# Lenguaje RTL

- Descripción del funcionamiento del procesador a nivel de transferencia de registros
  - Descripción a más bajo nivel que el nivel de instrucción (tema1)
- Lenguaje RTL
  - Herramienta que posibilita trasladar la especificación de un sistema a su implementación hardware

# Lenguaje RTL

- Secuencia ordenada de operaciones que deben realizarse en un solo ciclo de reloj
- Operación básica: Transferencia
  - Transferir información entre dos elementos (memoria, entrada o salida) pudiendo ser transformada
- Primitivas del lenguaje:
  - Puertas lógicas (AND, OR, NOT, NAND, …)
  - Bloques combinacionales (multiplexores, codificadores, comparadores, sumadores, …)
  - Bloques secuenciales (biestables, registros, memorias, …)

# Lenguaje RTL: Notación

- Transferencia
  - Fuentes de información:
    - Entradas y elementos de memoria (registro, biestable)
  - Destinos de información:
    - Salidas y elementos de memoria
  - Podemos representar:
    - señales simples (biestables, línea simple): nombre en minusculas
    - vectores de señales (buses, registros): nombre en mayúsculas

# Lenguaje RTL: Notación

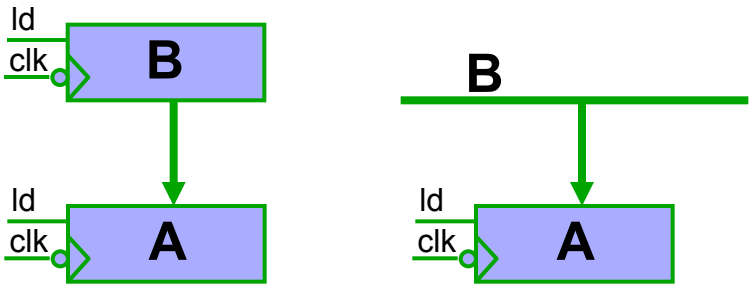
- Notación vectores de señales (buses, registros)
  - Todo el conjunto de señales (n bits):
    - A (para transferencias completas)
    - A[n] (para declaración del vector)
    - A[n-1:0] (enumeración de todas las posiciones)
  - Subconjunto de señales del vector:
    - Nombre[bit superior:bit inferior]
    - A[7:5] (los bits 7,6 y 5 de A)

# Lenguaje RTL: Notación

- Transferencias
  - Notación: **A ← B**
  - Reglas:
    - El número de bits en destino/fuente debe coincidir
    - El orden de los bits se mantiene (más significativo de B va a más significativo de A …)

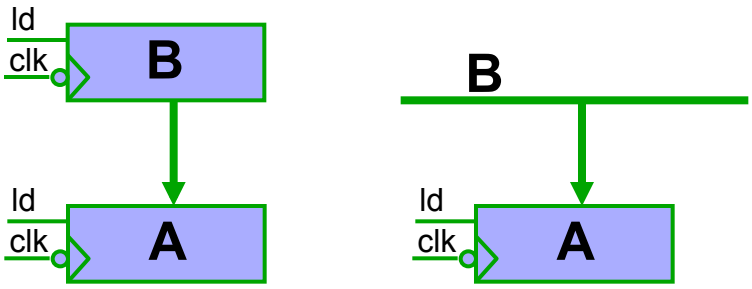
# Lenguaje RTL: Notación

- Transferencias
  - Notación: **A ← B**
  - Reglas:
    - El número de bits en destino/fuente debe coincidir
    - El orden de los bits se mantiene (más significativo de B va a más significativo de A …)



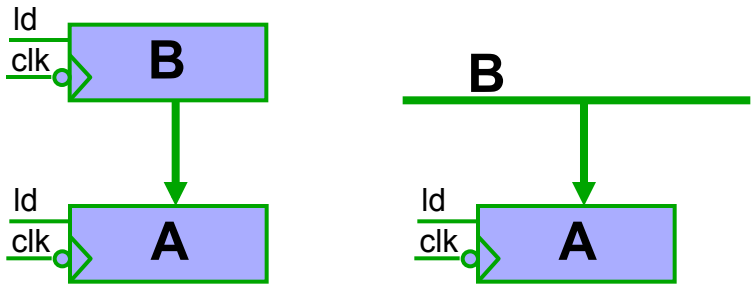
# Lenguaje RTL: Notación

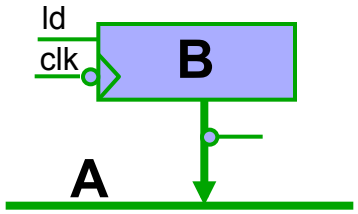
- Transferencias
  - Notación: **A ← B**
  - Reglas:
    - El número de bits en destino/fuente debe coincidir
    - El orden de los bits se mantiene (más significativo de B va a más significativo de A …)
  - Si el destino es una Salida se denomina CONEXIÓN
    - Notación: **A = B**



# Lenguaje RTL: Notación

- Transferencias
  - Notación: **A ← B**
  - Reglas:
    - El número de bits en destino/fuente debe coincidir
    - El orden de los bits se mantiene (más significativo de B va a más significativo de A …)
  - Si el destino es una Salida se denomina CONEXIÓN
    - Notación: **A = B**



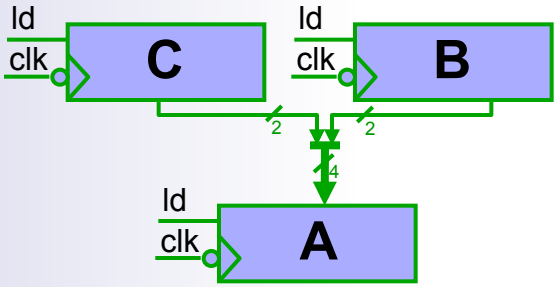



# Lenguaje RTL: Notación

- Transformaciones
  - Concatenación o unión:
    - Unión de varias fuentes en un vector de información
    - Notación: **A,B**
    - Ejemplos (registro A de 4 posiciones: A[4])
      - Desplazamiento izquierda: **A ← A[2:0],0**
      - Rotación derecha: **A ← A[0],A[3:1]**
      - Concatenación: **A ← C[1:0],B[1:0]**

# Lenguaje RTL: Notación

- Transformaciones
  - Concatenación o unión:
    - Unión de varias fuentes en un vector de información
    - Notación: **A,B**
    - Ejemplos (registro A de 4 posiciones: A[4])
      - Desplazamiento izquierda: **A ← A[2:0],0**
      - Rotación derecha: **A ← A[0],A[3:1]**
      - Concatenación: **A ← C[1:0],B[1:0]**

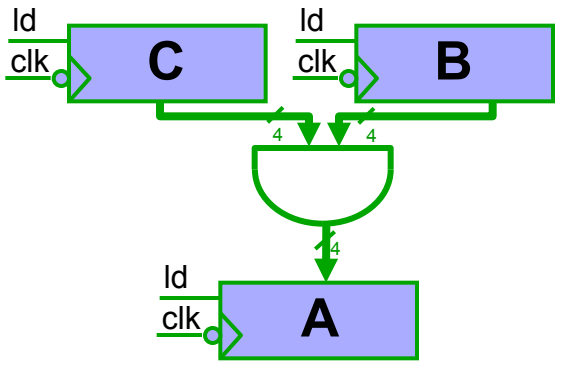


# Lenguaje RTL: Notación

- Transformaciones
  - Operaciones lógicas:
    - Complemento: $B \leftarrow \overline{A}$
    - OR: $A \leftarrow B \vee C$
    - AND: $A \leftarrow B \wedge C$

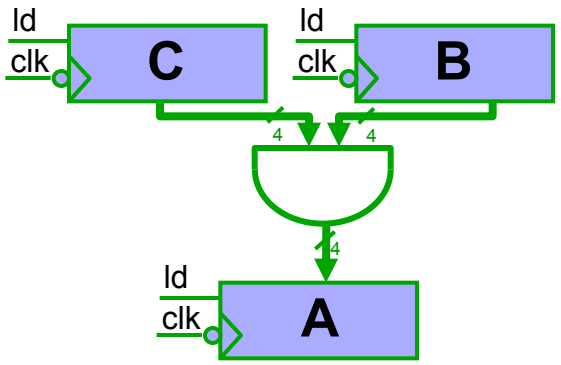
# Lenguaje RTL: Notación

- Transformaciones
  - Operaciones lógicas:
    - Complemento: $B \leftarrow \overline{A}$
    - OR: $A \leftarrow B \vee C$
    - AND: $A \leftarrow B \wedge C$



# Lenguaje RTL: Notación

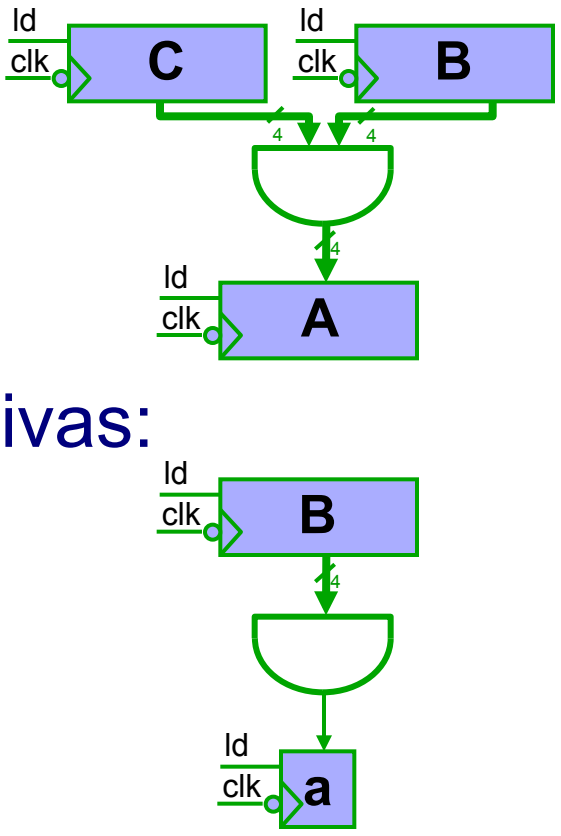
- Transformaciones
  - Operaciones lógicas:
    - Complemento: $B \leftarrow \overline{A}$
    - OR: $A \leftarrow B \vee C$
    - AND: $A \leftarrow B \wedge C$




  - Operaciones lógicas acumulativas:
    - OR acumulativo: $a \leftarrow \vee / B$
    - AND acumulativo: $a \leftarrow \wedge / B$

# Lenguaje RTL: Notación

- Transformaciones
  - Operaciones lógicas:
    - Complemento: **B** ← $\overline{\mathbf{A}}$
    - OR: **A** ← **B** ∨ **C**
    - AND: **A** ← **B** ∧ **C**
  - Operaciones lógicas acumulativas:
    - OR acumulativo: **a** ← ∨/ **B**
    - AND acumulativo: **a** ← ∧/ **B**

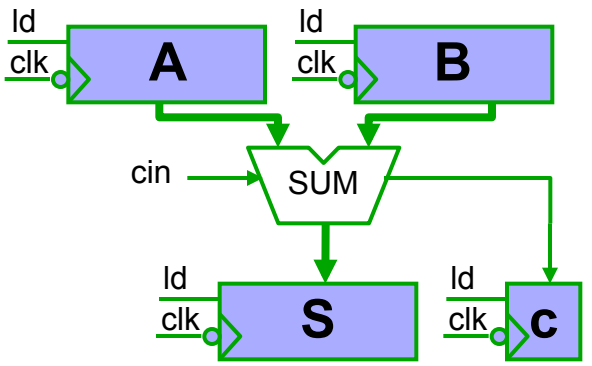


# Lenguaje RTL: Notación

- Transformaciones
  - Función combinacional general:
    - Tratamiento igual que vector de señales
    - Notación: NOMBRE(entrada1;entrada2;…;entradaN)
    - Ejemplo sumador: SUM
      - S ← SUM(A; B; cin)[n-1:0]
      - c ← SUM(A; B; cin)[n]

# Lenguaje RTL: Notación

- Transformaciones
  - Función combinacional general:
    - Tratamiento igual que vector de señales
    - Notación: NOMBRE(entrada1;entrada2;…;entradaN)
    - Ejemplo sumador: SUM
      - S ← SUM(A; B; cin)[n-1:0]
      - c ← SUM(A; B; cin)[n]



# Lenguaje RTL: Notación

- Transferencia condicional
  - Sólo se produce la transferencia de información si se cumple una Condición
  - Condición: cualquier expresión/elemento de un bit. (a, A[5], V/B, …)
  - Notación: **Destino*condición ← Fuente**
  - Ejemplos:
    - **A*MAYOR(X;A) ← X**
    - **A*(V/B) ← B**

# Lenguaje RTL

- Especificación de un sistema
  - Sección de datos
    - Declaración de elementos de memoria, entradas y salidas.

```
MODULE: EJEMPLO
  MEMORY: A[2];B[2].
  INPUTS: X[2].
  OUTPUTS: Z[2].

1. A ← X.
2. B ← A[0],A[1].
3. A ← B V A.
4. Z = A.

ENDSEQUENCE
  CONTROL RESET(1)
END
```

# Lenguaje RTL

- Especificación de un sistema
  - Sección de datos
    - Declaración de elementos de memoria, entradas y salidas.
  - Sección de control (síncrona)
    - Estados por los que pasa el sistema y acciones que tienen lugar
    - Cada línea en un ciclo de reloj

```
MODULE: EJEMPLO
  MEMORY: A[2];B[2].
  INPUTS: X[2].
  OUTPUTS: Z[2].

1. A ← X.
2. B ← A[0],A[1].
3. A ← B V A.
4. Z = A.

ENDSEQUENCE
  CONTROL RESET(1)
END
```

# Lenguaje RTL

- Especificación de un sistema
  - Sección de datos
    - Declaración de elementos de memoria, entradas y salidas.
  - Sección de control (síncrona)
    - Estados por los que pasa el sistema y acciones que tienen lugar
    - Cada línea en un ciclo de reloj
  - Sección de control (asíncrona)
    - Conexiones permanentes

```
MODULE: EJEMPLO
  MEMORY: A[2];B[2].
  INPUTS: X[2].
  OUTPUTS: Z[2].

1. A ← X.
2. B ← A[0],A[1].
3. A ← B V A.
4. Z = A.

ENDSEQUENCE
  CONTROL RESET(1)
END
```

# Lenguaje RTL

- Temporización
  - Todas las transferencias de un mismo estado suceden todas a la vez
  - Las transferencias ocurren a final del ciclo sincronizadas por flanco de bajada
  - Las conexiones ocurren durante todo el ciclo

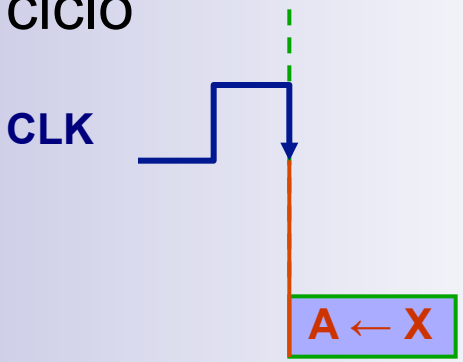
```
MODULE: EJEMPLO
  MEMORY: A[2];B[2].
  INPUTS: X[2].
  OUTPUTS: Z[2].

1. A ← X.
2. B ← A[0],A[1].
3. A ← B V A.
4. Z = A.

ENDSEQUENCE
  CONTROL RESET(1)
END
```

# Lenguaje RTL

- Temporización
  - Todas las transferencias de un mismo estado suceden todas a la vez
  - Las transferencias ocurren a final del ciclo sincronizadas por flanco de bajada
  - Las conexiones ocurren durante todo el ciclo




```
MODULE: EJEMPLO
  MEMORY: A[2];B[2].
  INPUTS: X[2].
  OUTPUTS: Z[2].

1. A ← X.
2. B ← A[0],A[1].
3. A ← B V A.
4. Z = A.

ENDSEQUENCE
  CONTROL RESET(1)
END
```

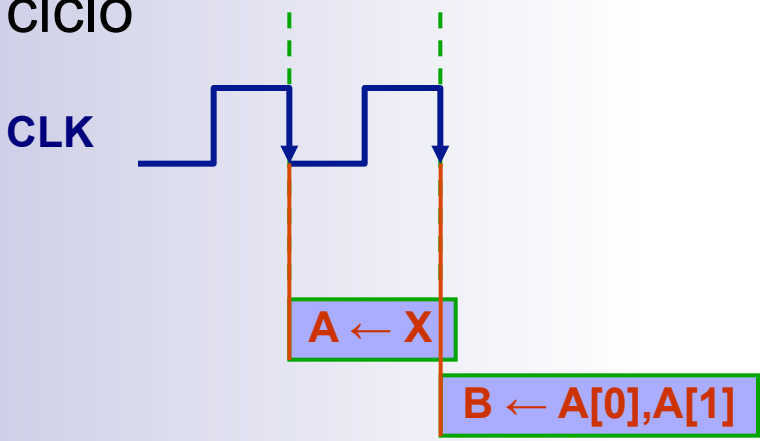
# Lenguaje RTL

- Temporización
  - Todas las transferencias de un mismo estado suceden todas a la vez
  - Las transferencias ocurren a final del ciclo sincronizadas por flanco de bajada
  - Las conexiones ocurren durante todo el ciclo

```
MODULE: EJEMPLO
  MEMORY: A[2];B[2].
  INPUTS: X[2].
  OUTPUTS: Z[2].

1. A ← X.
2. B ← A[0],A[1].
3. A ← B V A.
4. Z = A.

ENDSEQUENCE
  CONTROL RESET(1)
END
```



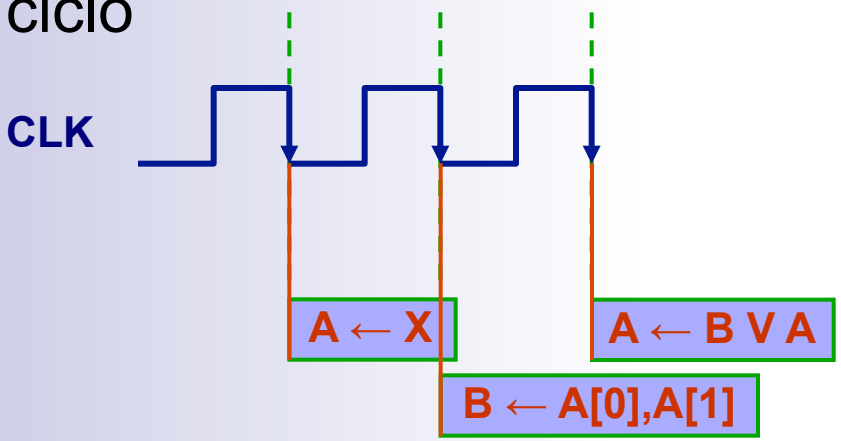
# Lenguaje RTL

- Temporización
  - Todas las transferencias de un mismo estado suceden todas a la vez
  - Las transferencias ocurren a final del ciclo sincronizadas por flanco de bajada
  - Las conexiones ocurren durante todo el ciclo

```
MODULE: EJEMPLO
  MEMORY: A[2];B[2].
  INPUTS: X[2].
  OUTPUTS: Z[2].

1. A ← X.
2. B ← A[0],A[1].
3. A ← B V A.
4. Z = A.

ENDSEQUENCE
  CONTROL RESET(1)
END
```



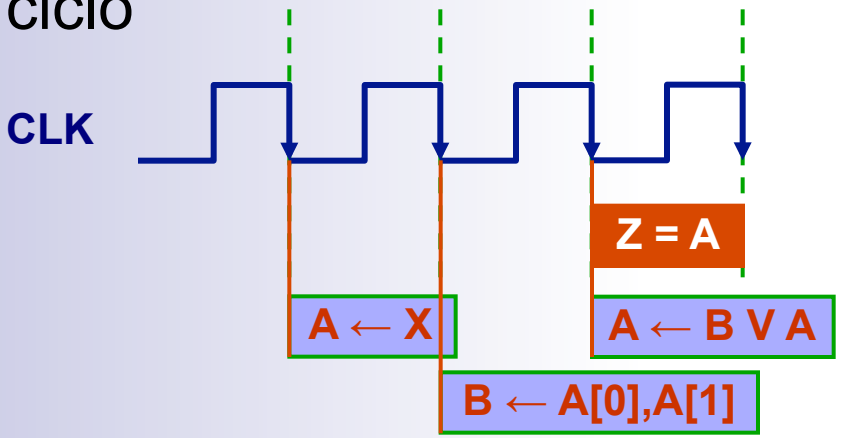
# Lenguaje RTL

- Temporización
  - Todas las transferencias de un mismo estado suceden todas a la vez
  - Las transferencias ocurren a final del ciclo sincronizadas por flanco de bajada
  - Las conexiones ocurren durante todo el ciclo

```
MODULE: EJEMPLO
  MEMORY: A[2];B[2].
  INPUTS: X[2].
  OUTPUTS: Z[2].

1. A ← X.
2. B ← A[0],A[1].
3. A ← B V A.
4. Z = A.

ENDSEQUENCE
  CONTROL RESET(1)
END
```



# Lenguaje RTL

- Control de la secuencia
  - Cada sentencia dura un ciclo
  - Después de cada sentencia se ejecuta la siguiente en la lista
  - Hay sentencias de salto que rompen esta secuencia
    - Salto incondicional: $\rightarrow (S)$
      - Siguiente sentencia S
    - Salto condicional: $\rightarrow (f_1,f_2,\ldots,f_n)/(S_1,S_2,\ldots,S_n)$
      - Si ($f_i$) la siguiente sentencia $S_i$