíz n-ésima

Si se sigue el mismo procedimiento para obtener el circuito de raíz cuadrada, podemos obtener una expresión para el circuito arriba mostrado:

$$I_{out} = \sqrt[n]{I_{bias_n}{}^{n-1}}\sqrt[n]{I_{in}} \tag{1.30}$$

Como se puede observar de la ecuación (1.30), el valor de la raíz depende del número de entradas en el transistor M_{FG}. Por ejemplo, si se requiere la implementación de la función raíz cúbica, el número de entradas en transistor M_{FG} será de tres. Por otro lado, si lo que se desea es obtener la familia de curvas para la función de la raíz n-ésima, el valor de la corriente I_{bias} deberá de variar de acuerdo con:

$$I_{bias_n} = \sqrt[n-1]{I_{bias_2}}^{*} \tag{1.31}$$

La Figura 1.11 muestra el resultado de la simulación en PSPICE para los circuitos de raíz cuadrada, cúbica y cuarta.

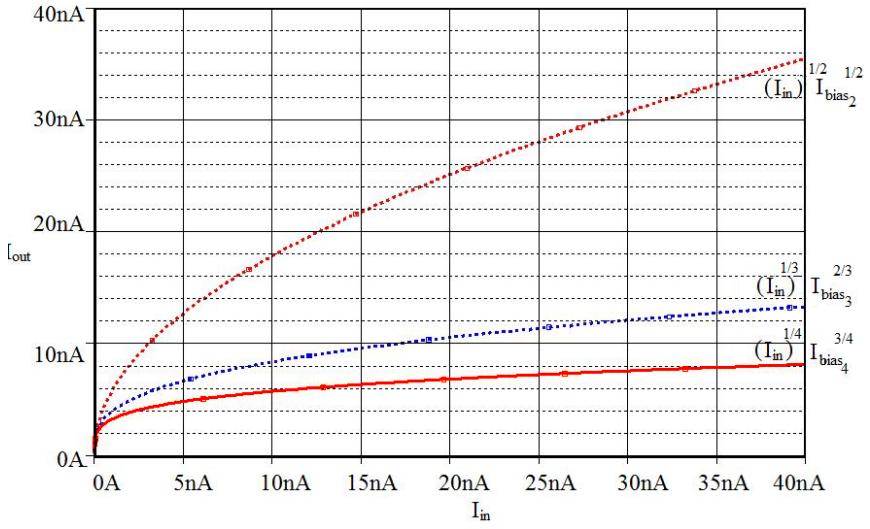


Figura 1.11 Resultado de la simulación del circuito de la fig. 1.10 para obtener la raíz cuadrada, cúbica y cuarta

De la Figura 1.12, parte el análisis para el segundo caso, es decir el circuito que implementa la función de potencia cuadrada.

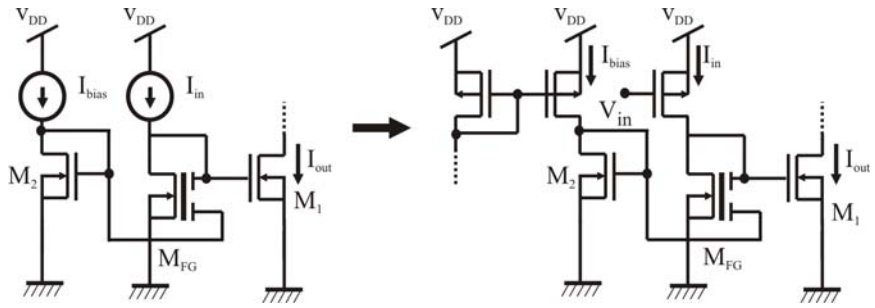


Figura 1.12 Circuito que implementa la función de potencia cuadrada

El voltaje de compuerta en M_2 está dado por:

$$V_{G2} = v_t \ln\left(\frac{I_{bias}}{I_0}\right) \tag{1.32}$$

El voltaje de compuerta de M_1 está dado por:

$$V_{FG} = v_t \ln\left(\frac{I_{out}}{I_0}\right) \tag{1.33}$$

Y el voltaje de compuerta de M_{FG} está dado por

$$v_t \ln\left(\frac{I_{in}}{I_0}\right) = v_t \ln\left(\frac{I_{out}}{I_0}\right)^{1/2} + v_t \ln\left(\frac{I_{bias}}{I_0}\right)^{1/2} \tag{1.34}$$

De las expresiones anteriores se puede obtener una ecuación para la corriente de salida y esta dada por:

$$I_{out} = \frac{I_{in}^{\ 2}}{I_{bias}} \tag{1.35}$$

De la misma manera que en el circuito para obtener la función de raíz cuadrada podemos hacer una generalización para este circuito como se muestra en la siguiente figura:

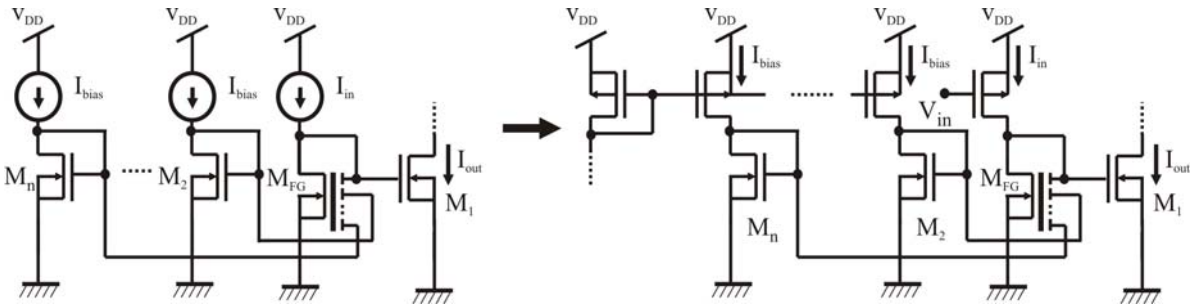


Figura 1.13 Generalización del circuito de la figura 1.12 para obtener la potencia n-ésima

Una expresión puede ser obtenida para el transistor M_{FG} de n entradas, dada por:

$$I_{out} = \frac{I_{in}^{\ n}}{I_{bias_n}^{\ n-1}} \tag{1.36}$$

Donde I_{bias_n} es obtenida de

$$I_{bias_n} = \sqrt[n-1]{I_{bias_2}^{\ *}} \tag{1.37}$$

Finalmente se muestra la simulación realizada en PSPICE para una tecnología de 0.5um de los circuitos para obtener la segunda, tercera y cuarta potencia de la corriente de entrada.

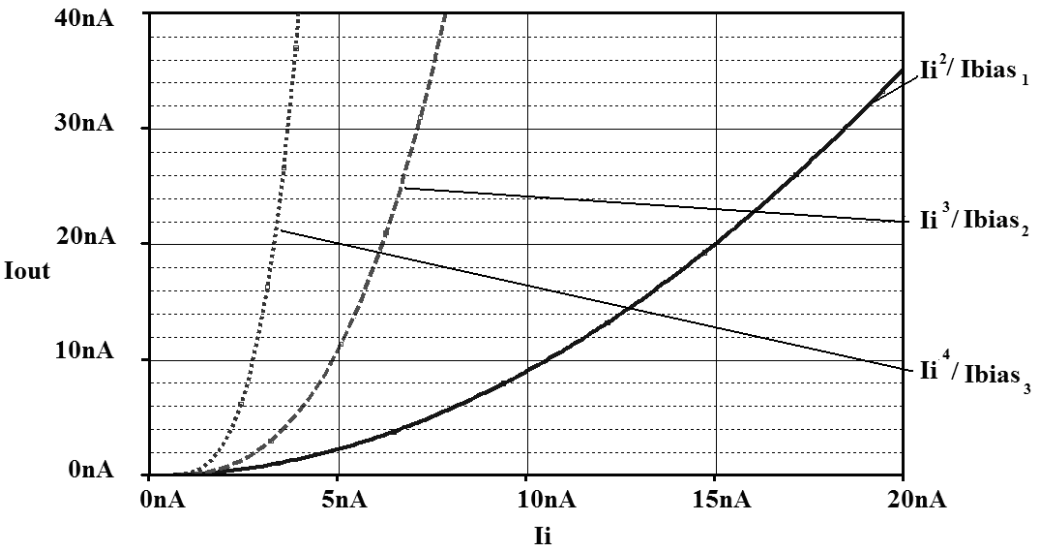


Figura 1.14 Resultado de la simulación del circuito de la fig. 1.13 para una potencia cuadrada, cúbica y cuarta

El análisis anterior demuestra la versatilidad que se tiene con el FGMOSFET para generar fácilmente circuitos no lineales, trabajando en el régimen de subumbral. Esta ventaja será aprovechada en este trabajo para la implantación de funciones no lineales que intervienen en la solución del *Problema de Asignación.*

REFERENCIAS

[1] D. Kahng and S.M. Sze, “A floating-gate and its application to memory devices,” The Bell System Technical Journal, vol. 46, no. 4, 1967, pp. 1288-1295.

[2] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically trainable artificial neural network with 10240 'floating gate' synapses," Proceedings of the International Joint Conference on Neural Networks, Washington, D.C., vol. II, 1989, pp. 191-196.

[3] S. Lai, “Flash memories: where we were and where we are going," IEEE International Electron Devices Meeting, San Francisco, 1998, pp. 971-974.

[4] T. Shibata, T. Ohmi, “A functional transistor MOS featuring gate-level weighted sum and threshold operations”, IEEE Transactions on Electron Devices, vol. 39, no. 6, 1992, pp. 1444-1455.

[5] Lopez-Martin, A.J.; Carlosena, A.; Ramirez-Angulo, J, “D/A Conversion based on multiple-input floating-gate MOST”, Circuits and Systems, 1999 42nd Midwest Symposioum on, vol. 1, pp. 149-152.

[6] Adil, F. Serrano, G. Hasler, P., “Offset removal using floating-gate circuits for mixed-signal systems”, Southwest Symposium on Mixed-Signal Design, Feb. 2003, pp 190-195.

[7] Hasler P., Minch B.A., Diorio C., “Floating-gate devices: they are no just for digital memories anymore”, Circuits and Systems, Proceedings of the 1999 International Symposium on, Vol. 2, pp. 388 – 391.

[8] B. A. Minch, C. Diorio, P. Hasler, C.A. Mead, “Translinear circuits using subthreshold floating-gate MOS transistors”, Analog Integrated Circuits and Siganl Processing, vol. 9, no. 9, 1999, pp. 167-179 .

[9] Hasler P., Minch D.W., Diorio C., “An autozeroing floating gate amplifier”, Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on, vol. 48, issue 1, 2001, pp. 74-82.

[10] Rodríguez V.E. “Low power and low voltage analogue design with the floating gate MOS”, IET, 2006, pp. 24-25.

[11] Yang K., Andreu A.G., “Subthreshold analysis of floating-gate MOSFET´s”, University/Government/Industry Microelectronic Symposium, Proceedings of the Tenth Biennial, 1993, pp. 141-144.

[12] Ochiai T., Hatanao H., “DC characteristic simulation for floating gate neuron MOS circuits”, Electronic Letters, vol. 35, issue 18, 1999, pp. 1505-1507.

[13] Koosh V.F., Goodman R. “Dynamic charge restoration of floating gate subthreshold MOS translinear circuits” Advanced Research in VLSI, 2001. ARVLSI 2001. Proceeding, 2001 Conference on**,** 2002, pp. 163 – 171.

[14] González-Carabarín L., Gómez-Castañeda F., Moreno-Cadenas J.A., “Generalized nth-power-law and nth-root circuits”, 6th International Conference on Electrical Engineering, Computing Science and Automatic Control, Nov. 2009.

2.1 Introducción

El concepto de Conmutación de Paquetes (en inglés Packet Switching) está basado en la división de llamadas o mensajes en piezas llamadas "paquetes". Estos paquetes se mueven a través de la red, desde un centro de conmutación a otro; cada centro de conmutación después de recibir un paquete lo almacena hasta que éste es recibido adecuadamente por otro centro de conmutación [1]. Dentro de las arquitecturas de Conmutación de Paquetes podemos encontrar la "Crossbar Switch", que consiste en un arreglo de interruptores que conecta *n* entradas con m salidas (donde $n = m$). Cuando un interruptor se cierra, el paquete conectado a la entrada *n* será dirigido a la salida *m*; la idea principal es presentada en la figura 2.1.

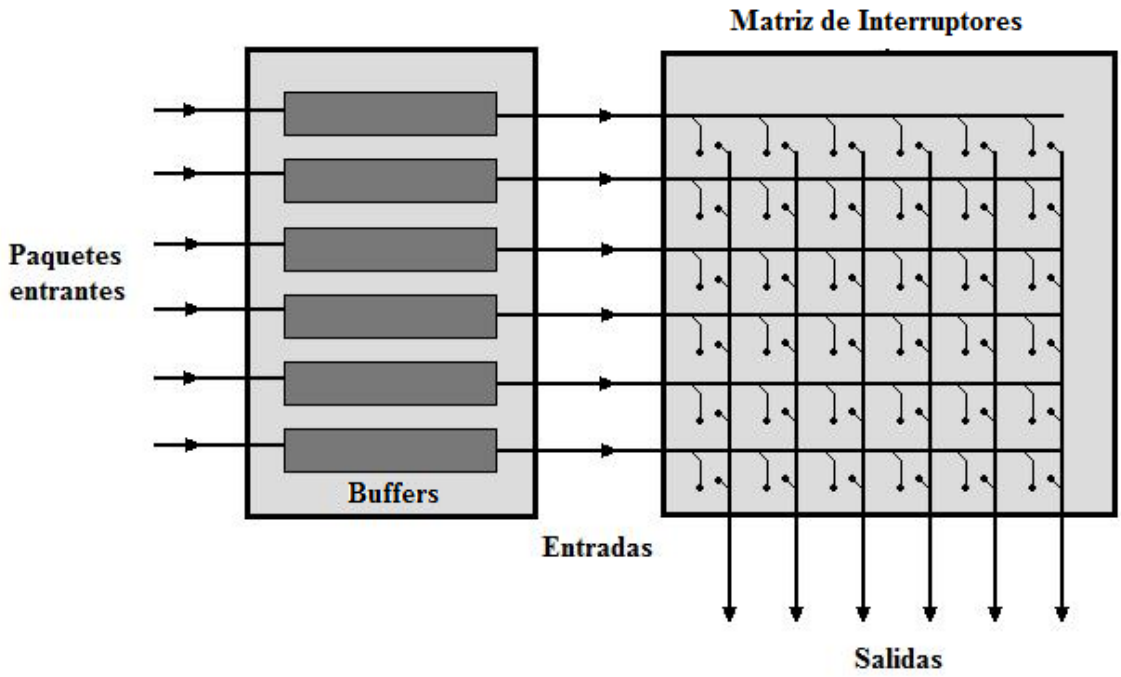


Figura 2.1. Arquitectura "Crossbar Switch"

Cuando dos o más entradas son asignadas a la misma salida, lo que es muy común debido al incremento exponencial del tráfico de información, el paquete de información es bloqueado sin importar el algoritmo de enrutamiento usado, lo que genera tiempos de retardo en el envío de información. Por medio de la solución del problema de asignación se puede resolver este problema examinando los requerimientos de las líneas de salidas y así minimizando los retardos. Este concepto se visualiza en la figura 2.2.

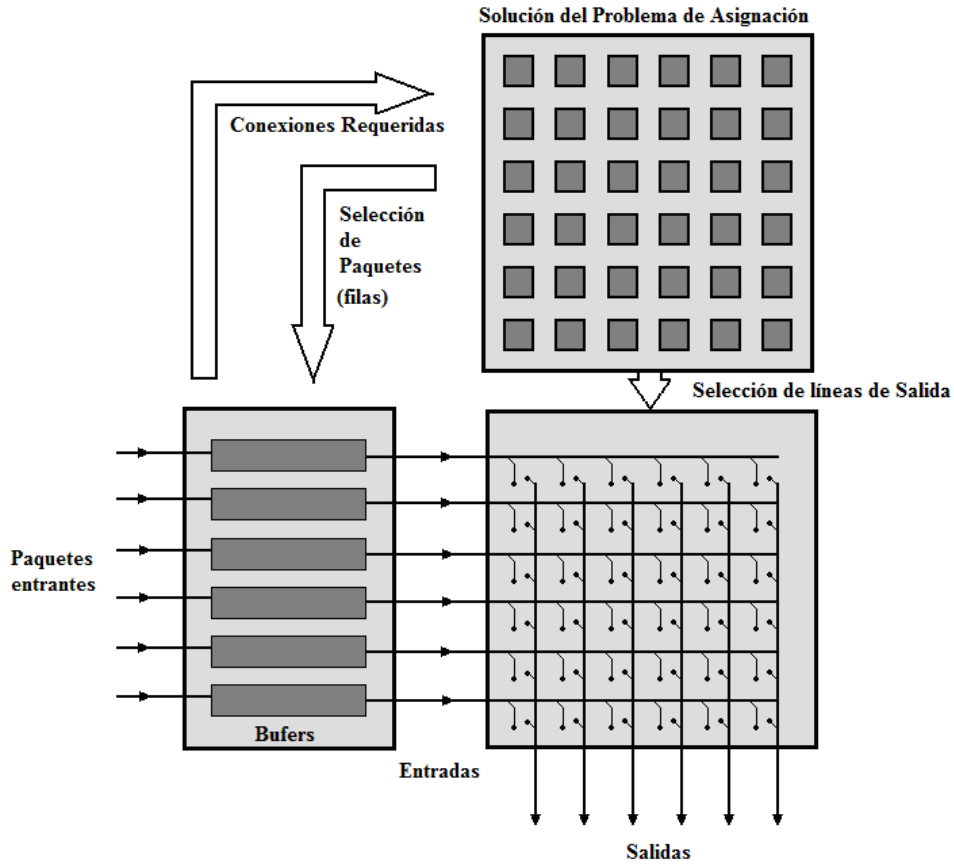


Figura 2.2. Control del flujo de información en una arquitectura “Crossbar Switch” a través de la solución del Problema de Asignación

2.2 El *Problema de Asignación*

La solución de muchos problemas, tanto teóricos como prácticos, implican la obtención de la "mejor elección" de parámetros para lograr cierto objetivo[2],así la optimización es la rama de las matemáticas que da respuesta a problemas en donde se desea elegir la mejor dentro de un conjunto de elementos. A su vez, los problemas de optimización se pueden dividir dentro de dos categorías: aquellos que involucran variables continuas y los que involucran variables discretas. En problemas que involucran variables continuas, se busca un conjunto de números reales o una función; en el segundo caso, se busca un objeto de un conjunto finito.

El *Problema de Asignación* pertenece a la rama de la programación lineal dedicada a la optimización combinatoria. Este problema se explica con el paradigma de "Agentes y Tareas", el cual tiene el siguiente enunciado:

Dado un grupo de N tareas que deben de ser asignadas a un grupo de N agentes, en donde cada agente "i" tiene un coeficiente de costo, denotado por el parámetro: s_{ij}, para realizar cada tarea "j" se debe de encontrar la distribución agente-tarea que optimice el desempeño del sistema, minimizando el costo global.

En la asignación de tareas a los agentes no se pueden asignar dos tareas al mismo agente, ni dos personas pueden ser asignadas a la misma tarea. El *Problema de Asignación* de 4 agentes y 4 tareas se ilustra en la figura 2.3, en donde los parámetros s_{ij} representan los costos. La solución hipotética se muestra resaltada en fondo oscuro.

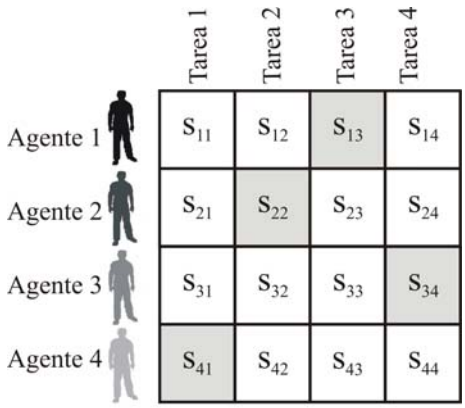


Figura 2.3

Así, este problema queda especificado como un problema de minimización, como sigue:

Minimizar la expresión: $\sum s_{ij}x_{ij}$ para i,j = 1,2,…,N (2.1)

Sujeta a: $\sum x_{ij} = 1$ para i = 1,2,…,N (2.2)

$\sum x_{ij} = 1$ para j = 1,2,…,N (2.3)

La solución está dada por la matriz de permutación X, cuyos elementos x_{ij} toman los valores discretos: 0,1. Alternativamente, el problema puede ser abordado como un problema de maximización considerando el complemento de cada parámetro s_{ij} como: $\hat{s}_{ij} = 1 - s_{ij}$. Uno de los métodos más conocidos para resolver el *Problema de Asignación* es el Método Húngaro, que se abordará en la secciones siguientes y para el cual es necesario definir algunos conceptos del área de Teoría de Grafos.

2.2.1 Introducción básica a la Teoría de Grafos

En matemáticas y ciencias computacionales, la Teoría de Grafos es el estudio de los grafos, que se definen como un par de conjuntos G=(V,E) [3]. Donde los elementos de V son los vértices (o nodos o puntos del grafo) y los elementos de E son los bordes (o líneas), como se puede ver en la figura 2.4.

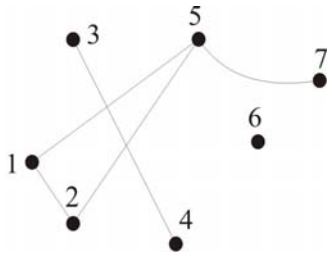


Figura 2.4 El conjunto V = {1,2,3,4,5,6,7} y el conjunto E = {{1,2} {1,5} {2,5} {3,4} {5,7}}

El conjunto de vértices en un grafo es referido como V(G) y el conjunto de bordes como E(G). Un vértice **v** es *incidente* con un borde **e**, si $\mathbf{v} \in \mathbf{e}$, entonces **e** es un borde en **v**. Podemos decir que dos vértices $\mathbf{v_1}$ y $\mathbf{v_2}$ son *adyacentes* si $\mathbf{v_1v_2}$ es un borde. Un par de vértices o bordes no adyacentes se les conoce como *independientes*.

También se define un *camino* como un grafo no-vacío P= (V,E) de la forma:

$$V = \{x_0, x_1, ..., x_k\} \quad E = \{x_0x_1, x_1x_2, ..., x_{k-1}x_k\}$$

Donde los vértices x_0 a x_k son unidos por P y éstos son llamados *vértices finales*.

- **Grafos bipartitos**

Sea r>=2 un entero, un grafo G=(V,E) es llamado r-partito, si V admite una partición de r clases tal que cada borde tenga su final en clases distintas. Los vértices en la misma clase no deben de ser adyacentes. Cuando r=2 se dice que tenemos un *grafo bipartito* (Figura 2.5).

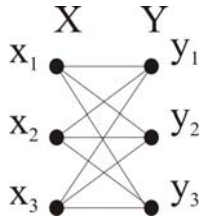


Figura 2.5 Ejemplo de grafo bipartito

- **Apareamiento en grafosj bipartitos**

El conjunto M de bordes independientes en un grafo G=(V,E) es llamado *matching* o *apareamiento*. M es un apareamiento de $U \subseteq V$ si cada vértice en U es incidente con un borde en M. Los vértices en U son entonces llamados *apareados*. Los vértices no incidentes con algún borde de M se les llama *no apareados*. Para la representación en una grafo bipartito de los bordes apareados será con una línea gruesa que los distinga de los demás bordes no apareados.

Consideremos un grafo bipartito con la partición de clases {X,Y}, donde los vértices denotados como "x" pertenecen a X y los denotados con "y" pertenecen a Y. Consideremos un apareamiento inicial M en G, un camino en G comienza en X en un vértice no apareado de E\M y luego de M y así alternadamente, a este camino se le conoce como un *camino alternado*. Un camino alternado que termina en un vértice no apareado de Y, se conoce como un *camino aumentado*, ya que podemos usarlo para cambiar M a un apareamiento mayor, es decir la diferencia simétrica de M con E(P) es un apareamiento y el conjunto de vértices apareados se incrementa por dos (los finales del camino P), como lo muestra la figura 2.6.

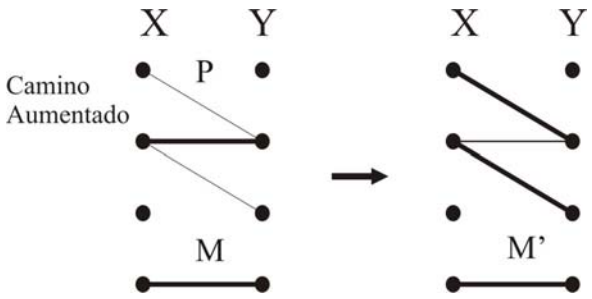


Figura 2.6 El camino alternado P es un camino aumentado, ya que incrementa el apareamiento M en dos, donde $M' = M \Delta E(P) = (M / E(P)) \cup \left(E(P) / M\right)$

Un apareamiento M satura un vértice **v**, y **v** se dice que es *M-saturado*; de otra forma **v** es *M-no saturado*. Si cada vértice de G es M-saturado, entonces el apareamiento es *perfecto*.

- **Teorema de Hall**

En el problema de apareamiento se desea encontrar un apareamiento en G que sature cada vértice en X. La condición necesaria y suficiente para la existencia de tal apareamiento esta dada por el Teorema de Hall [3]:
Sea G un grafo bipartito con bipartición (X,Y). Para cualquier conjunto S de vértices en G, se define N(S) como el conjunto de vértices adyacentes a S. Entonces G contiene un apareamiento que satura cada vértice en X si y solo si:

$$\left|\mathrm{N(S)}\right| > \left|\mathrm{S}\right| \text{ para toda } \mathrm{S} \subseteq \mathrm{X}$$

Sea **u** un vértice M-no saturado en X , y Z el conjunto de todos los vértices conectados a *u* por M-caminos alternados, entonces $\mathrm{S} = \mathrm{Z} \cap \mathrm{X}$ y $\mathrm{T} = \mathrm{Z} \cap \mathrm{Y}$ (Figura 2.7); se tiene que el teorema de Hall no se cumple si $\mathrm{N(S)} = \mathrm{T}$, es decir no existe un apareamiento que sature todos los vértices de X.

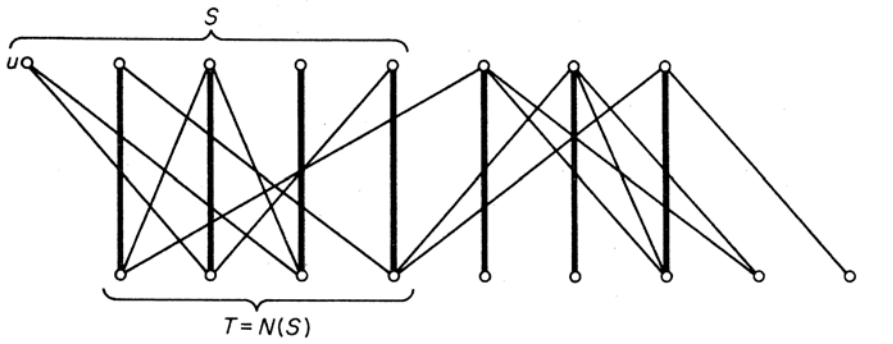


Figura 2.7. No se cumple el teorema de Hall pues $T = N(S)$

2.2.2 Algoritmo Húngaro para la solución del *Problema de Asignación*

El algoritmo Húngaro para la solución del *Problema de Asignación*, a veces se refiere como el Algoritmo Kuhn-Munkers.

Consideremos un grafo bipartito ponderado con una bipartición (X,Y), donde $X = \{x_1, x_2, ..., x_n\}$ representan los n-agentes y $Y = \{y_1, y_2, ..., y_n\}$ representan los n-empleos y cada borde $x_i y_i$ tiene un peso (o costo) $s_{ij} = s(x_i y_j)$. Debido a eso a veces se refiere al *Problema de Asignación* como un problema de *apareamiento ponderado*[3]. El problema de asignación es encontrar el apareamiento perfecto con un mínimo costo, es decir un ***apareamiento óptimo.***. Para la solución se utilizan variables auxiliares denominadas etiquetas que van actualizando el apareamiento hasta encontrar el óptimo.

Primero se define un *etiquetado de vértice factible* como una función ***l*** en el conjunto de vértices $X \cup Y$, tal que:

$$l(x) + l(y) \leq s(xy) \text{ para todo } x \in X \text{ y } y \in Y \qquad (2.4)$$

El número real ***l***(v) es llamado la *etiqueta* del vértice v. Si ***l*** es un etiquetado de vértice factible, denotamos E_l como el conjunto de bordes dados por:

$$E_l = \{xy \in E \,|\, l(x) + l(y) = c(xy)\} \qquad (2.5)$$

El bigrafo G con bordes E_l es referido como *grafo de igualdad* y denotada por G_l.

La idea básica del algoritmo es muy sencilla, la Figura 2.8 presenta el diagrama de flujo del algoritmo húngaro para la solución del *Problema de Asignación.*

Se comienza con un etiquetado de vértice factible ***l***, se determina G_l, y se elige un apareamiento M arbitrario. Si *X* es M-saturado, entonces M es un apareamiento perfecto y por lo tanto un apareamiento óptimo; en este caso el algoritmo se detiene. De otra manera, sea *u* un vértice M-no saturado, entonces $S=\{u\}$ y $T=\varnothing$. Si $N_{G_l}(S)=T$, se calcula la variable α_l de acuerdo con:

$$\alpha_l = \min_{\substack{x\in S \\ y\notin T}} \left\{ s(xy) - l(x) + l(y) \right\}$$

Y se calcula un nuevo etiquetado de vértices factibles $\hat{l}$ dada por:

$$\hat{l}(v) = \begin{cases} l(v) + \alpha_l & Si\ v \in S \\ l(v) - \alpha_l & Si\ v \notin S \\ l(v) - \beta_l & Si\ v \in T \\ l(v) + \beta_l & Si\ v \notin T \end{cases}$$

Después se reemplaza $l(v)$ por $\hat{l}(v)$, y se crean nuevos bordes de acuerdo a la ecuación (2.5) obteniendo un nuevo grafo $G_{\hat{l}}$ que reemplazará a G_l.

Por otro lado, si $N_{G_l}(S) \neq T$ entonces se elige un vértice *y* en $N_{G_l}(S)/T$, si *y* es M-saturado con $yz \in M$ (donde *z* sería cualquier elemento de X que estuviera apareado con el vértice *y*) y se reemplaza *S* por $S \cup \{z\}$ y *T* por $T \cup \{y\}$ y se vuelve a preguntar si $N_{G_l}(S)=T$ repitiéndose el ciclo anterior. En caso de que *y* no fuera M-saturado, el camino P (camino seguido desde *u* hasta *z*) es *yb* camino aumentado en G_l, y por tanto se reemplaza M por $\hat{M}$, donde $\hat{M} = M \Delta E(P)$, de ese punto el algoritmo vuelve a preguntar si *X* es M-saturado y se repite todo el ciclo.

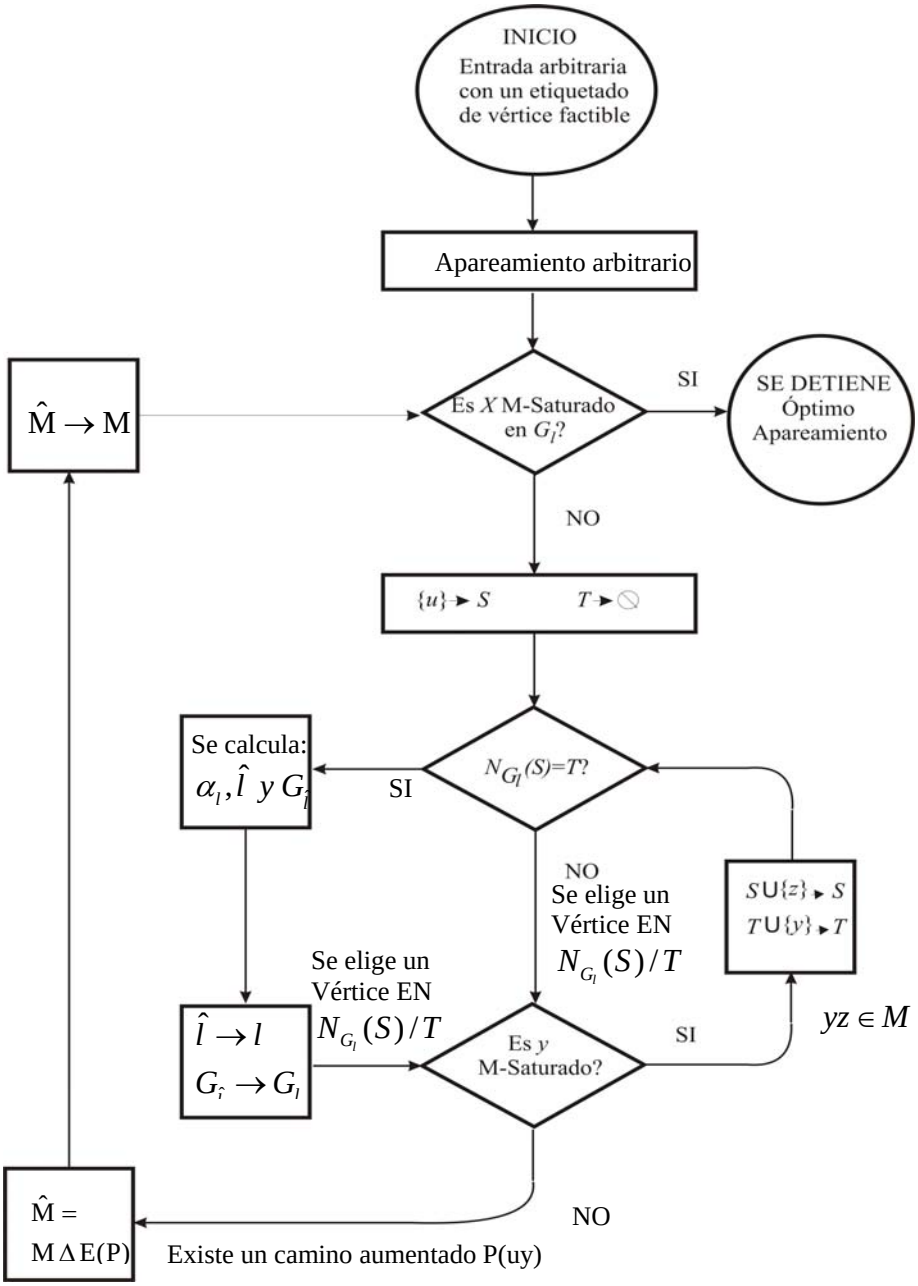


Figura 2.8 Algoritmo Húngaro para la Solución del Problema de Asignación

Con el objetivo de verificar los resultados obtenidos, se implementó el algoritmo Húngaro en Microsoft Visual Basic Versión 6.0. Se pueden simular matrices de hasta 64 x 64, generándolas ya sea con números aleatorios o bien introduciendo la matriz de costos. La Figura 2.9 muestra la ventana principal del simulador.

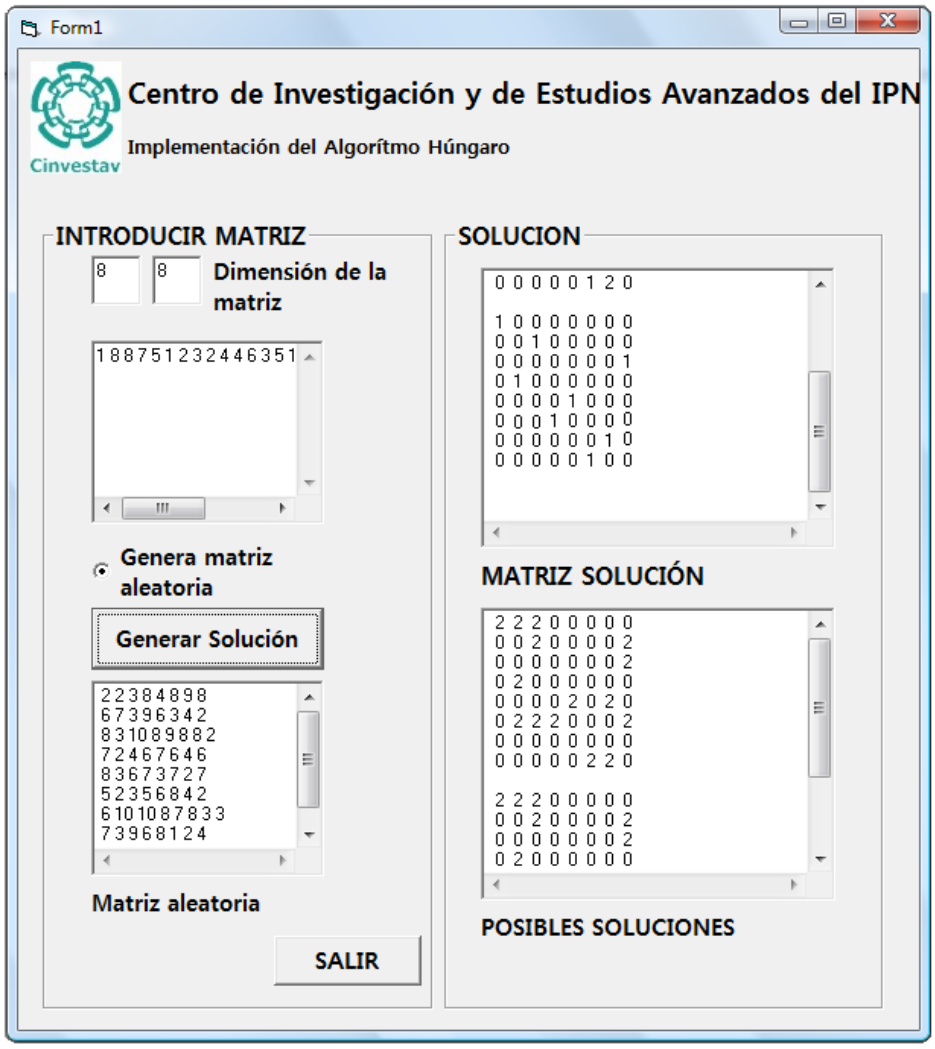


Figura 2.9. Ventana principal del simulador del algoritmo Húngaro implementado en Visual Basic versión 6.0

2.3 Soluciones alternativas al *Problema de Asignación*: implementación en circuitos electrónicos

En 1985 Hopfield y Tank [4] extendieron las aplicaciones de su modelo para solucionar problemas de optimización combinatoria; se dieron cuenta de que con una red de neuronas, éstas podían ser usadas para calcular la solución de un problema de optimización específico por su naturaleza analógica. Con base en el trabajo de Hopfield y Tank, se han propuestos varias soluciones a diversos problemas de optimización combinatoria con la finalidad de implementarlas en circuitos electrónicos. J. Wang [5] propuso una solución del

Problema de Asignación utilizando una modificación de la red de Hopfield, que consistía en un arreglo de n^2 neuronas, donde el estado estable de cada neurona representaba una variable de decisión; si bien la red encuentra la solución óptimamente, cada neurona es compleja, pues requiere de la implementación de varios amplificadores operacionales y una arreglo de resistencias y capacitores que incrementan su número a medida que las dimensiones de la matriz aumentan, lo cual conlleva gran área en chip, además de que el tiempo que le toma a la red llegar al estado estable es de casi 100µs. La Fig. 2.10 muestra la implementación de una celda para un arreglo de n x n.

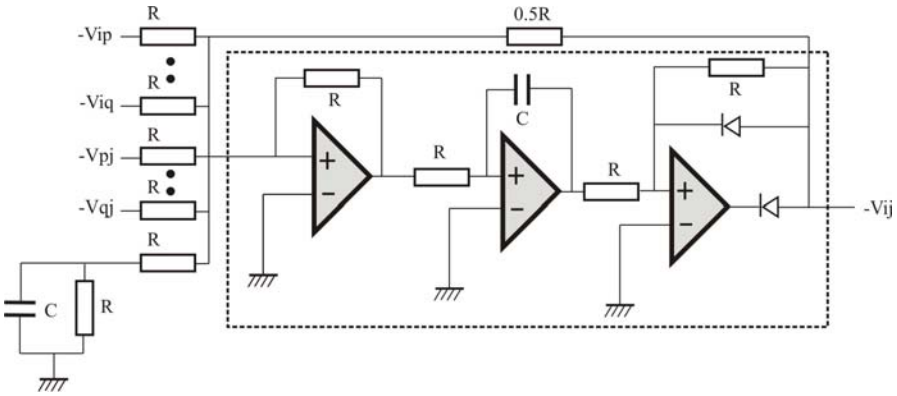


Figura 2.10 Implementación de la celda ij para el circuito de asignación propuesto en [5]

Symington-Waddie-Taghizadeh y Snowdon [6] proponen una solución por medio de redes neuronales cuyo procesamiento se realiza a través de un DSP (Digital Signal Processor).

Este sistema resulta bastante complejo de implementar pues consiste en un arreglo de redes neuronales que utiliza un sistema óptico que proporciona una matriz de "costos", posteriormente la información es procesada a través de una serie de DSP´s, que mediante un algoritmo iterativo encuentra la solución del *Problema de Asignación*. Este sistema resulta difícil de implementar, además de reportar tiempos de convergencia en el orden de milisegundos. K. Urahama [8] basado en sus trabajos anteriores [7], propone otra solución alternativa al *Problema de Asignación* utilizando el método de multiplicadores de Lagrange (Apéndice C). Primero abordaremos la solución del problema del *Ganador Toma Todo* utilizando el método de los multiplicadores de Lagrange; después, el *Problema de Asignación* se puede ver como una extensión en dos dimensiones del circuito del *Ganador Toma Todo.*

2.4 Solución del problema del *Ganador Toma Todo* utilizando el método de multiplicadores de Lagrange

Ya han sido muchos los trabajos dedicados a resolver el problema del *Ganador Toma Todo* debido a sus bastas aplicaciones en el campo de sistemas neuromórficos [9,10]. La idea básica es tratar de simular la competencia neuronal, donde la neurona mas fuerte será la ganadora; para su implementación en circuitos CMOS, se elige una señal de voltaje o corriente que posea la mayor intensidad de entre un conjunto de señales, mientras que las señales restantes serán pasivadas y la "ganadora" se enviará a un cierto valor de voltaje o corriente. De esta forma, el problema del *Ganador Toma Todo* puede ser definido como un problema de maximización de la función objetivo $F(x_i)$:

$$F(x_i) = \sum -s_i x_i \tag{2.6}$$

Sujeto a

$$-\sum_{i=1}^{n} x_i + 1 = 0 \tag{2.7}$$

Donde la ecuación (2.6) es la función objetivo y s_i son los coeficientes de costos, mientras que la ecuación (2.4) representa la restricción del sistema. Es decir, se busca el valor más grande de entre una serie de valores, con la restricción de que solo un elemento puede ser el "ganador". La solución consiste en la introducción del término de entropía a la función objetivo. Así la nueva función objetivo queda como:

$$\max \sum_i -s_i x_i + T\sum x_i \ln x_i \tag{2.8}$$

Esta modificación permite la solución de un problema de programación no lineal con un valor T suficientemente pequeño. Esta última expresión asegura que la solución no tenga mínimos locales, sino una única solución global, esto es por la introducción del término $T\sum x_i \ln x_i$ que es una función estrictamente convexa, y por lo tanto no tendrá soluciones locales, sino una solución global (Apéndice C).

Aplicando el método de los multiplicadores de Lagrange a la función objetivo $F(x_i)$, tenemos que la función de Lagrange asociada a $F(x_i)$ queda como:

$$L = -\sum_{i=1}^{n} s_i x_i + T\left[\sum_{i=1}^{n} x_i \ln x_i + (p-1)\left(-\sum_{i=1}^{n} x_i + 1\right)\right] \tag{2.9}$$

Donde p es el multiplicador de Lagrange; luego se deriva parcialmente la función de Lagrange con respecto a x_i y p, tal y como lo muestran las ecuaciones (2.10) y (2.11)

$$\frac{\partial L}{\partial x_i} = -s_i + T(\ln x_i - p) \tag{2.10}$$

$$\frac{1}{T}\frac{\partial L}{\partial p} = -\sum_{i=1}^{n} x_i + 1 \tag{2.11}$$

Del Apéndice C sabemos que para obtener un punto global (ya sea un mínimo o un máximo), las ecuaciones (2.10) y (2.11) deben de igualarse a cero, cuya solución está dada por las ecuaciones (2.12) y (2.13):

$$x_i = \exp\left(\frac{s_i}{T} + p\right) \tag{2.12}$$

$$\frac{dp}{dt} = -\sum_{i=1}^{n} x_i + 1 \tag{2.13}$$

De esta forma, tomando del conjunto $\{x_i\}$ la que converja a "uno" será la solución del problema del *Ganador Toma Todo.* Las expresiones (2.12) y (2.13) pueden ser implementadas electrónicamente. La Tabla 2.1 muestra las equivalencias de los términos de las ecuaciones (2.9) y (2.10) en variables eléctricas, las cuales se relacionan como lo muestra la Figura 2.11.

Término	Variable eléctrica
p	V_p
s_i	V_s
x_i	I_{Xn}

Tabla 2.1

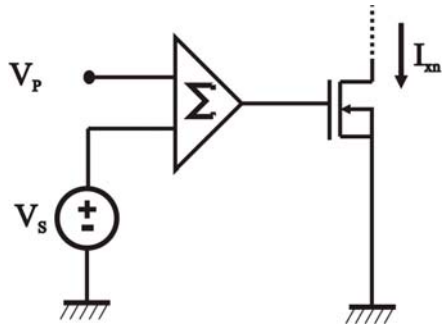


Figura 2.11. Implementación electrónica de (2.9)

Donde el transistor trabaja bajo el régimen de inversión débil, esto es para generar el término exponencial de la ecuación (2.12) y el bloque de suma puede ser implementado con un transistor MOS de compuerta flotante de dos entradas. La ecuación (2.13) se puede implementar como se muestra en la figura 2.12

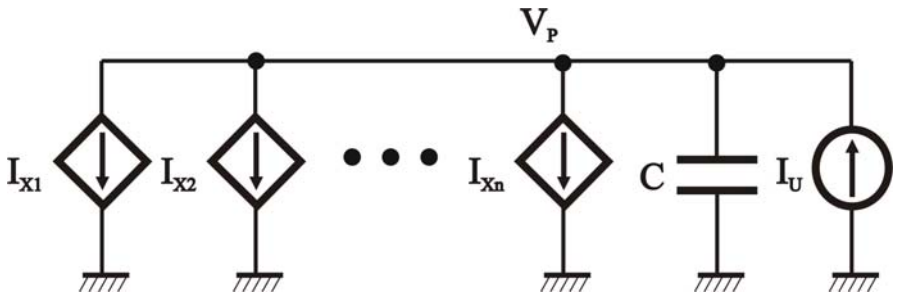


Figura 2.12. Implementación electrónica de (2.13)

Donde la corriente del capacitor está dada por:

$$I_c = C\frac{dV_c}{dt} \tag{2.14}$$

Por análisis de Kirchoff sabemos que:

$$I_c = I_U - \sum_{i=1}^{n} I_i \tag{2.15}$$

Si $V_c = V_P$ e igualando las ecuaciones anteriores tenemos:

$$\frac{dV_P}{dt} = \frac{1}{C}\left(I_U - \sum_{i=1}^{n} I_i \right) \tag{2.16}$$

La ecuación (2.16) demuestra que con el circuito de la figura 2.12 se puede implementar la ecuación 2.10. De esta manera, la figura 2.13 muestra un esquema general de la implementación electrónica del sistema anterior, para un vector de entrada de n elementos.

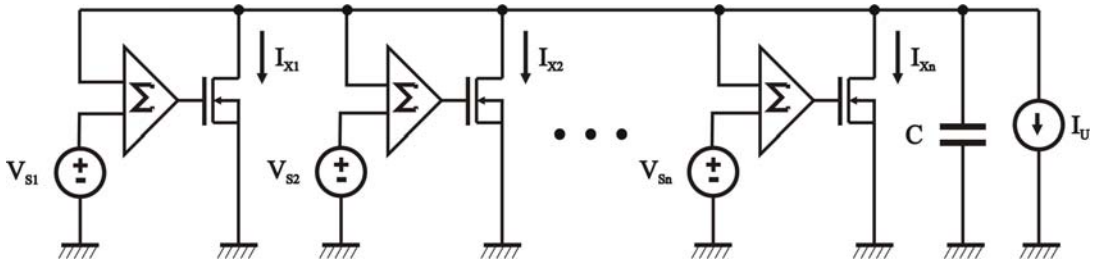


Figura 2.13 Esquema general de la implementación de las ecuaciones (2.9) y (2.10) simultáneamente para resolver el problema del *Ganador Toma Todo*

El valor I_U al cual convergerán las soluciones del problema del *Ganador Toma Todo* no será una matriz de ceros y unos como en el planteamiento inicial, sino estará dada en términos de la corriente de la fuente independiente I_U; entonces, la matriz solución será tal que los elementos no asignados estarán representados por una corriente de salida de 0A, mientras que los elementos asignados estarán representados por una corriente de salida igual a I_U, ya que los transistores estarán trabajando en la región de subumbral, este valor estará en el rango de decenas o centenas de nanoamperes.
Los voltajes Vs_i, son los voltajes del vector de entrada y deberán de ser ajustados a los niveles de voltaje que aseguren que el transistor trabaje en la región de subumbral.

K. Urahama[10] demuestra que la elección de T juega un papel muy importante en la solución del sistema. Para valores de T grandes, no se podrán discriminar las corrientes que representan la solución de las que tienden a cero. Idealmente T debe de ser 0 para que esto se logre, sin embargo en la práctica esto sería imposible, por lo tanto solo se buscan valores de T lo suficientemente pequeños para que las corrientes-solución puedan converger a valores cercanos a la corriente unitaria I_U. Esta cuestión y su implementación se abordarán en el siguiente capítulo.

Cabe mencionar que si se requiere la implementación del circuito LTA (Loser-Take-All o el Perdedor Toma Todo) [12], es necesario tratar el problema del *Ganador Toma Todo* como

un problema de minimización, esto se logra multiplicando por (-1) tanto la función objetivo como las restricciones.

2.5 Solución del *Problema de Asignación* utilizando el método de los multiplicadores de Lagrange

Ya se ha abordado la definición matemática del problema del *Ganador Toma Todo* y se partirá de la misma metodología, es decir utilizando los multiplicadores de Lagrange y tratando el problema como un problema de minimización; de esta manera, se agrega un término de entropía a la función objetivo (2.1) del problema de asignación:

$$\min \sum_{i,j} s_{ij} x_{ij} - T \sum \sum x_{ij} \ln x_{ij} \tag{2.17}$$

Así se obtiene la función Lagrangiana:

$$L = \sum_{i=1}^{n} \sum_{j=1}^{n} s_{ij} x_{ij} + T \left[\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \ln x_{ij} + \sum_{i=1}^{n} \left(p_i - \frac{1}{2} \right) \left(\sum_{j=1}^{n} x_{ij} - 1 \right) + \sum_{j=1}^{n} \left(q_j - \frac{1}{2} \right) \left(\sum_{i=1}^{n} x_{ij} - 1 \right) \right] \tag{2.18}$$

Donde p_i y q_j son los multiplicadores de Lagrange, obsérvese que el número de multiplicadores de Lagrange depende del tamaño de la matriz de entrada, por cada fila y columna de la matriz habrá un multiplicador de Lagrange, entonces los multiplicadores de Lagrange serán igual a *2n* de una matriz de costos de $n \times n$. Para obtener la solución es necesario derivar la ecuación (2.15) en función de x_{ij}, p_i, q_j e igualar con cero, quedando el siguiente sistema de ecuaciones diferenciales:

$$\frac{\partial L}{\partial x_{ij}} = s_{ij} + T\left(\ln x_{ij} + p_i + q_j \right) = 0 \tag{2.19}$$

$$\frac{1}{T} \frac{\partial L}{\partial p_i} = \sum_{j=1}^{n} x_{ij} - 1 = 0 \tag{2.20}$$

$$\frac{1}{T} \frac{\partial L}{\partial q_j} = \sum_{i=1}^{n} x_{ij} - 1 = 0 \tag{2.21}$$

La solución para las ecuaciones (2.19), (2.20) y (2.21) está dada por las ecuaciones (2.22), (2.23) y (2.24) respectivamente:

$$x_{ij} = \exp - \left(\frac{s_{ij}}{T} + p_i + q_j \right) \tag{2.22}$$

$$\frac{dp_i}{dt} = \sum_{j=1}^{n} x_{ij} - 1 \tag{2.23}$$

$$\frac{dq_j}{dt} = \sum_{i=1}^{n} x_{ij} - 1 \tag{2.24}$$

La tabla 2.2 muestra la variable eléctrica asignada a los términos de las expresiones anteriores.

Término	Variable eléctrica
p_i	V_{Pi}
q_j	V_{Qj}
s_{ij}	V_{Sij}
x_{ij}	I_{Xij}

Tabla 2.2

De esta forma, las x_{ij}´s que convergen a uno serán la solución del problema de asignación. Las expresiones (2.22), (2.23) y (2.24) pueden ser implementadas electrónicamente. La figura 2.14 muestra la idea principal para la implementación de la ecuación (2.19).

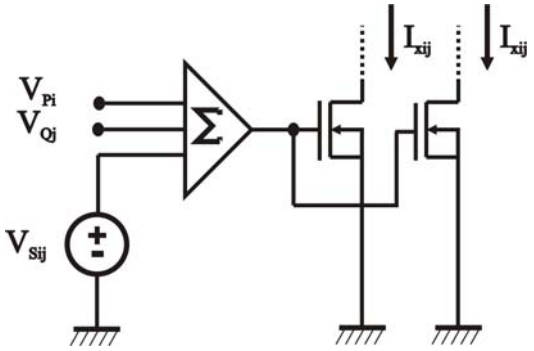


Figura 2.14 Implementación electrónica de (2.19)

Las ecuaciones 2.23 y 2.24 pueden ser implementadas de la misma forma como en el circuito del *Ganador Toma Todo*, solo que en este caso habrá dos fuentes dependientes de la corriente de cada celda x_{ij}, y el sentido de las corrientes cambiará.
La figura 2.15 muestra el diagrama para una matriz de *i x j* elementos, donde *i = j*.

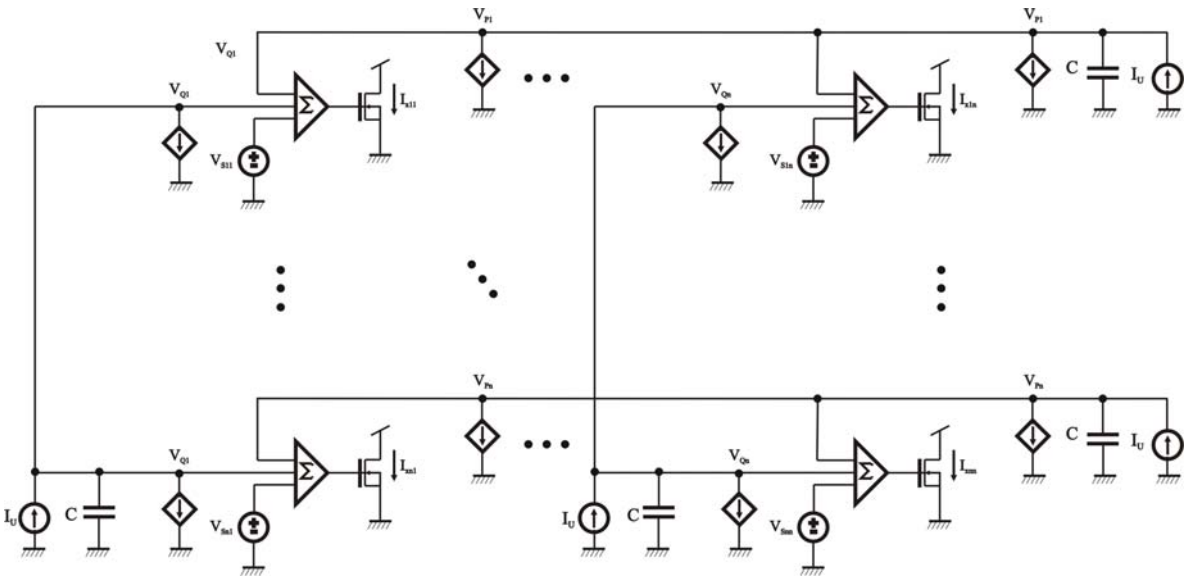


Figura 2.15. Implementación electrónica para la solución del problema de asignación de una matriz de *i x j* elementos (donde *i = j*)

Del esquema anterior, la corriente unitaria I_U estará dada en el orden de nanoamperios pues los transistores deberán de trabajar en el régimen de inversión débil para asegurar la implementación de la función exponencial.

Como se mencionó anteriormente, la convergencia de la red depende del valor de T. Entre mas pequeño sea el valor de T, el circuito convergerá a la solución óptima, esto se muestra en la figura 2.16.

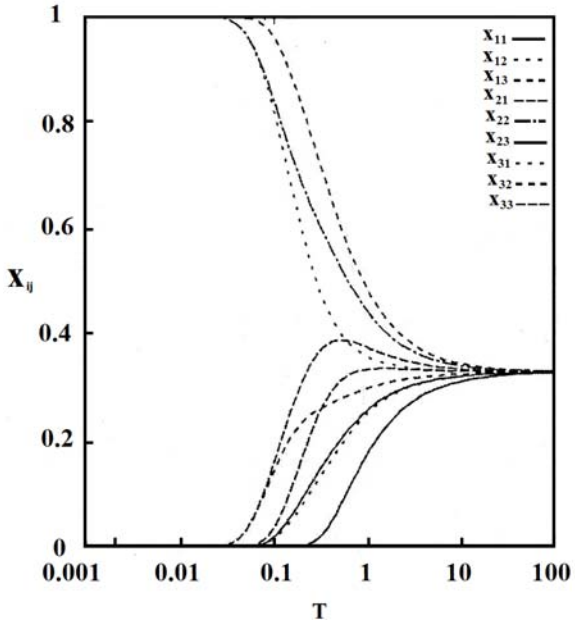


Figura 2.16. Ejemplo de la solución para una matriz de 3x3 para distintos valores de T

Para valores grandes de T, las soluciones tienden a un mismo valor o se mezclan sin hacer distinción de cuál es la solución. Para discriminar los valores solución de la matriz, el valor de T tiende a cero (esto para un caso ideal), en el caso práctico, sólo se busca un valor de T lo suficientemente pequeño como para poder discriminar las corrientes de salida.

En este capítulo se abordaron distintas soluciones al *Problema de Asignación* para su implementación en circuitos CMOS. Utilizando la solución con multiplicadores de Lagrange, se puede lograr una implementación sencilla en circuitos eléctricos, por lo cual se utilizará ese método para nuestro objetivo.

REFERENCIAS

[1] Roy D. Rosner, "Packet Switching: Tomorrow´s Communications Today", Lifetime Learning Publications, 1982.

[2] Christos H. Papadimitriou, Keneth Steiglitz, "Combinatorial Optimization and Complexity", General Publishing Company Ltd., 1982.

[3] J.A. Bondy, U.S.R. Murty, "Graph Theory with Applications", the Macmillian Press Ltd., 1976.

[4] J.J. Hopfield and D.W. Tank, "Neural computations of decisions in optimization problems" Biological Cybernetic ,vol. 52, pp. 141-152

[5] J. Wang, "Analog neural network for solving the assignment problem", Electronic Letters, vol. 28, issue 11, 1992, pp. 1047-1050.

[6] K. J. Symington, A.J. Waddie, M.R. Taghizadeh, J.F. Snowdon "A neural-network packet switching controller: scalability, performance, and network optimization", IEEE Transactions of Neural Networks, vol. 14, no.1, 2003, pp. 28-34.

[7] Urahama K., "Analog method for combinatorial optimization problems", IEICE Trans. Fundamentals, 1994, pp. 302-308.

[8] Urahama K., "Analog circuit for solving assignment problems", IEEE Trans. Circuits Syst., vol.41, 1994, pp. 426-429.

[9] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of O(n) complexity," in Advances in neural information processing systems, D. S. Touretzky, Ed., Vol. 2, pp. 703—711, Morgan Kaufmann, San Mateo, CA, 1989

[10] B. Sekerkiran and U. Cilingiroglu, "A CMOS K-winner-take-all circuit with O(n) complexity," IEEE Transactions on circuits and systems—II, Analog and Digital Signal processing, Vol. 46, No. 1, January 1999, pp. 1-5.

[11] N. Donckers, C. Dualibe, and M. Verleysen, "A current-mode CMOS loser-take-all with minimum function for neural computations," in Proceedings of IEEE International Symposium on Circuits and Systems**,** Vol. 4, pp. 415—418, Geneva, Switzerland, May 2000.

Antes de abordar la implementación del circuito del Asignación, se describirá la implementación del circuito del *Ganador Toma Todo* que servirá posteriormente para la implementación del circuito del *Problema de Asignación*.

3.1 Implementación CMOS del circuito del *Ganador Toma Todo*

El esquema general para implementar el circuito del *Ganador Toma Todo* de n entradas se muestra en la Figura 3.1, donde el diseño de las celdas se verá más adelante.

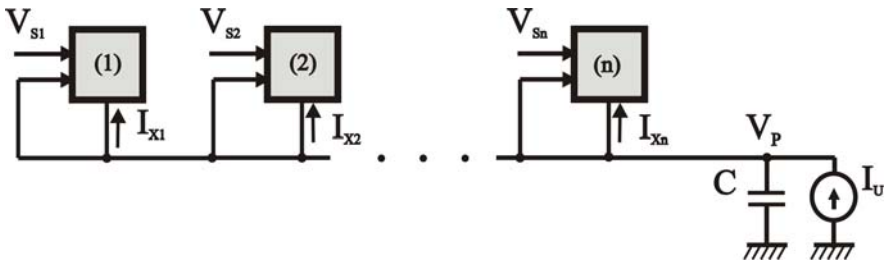


Figura 3.1 Esquema general de la implementación del circuito del *Ganador Toma Todo*

Como se mencionó en el capítulo anterior, para lograr la convergencia en la red es necesario que el término T sea lo más pequeño posible, o visto de otra manera, el voltaje V_{si} estará multiplicado por una ganancia, como se muestra en la Figura 3.2.

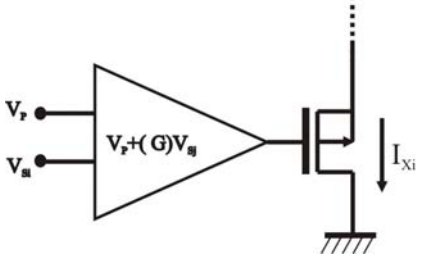


Figura 3.2

Cuando utilizamos el FGMOSFET como bloque sumador, dicha suma se multiplica por un factor de atenuación debido al factor de acoplamiento capacitivo, el cual siempre será menor que la unidad; es decir, entre mas entradas tenga el FGMOSFET, el voltaje en la compuerta flotante se atenuará más, esto es contrario a la condición de que el término T tienda a cero, provocando que la red no converja a los valores esperados. De esta manera se propone el uso de circuitos translineales que son muy útiles a la hora de implementar

diversas funciones no lineales, como se vió en el capítulo 1. Para este caso, se hizo uso de una configuración de espejos de corriente (que básicamente son circuitos translineales), agregando un FGMOSFET de múltiples entradas. Gracias a esto podemos obtener la ganancia deseada, como el siguiente análisis lo indica. Primero comenzaremos analizando el circuito de la Figura 3.3 que representa la celda n de la Figura 3.1.

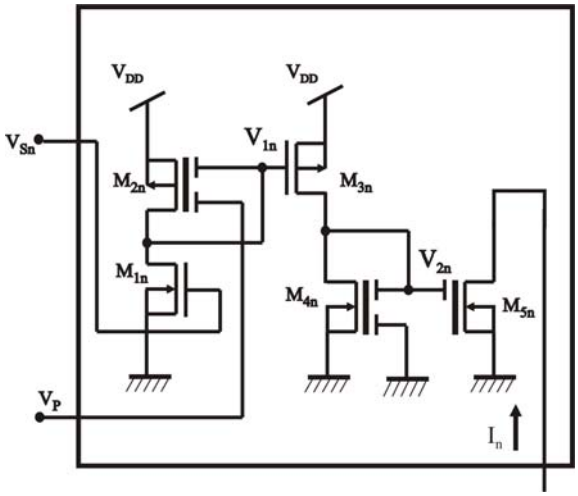


Figura 3.3

La tabla 3.1 muestra el significado de cada variable usada en el desarrollo matemático del circuito de la Figura 3.3

Variable	Significado
V_{Sn}	Voltaje del vector de entrada V_S
V_P	Voltaje proporcional al valor del multiplicador de Lagrange p
C_{1_M2}	Capacitor de entrada 1 del transistor de compuerta flotante M2
C_{2_M2}	Capacitor de entrada 2 del transistor de compuerta flotante M2
C_{T_M2}	Capacitancia total del transistor de compuerta flotante M2
C_{1_M4}	Capacitor de entrada 1 del transistor de compuerta flotante M4
C_{2_M4}	Capacitor de entrada 2 del transistor de compuerta flotante M4
C_{T_M4}	Capacitancia total del transistor de compuerta flotante M4
$I_{0M1,2,3,4,5}$	Factor de corriente de subumbral de los transistores M1, M2, M3, M4 y M5

Tabla 3.1

Para el siguiente análisis se considera que $C_{i\_Mn} >> C_{GD}$ y $Q_{FG} = 0$. La corriente que pasa por el transistor M_{1n}, está dada por la siguiente expresión:

$$I_1 = I_{0M1} \exp\left(\frac{V_{Sn}}{nv_T}\right) \tag{3.1}$$

Del transistor M_{2n} tenemos:

$$I_2 = I_{0M2} \exp\left[\frac{1}{nv_T}(V_{DD}) - \frac{1}{nv_T}\left((V_{1n})\frac{C_{1\_M2}}{C_{T\_M2}} + (V_P)\frac{C_{2\_M2}}{C_{T\_M2}}\right)\right] \tag{3.2}$$

Igualando las ecuaciones (3.1) y (3.2) y obteniendo el logaritmo natural en ambos lados tenemos:

$$\ln\left(\frac{I_{0M1}}{I_{0M2}}\right) + \left(\frac{V_{Sn}}{nv_T}\right) = \frac{1}{nv_T}(V_{DD}) - \frac{1}{nv_T}\left((V_{1n})\frac{C_{1\_M2}}{C_{T\_M2}} + (V_P)\frac{C_{2\_M2}}{C_{T\_M2}}\right) \tag{3.3}$$

Considerando que $I_{0M1} \approx I_{0M2}$, tenemos que $\ln\left(\frac{I_{0M1}}{I_{0M2}}\right) \approx 0$ y despejando V_{1n} de la ecuación (3.3) queda:

$$V_{1n} = -\left[V_P + \left(\frac{C_{T\_M2}}{C_{1\_M2}}\right)V_{Sn}\right] + \left(\frac{C_{T\_M2}}{C_{1\_M2}}\right)V_{DD} \tag{3.4}$$

La ecuación anterior muestra que el voltaje de entrada V_{Sn} está multiplicado por un factor igual a $\frac{C_{T\_M2}}{C_{1\_M2}}$ que es mayor que 1 ya que $C_{T\_M2} > C_{1\_M2}$, por lo tanto V_{Sn} es multiplicado por una ganancia, tal y como la ecuación 2.12 lo requiere.

La corriente de drenador en M_{3n} está dada por:

$$I_3 = I_{0M3} \exp\left(\frac{(V_{DD} - V_{1n})}{nv_T}\right) \tag{3.5}$$

Sin embargo, no se puede utilizar la corriente de salida del transistor M_{3n} para la implementación completa, pues es necesario cambiar el sentido de la corriente; eso se logra con otro espejo de corriente. El segundo espejo de corriente se aprovechará para agregar otra etapa extra de ganancia para asegurar la convergencia de la red. De la misma manera que en el análisis anterior, se utilizará otro transistor de compuerta flotante de dos entradas, con el fin de multiplicar la ganancia obtenida por una segunda ganancia.

La corriente que pasa por el transistor M_{4n} está dada por:

$$I_4 = I_{0M4} \exp\left[\frac{1}{nv_T}\left(\left(V_{2n}\right)\frac{C_{2\_M4}}{C_{T\_M4}}\right)\right] \tag{3.6}$$

Igualando las expresiones (3.5) y (3.6) tenemos:

$$\frac{I_{0M3}}{I_{0M4}}\exp\left(\frac{V_{DD}-V_{1n}}{nU_T}\right) = \exp\left[\frac{1}{nv_T}\left(V_{2n}\frac{C_{2\_M4}}{C_{T\_M4}}\right)\right] \tag{3.7}$$

Si consideramos que $\ln\left(\frac{I_{0M3}}{I_{0M4}}\right) \approx 0$ tenemos el siguiente desarrollo:

$$\frac{V_{DD}-V_{1n}}{nv_T} = \frac{1}{nv_T}\left(\frac{C_{2\_M4}}{C_{T\_M4}}V_{2n}\right) \tag{3.8}$$

Despejando V_{2n} de (3.8) queda:

$$V_{2n} = \frac{C_{T\_M4}}{C_{2\_M4}}\left(V_{DD}\right) - \frac{C_{T\_M4}}{C_{2\_M4}}\left(V_{1n}\right) \tag{3.9}$$

Luego, sabemos que la corriente I_5 está dada por:

$$I_5 = I_{0M5}\exp\left(\frac{1}{nv_T}\left(\frac{C_{1\_M5}}{C_{T\_M5}}\right)\left(V_{2n}\right)\right) \tag{3.10}$$

Sustituyendo la ecuación (3.4) en (3.9) tenemos que:

$$V_{2n}=\left(\frac{C_{T\_M4}}{C_{2\_M4}}\right)(V_{DD})\left[1-\left(\frac{C_{T\_M2}}{C_{1\_M2}}\right)\right]+\left(\frac{C_{T\_M4}}{C_{2\_M4}}\right)(V_P)+\left(\frac{C_{T\_M4}}{C_{2\_M4}}\right)\left(\frac{C_{T\_M2}}{C_{1\_M2}}\right)V_{Sn}$$

(3.11)

Finalmente, si sustituimos (3.11) en (3.10) y considerando que

$$V_k=\left(\frac{C_{1\_M5}}{C_{T\_M5}}\right)\left(\frac{C_{T\_M4}}{C_{2\_M4}}\right)(V_{DD})\left[1-\left(\frac{C_{T\_M2}}{C_{1\_M2}}\right)\right]$$

Tenemos que la corriente en el transistor M_{5n} está dada por la expresión:

$$I_5=I_{0M5}\exp\frac{1}{nv_T}\left[V_k+\left(\frac{C_{1\_M5}}{C_{T\_M5}}\right)\left(\frac{C_{T\_M4}}{C_{2\_M4}}\right)(V_P)+\left(\frac{C_{1\_M5}}{C_{T\_M5}}\right)\left(\frac{C_{T\_M4}}{C_{2\_M4}}\right)\left(\frac{C_{T\_M2}}{C_{1\_M2}}\right)V_{Sn}\right]$$

(3.12)

De donde $I_5=I_{X5}$. De (3.12), V_{Sn} está multiplicado por un factor igual a $\left(\frac{C_{1\_M5}}{C_{T\_M5}}\right)\left(\frac{C_{T\_M4}}{C_{2\_M4}}\right)\left(\frac{C_{T\_M2}}{C_{1\_M2}}\right)$, el cual es mucho mayor que 1, aumentando la ganancia a la cual debe de estar multiplicado V_{Sn}.

Finalmente, en la Figura 3.4 se muestra la implementación final del circuito del *Ganador Toma Todo* para un vector de n elementos.

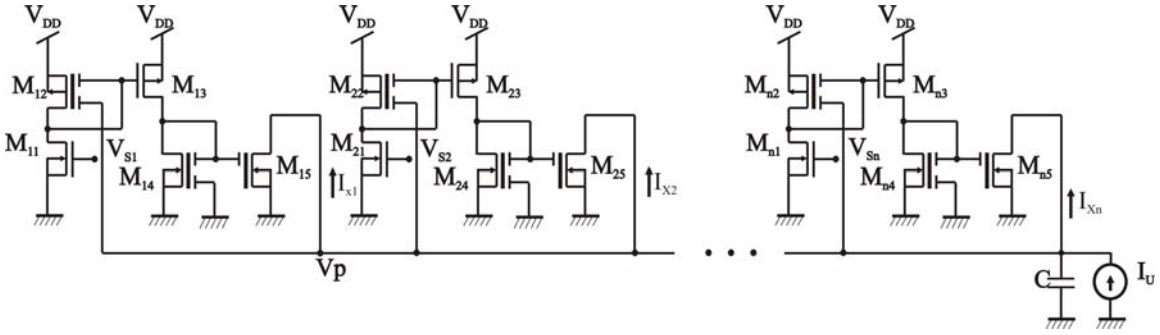


Figura 3.4 Circuito del *Ganador Toma Todo* de n elementos

Si se requiere la implementación del circuito LTA (Looser-Take-All o el perdedor gana todo) es necesario definir el problema del *Ganador Toma Todo* (problema de maximización) como un problema de minimización; como se mencionó en el capítulo anterior, el resultado no es mas que el intercambio de los transistores PMOS por NMOS y viceversa.

3.1.1 Simulación del circuito del *Ganador Toma Todo*

Los resultados de la simulación se obtuvieron para un circuito del *Ganador Toma Todo* de 2, 4 y 8 entradas; las simulaciones fueron realizadas en PSPICE para una tecnología de 0.5um y la corriente-solución para todos lo casos es de 200nA. La tabla 3.2 muestra los valores de ancho y largo de canal de los transistores MOS utilizados en la simulación, así como el voltaje $V_{DD.}$ El modelo usado para la simulación del FGMOSFET es el mencionado en la sección 1.8 y se muestra su listado de PSPICE en el apéndice A.

Variable	**Valor usado**
W/L de los transistores de canal n	3/0.9
W/L de los transistores de canal p	6/0.9
Voltaje de polarización	3V

Tabla 3.2

- **Simulación de un vector de 2 elementos**

Vector de entrada: $\begin{bmatrix}0.54 & 0.575\end{bmatrix}$

Solución: $\begin{bmatrix}0 & 1\end{bmatrix}$

La figura 3.5 muestra la corriente de salida, donde la corriente I_2 representa la solución del sistema.

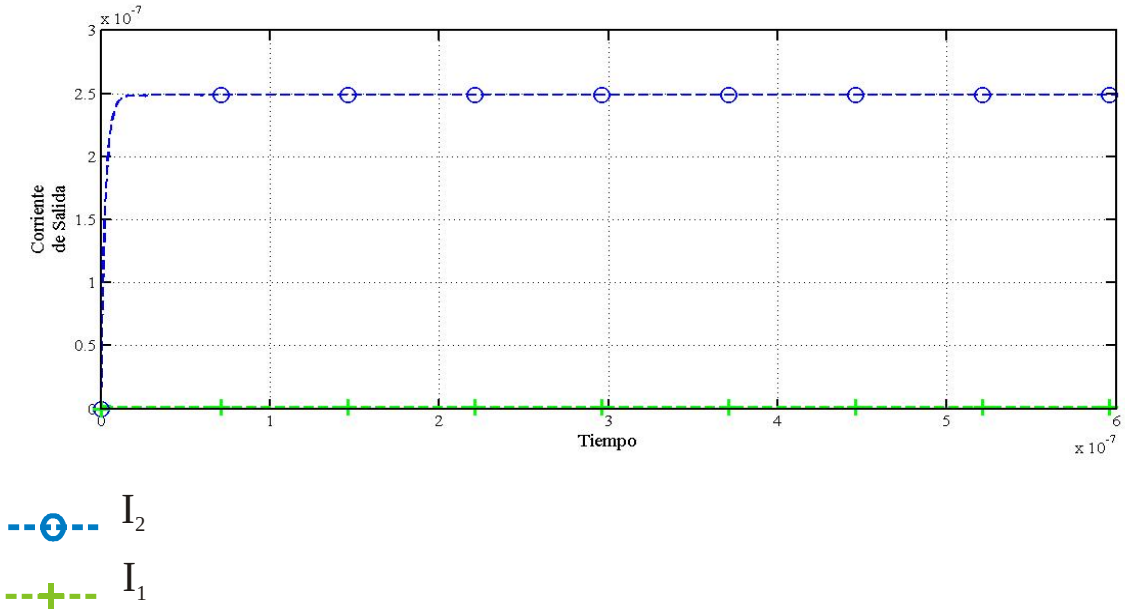


Figura 3.5 Solución del circuito del *Ganador Toma Todo* para un vector de 1x2

- **Simulación de un vector de 4 elementos**

Vector de entrada: $[0.54 \quad 0.565 \quad 0.495 \quad 0.585]$

Solución: $[0 \quad 0 \quad 0 \quad 1]$

La figura 3.6 muestra la corriente de salida, donde la corriente I_4 representa la solución.

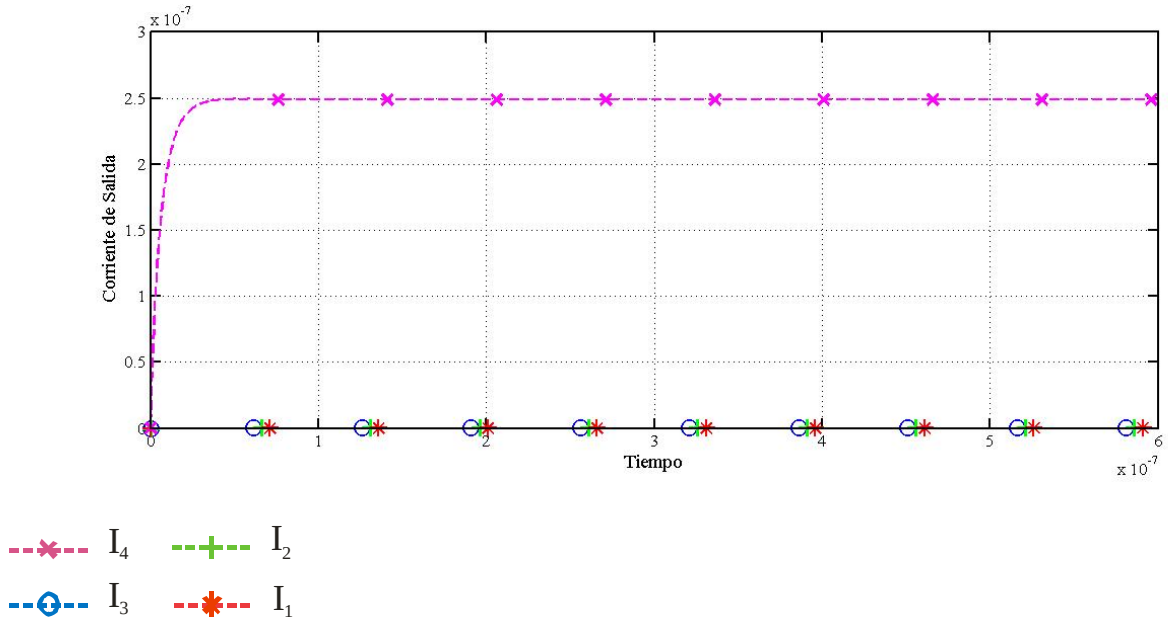


Figura 3.6 Solución del circuito del *Ganador Toma Todo* para un vector de 1x4

- **Simulación de un vector de 8 elementos**

Vector de entrada: $\begin{bmatrix}0.54 & 0.565 & 0.54 & 0.495 & 0.555 & 0.58 & 0.555 & 0.555\end{bmatrix}$

Solución: $\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}$

La figura 3.7 muestra la corriente de salida, donde la corriente I_6 representa la solución.

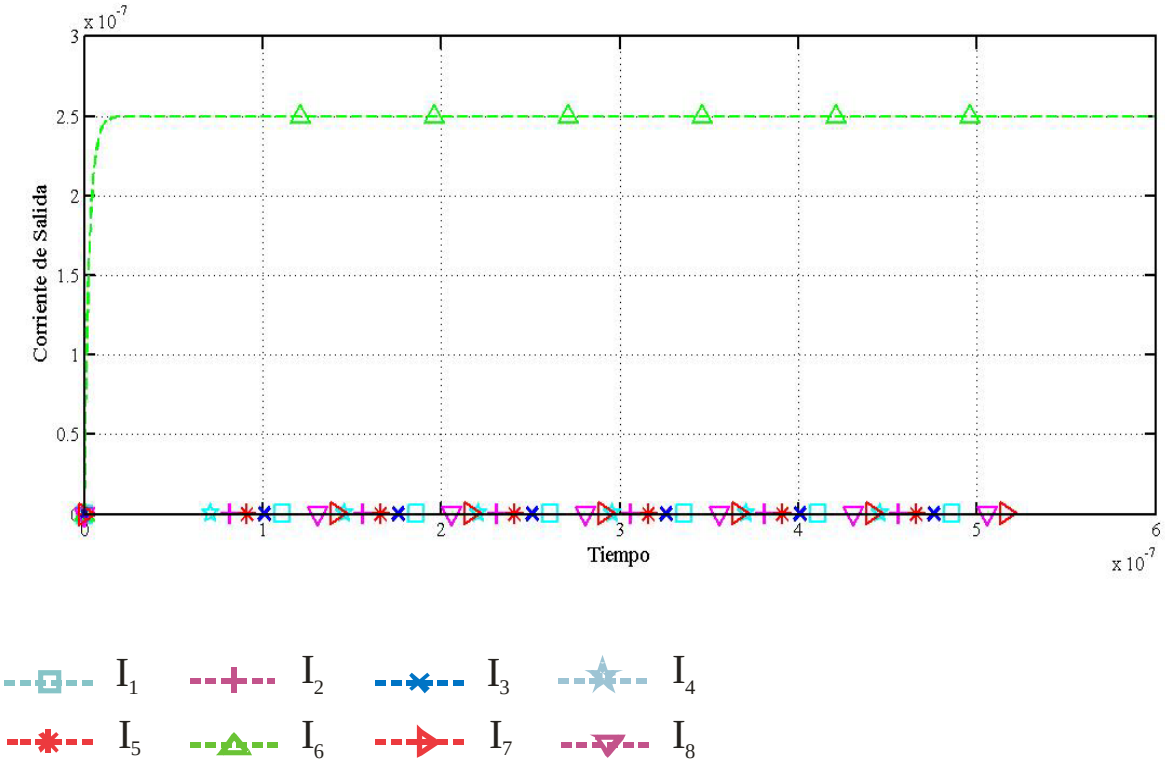


Figura 3.7 Solución del circuito del *Ganador Toma Todo* para un vector de 1x8

3.2 Implementación CMOS del circuito del *Problema de Asignación*

El esquema general del circuito de Asignación se muestra en la Fig. 3.8; el circuito de asignación puede ser visto como un circuito del *Ganador Toma Todo* en dos dimensiones.

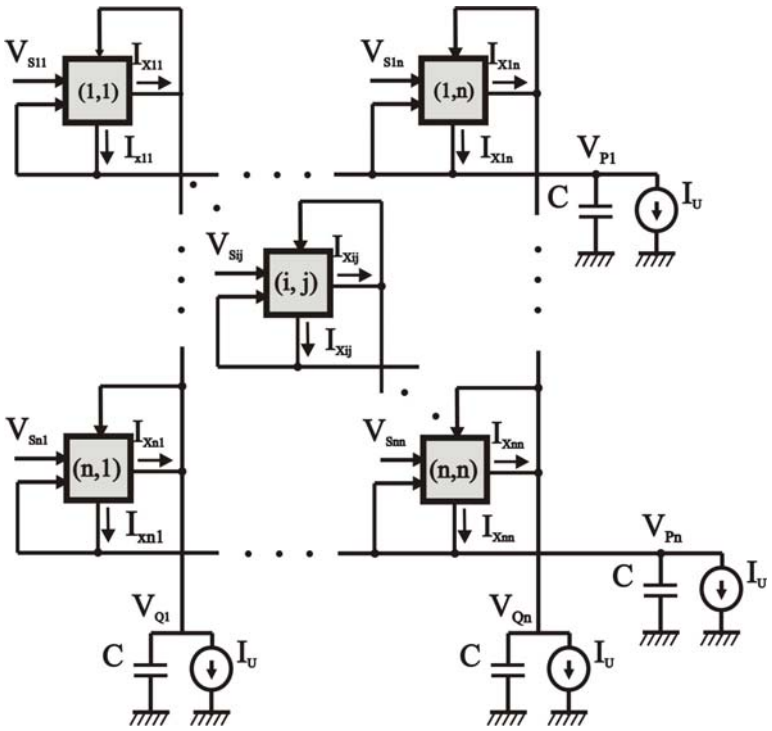


Figura 3.8 Arreglo de las celdas del circuito electrónico

De la misma manera que en el circuito del *Ganador Toma Todo*, no se puede utilizar directamente un FGMOSFET como bloque sumador, pues es necesario multiplicar el voltaje V_{sn} por una ganancia; la diferencia con el circuito del *Ganador Toma Todo*, es que el circuito de asignación requiere la suma de tres potenciales en vez de dos, como lo muestra la Figura 3.9:

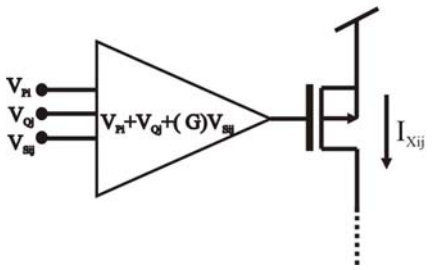


Figura 3.9 FGMOSFET de tres entradas con de sus entradas a un bloque de ganancia G

La Figura 3.10 muestra la celda utilizada para el circuito de Asignación, el cual es similar al utilizado para el circuito del *Ganador Toma Todo*, sólo que el transistor de compuerta flotante M2 debe de ser de tres entradas de acuerdo con la ecuación (2.22). Además, ya que estamos tratando de un problema se minimización se cambian los transistores NMOS por PMOS y viceversa.

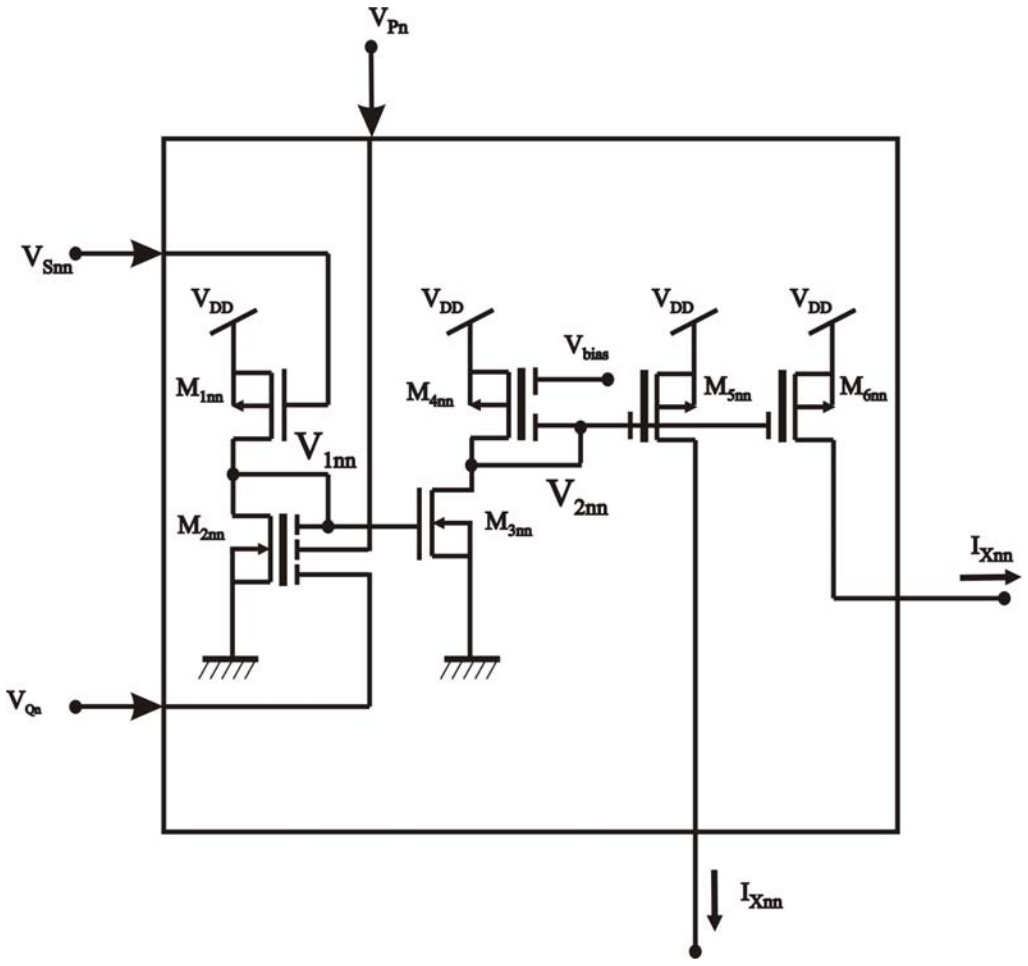


Figura 3.10 Celda propuesta

De acuerdo con la Figura 3.10, podemos tener el siguiente desarrollo matemático, el cual es similar al realizado para el circuito del *Ganador Toma Todo*, así que ya no se verá tan detalladamente; para la primera etapa, el voltaje V_{Snn} entra a un transistor de canal p con el fin de generar una corriente que polarizará al FGMOSFET de canal n de tres entradas, dos de las cuales servirán para la retroalimentación de los voltajes de V_{Pn} y de V_{Qn}.

La tabla 3.3 muestra el significado de cada variable usada en el desarrollo matemático del circuito de la Figura 3.10.

Variable	Significado
V_{Snn}	Voltaje de entrada de la celda (n,n)
V_{Pn}	Voltaje del multiplicador de Lagrange p
V_{Qn}	Voltaje del multiplicador de Lagrange q
C_{1_M2}	Capacitor de entrada 1 del transistor de compuerta flotante M_2
C_{2_M2}	Capacitor de entrada 2 del transistor de compuerta flotante M_2
C_{3_M2}	Capacitor de entrada 3 del transistor de compuerta flotante M_2
C_{T_M2}	Capacitancia total del transistor de compuerta flotante M_2
C_{1_M4}	Capacitor de entrada 1 del transistor de compuerta flotante M_4
C_{2_M4}	Capacitor de entrada 2 del transistor de compuerta flotante M_4
C_{T_M4}	Capacitancia total del transistor de compuerta flotante M_4
C_{1_M5}	Capacitor de entrada 1 del transistor de compuerta flotante M_5
C_{T_M5}	Capacitancia total del transistor de compuerta flotante M_5
C_{1_M6}	Capacitor de entrada 1 del transistor de compuerta flotante M_6
C_{T_M6}	Capacitancia total del transistor de compuerta flotante M_6
$I_{0M1,2,3,4,5}$	Corriente de subumbral de los transistores M_1, M_2, M_3, M_4 yM_5

Tabla 3.3

Para el análisis se considera que:

- $Q_{FG} = 0$
- $C_{GDij} << C_{1,2,3\_Mij}$

De esta manera el voltaje V_{1nn} está dado por:

$$V_{1nn} = \left(\frac{C_{T\_M2}}{C_{1\_M2}}\right) V_{DD} - V_{Pn} - V_{Qn} - \left(\frac{C_{T\_M2}}{C_{1\_M2}}\right) V_{Snn} \qquad (3.13)$$

Ahora, la corriente que pasa por el transistor M_{3nn} en subumbral está dada por:

$$I_3 = I_{0M3} \exp\left(\frac{1}{nv_T}\left(V_{1nn}\right) \right) \tag{3.14}$$

La expresión anterior muestra que el circuito de la figura 3.10 puede implementar la ecuación que interviene en la solución del *Problema de Asignación*. Como en el circuito del *Ganador Toma Todo*, en (3.13) se multiplica al voltaje de entrada V_{snn} por un factor mayor que uno, el cual proporciona una ganancia.

La segunda etapa del circuito, consiste en una ganancia adicional, para asegurar la convergencia de la red. Utilizando el mismo método usado anteriormente, se introduce un FGMOSFET canal p de dos entradas, una entrada sirve para la implementación del espejo de corriente y a la otra se le aplica un voltaje de polarización constante que sirve para el ajuste del nivel del voltaje de entrada del transistor M_{5nn}, además, sirve para asegurar que el transistor M_{5nn} trabaje siempre en la región de subumbral.

Siguiendo el mismo procedimiento anterior, se puede obtener el siguiente análisis de la Figura 3.10; despejando V_{2nn} tenemos:

$$V_{2nn} = \left(\frac{C_{T\_M2}}{C_{1\_M2}} \right)\left(\frac{C_{T\_M4}}{C_{1\_M4}} \right) V_{Snn} + \left(\frac{C_{T\_M4}}{C_{1\_M4}} \right) V_{Pn} + \left(\frac{C_{T\_M4}}{C_{1\_M4}} \right) V_{Qn} + V_{K1} \tag{3.15}$$

Donde:

$$V_{K1} = \left(\frac{C_{T\_M4}}{C_{1\_M4}} \right) V_{DD} \left(1 - \left(\frac{C_{T\_M2}}{C_{1\_M2}} \right) \right) - V_{bias}$$

Luego, sabemos que la corriente I_5 está dada por:

$$I_5 = I_{0M5} \exp\left(\frac{1}{nv_t} \left(V_{DD} - \left(V_{2nn}\right) \frac{C_{1\_M5}}{C_{T\_M5}} \right) \right) \tag{3.16}$$

Sustituyendo la ecuación (3.15) en (3.16) tenemos:

$$I_5 = I_{0M5}\exp\left\{\frac{1}{v_t}\left[\begin{array}{l} -\left(\frac{C_{T\_M4}}{C_{2\_M4}}\right)\left(\frac{C_{1\_M5}}{C_{T\_M5}}\right)\left(V_{Pnn}+V_{Qnn}\right)- \\ \left(\frac{C_{1\_M5}}{C_{T\_M5}}\right)\left(\frac{C_{T\_M4}}{C_{1\_M4}}\right)\left(\frac{C_{T\_M2}}{C_{1\_M2}}\right)V_{Snn}+V_{K2} \end{array}\right]\right\} \qquad (3.17)$$

Donde:

$V_{K2} = V_{DD} - V_{K1}$ y la corriente $I_5 = I_6 = I_{Xnn}$.

Los transistores M_{5nn} y M_{6nn} son de compuerta flotante de una entrada, esto con el objetivo de programar los voltajes de umbral.

Para la fuente de corriente constante, se utilizó un transistor canal n. En la figura 3.11 se muestra el circuito completo para una matriz de dos por dos elementos.

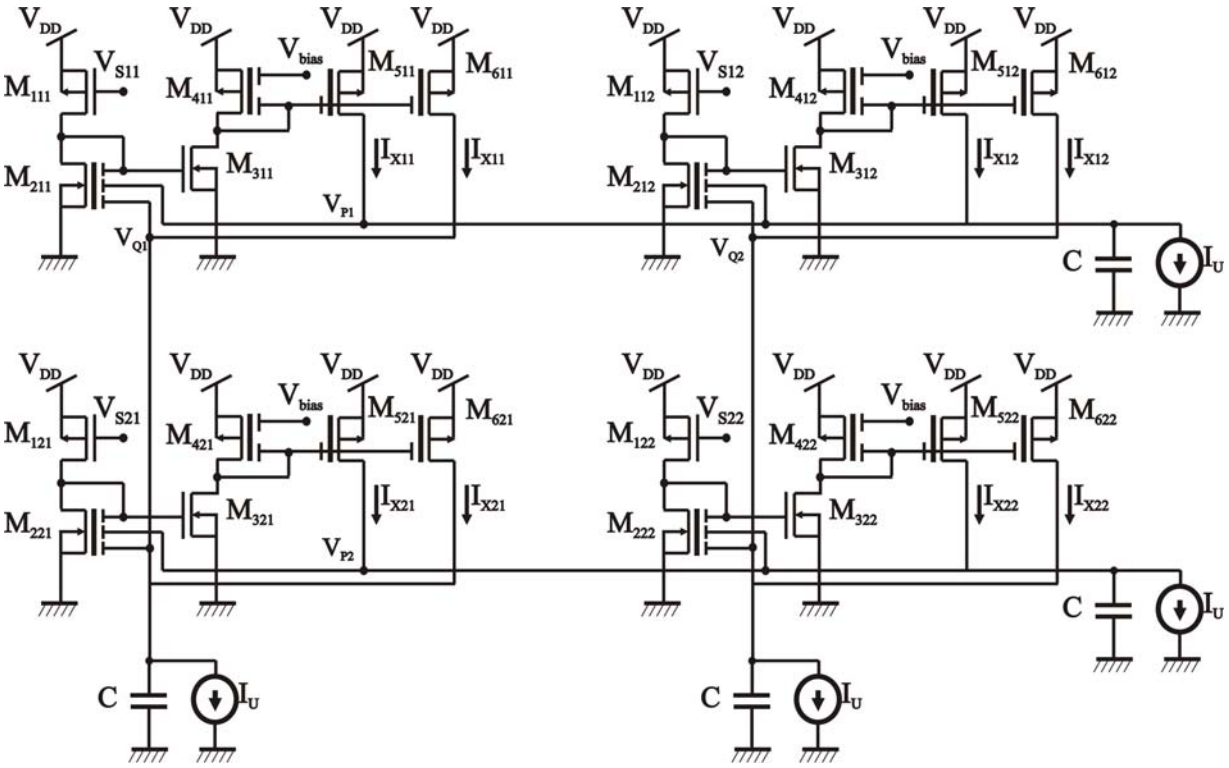


Figura 3.11 Diagrama electrónico para una matriz de 2x2

Más adelante se abordarán algunos de los métodos de programación de la compuerta flotante con el fin de que todos los transistores mantengan el mismo voltaje de umbral.

3.2.1 Simulación del circuito del *Problema de Asignación*

Se realizaron diversas simulaciones en PSPICE de matrices con elementos aleatorios de *n x n*, desde *n=2* hasta *n=8*; en este trabajo se presentan las simulaciones de matrices de 2x2, 4x4 y 8x8, tomadas del conjunto de simulaciones realizadas.

Se muestra la matriz de voltajes de entrada V_{Sn} y su respectiva matriz-solución obtenida a través del método Húngaro. La corriente a la cual converge la red es de 250nA y el tiempo que le toma a la red llegar a la solución es de menos de 10us. La tabla 3.4 muestra los valores de ancho y largo de canal de los transistores MOS utilizados en la simulación, así como el voltaje $V_{DD.}$

Variable	**Valor usado**
W/L de los transistores de canal n	5.4/0.9
W/L de los transistores de canal p	11.1/0.9
Voltaje de polarización	3V

Tabla 3.4

- **Simulación de una matriz de 2x2 elementos**

La Figura 3.12 muestra la simulación para resolver el *Problema de Asignación* dada la siguiente matriz de 2x2:

$$\begin{bmatrix} 2.157 & 2.195 \\ 2.168 & 2.178 \end{bmatrix}$$

Cuya solución es:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

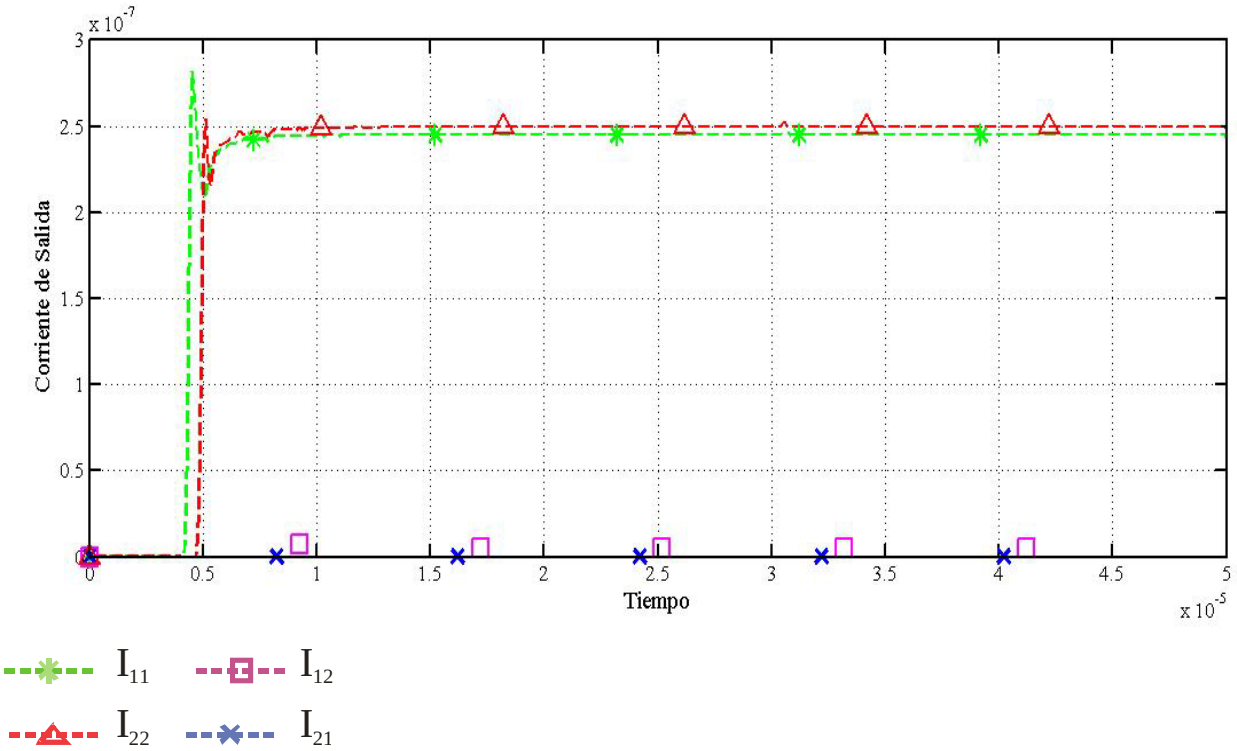


Figura 3.12 Resultado de la simulación en PSPICE para una matriz de 2x2

La solución representa las corrientes I_{11} e I_{22}, que concuerdan con la solución obtenida por el método Húngaro.

- **Simulación de una matriz de 4x4 elementos**

La siguiente simulación muestra la gráfica para la matriz de 4x4 siguiente

$$\begin{pmatrix} 2.16 & 2.165 & 2.18 & 2.175 \\ 2.14 & 2.11 & 2.145 & 2.13 \\ 2.14 & 2.13 & 2.135 & 2.115 \\ 2.165 & 2.15 & 2.11 & 2.15 \end{pmatrix}$$

La solución está dada por:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

La figura 3.13 muestra las corrientes de salida que convergen a aproximadamente a 250nA.

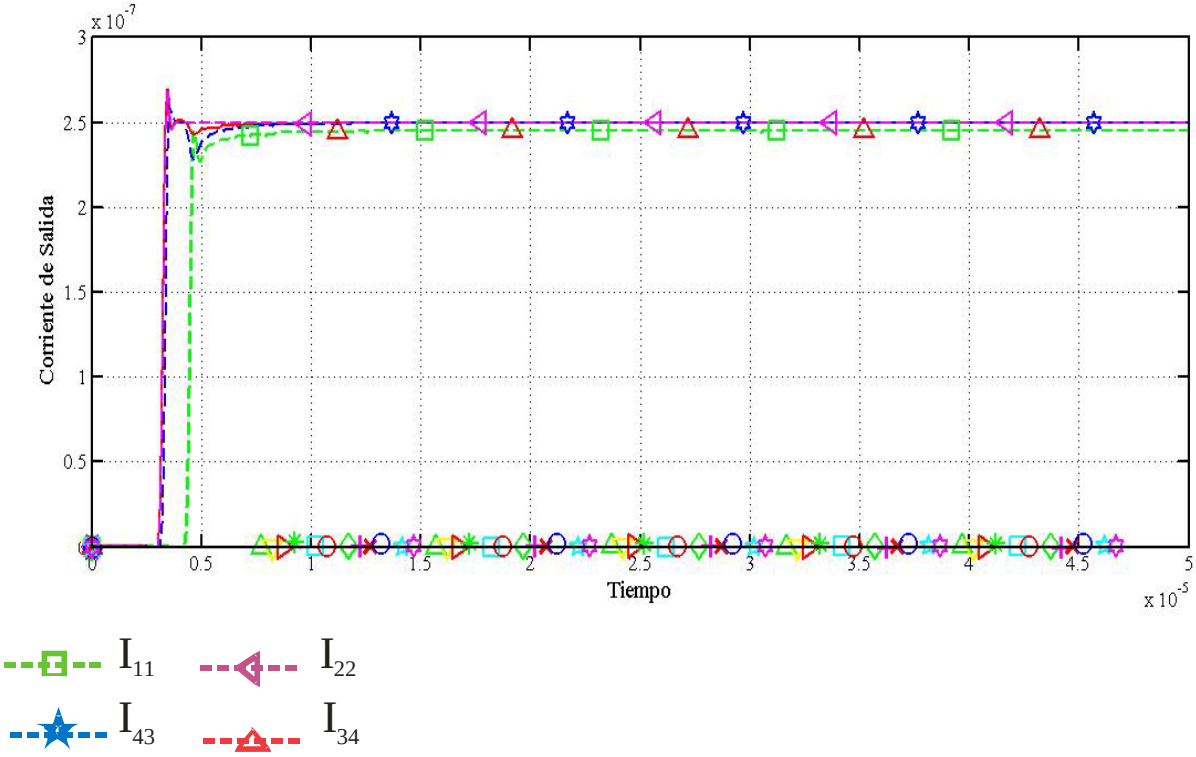


Figura 3.13 Simulación para una matriz de 4x4 en PSPICE

La solución representa las corrientes I_{11}, I_{22}, I_{43} e I_{34} que concuerdan con la solución obtenida por el método Húngaro

- **Simulación de una matriz de 8x8 elementos**

Finalmente se realizaron simulaciones para matrices de valores aleatorios de dimensión 8x8. La siguiente matriz se utiliza como ejemplo en este trabajo.

$$\begin{bmatrix} 2.115 & 2.12 & 2.155 & 2.105 & 2.11 & 2.19 & 2.19 & 2.175 \\ 2.185 & 2.175 & 2.195 & 2.13 & 2.13 & 2.155 & 2.155 & 2.145 \\ 2.185 & 2.125 & 2.14 & 2.12 & 2.12 & 2.19 & 2.19 & 2.175 \\ 2.16 & 2.1 & 2.17 & 2.115 & 2.185 & 2.13 & 2.115 & 2.13 \\ 2.1 & 2.155 & 2.185 & 2.12 & 2.155 & 2.125 & 2.19 & 2.13 \\ 2.14 & 2.19 & 2.105 & 2.145 & 2.16 & 2.14 & 2.11 & 2.17 \\ 2.14 & 2.13 & 2.2 & 2.155 & 2.155 & 2.155 & 2.14 & 2.16 \\ 2.19 & 2.185 & 2.11 & 2.11 & 2.19 & 2.17 & 2.19 & 2.105 \end{bmatrix}$$

Cuya solución es:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

El resultado de la simulación está dado por la figura 3.14:

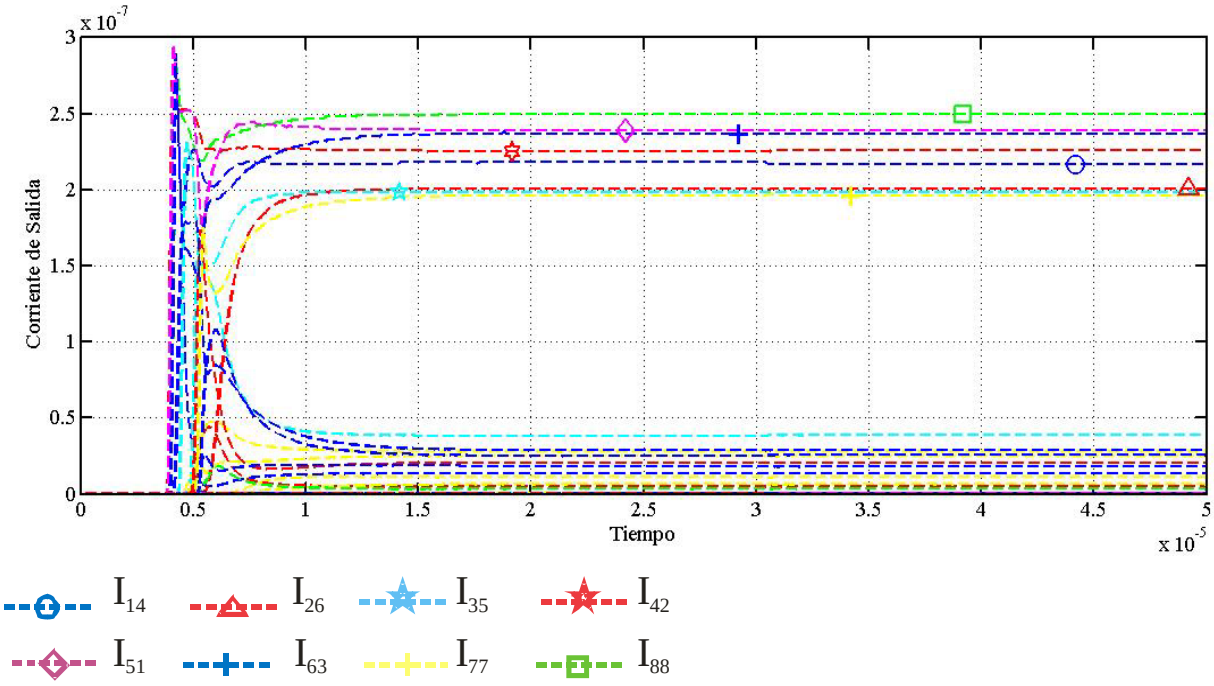


Figura 3.14 Simulación en PSPICE para una matriz de 8x8

De los resultados anteriores se muestra una convergencia satisfactoria de la red; sin embargo existe un inconveniente. A veces se tiene que no sólo es uno el camino óptimo o la solución óptima del problema de asignación, sino que existe más de una solución. En el ejemplo de la Fig. 3.15 para una matriz de 4x4 se muestra que se tienen dos soluciones:

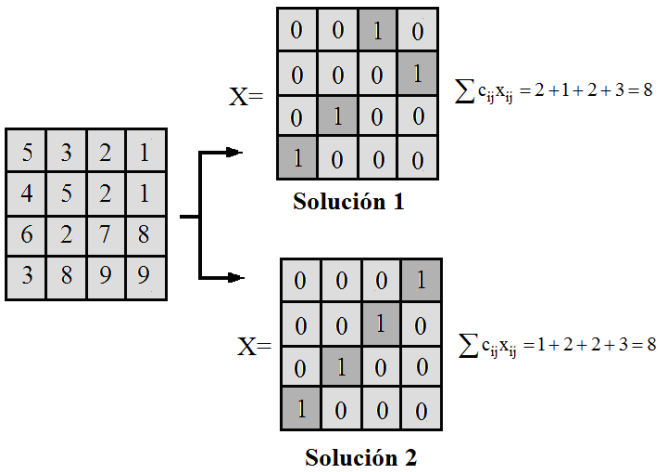


Figura 3.15 Una matriz con dos soluciones que minimizan $\sum c_{ij}x_{ij}$

Ambas soluciones representan el valor mínimo de $\sum c_{ij}x_{ij}$. Realizando la simulación en PSPICE de una matriz de este tipo, las corrientes I_{13}, I_{14}, I_{23} e I_{24}, que para el caso del ejemplo corresponden a las x_{13}, x_{14}, x_{23} y x_{24} convergen a 125nA (o sea la mitad de I_U, donde I_U es 250nA), como lo muestra la Figura 3.16.

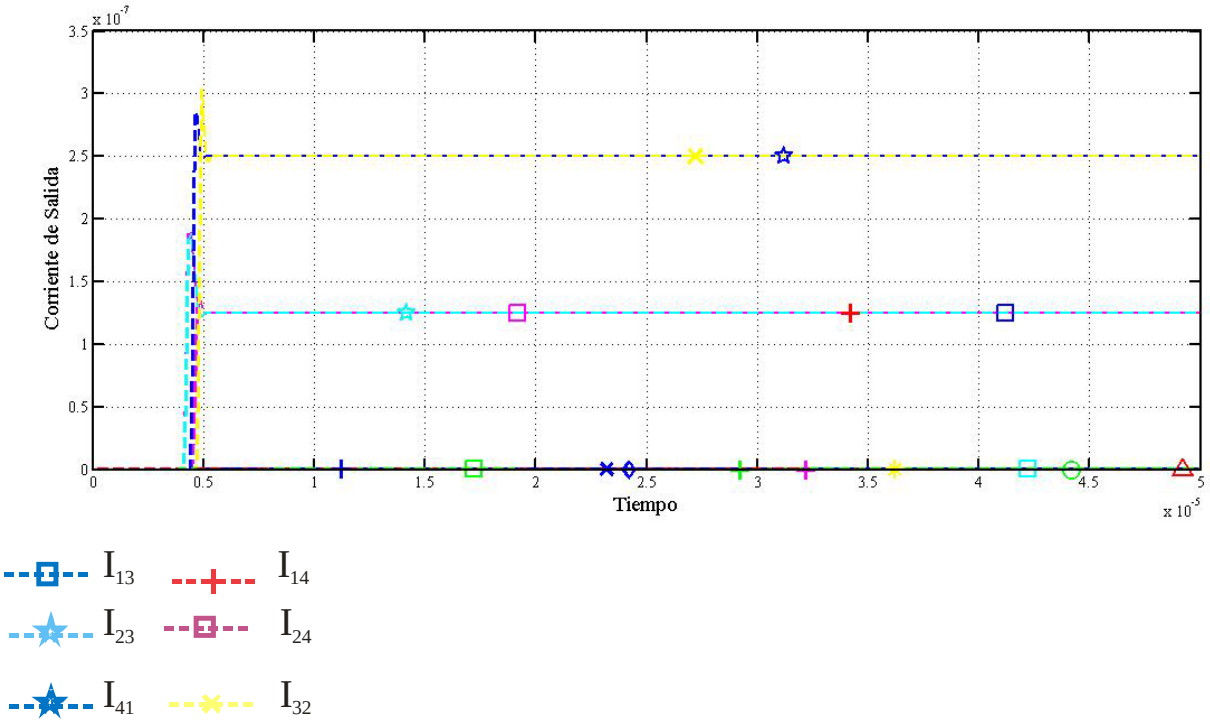


Figura 3.16 Simulación para una matriz de 4x4 con dos soluciones. Ambas soluciones convergen a 125nA, la mitad de la corriente de salida

4.1 Introducción

Ciertamente en los últimos años los circuitos analógicos han venido ganando más campos de aplicación que antes eran del dominio digital, como se abordó en los capítulos anteriores. De igual forma es bien conocida una de las principales desventajas que ha impedido a los diseñadores fabricar circuitos integrados analógicos con un alto grado de precisión, esto es, el desapareamiento entre dos transistores o "mismatch" que es el término anglosajón. El desapareamiento lo podemos definir como la variación de parámetros entre dos o más dispositivos idénticos lo cual provoca un comportamiento diferente ante estímulos iguales [1], provocando el mal funcionamiento del circuito analógico, inclusive puede llegar a afectar el desempeño de circuitos digitales [2]. El desapareamiento puede ser el resultado de efectos sistemáticos o de efectos aleatorios. Los primeros son causados por un mal diseño topológico o de variaciones incontrolables durante el proceso de fabricación. Los efectos aleatorios que originan el desapareamiento son provocados por variaciones en los parámetros tales como la concentración del dopado, la movilidad de los portadores de carga y el espesor de óxido.

De esta forma, se han propuesto diversos métodos para eliminar el desapareamiento entre dos transistores. Podemos diferenciar entre las usadas durante el diseño topológico del circuito integrado, con las técnicas de programación de la inyección de electrones calientes y el tuneleo Fowler-Nordheim, las cuales nos permiten establecer un mismo valor del voltaje de umbral para un arreglo de transistores.

4.2 Mecanismo de tuneleo Fowler-Nordheim

Para generar el mecanismo de tuneleo FN en un transistor es necesario modificar su estructura agregando un capacitor adicional en donde se aplicará el voltaje de tuneleo como se muestra en la Fig. 4.1, donde se ilustra el diseño topológico que se puede realizar en una tecnología de pozo n y dos polisilicios.

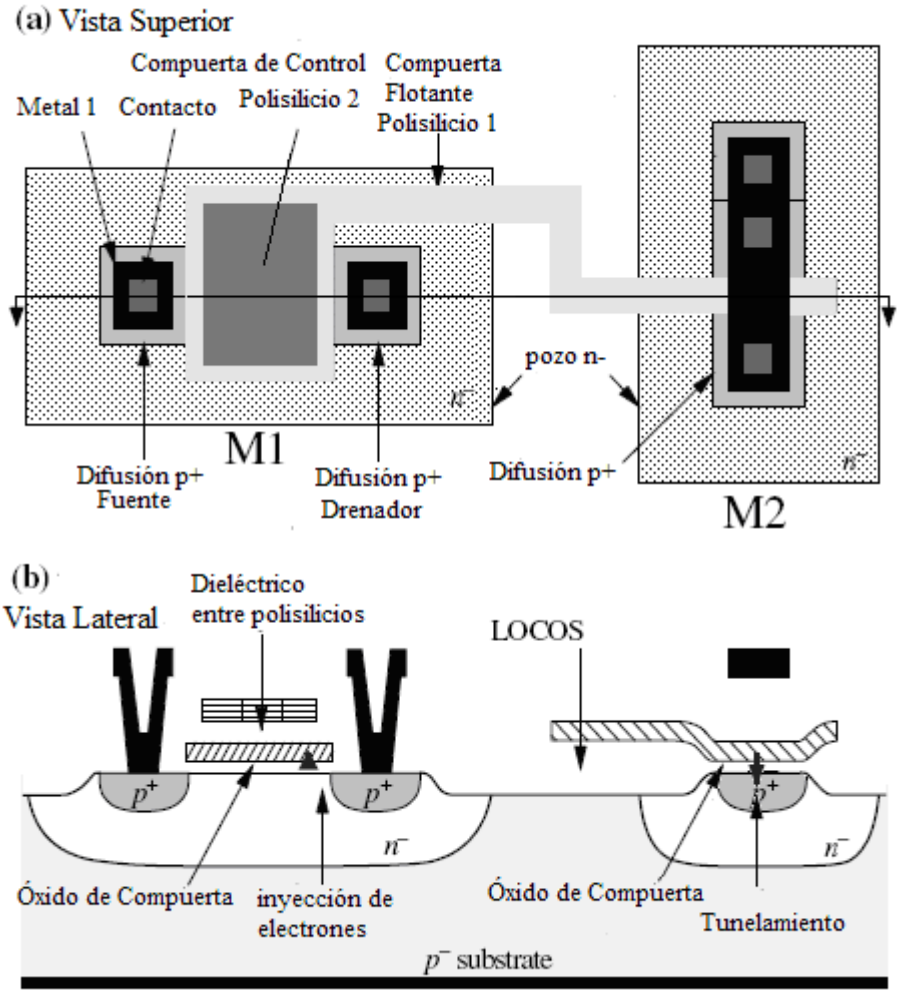


Figura 4.1 Estructura de un transistor PMOS con terminal de tuneleo a) Vista Superior b) Vista Lateral

El mecanismo de tuneleo sucede cuando el voltaje en la terminal de tuneleo es lo suficientemente grande como para reducir el ancho de la barrera de potencial del SiO_2 (Fig. 4.2), de esta manera la probabilidad de que un electrón atraviese la barrera de SiO_2 se incrementa.

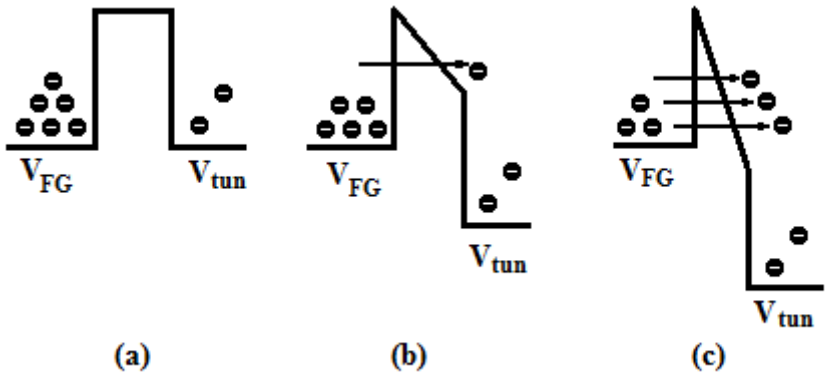


Figura 4.2 (a) $V_{tun}=V_{FG}$ (b) $V_{tun}>V_{FG}$ (c) $V_{tun}>>V_{FG}$ la probabilidad de que un electrón atraviese la barrera de potencial de SiO_2 es muy grande

Cuando ocurre el fenómeno de tuneleo, los electrones de la compuerta flotante tunelean hacia el pozo n-, donde la corriente de tuneleo está dada por [3]:

$$I_{tun} = I_{tun0} \exp\left(\left(\Delta V_{tun} - \Delta V_{FG}\right) / V_X\right) \tag{4.1}$$

Donde V_x es un parámetro relacionado a los voltajes en equilibrio de la compuerta flotante y de tuneleo, ΔV_{tun} es el cambio en el voltaje de tuneleo y ΔV_{FG} es el cambio en el voltaje de la compuerta flotante con relación al voltaje de la compuerta flotante en equilibrio.

Esto quiere decir que el potencial en la compuerta flotante se hace menos negativo y por lo tanto si trabajamos con transistores de canal n, el mecanismo de tuneleo disminuye su voltaje de umbral, mientras que si trabajamos con transistores canal p, el mecanismo de tuneleo aumenta el voltaje de umbral, es por eso que durante el proceso de programación, el mecanismo de tuneleo es utilizado como borrado general, pues elimina la carga inicial de un transistor de compuerta flotante.

4.3 Inyección de electrones calientes

Para el transistor de canal p, el mecanismo de inyección de electrones calientes sucede cuando existe un voltaje entre el drenador y la fuente lo suficientemente grande para

acelerar a los huecos del canal; cuando esto sucede, los huecos acelerados chocan contra los átomos de la red del pozo n, generando pares electrón-hueco; los electrones generados por impacto pueden ser dirigidos hacia la compuerta flotante cuando un voltaje en la compuerta es aplicado, esto se puede ver a través de la Figura 4.3. Este mecanismo "inyecta" electrones a la compuerta flotante del transistor p y por tanto se puede decir que hace "más negativo" el potencial de la compuerta. De esta forma se puede disminuir el voltaje de umbral del transistor p a través de la inyección de electrones calientes.

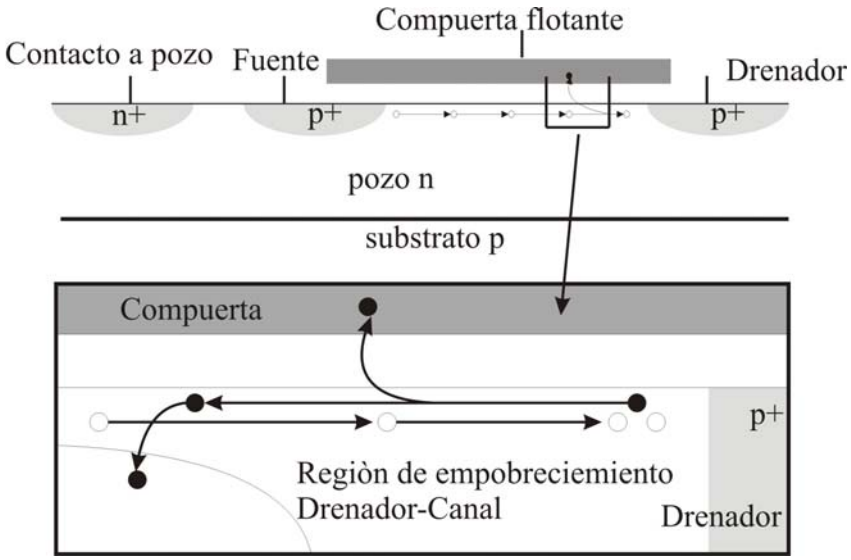


Figura 4.3 Mecanismo de inyección de electrones calientes

La corriente de inyección está dada por:

$$I_{iny} = I_{iny0}\left(\frac{I_s}{I_{s0}}\right)^{\alpha} \exp\left(\frac{-\Delta V_{SD}}{V_{iny}}\right) \tag{4.2}$$

Donde I_{s0} es la corriente inicial, I_s es la corriente final, α es $1-\left(\frac{v_t}{V_{iny}}\right)$, V_{iny}, I_{iny} son parámetros físicos del dispositivo, ΔV_{SD} es el voltaje fuente-drenador.

Para el caso de la inyección de electrones en los transistores de canal n, es necesario realizar un implante adicional en el pozo, lo cual genera un costo adicional para la

fabricación del chip, es por eso que la programación usando el mecanismo de inyección es usada sólo para transistores de canal p; mas adelante se verá que la técnica de *programación indirecta* se puede utilizar para programar transistores de canal n a través de un transistor p de compuerta compartida.

Para la simulación del mecanismo de inyección y de tuneleo se utilizó el modelo [4] descrito detalladamente en el Apéndice D.

4.4 Técnicas de programación usando la inyección de electrones calientes y el tuneleo Fowler-Norheim

Una de las técnicas más populares y efectivas que se ha venido usando en el área de diseño de circuitos integrados para eliminar el desapareamiento entre transistores, es la técnica de programación con electrones calientes y tuneleo Fowler-Nordheim. Esta técnica permite la programación múltiple de un arreglo de transistores con la finalidad de establecer un mismo voltaje de umbral. El mecanismo de tuneleo se utiliza como borrado general de los dispositivos, mientras que el mecanismo de inyección es usado para la programación del voltaje de umbral. Una vez que se ha logrado el borrado general, se introducen pulsos para generar el mecanismo de inyección de una manera controlada.

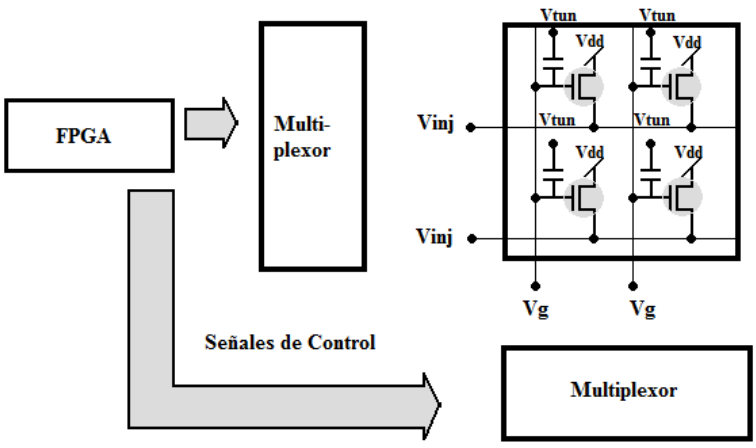


Figura 4.4 Sistema general de programación

La Fig. 4.4 muestra el sistema general utilizado para programar un arreglo de transistores, el control de las señales se realiza por medio de un FPGA, que genera los pulsos de programación al igual que la selección del transistor a programar. Existen dos algoritmos que permiten la programación óptima del voltaje de umbral. El primero usa la modulación del ancho de pulso [5] aplicado a la terminal de inyección, mientras que el otro usa la variación del voltaje V_{DS} del transistor programado [6], estos algoritmos son implementados en el FPGA.

También podemos distinguir dos métodos de programación: directa e indirecta, abordaremos ambos métodos en las siguientes secciones.

4.5 Método de programación directa

El método de programación directa permite la programación del voltaje de umbral para un arreglo de transistores de compuerta flotante. Ya han sido varios los trabajos publicados utilizando este método [7].

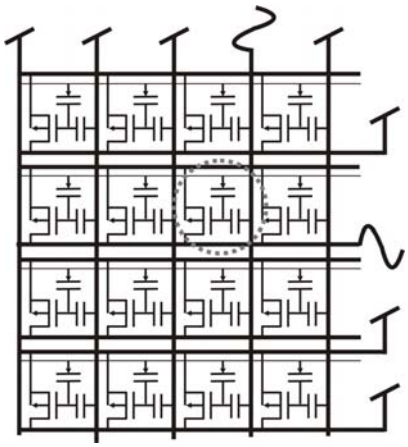


Figura 4.5 Programación del voltaje de umbral de una matriz de transistores de compuerta flotante

Para la programación se sigue la siguiente secuencia: primero se realiza un borrado general utilizando el mecanismo de tuneleo FN, con el cual se establece el voltaje inicial en la compuerta flotante de cada transistor. Luego, se debe seleccionar cada transistor a programar (Fig. 4.5); esto se logra a través de su aislamiento del circuito el cual opera por medio de compuertas de transmisión (Fig. 4.6) que son controladas por un FPGA.

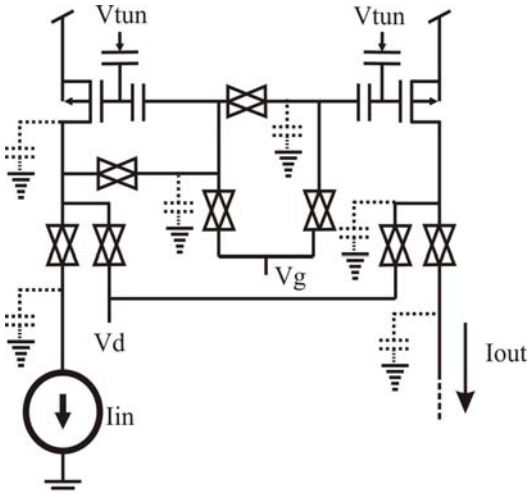


Figura 4.6 Programación de un espejo de corriente utilizando la técnica de programación directa

Posteriormente, se introduce un voltaje de inyección en la terminal de drenador. Para tener el control sobre la inyección, este voltaje se aplicará de manera pulsada. Se pueden utilizar dos algoritmos para el control de inyección, uno controla el tiempo del pulso en estado alto, mientras que el otro controla el nivel de voltaje en estado alto de cada pulso, como se mencionó anteriormente; después de inyectar un pulso, la corriente en el transistor es medida y comparada con una corriente objetivo; cuando se ha alcanzado la corriente deseada, se prosigue a la programación de otro transistor. El procedimiento se repite hasta que todos los transistores hayan sido programados para una misma corriente de salida. Si bien este método logra que un arreglo de transistores tenga el mismo voltaje de umbral, la adición de interruptores para aislar cada transistor, además de hacer más complejo el circuito, introduce capacitancias y resistencias parásitas que pueden afectar drásticamente el desempeño del circuito

4.6 Método de programación indirecta

El método de programación indirecta [8] permite la programación del voltaje de umbral de una serie de transistores, la diferencia con el método de programación directa radica en que la programación se realiza a través de un segundo transistor (transistor de programación) que comparte la compuerta flotante con el transistor que se desea programar (transistor agente), tal y como lo muestra la figura 4.7 para un espejo de corriente.

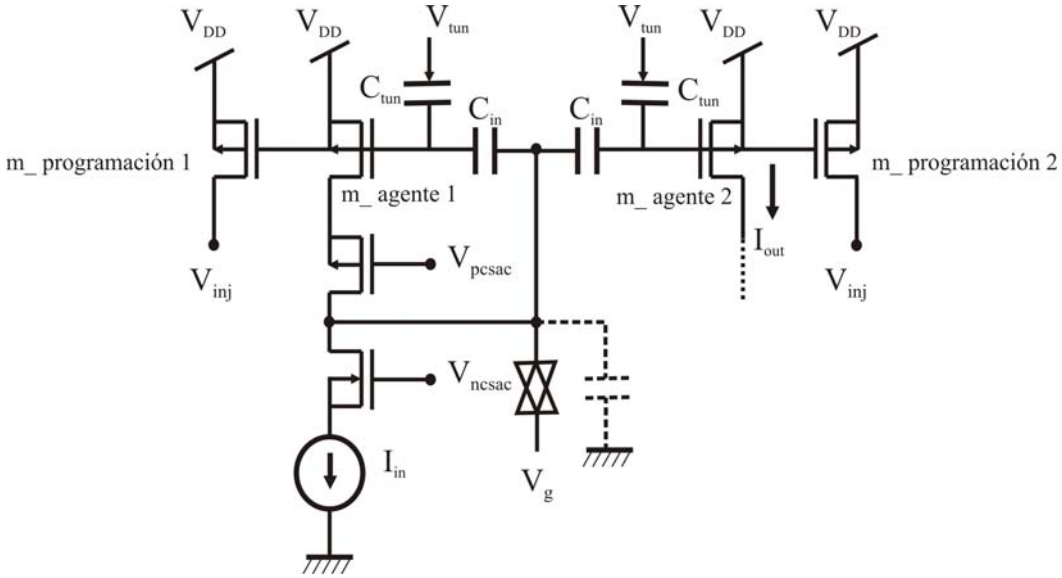


Figura 4.7 Implementación del circuito de programación indirecta para la programación de los transistores de un espejo de corriente

Como se puede observar, la programación se realiza a través de los transistores de programación; la corriente del transistor agente es medida indirectamente a través del transistor de programación. La ventaja es que este método permite una reducción notable en el número de interruptores o compuertas de transmisión. Otro aspecto importante es que con este método se hace posible la programación indirecta de transistores NMOS de compuerta flotante. Esto se hace a través un transistor PMOS que comparte su compuerta flotante con el NMOS.

De esta forma podemos listar las ventajas que presenta el método de programación indirecta:

- Permite la programación de transistores de canal n
- Decrementa el número de capacitancias parásitas para aplicaciones donde se requieran velocidades de respuesta rápidas
- Decrementa las resistencias parásitas
- Decrementa la potencia consumida

- Permite la programación mientras el circuito está en funcionamiento

Debido a estas ventajas, se podría usar este método para programar el circuito de Asignación para tener una respuesta precisa.

4.7 Simulación del método de programación indirecta

A continuación, se muestra la simulación de la programación de los voltajes de umbral en un espejo de corriente usando el método de programación indirecta. La simulación fue realizada para una tecnología de 0.5um en PSPICE. Además, fueron utilizados varios valores de voltaje de umbral, diferentes para los dos transistores a programar. Los voltajes de umbral fueron variados +/- 10% del valor nominal proporcionado por la tecnología de 0.5um. La siguiente figura muestra el circuito completo utilizado para la programación.

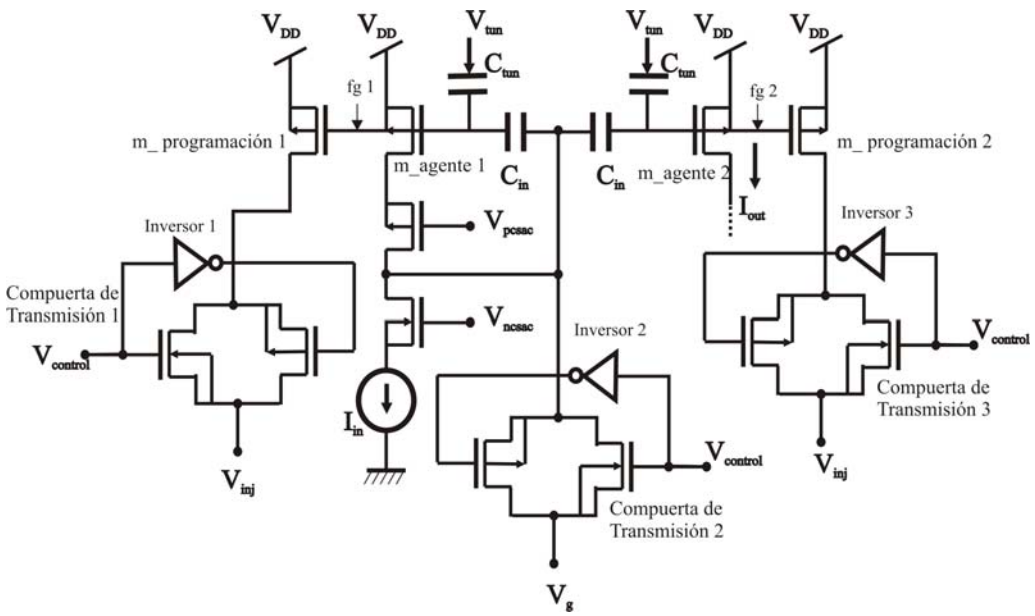


Figura 4.8 Espejo de corriente programado con el método de programación indirecta

La figura 4.9 muestra la variación de corriente de salida que existe en un espejo de corriente convencional cuando existe desapareamiento en el voltaje de umbral de +/- 10%

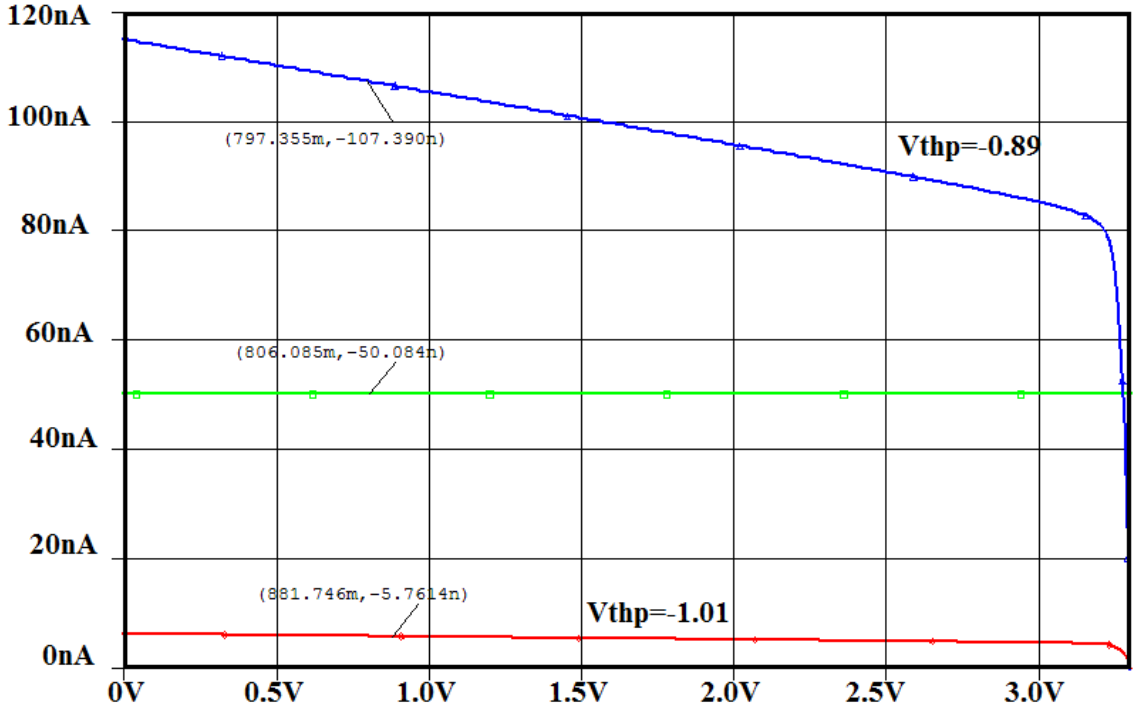


Figura 4.9 Corriente de salida de un espejo de corriente ante una variación de +/- 10% del voltaje de umbral en un par de transistores

La programación indirecta permite obtener una corriente de salida casi idéntica a la copiada (cuando consideramos que los potenciales de drenador de ambos transistores son casi iguales). Las siguientes figuras muestran la simulación en PSPICE del circuito de la Figura 4.8. Ambos transistores presentan un voltaje de umbral distinto. MP1 tiene un voltaje de umbral de 0.95V que es el voltaje de umbral nominal dado por la tecnología de 0.5um, mientras que para MP2 se ha variado su voltaje de umbral a +10% . Ambos se programan para tener corrientes de salidas similares. Para este caso la corriente objetivo es de 180nA. Después de que ambos han sido programados a la corriente objetivo, los transistores agentes se conectan al circuito original. La Fig. 4.10 (a) muestra los voltajes en las compuertas flotantes de los transistores agentes, donde debe de existir una diferencia de potencial entre ambos aproximadamente igual a la diferencia entre los voltajes de umbral. La Fig. 4.10 (b) muestra las corrientes de drenador de los transistores agentes durante todo el proceso de programación. Después de 550ms, ambos transistores han sido programados y la corriente copiada (50nA) al transistor agente 2 es aproximada a la corriente de entrada; la diferencia que hay entre ambas se debe a la diferencia de voltajes de drenador entre ambos transistores.

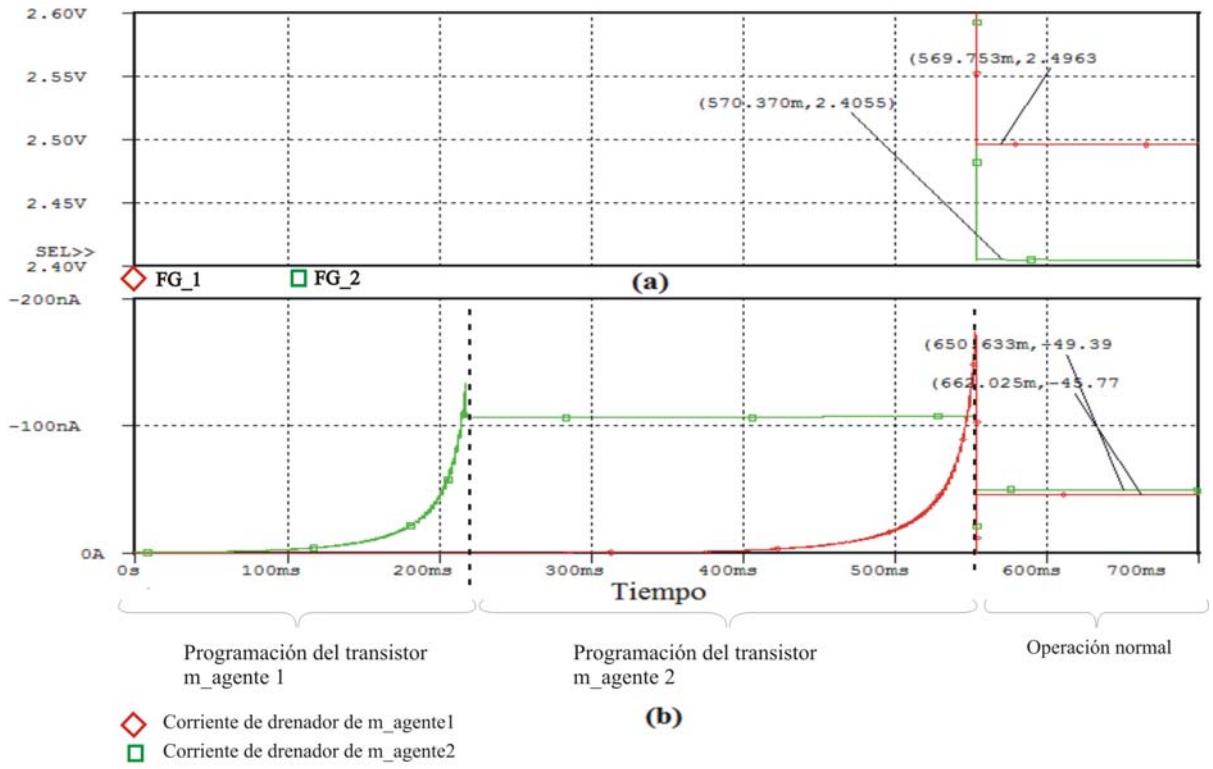


Figura 4.10 Comportamiento de la corriente de drenador de los transistores de programación durante el proceso de programación.

La figura 4.11, muestra el mismo proceso mostrado anteriormente, solo que ahora la variación del voltaje de umbral es de -10%, es decir el voltaje de umbral de uno de los transistores fue puesto a 0.81V. La figura 4.11 (a) muestra los voltajes en las compuertas flotantes, la diferencia entre esos voltajes debe de ser aproximadamente la diferencia que existe entre sus respectivos voltajes de umbral. La figura 4.11 (b) muestra la corriente de drenador de los transistores agente. Como se observa, la corriente copiada es casi igual a la corriente de entrada, la diferencia entre sus valores se debe a la diferencia entre los voltajes de drenador de ambos transistores. Como es de esperarse el tiempo que tarda programar el segundo transistor agente debe de ser menor.

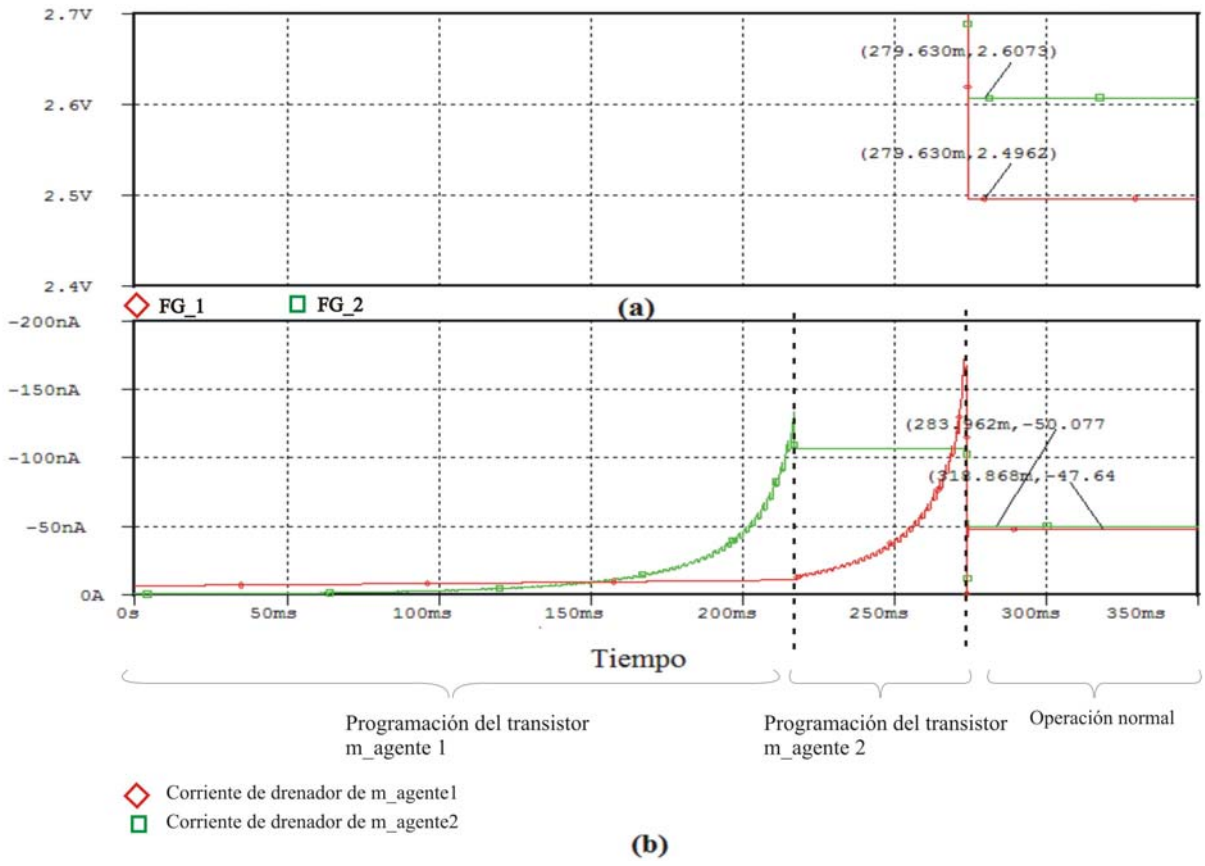


Figura 4.11 (a) Voltaje en las compuertas flotantes (b) Corriente de drenador de los transistores agentes

Finalmente, se muestran los pulsos utilizados durante el proceso de programación con el mecanismo de inyección. Como se ha mencionado, para una tecnología de 0.5um se necesita un voltaje Vds de entre 4.5V y 6.5V para que pueda existir la inyección de electrones calientes, es por eso que el voltaje Vdd se sube a un potencial de 6.5, y los pulsos aplicados a la terminal de drenador van de 0V (para que exista inyección) a 2.5V (para que no exista inyección), y el voltaje Vg es de 4V durante el proceso de inyección, tal y como se muestra en la figura 4.12.

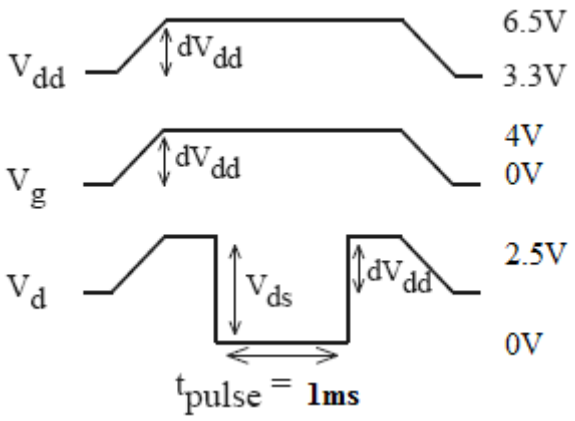


Figura 4.12 Pulsos de programación usados

REFERENCIAS

[1] "The Art of Analog Layout", Alan Hastings. 2001, Prentice Hall.

[2] Klimach, H. Arnaud, A. Galup-Montoro, C. Schneider, M.C., "MOSFET mismatch modeling: a new approach", Design & Test Computers, IEEE, Feb. 2006, vol. 1, issue 1, pp 20-29.

[3] Cruz Alejo, Jesús de la, "El transistor MOS de compuerta flotante como memoria no Volátil en Circuitos Analógicos", Dic. 2008, Tesis de Doctorado

[4] K. Rahimi, C. Diorio, C. Hernandez, M. Dean Brockhausen "A simulation model for floating-gate MOS synapse transistors", IEEE International Symposium on Circuits and Systems, Phoenix-Scottsdale, AZ, USA, Ago. 2002, pp. 532-535, vol. 2.

[5] Bandyopadhyay, A. Serrano, G.J. Hasler, P. "Adaptive algorithm using hot-electron injection for programming analog computational memory elements within 0.2% of accuracy over 3.5 decades", IEEE Journal of Solid-State Circuits, 2006, pp 2107-2114.

[6] Smith, P.D. Kucic, M. Hasler, P. "Accurate programming of analog floating-gate arrays", IEEE International Symposium on Circuits and Systems, Phoenix-Scottsdale, AZ, USA, Ago. 2002, pp 489-492.

[7] G. Serrano, P.D. Smith, H.J. Lo, R. Chawla, T. S. Hall, C. M. Twigg and P.Hasler, "Automatic rapid programming of large arrays of floating-gate elements" Proceedings of the International Symposium on Circuits and Systems, Vancouver, Canadá, May. 2004, vol. 1 pp. 373-376.

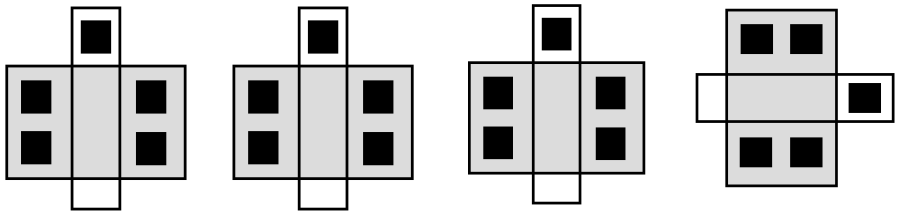
[8] Graham, D.W. Farquhar, E. Degnan, B. Gordon, C. Hasler, P., " Indirect programming of floating-gate transistors", IEEE Transactions on Circuits and Systems, May. 2005, Kobe, Japón vol. 3 pp. 2172 – 2175.

5.1 Técnicas layout para reducir el desapareamiento

La forma, el tamaño y la orientación de los transistores MOS afectan su apareamiento. Es importante tomar en consideración ciertas reglas para mantener el apareamiento entre dos transistores durante el diseño topológico y así evitar el desapareamiento debido a efectos sistemáticos. A continuación se hace un resumen de las reglas mas importantes que se deben de seguir.

1) Usar “dedos” idénticos.- Muchos transistores requieren un ancho relativamente grande, es por eso que se dividen en secciones o “dedos”, los cuales deben de ser idénticos entre sí.
2) Usar áreas activas grandes.- El desplazamiento eléctrico residual debido a fluctuaciones aleatorias es inversamente proporcional con el cuadrado del área del dispositivo.
3) Orientar los transistores en la misma dirección.- Los transistores que no están orientados en la misma dirección pueden ser vulnerables a variaciones en la movilidad inducida por el esfuerzo mecánico o eléctrico.
4) Colocar a los transistores lo más cerca posible.- Los transistores MOS son vulnerables a gradientes en temperatura, stress y espesor de óxido, por eso deben de colocarse lo mas cerca uno de otro.
5) El diseño geométrico debe de ser lo mas compacto posible.- Los diseños geométricos son vulnerables a distintos tipos de gradientes.
6) Usar técnicas como la del centroide común.
7) Colocar transistores “fantasma” al final de un arreglo de transistores
8) Colocar el pozo de los transistores lejos de dispositivos de potencia.
9) No colocar contactos en la parte superior del área activa de la compuerta.
10) No colocar metal a través de la región activa de la compuerta.
11) Mantener todas las uniones de difusiones profundas lejos del área activa de la compuerta.
12) Colocar los transistores que se deseen aparear en un mismo eje de simetría a lo largo de la oblea.
13) Usar dispositivos de óxido delgado en vez de óxido grueso.

Las figuras 5.1 y 5.2 ilustran algunas de las ideas anteriores.



(a) (b)

Fig. 5.1 (a) Transistores colocados en la misma dirección (b) Evitar el poner transistores orientados en diferentes direcciones

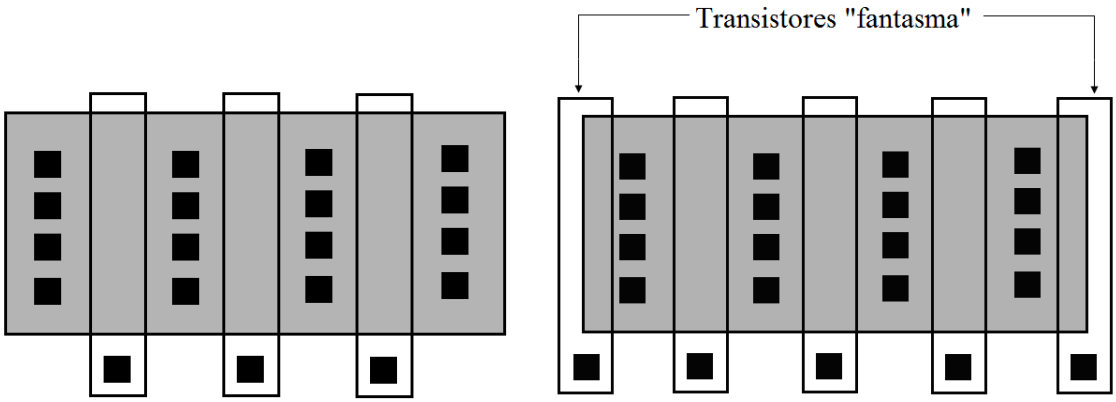


Figura 5.2 Colocar transistores “fantasma” en la parte lateral de un arreglo de transistores

5.2 Diseño geométrico de la celda de programación indirecta

Utilizando el programa L-Edit versión 13.0, se realizó el diseño geométrico de la celda de programación indirecta de la figura 4.7, usando las reglas de diseño para una tecnología de 0.5um. En la figura 5.3 se muestra el diseño geométrico de la celda.

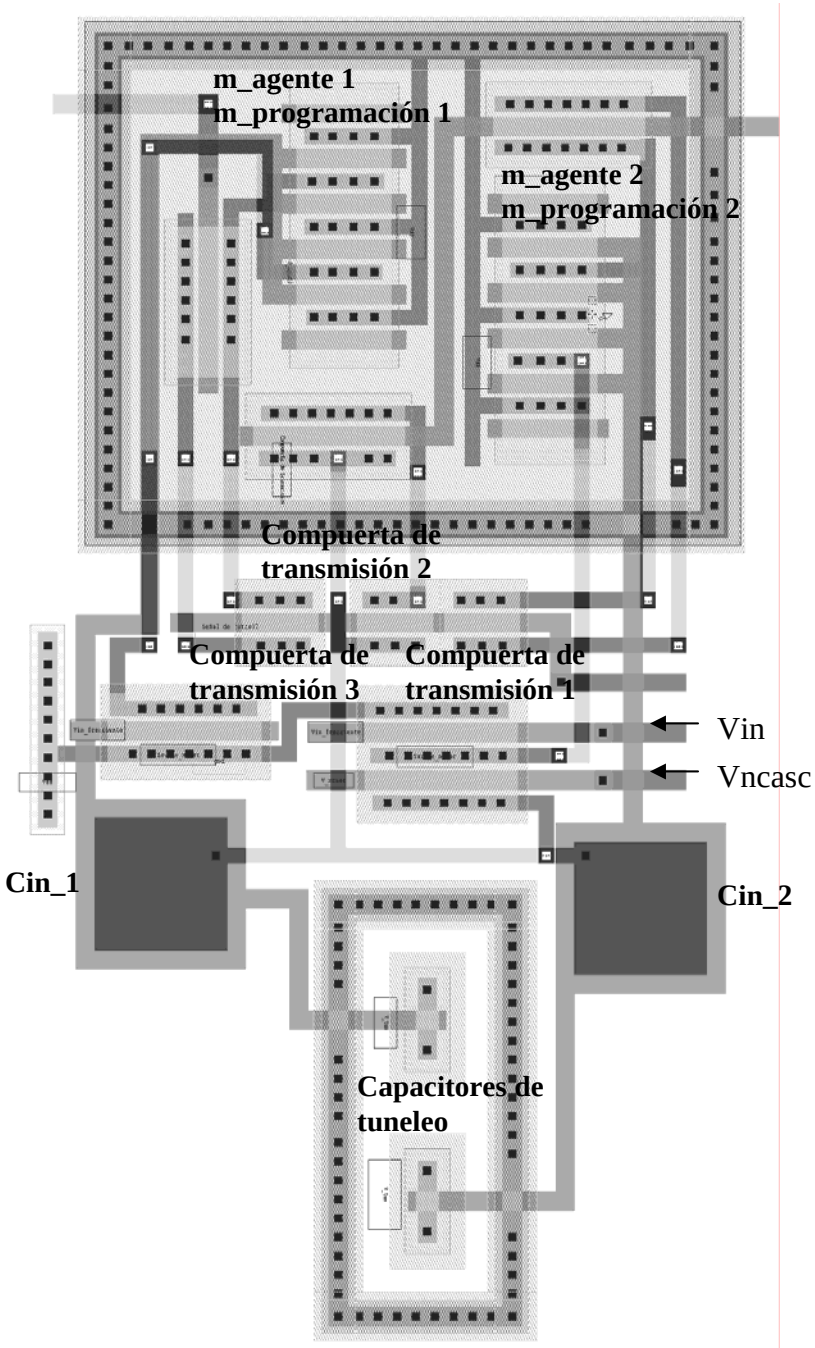


Figura 5.3 Diseño geométrico del circuito de la Fig, 4.7

CONCLUSIONES

Debido a las propiedades inherentes del transistor de compuerta flotante, se han podido reducir topologías complejas a unas más simples; además se puede usar como elemento de programación en algunas técnicas para la reducción del desacoplamiento. Esas ideas se trasladaron a este trabajo para la solución del *Problema de Asignación.* Se estudiaron algunos trabajos anteriores donde se proponía una solución analógica al *Problema de Asignación*, sin embargo estos resultaban complejos para su implementación electrónica. El método de los multiplicadores de Lagrange permitió la solución del *Problema de Asignación* y a su vez una implementación sencilla por medio del uso del transistor de compuerta flotante. Se usó el transistor de compuerta flotante como bloque sumador, pero además se aprovecharon las características del FGMOS como elemento de un circuito translineal, lo cual permitió la implementación de las ecuaciones diferenciales que intervenían en la solución del *Problema de Asignación*. Los resultados de la simulación hechos en PSPICE fueron satisfactorios, pues la red propuesta resuelve el problema óptimamente hasta para matrices de 8x8 en un tiempo menor de los 10us, mientras que las otras topologías propuestas les tomaba un tiempo mucho mayor en converger a la solución.

Otro aspecto importante que se debe de considerar en el diseño de circuitos analógicos es el desacople entre transistores (que le puede costar al diseñador ya sea circuitos con baja precisión hasta el mal funcionamiento del chip). Tomando esto como aspecto esencial, se consideran diversas técnicas para disminuir el desacoplamiento, una de ellas es la programación indirecta, que utiliza la ventaja del transistor de compuerta flotante que nos permite programar la carga en su compuerta. El método de programación indirecta resulta efectivo para eliminar el desacoplamiento entre transistores; además lo que hace atractivo al método, es la disminución del número de compuertas de transmisión para aislar el transistor que se desea programar, lo que permite que las características del circuito original no se vean afectadas, tal como el tiempo de respuesta.

PERSPECTIVAS A FUTURO

- Fabricación del chip del circuito de Asignación, considerando los elementos de programación indirecta para el ajuste de voltaje de umbral de los transistores de la red.
- Programación en el FPGA del algoritmo de programación indirecta.
- Utilizar el método de programación indirecta para el ajuste de los voltajes de umbral de los transistores del circuito de Asignación.
- Caracterización del chip del circuito de Asignación.
- Implementar el circuito de Asignación en una arquitectura de Conmutación de Paquetes para observar su desempeño en una aplicación real.

APÉNDICE A

Macromodelo para simulación de las características en DC del FGMOS

El siguiente listado muestra el macromodelo utilizado en esta tesis para la simulación del transistor de compuerta flotante. El programa fue realizado en PSPICE versión 10.0 para un transistor de compuerta flotante de tres entradas.

```
.subckt fgmos2 in1 in2 in3  phif
*vin1___________|    |   |    |
*vin2___________________|   |    |
*vin3 _____________________|    |
*fg  ______________________________|

g1 0 1 in1 0 1
g2 0 2 in2 0 1
g3 0 5 in3 0 1

r1 1 0 50.56
r2 2 0 50.56
r3 5 0 50.56

E 3 0  poly(3) 1 0 2 0 5 0 0 1 1 1
vz 3 4 0
rtot 4 0 132.45
h phif 0 vz 1
.ends fgmos2
```

Donde in1, in2 e in3 son las tres entradas y phif es el voltaje en la compuerta flotante.

Para el cálculo del coeficiente de acoplamiento capacitivo que está dado por la relación $\frac{R_i}{R_{TOT}}$ se realizó un programa en Visual Basic v6.0, donde dados los datos de entrada (tipo de transistor, tamaños, número de capacitores en la compuerta flotante, el valor de los capacitores de entrada o en su defecto el tamaño), lee los parámetros de los modelos para transistores PMOS y NMOS para tecnología de 0.5um. La figura A.1 muestra la ventana principal del programa.

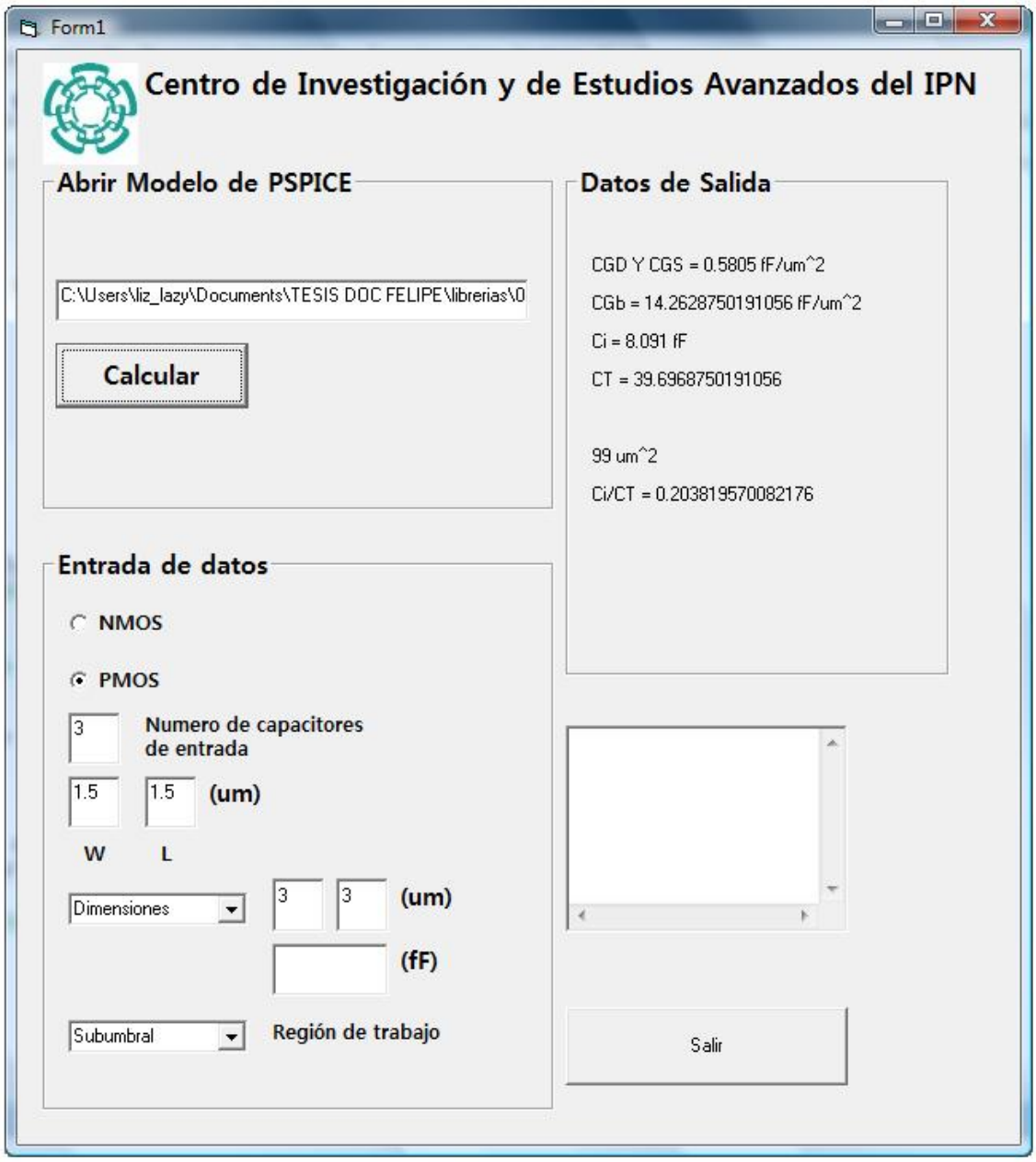


Figura A.1 Ventana principal del programa realizado en Visual Basic 6.0 para el cálculo de las capacitancias de un FGMOSFET

APÉNDICE B

El siguiente listado muestra el programa de implementación del algoritmo Húngaro en Visual Basic Versión 6.0

Programa principal

```
'********************************** VALORES INICIALES **********************************
auxiliar_1(1, 1) = 11
n = 0
'********************************** SE INTRODUCE MATRIZ **********************************
If Option1.Value = True Then ' SE GENERAN NUMEROS ALETORIOS DE ENTRE 1 Y 10
Randomize
For j = 1 To a
  For i = 1 To a
  matriz(j, i) = 1 + (9 * Rnd())
  matriz(j, i) = Round(matriz(j, i), 0)
  mate_6(j, i) = matriz(j, i)            ' ********** MATRIZ USADA EN BUSQUEDA
  mate_7(j, i) = matriz(j, i)            ' ********** MATRIZ USADA EN BUSQUEDA
   Text11.Text = Text11.Text & " " & matriz(j, i)
   Next
   Text11.Text = Text11.Text & Chr(13) & Chr(10)
Next

Else '*****************************************DE LO CONTRARIO SE INTRODUCE UNA MATRIZ

Dim strAryWords
Dim strValue
Max = 0
strValue = Text1.Text
strAryWords = Split(strValue, " ")
' - strAryWords is now an array
For i = 0 To (a - 1)
For j = 0 To (a - 1)
    matriz(i + 1, j + 1) = Val(strAryWords(j + a * i))
    mate_6(i + 1, j + 1) = matriz(i + 1, j + 1)    ' ********** MATRIZ USADA EN BUSQUEDA
    mate_7(i + 1, j + 1) = matriz(i + 1, j + 1)    ' ********** MATRIZ USADA EN BUSQUEDA

Next
Next
End If

'******* SE LOCALIZA NUMERO MAS PEQUEÑO Y SE ESTABLECEN CONDICIONES INICIALES *************

For i = 1 To a
beta(1, i) = 11
    For j = 1 To a
    mate(i, j) = 0
    edge(i, j) = 0
    mate_2(i, j) = 0
    mate_4(i, j) = 0
If beta(1, i) > matriz(j, i) Then
beta(1, i) = matriz(j, i)
End If
Next
Next
aux6(1, 0) = 11

'***************** SE RESTA EL VALOR MAS PEQUEÑO DE CADA COLUMNA A LA MATRIZ **************
 For i = 1 To a
    For j = 1 To a
    mate_main(j, i) = matriz(j, i) - beta(1, i)
    mate_6(j, i) = matriz(j, i) - beta(1, i)
    mate_7(j, i) = matriz(j, i) - beta(1, i)
     Next
 Next
'**********************INICIO DEL CICLO FOR PRINCIPAL DEL MÉTODO *************************
For main = 1 To 1
For i = 1 To a
    For j = 1 To a
    If mate_main(j, i) = 0 Then
```

```
       edge(j, i) = 2
       mate(j, i) = 2
     End If
     Next
 Next

'************************** MUESTRA LA MATRIZ FINAL DE EDGE ******************************

 For i = 1 To a
 For j = 1 To a
 Text2.Text = Text2.Text & "  " & edge(i, j)
 Next
 Text2.Text = Text2.Text & Chr(13) & Chr(10)
fila_mate2(1, i) = 0    ' VECTOR AUXILIAR PARA UTILIZAR EN EL MODULO AUGMENTINGPATH
fila_mate3(1, i) = 0    ' VECTOR AUXILIAR PARA UTILIZAR EN EL MODULO HUNGARO
columna_mate2(1, i) = 0 ' VECTOR AUXILIAR PARA UTILIZAR EN EL MODULO AUGMENTINGPATH
columna_mate3(1, i) = 0 ' VECTOR AUXILIAR PARA UTILIZAR EN EL MODULO HUNGARO
 Next
 Text2.Text = Text2.Text & Chr(13) & Chr(10)

'***************************** ENCUENTRA MATCHING'S Y EDGES *****************************
 For i = 1 To a
 For j = 1 To a
 mate_2(i, j) = mate(i, j)  ' SE UTILIZA EN match1
 mate_3(i, j) = mate_2(i, j)
 Text3.Text = Text3.Text & "  " & mate(i, j)
 Next
 Text3.Text = Text3.Text & Chr(13) & Chr(10)
 Next
 Text3.Text = Text3.Text & Chr(13) & Chr(10)

Call augmenting

For i = 1 To a
For j = 1 To a
Text4.Text = Text4.Text & "  " & mate_2(i, j)
mate_4(i, j) = mate_2(i, j)  '
Next
Text4.Text = Text4.Text & Chr(13) & Chr(10)
Next
Text4.Text = Text4.Text & Chr(13) & Chr(10)
For i = 1 To aux_16 - 1
Text8.Text = Text8.Text & "  " & auxiliar(i, 1)
Text8.Text = Text8.Text & "  " & auxiliar(i, 2)
Next
Text8.Text = Text8.Text & Chr(13) & Chr(10)

Call ELIMINACION

For i = 1 To a
Text10.Text = Text10.Text & " " & el_fila(1, i)
'PREGUNTA POR EL NUMERO DE 1´s EN LOS VECTORES AUXILIARES, SI EL NUMERO DE 1's ES IGUAL A
N, EL PROBLEMA ESTA RESUELTO!!!!

If el_fila(1, i) = 1 Then
main_counter = main_counter + 1
End If
Next
Text10.Text = Text10.Text & "  "

For i = 1 To a
Text10.Text = Text10.Text & " " & el_columna(1, i)
If el_columna(1, i) = 1 Then
main_counter = main_counter + 1
End If
Next
Text10.Text = Text10.Text & Chr(13) & Chr(10)

If main_counter = a Then
 main = 1                   'EL PROBLEMA ESTA RESUELTO !!!!!
Else
  main = 0             ' ****** INICIALIZA LOS VECTORES
  main_counter = 0     ' ****** INICIALIZA LOS VECTORES
  Call buscar
End If

  For i = 1 To a
```

```
  For j = 1 To a
  Text7.Text = Text7.Text & " " & mate_7(i, j)
  mate_main(i, j) = mate_7(i, j)
  mate_6(i, j) = mate_7(i, j)

  Next
  Text7.Text = Text7.Text & Chr(13) & Chr(10)
  Next
  Text7.Text = Text7.Text & Chr(13) & Chr(10)

Next 'main
End Sub
Private Sub Command2_Click()
End
End Sub
```

Modulo1: Busca un camino aumentado

```
'***************************  INICIALIZANDO MATRICES AUXILIARES ************************
For ini_2 = 1 To a
For ini_1 = 1 To a
'mate_3(ini_2, ini_1) = 0
Next
auxiliar2(1, ini_2) = 0
auxiliar3(1, ini_2) = 0
Next
For AUX_3 = 1 To a

'************************SE VERIFICA QUE EXISTA UN NODO NO SATURADO **********************
If fila_mate2(1, AUX_3) = 0 Then
' EXISTE UN NODO NO SATURADO
   fila = AUX_3
   aux_12 = 0
   m_fila = AUX_3
           For aux_11 = 1 To a
              If mate_3(fila, aux_11) = 2 Then
              aux_12 = 1 ' EXISTE POR LO MENOS UNA LINEA DE CONEXIÓN
              aux_11 = a ' SE ROMPE EL CICLO FOR
              End If
           Next
If aux_12 = 1 Then                                      ' EXISTE POR LO MENOS UNA CONEXION,
PROSIGUE CON EL PROGRAMA
nodo_ini = fila ' GUARDA EL VALOR DEL NODO INICIAL, PARA ASEGURAR QUE NO SE GENERE UN
CAMINO CERRADO
   aux_5 = 1                                                                  ' VALOR INICIAL
   For aux_4 = 1 To a
'*******************************************************************************************
If aux_5 Mod 2 = 1 Then  '* IMPAR:FILA
       For aux_6 = 1 To a
       If mate_3(fila, aux_6) = 2 Then
            If auxiliar3(1, aux_6) = 1 Then  ' ya paso por ese edge
            patito = 0
            mate_3(fila, aux_6) = 0   ' SE ELIMINA EL EDGE = 2 ANTERIOR
            AUX_3 = 1  ' SE REINICIA NUEVAMENTE LA BUSQUEDA
            aux_6 = a
            aux_8 = a
          Else
             aux_5 = aux_5 + 1            '  **************************** CONTADOR AUXILIAR
            columna = aux_6               '  ************* SE GUARDA EL VALOR DE LA COLUMNA
            fila_auxiliar = fila          '  ******************** SE GUARDA EL VALOR ACTUAL
            aux_6 = a                     '  ************************* TERMINA EL CICLO FOR
            auxiliar2(1, fila) = 1
            auxiliar3(1, columna) = 1
            patito = patito + 1
            auxiliar(patito, 1) = fila
            auxiliar(patito, 2) = columna
          End If
         'Next
       End If
       Next
Else

'EL SIGUIENTE EDGE NO ES 1! PUEDES SER:
'1. NO ES UN AUGMENTING PATH (SOLO SI AUX_5 = 1)
```

```
'2. SER UN AUGMENTING PATH QUE TERMINO EN EL CICLO ANTERIOR (AUX_5 = 3, 5, 7 ...)
  If aux_5 = 1 Then             ' ***************** SI AUX_5 = 1 NO ES UN AUGMENTING PATH !
  patito = 0
  mate_3(fila_auxiliar, columna) = 0   ' ******************SE ELIMINA EL EDGE = 2 ANTERIOR
  AUX_3 = 1 ' SE REINICIA NUEVAMENTE LA BUSQUEDA
  aux_4 = a
          ' *********** SE REINICIA LA BUSQUEDA
          For cont = 1 To a
          auxiliar3(1, cont) = 0
          auxiliar2(1, cont) = 0
          Next
  ' ***************** T E N E M O S   U N   A U G M E N T I N G   P A T H   !!!!!! *********
  Else
  aux_13 = 1
  aux_16 = aux_5 ' PARA MAS ADELANTE
  aux_5 = 1   ' IMPIDE QUE ENTRE AL SIGUIENTE IF
  End If
End If
If aux_5 Mod 2 = 0 Then
       For aux_7 = 1 To a
       If mate_3(aux_7, columna) = 1 Then '  ***** FILA VARIABLE(aux_T), COLUMNA FIJA(fila)
             If auxiliar2(1, aux_7) = 1 Then
            Else
              'mate_3(aux_7, columna) = 1
              aux_5 = aux_5 + 1              '  ************************ CONTADOR AUXILIAR
              fila = aux_7                   '  ************* SE GUARDA EL VALOR DE LA FILA
              columna_auxiliar = columna     '  ***************** SE GUARDA EL VALOR ACTUAL
              aux_7 = a                      '  ******************* SE TERMINA EL CICLO FOR
              auxiliar2(1, fila) = 1
              auxiliar3(1, columna) = 1
              path = path + 1
              auxiliar(path, 1) = columna
              auxiliar(path, 2) = fila
            End If
       End If 'If mate_3(aux_7, columna) = 1
       Next
Else
        If aux_13 = 1 Then
        aux_4 = a ' SE ROMPE EL CICLO FOR
        AUX_3 = a
        Else
        path = 0
          mate_3(fila_auxiliar, columna_auxiliar) = 0   ' SE ELIMINA EL EDGE = 2 ANTERIOR
          AUX_3 = 1  ' SE REINICIA NUEVAMENTE LA BUSQUEDA
          aux_4 = a '
          For cont = 1 To a
          auxiliar3(1, cont) = 0
          auxiliar2(1, cont) = 0
          Next
        End If
End If
Next    ' AUX_4
End If
End If
Next    ' AUX_3
'***************************** PREGUNTA SI EXISTE UN CAMINO AUMENTADO ******************
If aux_13 = 1 Then
aux_15 = 1
For aux_14 = 1 To aux_16 - 1
If aux_15 Mod 2 = 1 Then
mate_2(auxiliar(aux_14, 1), auxiliar(aux_14, 2)) = 1
fila_mate3(1, auxiliar(aux_14, 1)) = 1
aux_15 = aux_15 + 1
Else
mate_2(auxiliar(aux_14, 2), auxiliar(aux_14, 1)) = 2
aux_15 = aux_15 + 1
End If
Next
End If
End Sub
```

Modulo 2

```
Sub buscar()
Dim aux_25, aux_26, aux_27, aux_28, aux_29, aux_30, aux_31, aux_32, mate_8() As Integer
ReDim mate_8(1 To a, 1 To a)
For aux_25 = 1 To a
 For aux_26 = 1 To a
If el_fila(1, aux_25) = 1 Then
mate_7(aux_25, aux_26) = 0
ElseIf el_columna(1, aux_26) = 1 Then
mate_7(aux_25, aux_26) = 0
End If

'**SI EXISTE INTERSECCIÓN ENTRE UNA FILA Y COLUMNA SE VUELVE A COLOCAR EL VALOR ORIGINAL***
If el_fila(1, aux_25) = 1 Then
  If el_columna(1, aux_26) = 1 Then
  mate_7(aux_25, aux_26) = mate_6(aux_25, aux_26)
  mate_8(aux_25, aux_26) = 1
  End If
End If
Next
Next
aux_31 = 11

'*****SE BUSCA EL NUMERO MAS PEQUEÑO MAYOR DE CERO DE LOS QUE RESTAN EN LA MATRIZ mate_6 **
For aux_29 = 1 To a
For aux_30 = 1 To a

If mate_7(aux_29, aux_30) > 0 Then  ' ****************** SE VERIFICA QUE SEA MAYOR DE CERO
If mate_8(aux_29, aux_30) <> 1 Then '******SE VERIFICA QUE NO SEA UNA INTERSECCION DE NODOS
If mate_7(aux_29, aux_30) < aux_31 Then
aux_31 = mate_7(aux_29, aux_30)
End If
End If
End If
Next
Next

For aux_29 = 1 To a
For aux_30 = 1 To a
If mate_8(aux_29, aux_30) = 1 Then ' INTERSECCION DE NODOS
mate_7(aux_29, aux_30) = (mate_7(aux_29, aux_30) + aux_31)
Else
  If mate_7(aux_29, aux_30) > 0 Then  ' MAYOR DE 1
  mate_7(aux_29, aux_30) = (mate_7(aux_29, aux_30) - aux_31)
  Else
  mate_7(aux_29, aux_30) = mate_6(aux_29, aux_30)
' DONDE HABÍA CEROS SIN INTERSECCIÓN SE COLOCA NUEVAMENTE EL VALOR ORIGINAL
End If
End If
Next
Next
' SE HA GENERADO UNA NUEVA MATRIZ !
End Sub
```

Modulo 3: Eliminación

```
Sub ELIMINACION()
' ******************************** INICIALIZANDO VALORES *********************************
For ini_2 = 1 To a
For ini_3 = 1 To a
mate_5(ini_2, ini_3) = 0
Next
el_fila(1, ini_2) = 1                                          ' ********** VALORES INICIALES
el_columna(1, ini_2) = 0                                       ' ********** VALORES INICIALES
Next
fila_auxiliar2 = 0
columna_auxiliar2 = 0
aux_20 = 0

For aux_16 = 1 To a    ' CICLO FOR PRINCIPAL
' Se pregunta si existe un nodo no saturado
If fila_mate3(1, aux_16) = 0 Then
fila_auxiliar2 = aux_16
```

```
                    ' SE PONE A CERO EL VECTOR AUXILIAR PARA CAMINOS CERRADOS
                     For aux_33 = 1 To a
                       rec_vec(aux_33, 1) = 0
                       rec_vec(aux_33, 2) = 0
                      Next
                      aux_34 = 0
                      aux_35 = 0
'************SE VERIFICA QUE EL NODO NO SATURADO TENGA CONEXIÓN CON ALGUNA COLUMNA*********
For aux_19 = 1 To a
If mate_4(fila_auxiliar2, aux_19) = 2 Then
aux_20 = 1   ' si lo tinen se le asigna a aux_20 '1'
End If
Next
If aux_20 = 1 Then
aux_23 = 0
For aux_17 = 1 To a
If aux_18 Mod 2 = 1 Then                                          '********** IMPAR: FILA
el_fila(1, fila_auxiliar2) = 0
contador1 = 0
'***************************** VERIFICA QUE TENGA UNA CONEXIÓN = 2 ***********************
     For aux_20 = 1 To a
     If mate_4(fila_auxiliar2, aux_20) = 2 Then
            ' SE CHECA SI NO SE ESTA EN UN CAMINO CERRADO (EVITANDO QUE LA RUTINA SE CICLE)
          If rec_vec(fila_auxiliar2, 1) = 1 Then
              If rec_vec(aux_20, 2) = 1 Then
               aux_34 = 1 ' INDICA QUE YA NO SE DEBE SEGUIR LA BUSQUEDA
               mate_4(fila_auxiliar2, aux_20) = 0
               aux_20 = a ' ROMPE EL CICLO
              End If
           End If
         If aux_34 = 0 Then
     rec_vec(fila_auxiliar2, 1) = 1 '********* VECTOR AUXILIAR PARA CAMINOS CERRADOS (FILA)
     rec_vec(aux_20, 2) = 1         '****** VECTOR AUXILIAR PARA CAMINOS CERRADOS (COLUMNA)
     el_fila(1, fila_auxiliar2) = 0 '*************** SE MARCA LA FILA A "NO SER ELMININADA"
     'mate_4(fila_auxiliar2, aux_20) = 0
     columna_auxiliar2 = aux_20                ' ********* SE GUARDA EL VALOR DE LA COLUMNA
     aux_20 = a                                ' ******************** SE ROMPE EL CICLO FOR
     aux_18 = aux_18 + 1                       ' ******* SE INCREMENTA EL CONTADOR AUXILIAR
     aux_16 = 0                                ' ************* VUELVE A INICIAR LA BÚSQUEDA
        End If
     Else
     contador1 = contador1 + 1
     If contador1 = a Then
     contador1 = 0
     mate_4(fila_auxiliar2, columna_auxiliar2) = 0
     End If
     End If  'mate_4(fila_auxiliar2, aux_20) = 2
     Next 'For aux_20 = 1 To a
Else    ' YA NO EXISTE CONEXIÓN SIGUIENTE
aux_17 = a     ' SE INTERRUMPE EL CICLO FOR DE AUX_17
aux_18 = 1     ' IMPIDE QUE CONTINUE EN EL SIGUIENTE IF
End If
If aux_18 Mod 2 = 0 Then                                  ' *************** PAR: COLUMNA
el_columna(1, columna_auxiliar2) = 1
contador2 = 0
     For aux_21 = 1 To a
     If mate_4(aux_21, columna_auxiliar2) = 1 Then
          If rec_vec(aux_21, 1) = 1 Then
              If rec_vec(columna_auxiliar2, 2) = 1 Then
               ' SE PASO POR ESE EDGE ANTES
               aux_21 = a ' ROMPE EL CICLO
               aux_35 = 1 ' INDICA QUE YA NO SE DEBE SEGUIR LA BUSQUEDA
               mate_4(aux_21, columna_auxiliar2) = 0
              End If
           End If
          If aux_35 = 0 Then
     rec_vec(aux_21, 1) = 1         '********* VECTOR AUXILIAR PARA CAMINOS CERRADOS (FILA)
     rec_vec(columna_auxiliar2, 2) = 1  '***VECTOR AUXILIAR PARA CAMINOS CERRADOS (COLUMNA)
     el_columna(1, columna_auxiliar2) = 1     '**********SE MARCA COLUMNA A SER "ELIMINADA"
     'mate_4(aux_21, columna_auxiliar2) = 0
     fila_auxiliar2 = aux_21                  ' ********** SE GUARDA EL VALOR DE LA COLUMNA
     aux_21 = a                               ' ********************* SE ROMPE EL CICLO FOR
     aux_18 = aux_18 + 1                      ' ******** SE INCREMENTA EL CONTADOR AUXILIAR
     aux_16 = 0
         End If
         Else
     contador2 = contador2 + 1
```

```
     If contador2 = a Then
     mate_4(fila_auxiliar2, columna_auxiliar2) =      End If
     End If 'If mate_4(aux_21, columna_auxiliar2) = 2 Then
     Next   'For aux_21 = 1 To a
Else           ' YA NO EXISTE CONEXIÓN SIGUIENTE
aux_17 = a     ' SE INTERRUMPE EL CICLO FOR DE AUX_17
End If ' If aux_18 Mod 2 = 1 Then
Next ' aux_17
Else 'If aux_20 = 1 Then
End If 'If aux_20 = 1 Then
End If 'If fila_mate3(1, aux_16) = 0
Next

End Sub
```

Modulo 4: Busca apareamiento de nodos

```
Sub Completematch()
For aux1 = 1 To a
For aux2 = 1 To a
If edge(aux1, aux2) = 2 Then
   If fila_mate(1, aux1) = 0 Then  ' NO EXISTE ASIGNACIÓN EN ESA FILA
       If columna_mate(1, aux2) = 0 Then ' NO EXITE ASIGANCION EN ESA COLUMNA
       mate(aux1, aux2) = 1
       mate_2(aux1, aux2) = 1   ' MATRIZ AUXILIAR PARA UTILIZAR EN EL MODULO AUGMENTINGPATH
       mate_3(aux1, aux2) = 1   ' MATRIZ AUXILIAR PARA UTILIZAR EN EL MODULO AUGMENTINGPATH
       fila_mate(1, aux1) = 1
       fila_mate2(1, aux1) = 1  ' VECTOR AUXILIAR PARA UTILIZAR EN EL MODULO AUGMENTINGPATH
       fila_mate3(1, aux1) = 1         ' VECTOR AUXILIAR PARA UTILIZAR EN EL MODULO HUNGARO
       columna_mate(1, aux2) = 1
       columna_mate2(1, aux2) = 1         columna_mate3(1, aux2) = 1 ' VECTOR AUXILIAR PARA
UTILIZAR EN EL MODULO HUNGARO
       End If
    End If
End If
Next
Next
End Sub
```

Modulo 5: Declaración de variables globales

```
Public a, d, c, n, x, y, f, aux, aux2, aux7, ploscar, aux_5, aux18, aux_16, moose As
Integer ' DIMENSION DE LA MATRIZ
Public mate(), mate_2(), edge_aux(), edge(), aux4(), aux5(), aux6(), slack(), alfa(),
path(), edge_2() As Integer
Public fila_2(), columna_2(), mate_3(), mate_4(), edge_3() As Integer
Public fila_mate2(), columna_mate2(), auxiliar(), fila_mate3(), columna_mate3(), el_fila(),
el_columna() As Integer
Public aux_13, columna_auxiliar, fila_auxiliar, fila, columna, m_fila As Integer


' **********************  VARIABLES DECIMALES **********************
Public beta(), matriz(), mate_6(), mate_7(), mate_main() As Double
```

APÉNDICE C

Método de los multiplicadores de Lagrange

El método de los multiplicadores de Lagrange es utilizado en el área de la optimización para maximizar o minimizar una función sujeta a una o más restricciones. El método consiste en la introducción de una variable adicional por cada restricción, así, si el problema original tiene n variables y m restricciones, se agregan m nuevas variables, de tal forma que el problema se convierta en un problema de n + m variables sin ninguna restricción. El concepto de multiplicadores de Lagrange aplica a problemas de optimización con funciones multivariables $f(x_1, x_2, ..., x_n)$ sujeta a un conjunto de restricciones $g_1(x_1, x_2, ..., x_n)$. El resultado es la función Lagrangiana que está definida como:

$$L(x_1, x_2, ...x_n, \lambda_1, \lambda_2, ...\lambda_m) = f(x_1, x_2, ...x_n) + \sum_{i=1}^{m} \lambda_i g(x_1, x_2, ...x_n) \qquad C.1$$

Donde λ_i son los multiplicadores de Lagrange. Ya que la función de Lagrange es un problema sin restricciones, las condiciones necesarias para encontrar su mínimo son que las derivadas con respecto a todas sus variables deben de ser igual a cero, tal y como se muestra en la siguiente expresión:

$$\frac{\partial L}{\partial x_k} = \frac{\partial y}{\partial x_k} + \sum_{i=1}^{m} \lambda_i \frac{\partial g_i}{\partial x_k} = 0, \qquad k = 1, 2, ..., n \qquad C.2$$

$$\frac{\partial L}{\partial \lambda_i} = g_i = 0, \qquad i = 1, 2, ..., m \qquad C.3$$

Las ecuaciones A.2 y A.3 son un sistema de n+m ecuaciones con n+m incógnitas. Los valores de $x_1, x_2, ..., x_n$ que satisfacen las condiciones necesarias para la función Lagrangiana $L(x_1, x_2, ...x_n, \lambda_1, \lambda_2, ...\lambda_m)$, también satisfacen las condiciones para obtener los valores extremos de $f(x_1, x_2, ...x_n)$. Sin embargo, ésta solución no asegura que el punto extremo sea un punto meseta (saddle point), a menos que la función de Lagrange se trate de una función convexa.

- **Condiciones para puntos extremos**

  - **La condición necesaria de Primer-Orden**

El punto x* es llamado mínimo local de f(x) en $\mathbf{R^n}$ si podemos encontrar un $\varepsilon > 0$ tal que $f(x) \geq f(x^*)$ para toda x en el vecindario ε de x*. En este caso, x* es un punto mínimo, pero no necesariamente será un punto mínimo global. Así el Teorema de Fermat dice:

Sea x un punto mínimo de f(x) en* $\boldsymbol{R^n}$ *y f(x) es diferenciable en x*. Entonces:*

$$\nabla f(x^*) = 0$$

  - **La condición suficiente de Primer Orden**

Podemos encontrar que dada una función, pueden existir puntos estacionarios (el gradiente de la función es cero en éstos puntos) que no son puntos mínimos. Es decir puede existir un punto máximo o un punto meseta.

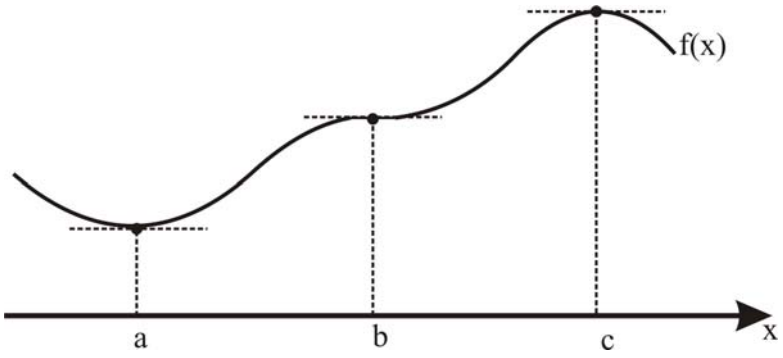


Figura A.1 **a** es un punto mínimo, **b** es un punto de inflexión y **c** es un punto máximo

Sin embargo para funciones convexas las situaciones anteriores son imposibles, por tanto se puede establecer que:

Sea una función convexa y diferenciable f(x) *en el punto* x* *y sea* $\nabla f(x^*) = 0$**,** *entonces x* es un punto mínimo global de* f(x) *en* $\boldsymbol{R^n}$**.** *Entonces para una función convexa la condición necesaria de Primer Orden también es una condición suficiente.*

APÉNDICE D

Macromodelo de simulación de los mecanismos de tuneleo Fowler-Nordheim e inyección de electrones calientes

La simulación en PSPICE de los mecanismos de inyección de electrones calientes y tuneleo Fowler-Nordheim no era posible debido a su complejidad y a que el potencial de canal se usa como una variable explícita. Así se propone un modelo de simulación empírico de los mecanismos de inyección de electrones calientes y tuneleo FN para una tecnología de 0.5µm; debido a que el modelo está basado en transistores y fuentes de corriente y voltaje dependientes, se puede utilizar para simulaciones en PSPICE. La figura D.1 muestra el macromodelo utilizado para simular los métodos de programación indirecta en PSPICE del capítulo 4.

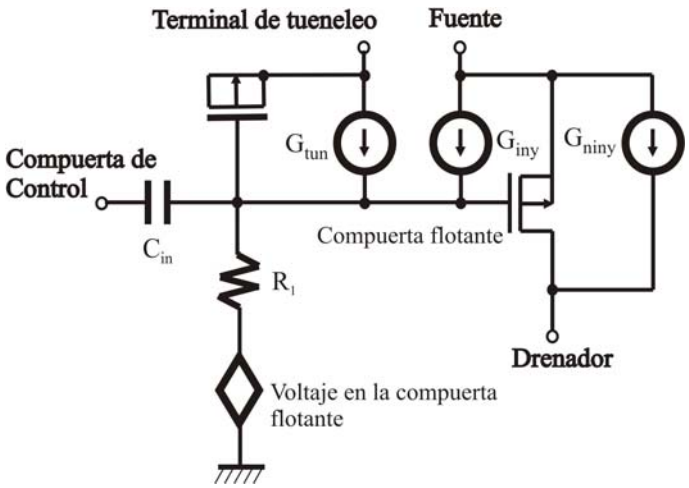


Figura D.1 Macromodelo de los mecanismo de inyección y tuneleo para su simulación en PSPICE

Donde la corriente de tuneleo (G_{tun})se aproxima a:

$$I_{tun} = -I_{tun0} WL \exp\left(-\frac{V_f}{V_{ox}}\right) \qquad \text{D.1}$$

Donde I_{tun0} es una corriente pre-exponencial, V_{ox} es el voltaje de óxido y V_f es un voltaje constante que depende del espesor de óxido.

También se propone una ecuación semi-empírica para la corriente de inyección (G_{iny}) dada por:

$$I_{inj} = \alpha I_s \exp\left(-\frac{\beta}{\left(V_{gd} + \delta\right)^2} + \lambda V_{SD} \right) \qquad \text{D.2}$$

Donde I_s es la corriente de fuente, V_{GD} es el voltaje compuerta – drenador y V_{SD} es el voltaje fuente – drenador. α, β y δ son parámetros de ajuste.

De igual forma se propone otro modelo empírico para modelar la corriente no inyectada (G_{niny}), dado por:

$$I_B = \eta I_S \left(\gamma V_{SD} - kV_{SG} + v_t\right) \exp\left(\frac{\lambda}{\gamma V_{SD} - kV_{SG} + V_T} \right) \qquad \text{D.3}$$

Donde I_s y I_B son las corrientes de substrato y fuente, V_{SD} y V_{SG} son los voltajes de fuente-drenador y fuente-compuerta respectivamente.