1. Circuitos aritméticos-lógicos

1.1. Operaciones aritméticas

Los **circuitos aritméticos** son elementos fundamentales en la construcción de computadoras. Estos circuitos realizan **operaciones aritméticas** como la suma, resta, comparación, multiplicación y división.

Un **semi-sumador** (1 bit) admite dos bits A y B de entrada, y genera salidas de suma Σ y acarreo C_{out}.

Un **sumador completo** (1 bit) admite dos bits A y B de entrada, un acarreo de entrada C_{in}, y genera salidas de suma Σ y acarreo C_{out}.

Para sumar dos números binarios de n dígitos se necesitan n sumadores completos. Un **sumador binario en paralelo** (n bits) se forma conectando n sumadores completos en paralelo para ampliar el tamaño de los números que se pueden sumar.

Un **sumador de acarreo serie** es aquel en el que la salida de acarreo C_{out} de cada sumador completo i se conecta a la entrada de acarreo C_{in} de cada sumador completo de orden superior $i+1$. Esto genera un retardo de propagación.

Un **sumador de acarreo anticipado** predice el acarreo de salida de cada etapa en función de los bits de entrada utilizando un circuito combinacional externo. De esta forma, se evita el retardo de propagación.

$$G_i = A_i \cdot A_b \qquad P_i = A_i + B_i \qquad C_{out_i} = G_i + P_i \cdot C_{out_{i-1}}$$

Un **restador completo** (1 bit) admite dos bits A y B de entrada, un préstamo de entrada B_{in}, y genera salidas de suma Σ y préstamo de salida B_{out}.

La **resta** de dos números binarios se puede implementar con sumadores al calcular el complemento a dos del sustraendo y tratando a los acarreos como préstamos.

La **multiplicación** de dos números binarios se pueden implementar con sumadores, mediante la suma de multiplicandos trasladados.

Un **comparador** combina dos operandos A y B, y genera tres salidas que indican la relación entre los operandos: $A < B$, $A = B$ y $A > B$.

1.2. Operaciones bit a bit

Una operación **bit a bit** opera sobre datos binarios a nivel de sus bits individuales. Se clasifican en operaciones lógicas, de desplazamiento y de rotación.

Las operaciones bit a bit **lógicas** toman dos números binarios y realizan la operación lógica correspondiente en cada par de bits. Las más comunes son la negación (NOT), el producto lógico (AND), la suma lógica (OR) y la OR exclusiva (XOR).

Un **desplazador lógico** desplaza un número hacia la izquierda o la derecha y completa los espacios vacíos con 0. Un **desplazador aritmético** hacia la derecha completa los espacios vacíos con el bit más significativo del número original, y hacia la izquierda completa los espacios vacíos con 0. Desplazar n posiciones hacia la izquierda o derecha equivale a la multiplicación o división por 2^n del número original.

La **rotación** es una operación de reordenamiento de las entradas de una n-upla. Un **rotador** desplaza un número hacia la izquierda o derecha como si los extremos estuviesen conectados. En la rotación se conservan los datos originales.

Un ***barrel shifter*** es un circuito digital que puede desplazar o rotar una palabra utilizando sólo lógica combinacional.

1.3. Unidad aritmético-lógica (ALU)

Una **unidad aritmético-lógica (ALU)** es un circuito digital que realiza operaciones aritméticas y lógicas entre uno o dos argumentos, lo que la vuelve una pieza fundamental para la construcción de una computadora.

- **Datos:** una ALU tiene tres buses de datos paralelos que consisten en dos operandos de entrada (A y B) y una salida de resultado (R).

- **Código de operación:** un bus de entrada (Op) transmite un código que especifica la operación a realizar. Su tamaño determina el número máximo de operaciones diferentes que la ALU puede realizar.
- **Estados:** son señales individuales que transmiten información complementaria sobre el resultado de la operación actual (F), como puede ser un bit de acarreo, cero, signo, desbordamiento o paridad.

Las ALU tienen señales para propagar estados, lo que permite el uso de múltiples ALU interconectadas para crear una ALU con un tamaño de palabra más amplio. Cada módulo procesa un **segmento** de los operandos. Esta técnica se llama **bit slicing**.

Una alternativa al bit slicing es el uso de una misma ALU para procesar distintos segmentos en diferentes ciclos de reloj.

Las ALU tienen una **unidad aritmética**, una **unidad lógica**, una **unidad de corrimiento**, una **lógica de banderas** y un **multiplexor de salida**.

La **arquitectura** de una ALU depende de los objetivos de diseño:

- **Arquitectura secuencial:** Múltiples ALU se conectan de forma secuencial con sus señales de propagación de datos. Los operandos A y B se conectan en paralelo a las ALU, que tienen salidas R en paralelo.
- **Arquitectura paralela simultánea**: Los operandos A y B se conectan en paralelo a los módulos de las unidades aritméticas y lógicas. Las salidas de estos módulos están conectadas a multiplexores, cuya selección controlada por el Op será la salida de la ALU.
- **Arquitectura paralela de bajo consumo:** Los operandos A y B se conectan en paralelo a demultiplexores para cada unidad, cuya selección controlada por el Op será la entrada de cada módulo. Las salidas de estos módulos están conectadas a multiplexores, cuya selección controlada por el Op será la salida de la ALU.

2. Circuitos secuenciales

Un **circuito secuencial** es un circuito cuyas salidas dependen de las entradas actuales, las entradas previas y las salidas previas. Para implementar esto, los circuitos secuenciales tienen **memoria** y **retroalimentación**.

Un circuito secuencial puede ser:

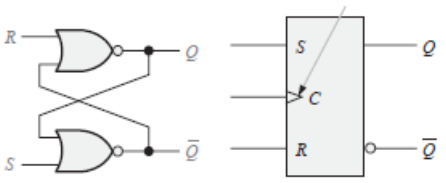
- **Controlado por eventos:** asincrónico, cuyos estados cambian cuando cambian sus entradas.
- **Controlado por reloj:** sincrónico, cuyos estados cambian cuando cambian sus entradas y la señal de reloj.
- **Controlado por pulsos:** combinación de asincrónicos y sincrónicos.

El elemento fundamental para la construcción de memorias es el **biestable**, un elemento con dos estados estables.

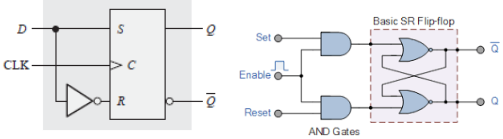
Un **latch** es un biestable que cambia de estado de forma asincrónica.

Un **flip-flop** es un biestable que cambia de estado de forma sincrónica, con entrada de reloj, pero que también puede tener entradas asíncronas.

El **latch SR** tiene entradas S y R y salida Q. Cuando S y R son 0, la salida no cambia. Cuando S es 1 y R es 0, la salida es 1. Cuando S es 0 y R es 1, la salida es 0. Cuando S y R son 1, pueden producirse errores. A partir de este, se construyen los demás latches y flip-flops. El **flip-flop SR** es un latch SR que actualiza su estado sólo en flanco positivo de reloj.



El **flip-flop D** tiene una única entrada D, y la salida toma el valor de esa entrada en cada flanco positivo de reloj.



El **flip-flop JK** funciona igual que el flip-flop SR en seteo, reseteo y no cambio, y no tiene condiciones no válidas de entrada. Cuando ambas entadas son 1, el flip-flop alterna sus valores de salida.

El **retardo de propagación** es el intervalo de tiempo requerido para que se produzca un cambio en la salida una vez que se ha aplicado una señal en la entrada.

El **tiempo de establecimiento** es el intervalo mínimo que los niveles lógicos deben mantener constantes en las entradas antes de que llegue el flanco de disparo del impulso de reloj, de modo que dichos niveles sincronicen correctamente en el flip-flop.

El **tiempo de mantenimiento** es el intervalo mínimo que los niveles lógicos deben mantenerse constantes en las entradas después de que haya pasado el flanco de disparo del impulso de reloj, de modo que dichos niveles se sincronicen correctamente en el flip-flop.

Los flip-flops están sujetos a un problemas de **metaestabilidad**. Cuando dos entradas cambian aproximadamente al mismo tiempo y el orden no es claro, la salida puede comportarse de manera impredecible.

Los problemas de metaestabilidad pueden evitarse asegurando que los datos y las entradas de control se mantengan válidas y constantes durante el **período de apertura**.

Otra forma de evitar problemas de metaestabilidad consiste en conectar múltiples flip-flops en serie, todos compartiendo un reloj común.

2.1. Registros

Un **registro** es un circuito formado por n flip-flops junto a una lógica combinacional de control.

Las entradas y salidas de un registro pueden ser en **serie** o en **paralelo**. Además, pueden tener entradas asincrónicas de seteo y reseteo.

Se los puede utilizar como **almacenamiento** de datos, como líneas de **retardo** digitales en tareas de **sincronización**, o **conversores** de datos en formato serie-paralelo.

Un registro **serie-serie** tiene una única entrada que se conecta a la entrada del primer flip-flop. La salida de cada flip-flop se conecta a la entrada del siguiente flip-flop. La salida del último flip-flop es la salida del registro.

Un registro **serie-paralelo** tiene una única entrada que se conecta a la entrada del primer flip-flop. La salida de cada flip-flop se conecta a la entrada del siguiente flip-flop y también a su salida en paralelo correspondiente del registro.

Un registro **paralelo-serie** tiene n entradas que se conectan a la entrada de su correspondiente flip-flop. La salida de cada flip-flop se conecta a la entrada del siguiente flip-flop. La salida del último flip-flop es la salida del registro. Una señal de carga/desplazamiento determina si se guardan los valores de entrada en los flip-flops o si se desplazan a la salida. Se puede implementar con multiplexores en las entradas sincrónicas de cada flip-flop, o con flip-flops con entradas asincrónicas.

Un registro **paralelo-paralelo** tiene n entradas que se conectan a la entrada de su correspondiente flip-flop y n salidas que se conectan a la salida de su correspondiente flip-flop.

Un registro **serie-serie bidireccional** permite desplazar datos en ambas direcciones. Se implementan con multiplexores que permiten la transferencia de un bit de datos de una etapa a la siguiente de la izquierda o de la derecha, dependiendo del nivel de una línea de control.

Un registro **universal** tiene capacidad de entrada y salida en serie y paralelo. Su funcionamiento se controla con una entrada de selección de operación.

2.2. Memorias

Una **memoria** es un dispositivo que almacena datos de forma estructurada.

La **persistencia** de una memoria es su capacidad de retener información.

- **Volátil:** la integridad de la información depende de la alimentación.
- **No volátil:** la información se preserva aún en ausencia de energía.

La a forma de lectura y escritura de los datos una matriz puede ser en **paralelo** o en **serie**.

El **mecanismo de direccionamiento:** la forma en la que se accede a un dato en particular.

- **Aleatoria:** los datos pueden ser accedidos en cualquier orden.
- **Secuencial:** para acceder a un dato, se debe acceder a todos los que lo preceden.

Las **operaciones** que puede realizar una memoria pueden ser de **lectura y escritura** o de **solo lectura**.

Las **memorias de sólo lectura (ROM)** almacenan de forma permanente o semipermanente los datos almacenados. Se utilizan en la microprogramación de sistemas.

- **PROM** (ROM programable): sólo puede programarse una vez, compuesta por fusibles que se destruyen durante la programación.

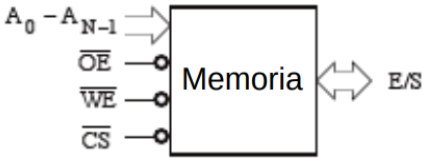
- **EPROM** (PROM borrable): PROM que puede programarse, borrarse y reprogramarse. El borrado es mediante exposición a luz UV.
- **EEPROM** (PROM borrable eléctricamente): PROM que puede programarse, borrarse y reprogramarse. El borrado es mediante un pulso eléctrico.
- **Flash EEPROM**: EEPROM con una estructura más sofisticada. Puede modificarse sin usar dispositivos especiales.

Las **memorias de acceso aleatorio (RAM)** son memorias de lectura-escritura en las que los datos se pueden escribir o leer en cualquier dirección de memoria. Se utilizan habitualmente para almacenamiento de datos a corto plazo porque son volátiles.

Una **palabra de memoria** o **celda** es un conjunto de n bits adyacentes. Cada posición de memoria se identifica a través de una **dirección** de m bits. Con un decodificador se pueden direccionar 2^m posiciones. En general, las RAM se implementan con una **matriz de celdas**, teniendo un decodificador para la línea y otro para la columna.

La **capacidad de memoria** se puede expresar como el número de posiciones (2^m celdas) o el número de bits ($n2^m$ bits).

Las memorias están diseñadas para conectar múltiples a un mismo bus de datos. Cada chip de memoria posee una **entrada de selección** $\overline{CS}$ que la conecta/desconecta del bus de datos, una **entrada de lectura** $\overline{OE}$ que habilita/bloquea la lectura y una **entrada de escritura** $\overline{WE}$ que habilita/bloquea la escritura.



En una **memoria con múltiples puertos**, cada puerto es una ruta de acceso independiente para leer o escribir datos en la matriz de la memoria. Se puede acceder a la matriz de forma aleatoria a través de cada puerto.

Cada **puerto** tiene un bus de direcciones, una señal de lectura/escritura y un bus bidireccional de datos.

Una memoria de n puertos actúa como n memorias independientes de idéntico contenido, permitiendo leer n veces más rápido que procesando los accesos secuencialmente.

Una memoria con múltiples puertos tiene un conflicto cuando se quiere escribir una celda desde dos puertos distintos en simultáneo.

Las memorias multipuerto se pueden clasificar de acuerdo a su modo de operación y organización interna en:

- **Memoria multipuerto asíncrona:** responde a los cambios de dirección y control sin la necesidad de relojes.
- **Memoria multipuerto síncrona:** responde de forma sincrónica a los cambios en las señales de dirección y control en relación a un reloj.
- **Memoria multipuerto conmutable por banco:** se divide la memoria en bancos, a los que se puede acceder desde un sólo puerto a la vez.
- **Memoria multipuerto de acceso secuencial y aleatorio:** tiene una interfaz de acceso aleatorio en un puerto y una interfaz secuencial tipo *FIFO* en el otro. Se utilizan como interfaz entre sistemas asíncronos y síncronos.

En una **memoria de acceso secuencial *FIFO***, la información que entra primero será la primera en salir (*first in, first out*). Se utilizan como interfaces en sistemas que operan a velocidades diferentes.

Para manejar la información tiene dos punteros (de lectura y de escritura) y dos banderas que reflejan su estado (llena, vacía).

La implementación basada en registros utiliza un conjunto de registros de desplazamiento que se conectan entre sí de manera que los datos se empujan cuando se leen/eliminan de la memoria.

La implementación con una RAM utiliza dos registros adicionales para almacenar las direcciones del lectura y escritura que se actualizan en cada lectura/escritura.

En una **memoria de acceso secuencial *LIFO***, la información que entra último será la primera en salir (*last in, first out*).

La implementación basada en registros utiliza un conjunto de registros de desplazamiento que se conectan entre sí de manera que los datos se empujan hacia abajo o hacia arriba cuando los datos se almacenan o eliminan de la memoria.

La implementación con una RAM utiliza un puntero de pila que almacena la dirección de la parte superior de la pila que se actualiza tras cada lectura/escritura.

Una **memoria asociativa** compara los datos de entrada con los que están almacenados para generar un indicador de correspondencia y devolver la dirección de los datos coincidentes.

Para realizar una escritura se necesita el dato y la dirección donde se almacena. Para realizar una lectura, ingresa el dato y se obtiene su dirección e información asociada a través de **asociación por comparación**.

3. Máquinas de estados finitos

Una **máquina de estados finitos (*FSM*)** es una máquina abstracta que puede estar, en cualquier momento, en uno de un número finito de estados.

Un **estado** es una descripción de las variables internas de un sistema. El estado actual depende del **estado inicial** y las entradas anteriores. El **estado final** detiene la FSM.

Una **transición** es un conjunto de acciones que se ejecutan cuando se cumple una condición o cuando sucede un evento. Una FSM puede cambiar de un estado a otro en una transición.

Una FSM se define a partir de un alfabeto Σ de entrada, un alfabeto Γ de salida, un conjunto finito S de estados, una función T de transición de estados y una función G de salida.

Un **semiautómata** es una FSM sin función de salida.

En una **máquina de Moore**, la salida depende sólo del estado actual.

En una **máquina de Mealy**, la salida depende del estado actual y de la entrada.

Una FSM se puede representar con:

- **Diagrama de estados:** grafo orientado donde los estados son vértices y las transiciones son arcos.
- **Tabla de transición de estados:** esencialmente una tabla de verdad que muestra la salida y a qué estado se moverá la FSM en función del estado actual y las entradas.
- **Diagrama UML**.

Las FSM se clasifican en:

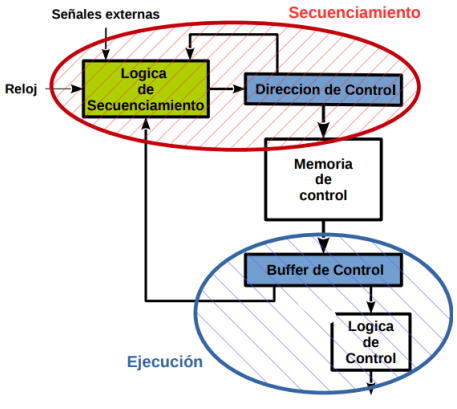
- **Receptores:** producen una salida binaria que indica si se acepta o no la entrada recibida.
- **Transductores:** producen salida y entrada de lectura.
- **Secuenciadores:** tienen un alfabeto de entrada de una sola letra y producen solo una secuencia de salidas.

3.1. Microprogramación

Un **autómata** es una máquina abstracta que representa el comportamiento de un sistema, proceso o algoritmo en términos de estados y transiciones.

Un **lenguaje de máquina** es un sistema de códigos interpretable por el controlador de un autómata. Está compuesto por un conjunto de **microinstrucciones** que determinan las operaciones que puede ejecutar el autómata.

Un **sistema microprogramado** consiste en un conjunto de elementos que permiten ejecutar una secuencia de microinstrucciones.



La **unidad de control microprogramada** incluye las lógicas para **secuenciar** a través de microoperaciones, **ejecutar** las microoperaciones y tomar decisiones basadas en entradas externas.

- **Memoria de programa:** ROM que almacena las operaciones a realizar, codificadas como una secuencia de micro-instrucciones.
- **Lógica de secuenciamiento:** circuito secuencial encargado de elegir la próxima micro-instrucción a ejecutar.
- **Lógica de control:** circuito combinacional encargado de generar las señales de control a partir de información almacenada en las microinstrucciones y señales externas.

El **ciclo de microinstrucción** es el evento básico en un sistema microprogramado y consiste en la búsqueda y ejecución de las microinstrucciones.

Hay dos formas en las que se puede organizar la información en una microinstrucción:

- **Microprogramación vertical:** cada micro-instrucción especifica sólo una micro-operación a realizar.
- **Microprogramación horizontal:** cada micro-instrucción especifica múltiples micro-operaciones a realizar en paralelo.

El **micro-secuenciador** es un contador al que se le puede cargar una dirección arbitraria para modificar el flujo de ejecución. Hay tres arquitecturas básicas para generar la **lógica de secuenciamiento**:

- **Campo con dos direcciones:** dos campos de direcciones en cada microinstrucción, uno por cada rama de ejecución.
- **Campo de dirección única:** las señales de selección de dirección determinan qué opción se selecciona.
- **Formato variable:** dos formatos de microinstrucción diferentes, que se selecciona con un bit que determina el formato utilizado en cada instrucción.

4. Máquinas algorítmcias de estados

4.1. Máquinas abstractas

Un **algoritmo** es una secuencia finita de instrucciones finitas, no ambiguas, ordenadas e implementables.

- **Tiempo secuencial:** un algoritmo define una secuencia de estados computacionales por cada entrada válida.
- **Estado abstracto:** cada estado computacional se describe formalmente utilizando una estructura de primer orden.
- **Exploración acotada:** la transición de un estado al siguiente queda completamente determinada por una descripción fija y finita.
- **Aritmetizabilidad:** solo las operaciones calculables están disponibles en el paso inicial.

La descripción de un algoritmo se hace en tres niveles:

- **Alto nivel:** se explica el algoritmo con ilustraciones y omitiendo detalles.
- **Formal:** se usan formas estructuradas e independientes de las técnicas de implementacion para describir la secuencia de tareas.
- **Implementación:** se selecciona las técnicas específicas para implementar el algoritmo.

Los **modelos de computación** se utilizan para estudiar la **complejidad computacional** de un algoritmo. Se clasifican en tres categorías:

- **Secuenciales:** una tarea se ejecuta después de otra. Son fáciles de entender e implementar.
- **Funcionales:** se usan funciones matemáticas para representar las tareas a realizar.
- **Concurrentes:** varias tareas ocurren simultáneamente.

Una **máquina abstracta** es un modelo de un sistema informático, construido para realizar un análisis del funcionamiento de un algoritmo. El modelo consiste en entradas, salidas y las operaciones que se pueden realizar.

Un **autómata finito** tiene un número finito de estados. Cada estado acepta un número finito de entradas tiene reglas que describen la acción para cada entrada. Al mismo tiempo, una entrada puede hacer que la máquina cambie de estado. Para cada entrada hay exactamente una transición. Debido a la restricción en el número de estados posibles, en un autómata finito sólo se pueden implementar algoritmos que requieran un número finito de estados.

Un **autómata con pila** es un autómata finito que tiene una memoria LIFO (pila). Las transiciones dependen de la entrada, el estado activo y la pila, a la que puede leer o escribir. Si bien tiene la restricción en el número de estados posibles, la pila le provee de una memoria que hace que pueda implementar más algoritmos que un autómata finito.

Un **autómata con procesador de datos** está compuesto por un **autómata finito**, que controla el flujo de tareas a ejecutar, y un **camino de datos**, que realiza las operaciones de procesamiento de datos. Una ruta de datos es una colección de unidades funcionales que realizan operaciones de procesamiento de datos, registros y buses.

Una **máquina de Turing** consiste en una relación de entrada–salida. La entrada se da en la cinta de entrada de la máquina, y la salida consiste en el contenido de la cinta de salida cuando la máquina se detiene. La máquina de Turing se comporta como un computador con un programa fijo, por lo que se debe construir uno diferente para cada cálculo.

La **máquina universal de Turing** es una máquina capaz de simular cualquier otra máquina de Turing.

Una **máquina con registros** utiliza registros con direcciones exclusivas, cada uno de ellos identificado por un único número entero positivo.

De acuerdo con su complejidad, se clasifican en cuatro subclases:

- **Máquina de conteo:** comprende un conjunto de registros y un conjunto de instrucciones para construir los programas. Un autómata, separado de los registros, ejecuta el programa almacenado en las instrucciones.
- **Máquina de puntero:** comprende un conjunto de registros y un autómata que ejecuta el programa. El conjunto de instrucciones no tiene operaciones aritméticas, sino operaciones de control que modifican la estructura de almacenamiento.
- **Máquina de acceso aleatorio:** comprende un conjunto de registros, un conjunto de instrucciones y direccionamiento indirecto. Dependiendo de la instrucción, el autómata obtiene la dirección de un registro **directamente** de la instrucción o **indirectamente** del contenido de un puntero especificado en la instrucción a partir de un parámetro de la misma.
- **Máquina de acceso aleatorio con programa almacenado:** es una máquina de acceso aleatorio cuyo programa está almacenado en sus registros junto con sus entradas. Como el algoritmo esta almacenado en memoria, esta máquina tiene la capacidad de auto-modificar instrucciones de programa.

4.2. Máquinas algoritmicas de estados

Un **diagrama de flujo** es la representación gráfica de un algoritmo.

Las **máquinas algorítmicas de estados (ASM)** son un método para diseñar máquinas de estado finito a partir de la descripción de la secuencia de operaciones de un sistema digital.

Utilizan tres tipos de bloques:

- **Estado:** tiene una entrada y una salida. Especifica una o más operaciones que podrían completarse simultáneamente en un ciclo de reloj.
- **Decisión:** tiene una entrada y múltiples salidas. Especifica una serie de rutas alternativas que se pueden seguir.
- **Condición:** Sigue a un bloque de decisión y contiene operaciones condicionales que se invocan cuando el bloque de decisión selecciona la ruta que lo contiene.

El **controlador** se determina a partir de los bloques de decisión y las transiciones de estado. Se puede implementar utilizando una FSM o microprogramación.

El **procesador de datos** se obtiene de las operaciones especificadas en los bloques de estados y condicionales.