Arquitecturas de SE y Alternativas de Diseño

Dr. Óscar Rodríguez Polo
Dr. Sebastián Sánchez Prieto
Dr. Pablo Parra Espada

Bibliografía:

Embedded Systems Architecture : A Comprehensive Guide for Engineers and Programmers.- Tammy Noergaard Elsevier Science Publication. ISBN:0750677929

Embedded System Design: A Unified Hardware/Software Introduction Frank Vahid and Tony Givargis John Wiley & Sons; ISBN: 0471386782.

2017 Embebedded Market Study

Esquema

- Arquitectura de un SE
- Ciclo de Vida de un SE
  - Métricas de diseño
- Alternativas de diseño e implementación de la CPU de un SE
- Técnicas de Diseño y Validación de un SE

Arquitectura de un SE

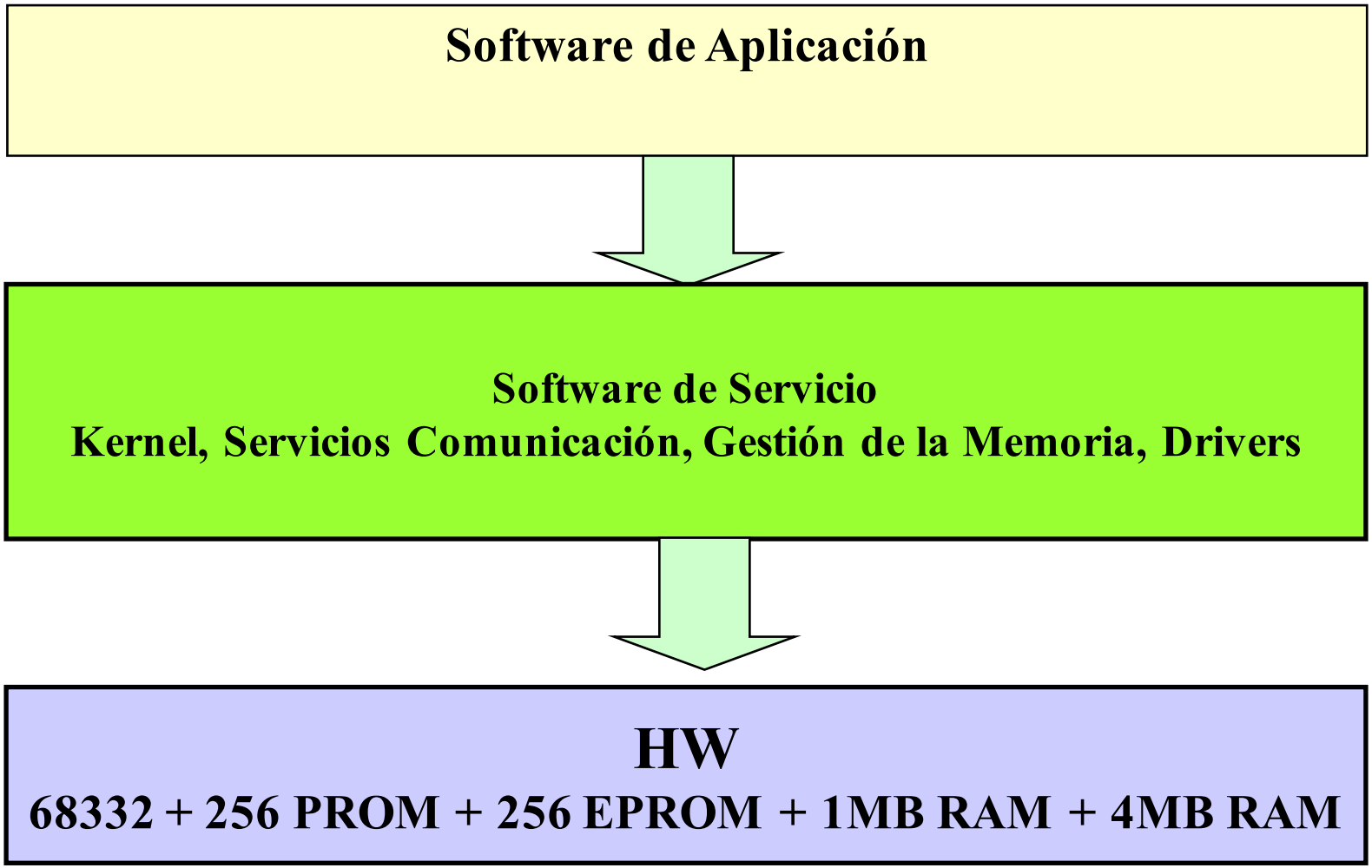
Arquitectura de un SE

- Definición:
  - Abstracción que aporta la información necesaria para comprender el funcionamiento del sistema, los elementos que lo constituyen y las interacciones entre ellos.
  - La arquitectura no entra en detalles de implementación de los distintos elementos.
  - Generalmente, para definir la arquitectura de un SE, se emplean distintos esquemas o diagramas denominados **estructuras**
  - Cada estructura muestra una vista que describe parte de la funcionalidad del SE.

Arquitectura = Σ Estructuras

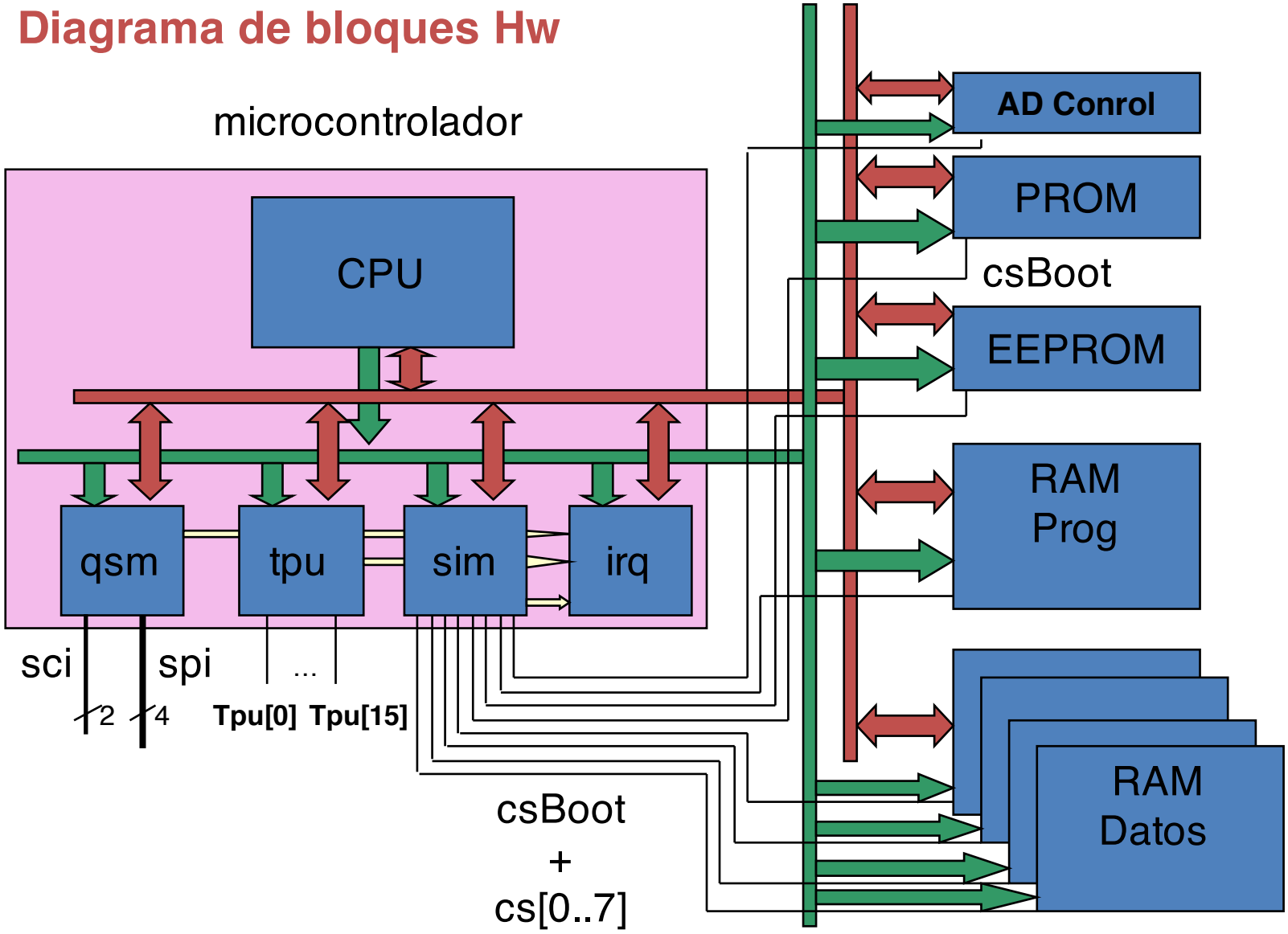
Ejemplo Estructuras OBDH

Estructura Global del sistema



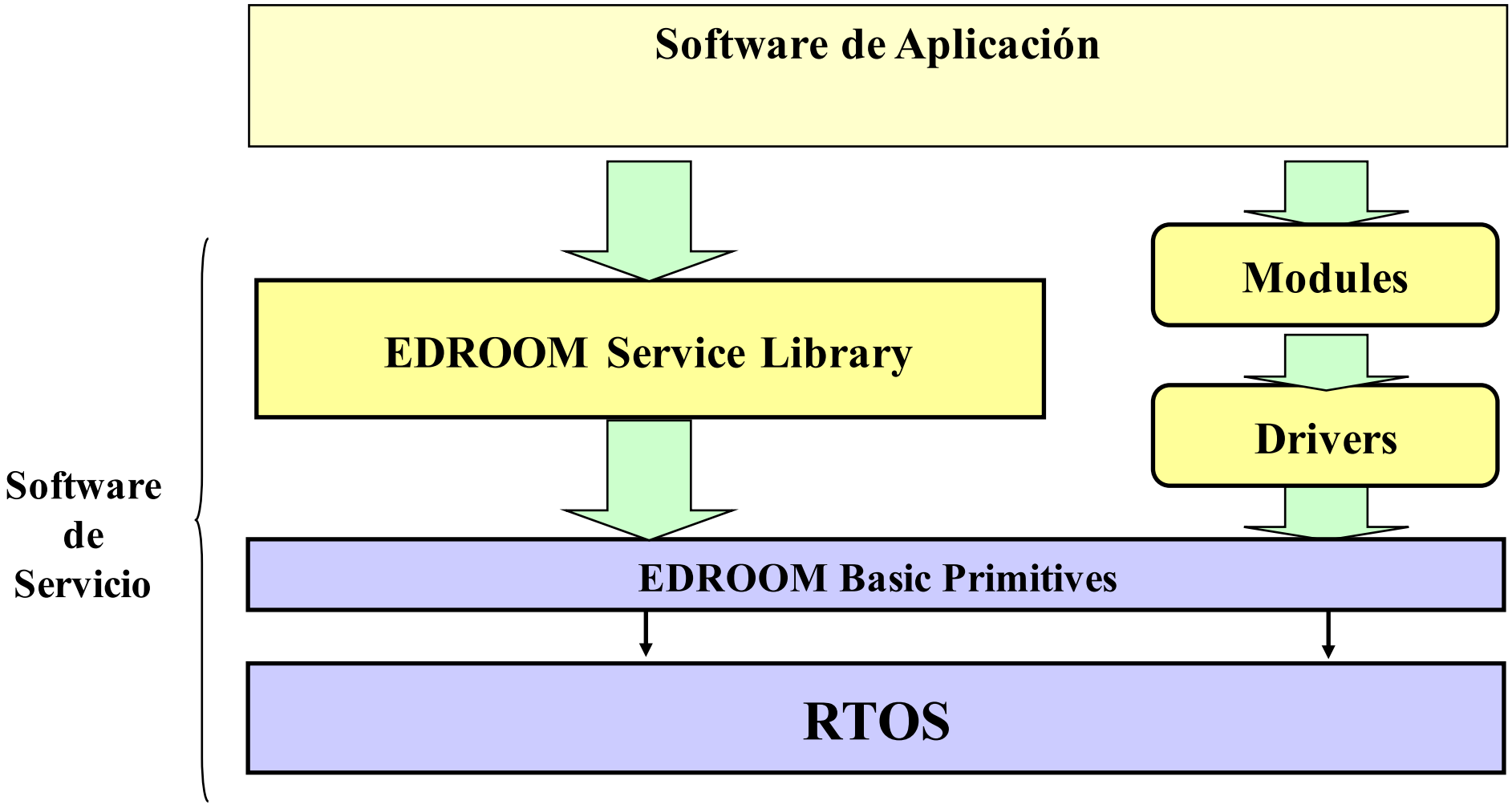
Ejemplo Estructuras OBDH

Diagrama de bloques Hw



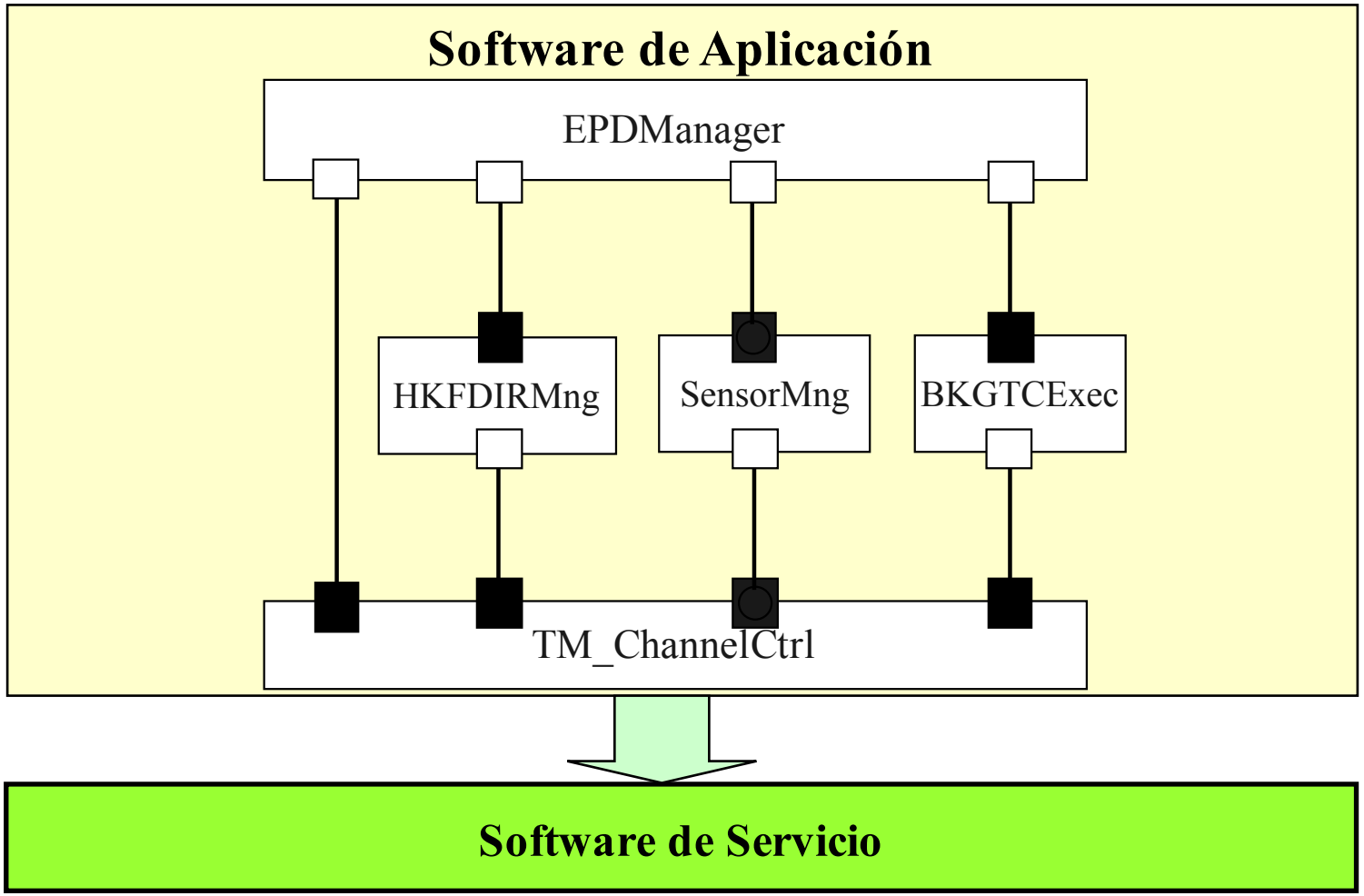
Ejemplo Estructuras OBDH

Estructura de Niveles del Software



Ejemplo Estructuras OBDH

Estructura de tareas concurrentes



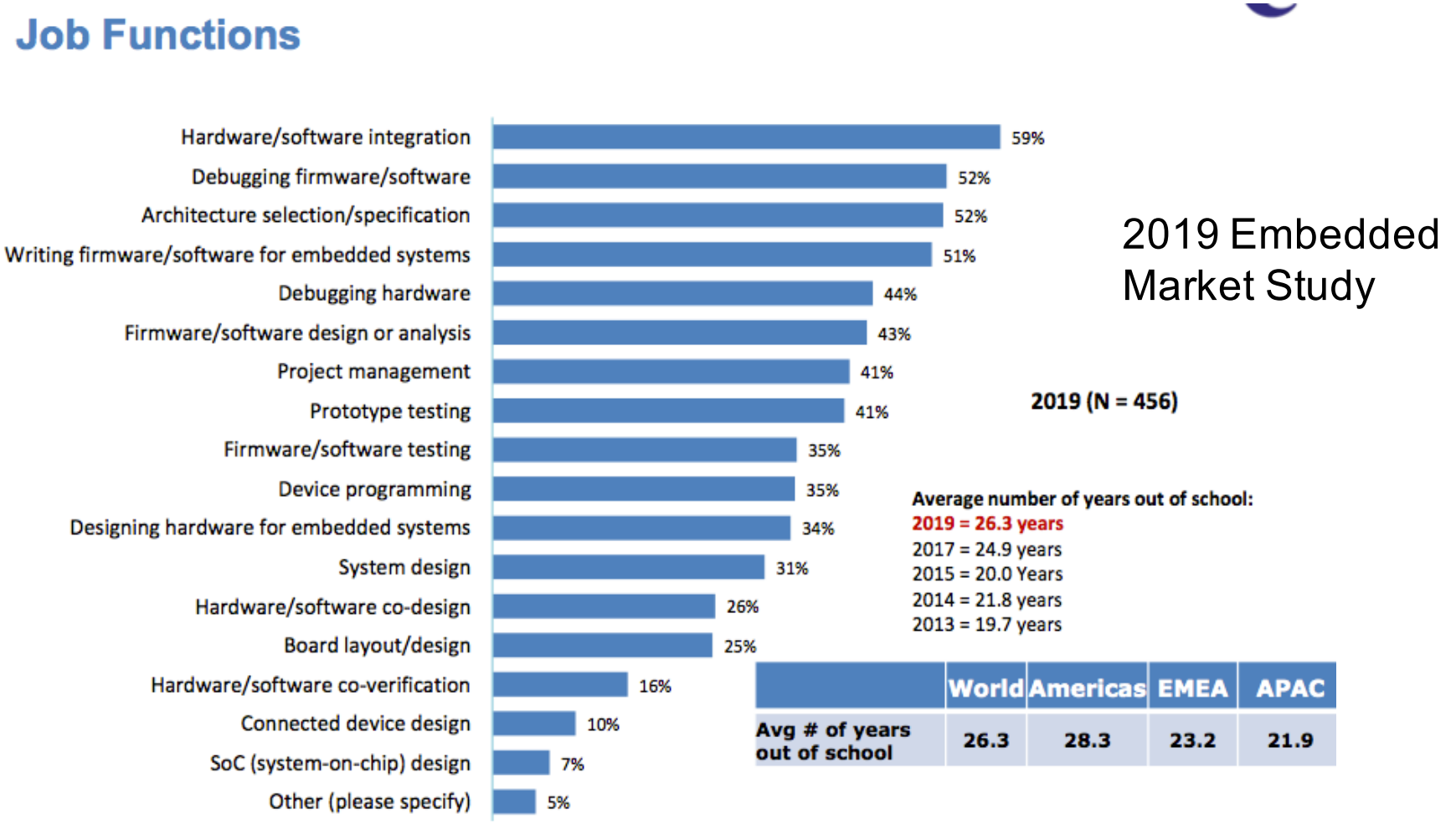
Ciclo de vida de un SE

Ciclo de Vida del SE

- El desarrollo de SE debe abordarse como una **Ingeniería:**
  - El ciclo de vida requiere del uso de Metodologías de Desarrollo: Sw, Hw y Co-diseño Hw-Sw.
  - Para llevar a cabo el desarrollo es necesario contar con **profesionales cualificados**.

Ciclo de Vida del SE

- Actividades del ingeniero durante el ciclo de vida:



Ciclo de Vida del SE

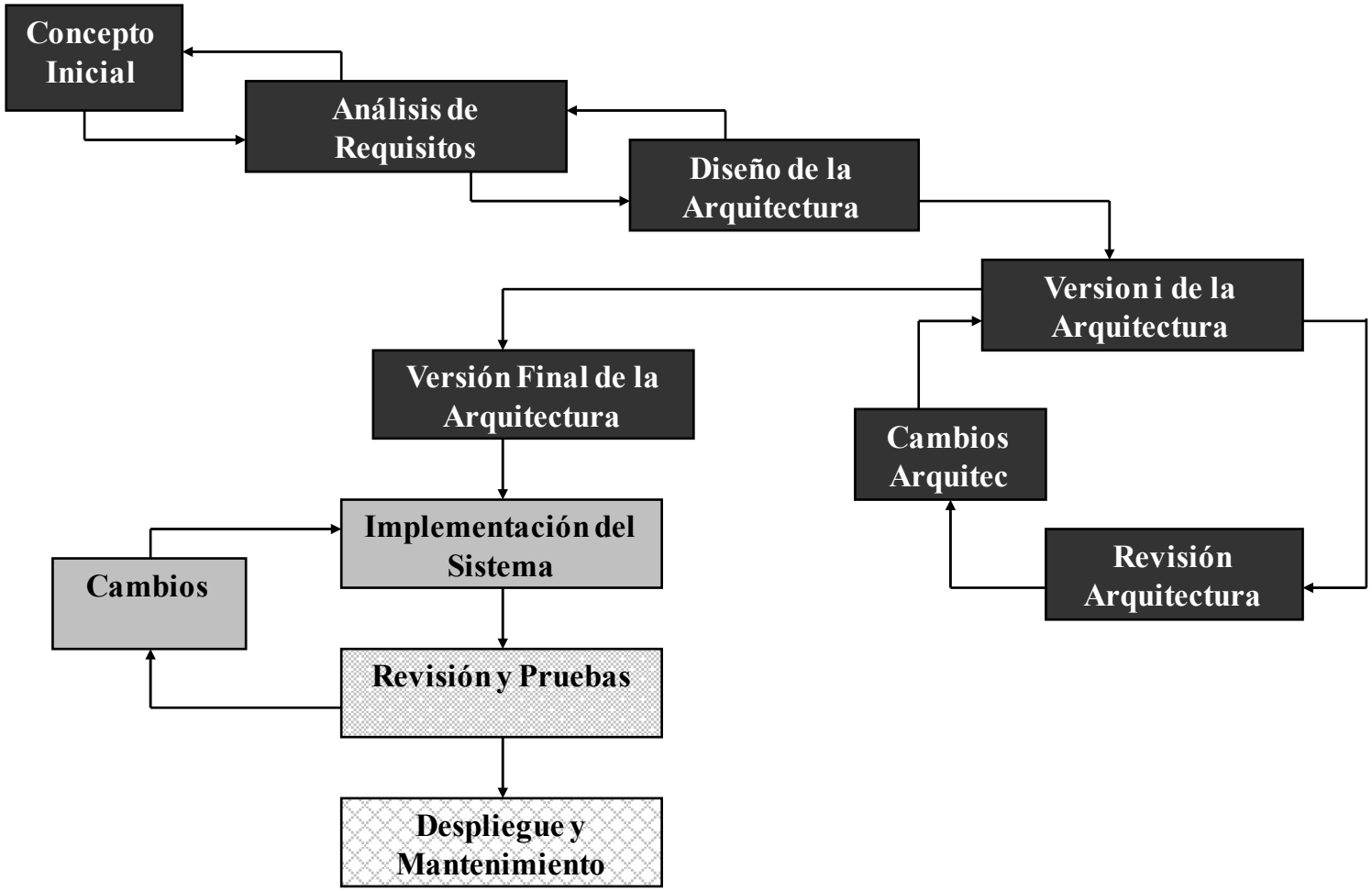
Fase 1 : Especificación de la Arquitectura

Fase 2 : Implementación de la Arquitectura

Fase 3: Pruebas

Fase 4: Mantenimiento del Sistema

Ciclo de Vida del SE



Análisis de Requisitos

- Durante el análisis de requisitos hay que tener en cuenta la incorporación de diferentes **métricas** que se marcan como requisitos.

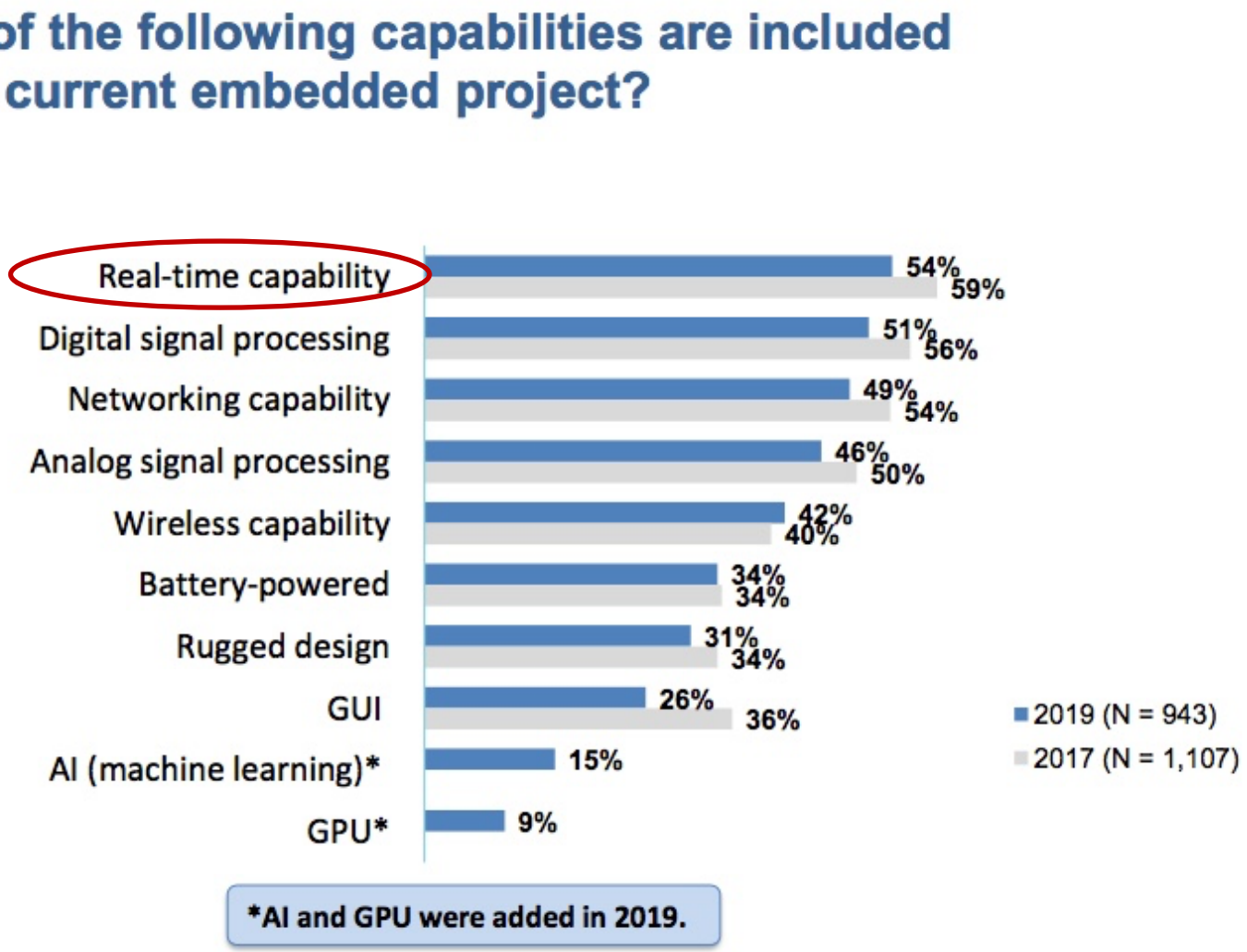
Métricas de los SE

- Una métrica es un aspecto del producto desarrollado que es capaz de medirse.
- Obtener un valor óptimo en las métricas objetivo resulta clave para que el proceso de desarrollo garantice la calidad y rentabilidad del producto.
- Métricas comunes en SE:
  - **Tamaño**: Espacio físico que ocupa el sistema.
  - **Consumo**:
    - Milivatio (pico de consumo)
    - Milivatio-hora (consumo medio)
  - **Rendimiento:**
    - Tiempo de respuesta
    - Capacidad de procesamiento del sistema

Métricas de Rendimiento

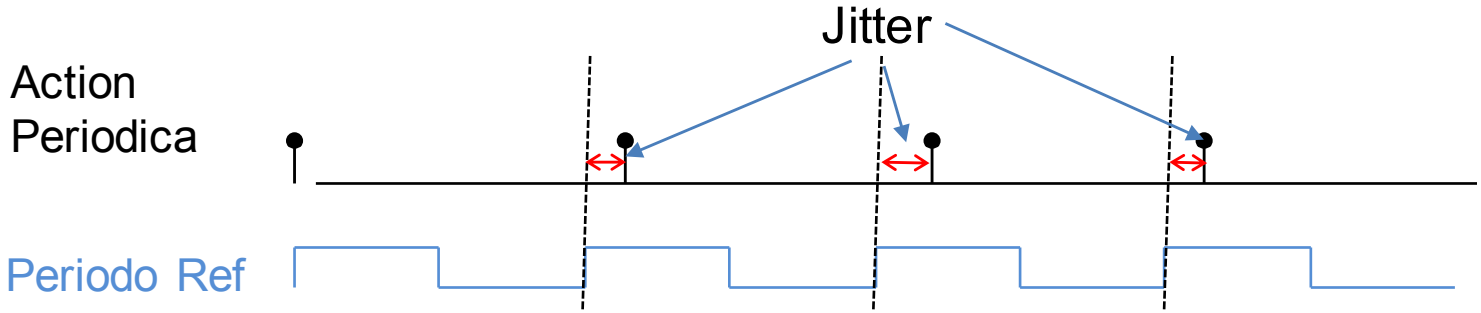
- En los sistemas empotrados, algunas de los parámetros utilizados con frecuencia en computación (MIPs o frecuencia de reloj) no reflejan bien el rendimiento real del producto.
- Parámetros significativos en los sistemas empotrados
  - **Tiempo de Respuesta:** Tiempo desde necesario para ejecutar una tarea**.**
    - Ejemplo: Tiempo de captura de una imagen de una cámara
  - **Capacidad de Procesamiento:** Numero de tareas completadas por unidad de tiempo.
    - Ejemplo:
      - Numero de imágenes por unidad de tiempo
      - 320 Deep Learning TOCs ENVIDIA DRIVE AGX PEGASUS
    - La capacidad de procesamiento puede ser mayor que (Unidad de Tiempo)/ (Tiempo de Respuesta) incorporando paralelismo en el diseño del sistema.

Métricas de rendimiento

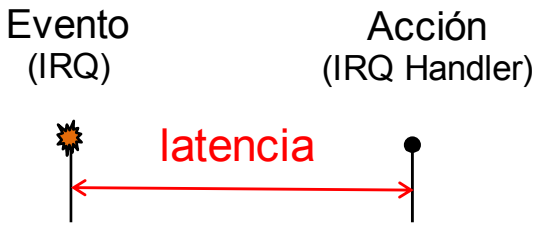


Métricas de Rendimiento

- Caracterización de Sistemas de tiempo real (SRT).
  - **Jitter:** Desviación respecto al periodo prefijado.



  - **Latencia:** Retardo en la atención de un evento.
    - En STR la latencia hace referencia generalmente a la latencia máxima



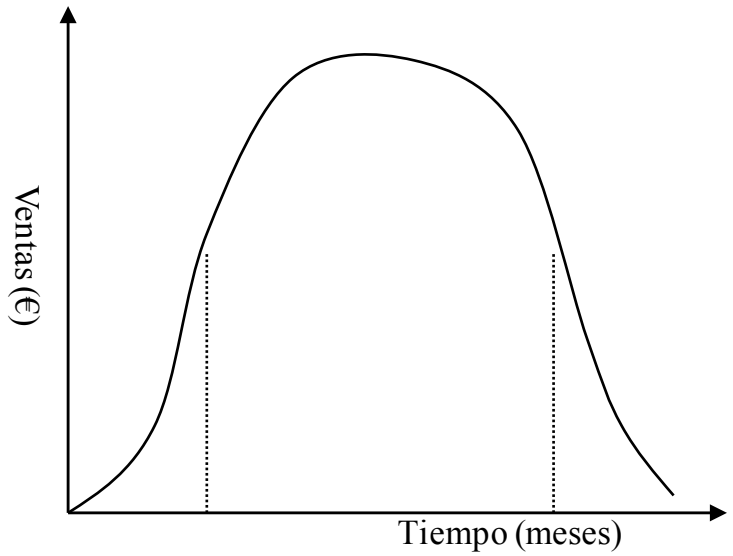
https://robohub.org/a-practical-look-at-latency-in-robotics-the-importance-of-metrics-and-operating-systems/

Métricas de los SE

- Métricas comunes en SE (2):
  - **Flexibilidad**: Capacidad de adaptarse a nuevos requisitos.
  - **Seguridad (Safety)**: Probabilidad de que el sistema no cause daño.
  - **Mantenibilidad (Maintainability)**: Facilidad para mantener el sistema.
  - **Métricas de Tiempo**: Time-to-Market, Time-to-Prototype
  - **Métricas de Coste**:
    - Coste NRE (Non-Recurring Engineering Cost):
    - Coste por Unidad.
    - Coste Total, Coste Producto.

Métricas

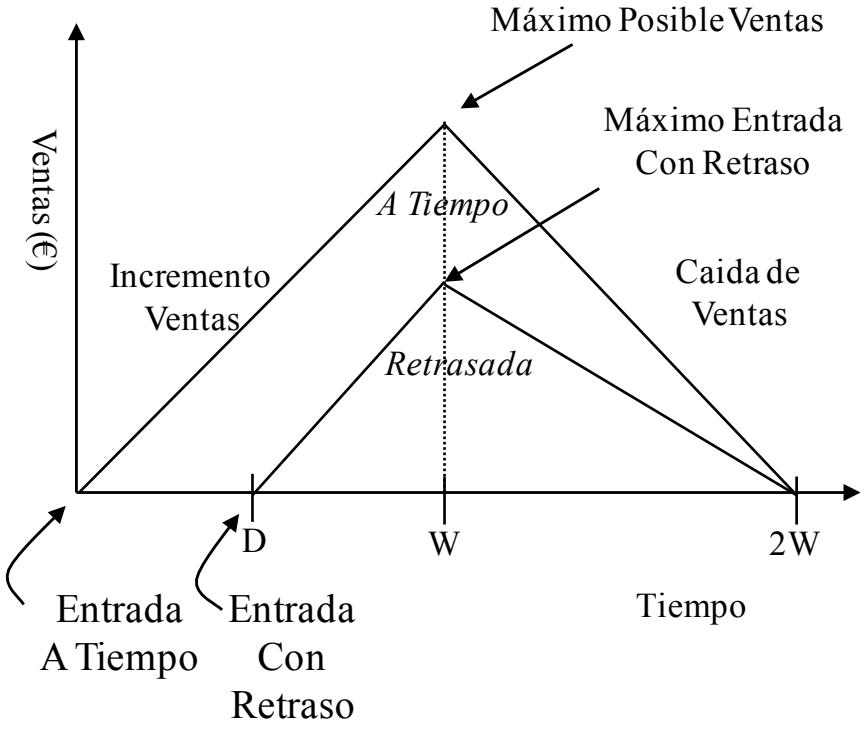
- Tiempo de puesta en el mercado (*Time-to-Market*)



- El tiempo desde el inicio del desarrollo hasta su puesta en el mercado.
- Ventanta de Mercado:
  - Tiempo durante el que el producto alcanzará sus mayores ventas.
  - La media de la ventana de mercado es de 8 meses
- Un **retraso** en la puesta en el mercado del producto puede implicar grandes **pérdidas**

Métricas

Tiempo de puesta en el mercado (*Time-to-Market*) (2)



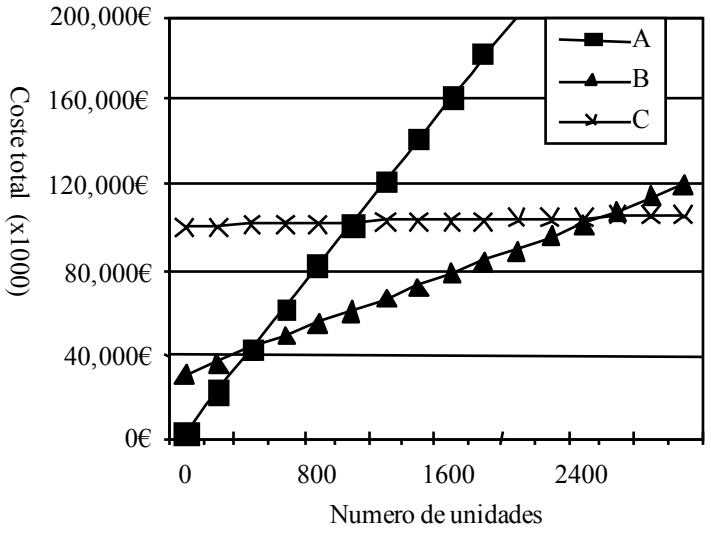
- Modelo de ventas simplificado:
  - 2W = Ventana de Mercado
  - La penetración en el mercado de un producto = **area del triángulo**
  - Las **pérdidas** debidas al retraso = **diferencia** entre las **areas** de los triángulos.
- Ejemplos para 2W=1 año:
  - Retraso 4 semanas => 22% pérdidas
  - Retraso **10 semanas** => **50% pérdidas**

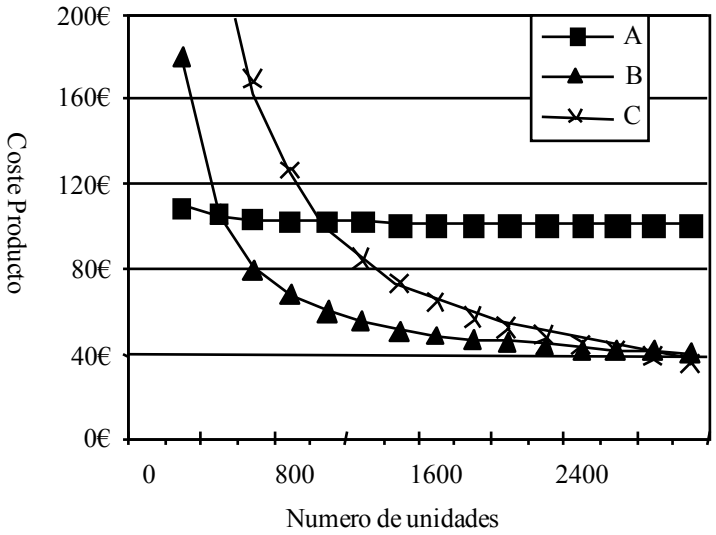
Métricas de Coste

- **Coste NRE** (Non-Recurring Engineering Cost): Es el coste de diseño y verificación del sistema. Sólo se computa una vez y es independiente del número de unidades.
- **Coste por Unidad**: Es el coste de fabricación de una unidad, no incluye el coste NRE.
- **Coste Total**: Coste NRE + (Coste por Unidad) x nº unidades
- **Coste Producto**: Coste Total/(nºunidades)
  - Ejemplo de Coste Producto
    - NRE=2000€, Coste por Unidad=100€
    - Si se producen 10 unidades
      - Coste Producto= 2000€/10 + 100€ = 300€
    - Para 100 unidades
      - Coste Producto= 2000€/100 + 100€ = 120€

Métricas de Coste

- La inversión en diseño puede reducir el Coste Producto
  - –Diseño A: NRE=2,000€, Coste por unidad=100€
  - –Diseño B: NRE=30,000€, Coste por unidad=30€
  - –Diseño C: NRE=100,000€, Coste por unidad=2€

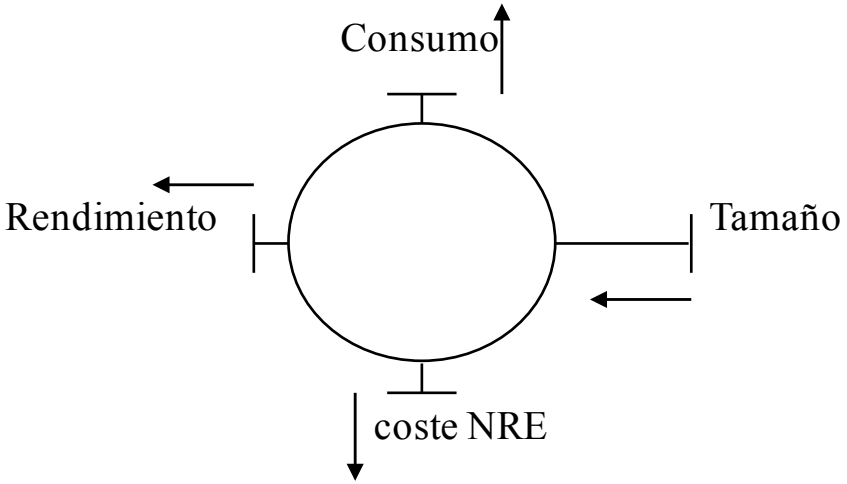




- ¡Pero siempre hay que tener en cuenta el **Tiempo de puesta en el mercado!**

Métricas

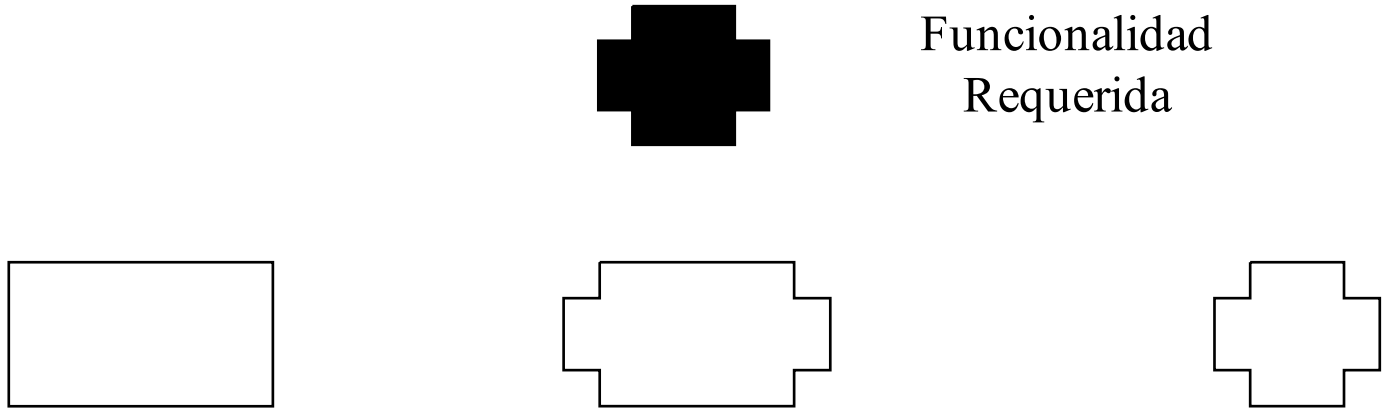
- Compromiso (Trade-off)



- Mejorar los resultados en una de las métricas puede afectar negativamente al resto de métricas =>
- Es necesario establecer un **compromiso** al marcar los objetivos de las métricas

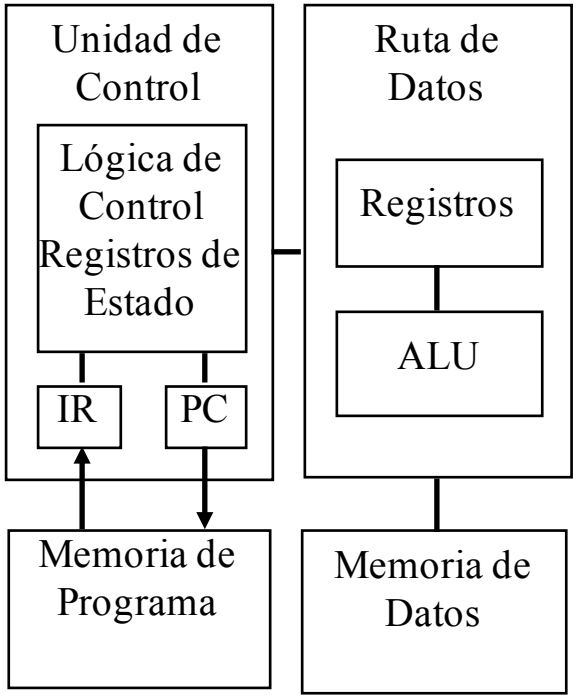
Alternativas de diseño de CPU para SEs

- Planteamiento del problema
  - Existen distintas alternativas de diseño de CPU para desarrollar un sistema empotrado:
    - Desarrollar un procesador "ad hoc" o de propósito único .
    - Utilizar procesadores de propósito general
    - Utilizar procesadores de aplicación específica: DSPs y microcontroladores



Alternativas de diseño de CPU para SEs

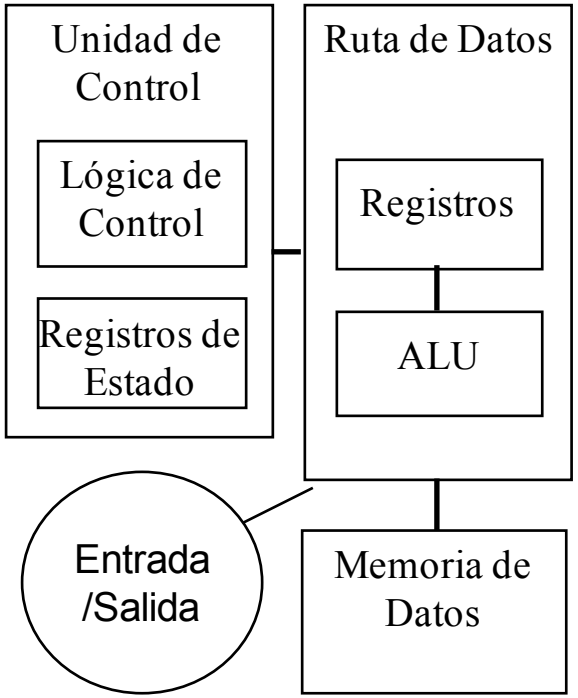
- Procesador de propósito general
  - Características
    - Programable vía Software: C, Java
    - Ruta de datos general
    - Registro de datos generales
    - Tipos de datos generales
    - ALU general
  - Ventajas
    - Minimización del Coste NRE
    - Minimización del Tiempo de Puesta en el Mercado.
    - Gran flexibilidad: software
  - Desventajas
    - Coste por Unidad: ¿mínimo?
    - Rendimiento: ¿óptimo?



Alternativas de diseño de CPU para SEs

Procesador "ad hoc" o de propósito único

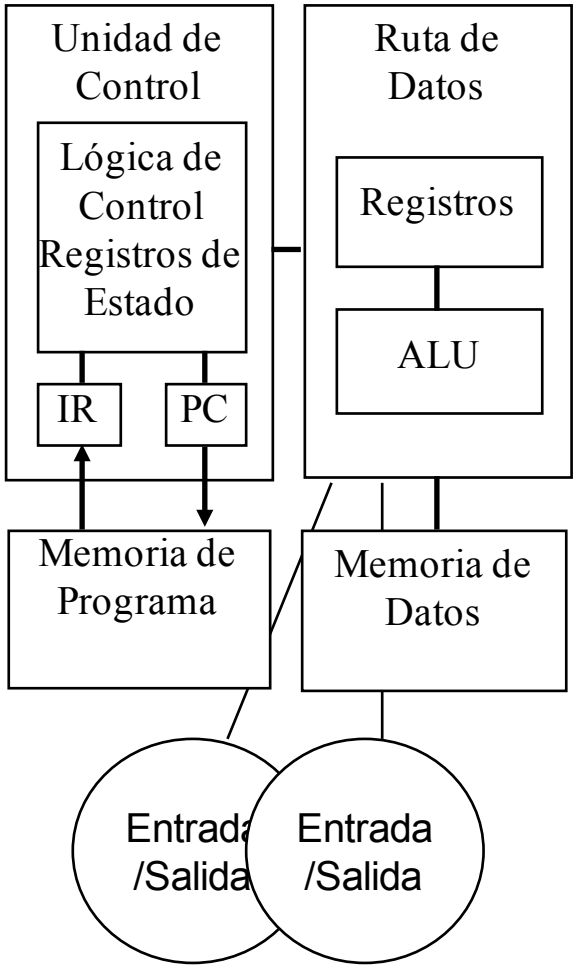
- Características
  - Diseñado para resolver un único problema.
  - No emplea memoria de programa
  - Tipos de datos y ALU "ad hoc"
  - Integra Entrada/Salida "ad hoc"
- Ventajas
  - Minimiza el Coste por unidad
  - Minimiza Tamaño y Consumo
  - Optimización del Rendimiento
- Desventajas
  - Alto Coste NRE
  - Baja Flexibilidad.
  - Tiempo de Puesta en el mercado elevado



Alternativas de diseño de CPU para SEs

Procesador de propósito específico

- Características
  - Diseñado para resolver un conjunto de problemas.
  - Programable vía Software
  - Tipos de datos y ALU específicos
  - Integra Entrada y Salida
- Ventajas y Desventajas
  - Tiene las ventajas y desventajas de los procesadores de propósito general **mejorando**:
    - Rendimiento para un conjunto de problemas
    - Tamaño, Coste, Consumo (integra E/S)
- Ejemplos: DSP, microcontroladores



Alternativas de Implementación de CPU

- Tecnologías de Circuitos Integrados (CIs)
  - La implementación del procesador en un CI está soportada por distintas tecnologías:
    - Full Custom/ VLSI
    - Semi Custom ASIC (Gate Array o Standard Cell)
    - FPGAs
  - Todas las tecnologías está basadas en el diseño por capas de un CI:
    - A partir de un sustrato semiconductor se diseñan las capas destinadas a implementar los transistores, las puertas y sus conexiones que constituirán finalmente el procesador.
    - Mediante técnicas de **enmascaramiento** se consigue el tratamiento selectivo que diferencia las zonas de cada capa.

Alternativas de Implementación de CPU

- Cada tecnologías se basa en la personalización de un número de capas.
- Las capas del CI definen:
  - La ubicación de los transistores
  - Composición de los transistores para construir puertas
  - Definición de las conexiones entre puertas

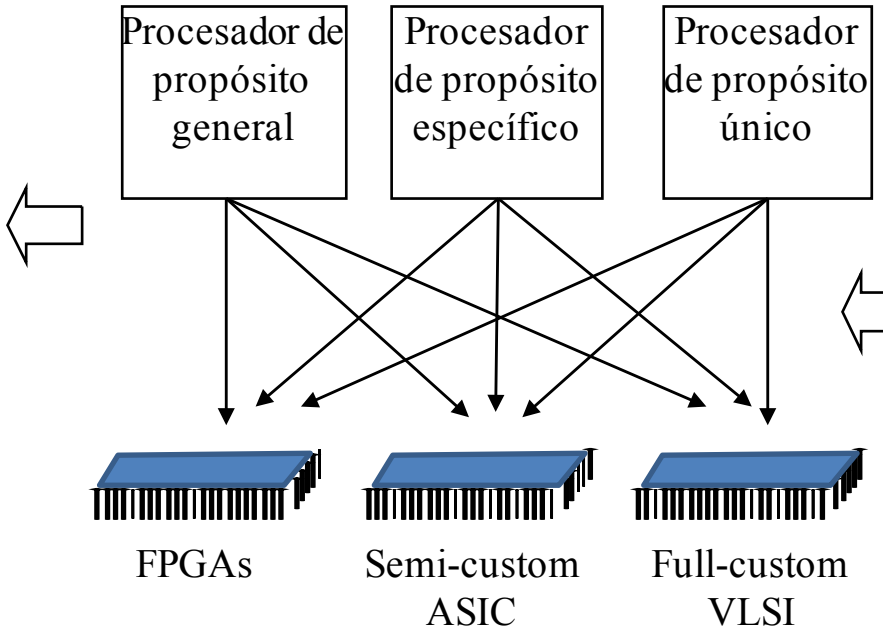
- Full Custom/ VLSI personaliza todas las capas
- Semi Custom ASIC. Dos alternativas:
  - **Gate Array:** Las capa de puertas está ya fijada formando un array. Personaliza conexiones.
  - **Standar Cell**: Las máscaras que definen una puerta, o una red de puertas (and-or-invert) están ya diseñadas, pero deben combinarse para definir la máscaras completa de esa capa
- FPGA no hay personalización en la fabricación. La capa de conexiones es configurable/reprogramable por el cliente después de la fabricación

Diseño e implementación de la CPU

- Tecnologías de CI y el diseño de la CPU
  - Son decisiones independientes:
    - Cualquier combinación de ambas es posible
  - Análisis de ventajas e inconvenientes:
    - Soluciones basadas en diseño personalizado
    - Utilización de soluciones de propósito general.

Propósito General
Ventajas:

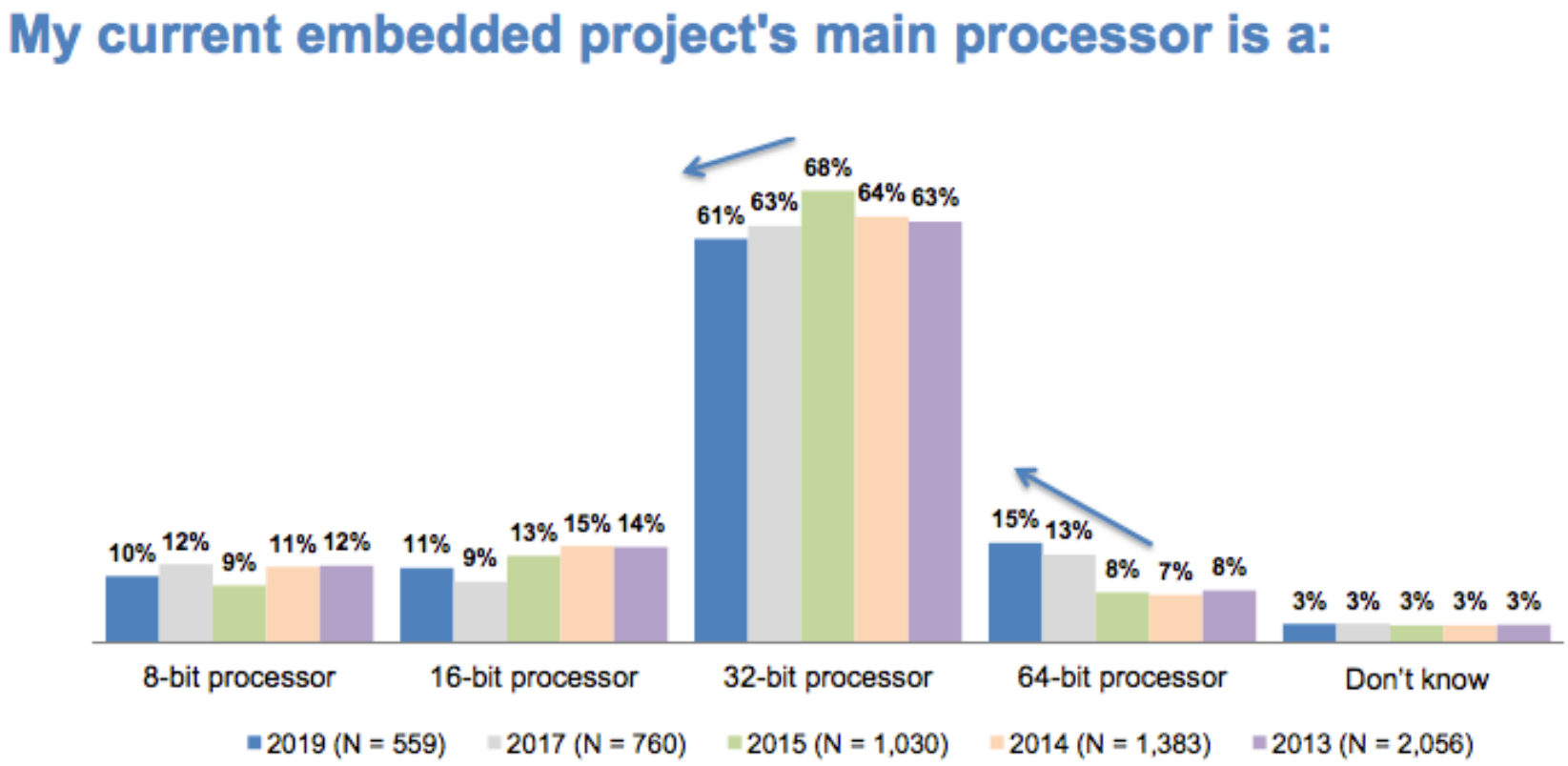
Flexibildad
Mantenimiento
Bajo Coste NRE
Minimiza Tiempo de puesta en mercado
Minimiza Tiempo de prototipado
Bajo Coste para volumen de ventas bajo



Diseño Personalizado
Ventajas:

Eficiencia de Potencia
Rendimiento
Tamaño
Bajo Coste volume de ventas altos

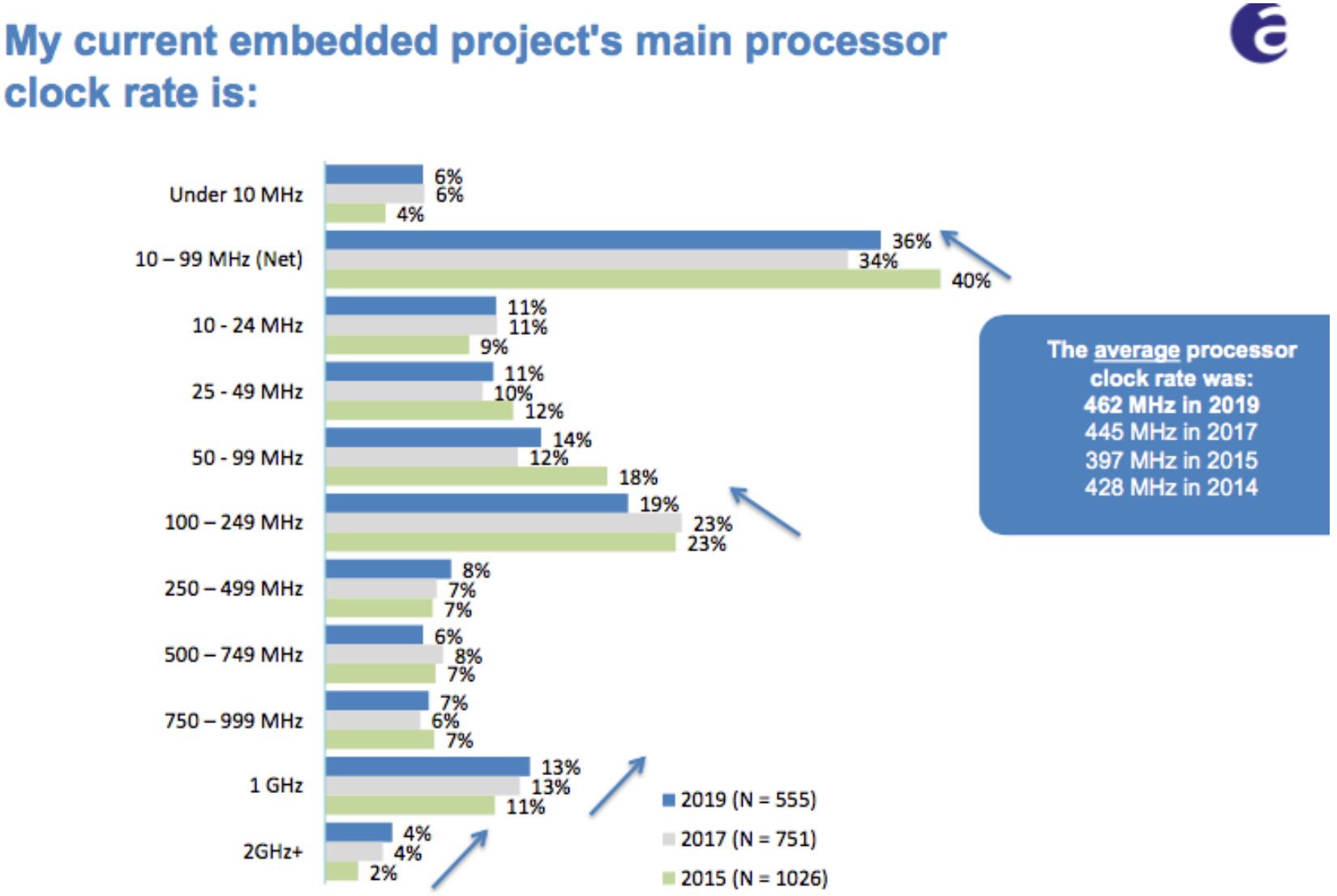
Diseño e implementación de la CPU



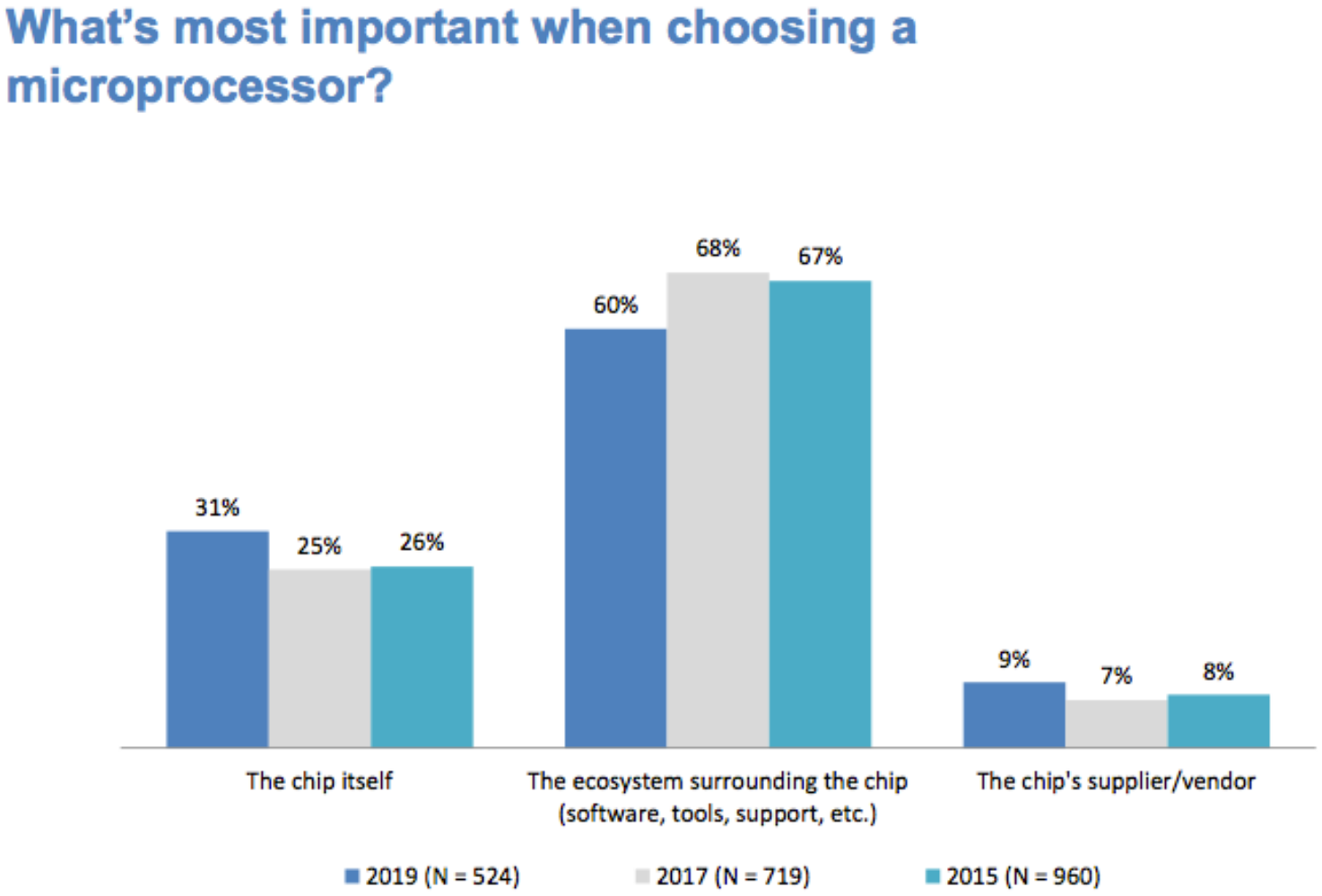
71% of **EMEA** users use 32-bit chips as their main processor.

Additional chips to the main processor	
Primarily 8-bit processors	19%
Primarily 16-bit processors	15%
Primarily 32-bit processors	55%
Primarily 64-bit processors	12%

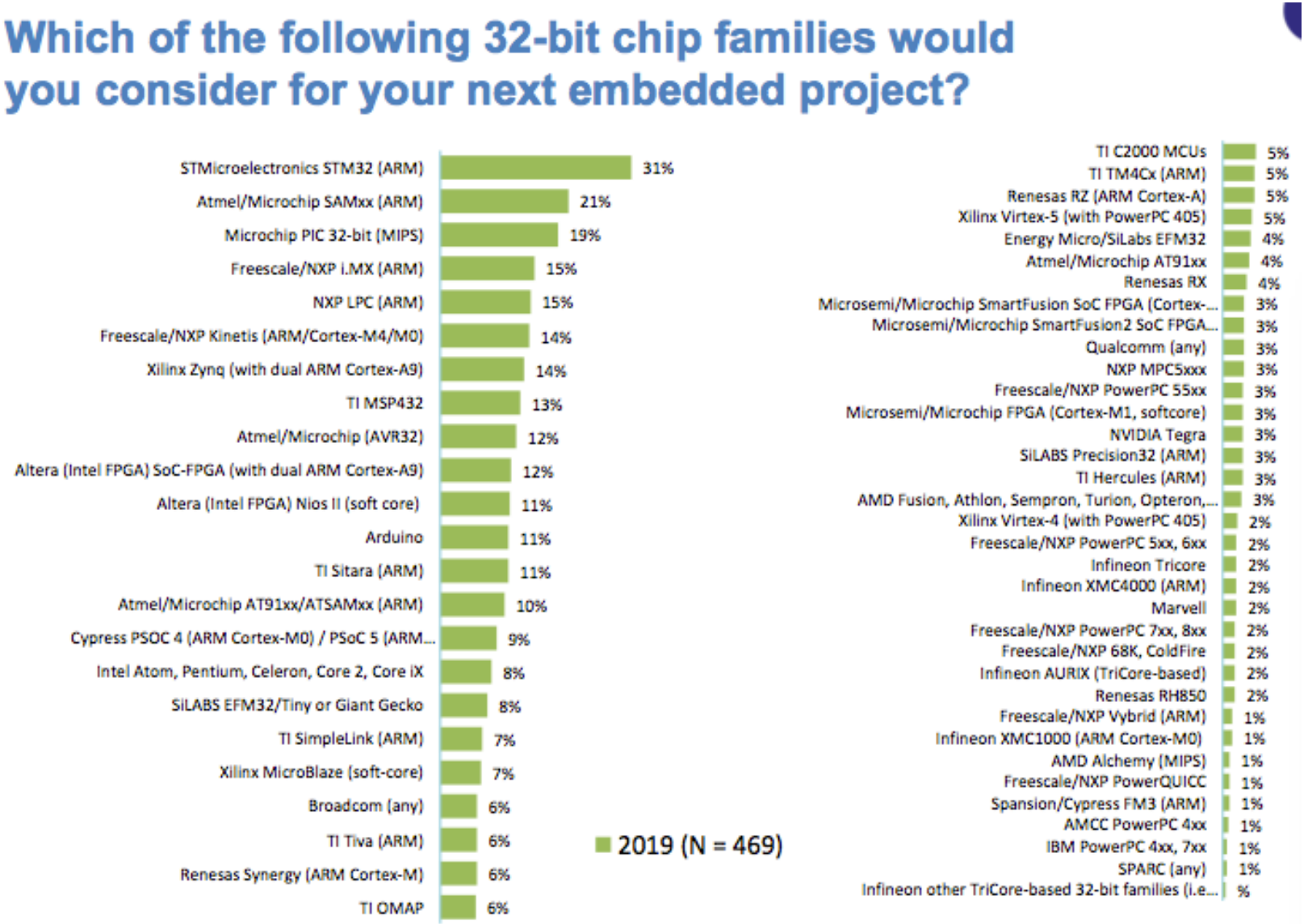
Diseño e implementación de la CPU



Diseño e implementación de la CPU

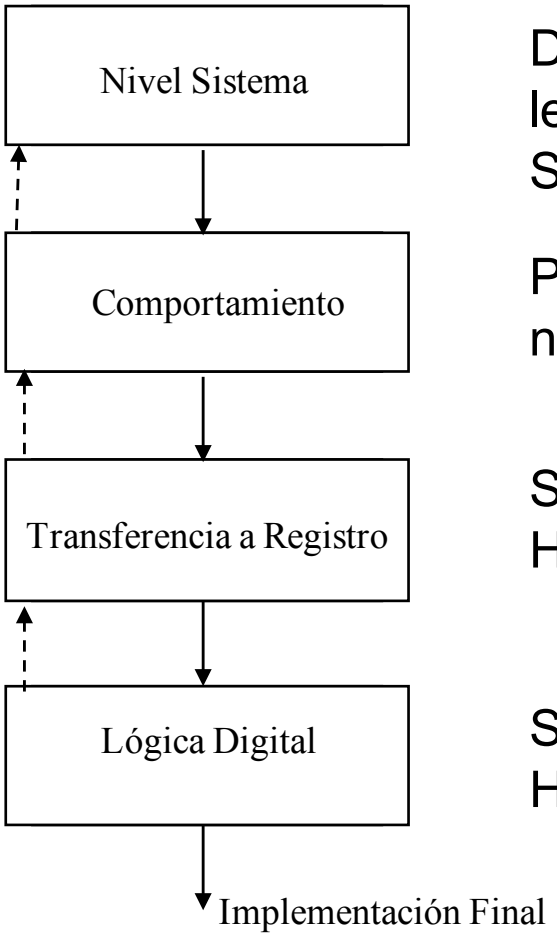


Diseño e implementación de la CPU



Alternativas de Diseño de SE

Niveles de Diseño de un SE

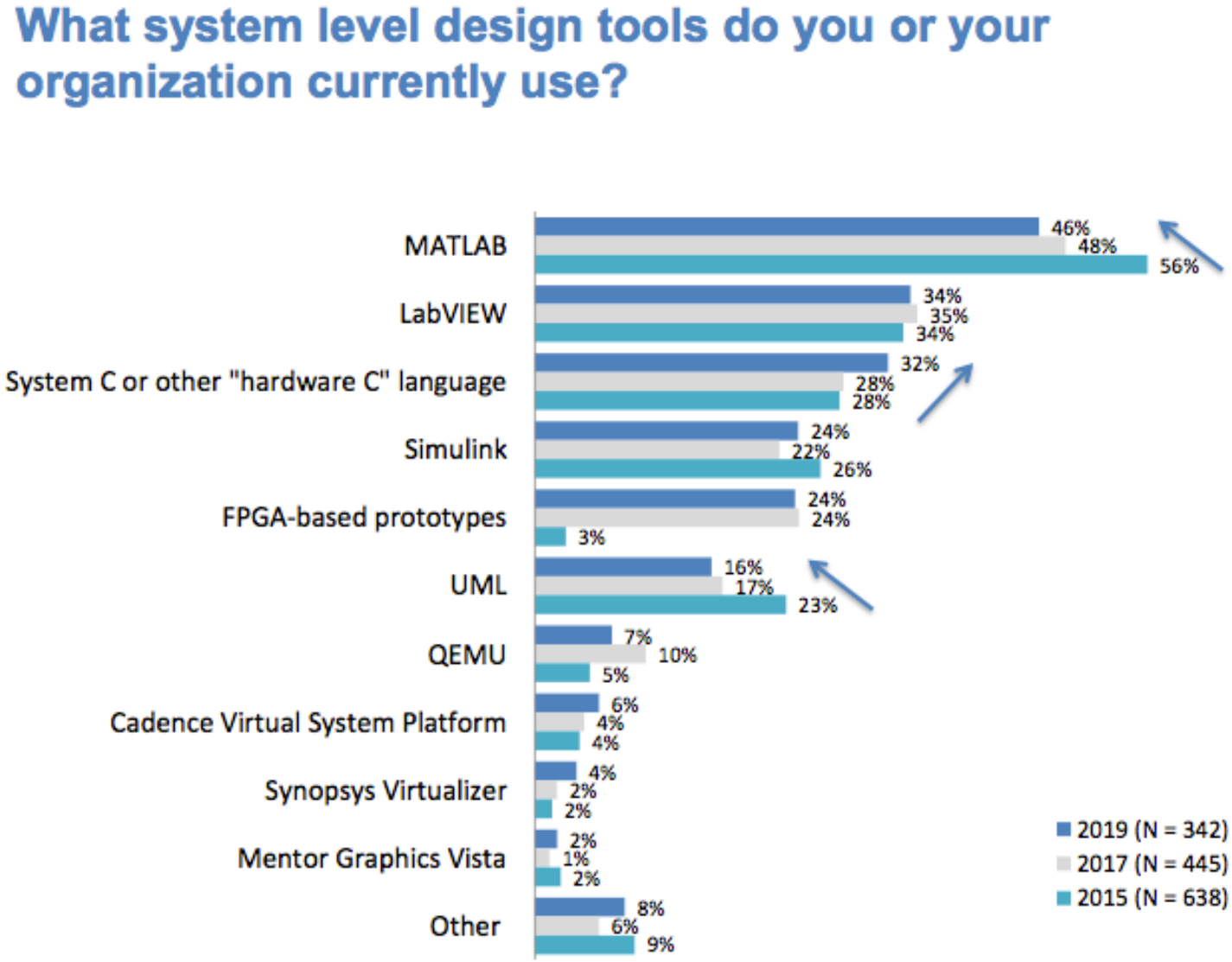


Descripción de la funcionalidad del sistema en lenguaje/s de alto nivel: MATLAB, UML, C, VHDL, SystemC

Particionado y definición del comportamiento a nivel de detalle.

SW: Obtención de código ensamblador
HW: Especificación basada en módulos combinaciones/secuenciales (Mx, Reg, Sum)
SW: -
HW: Ecuaciones booleanas, puertas lógicas

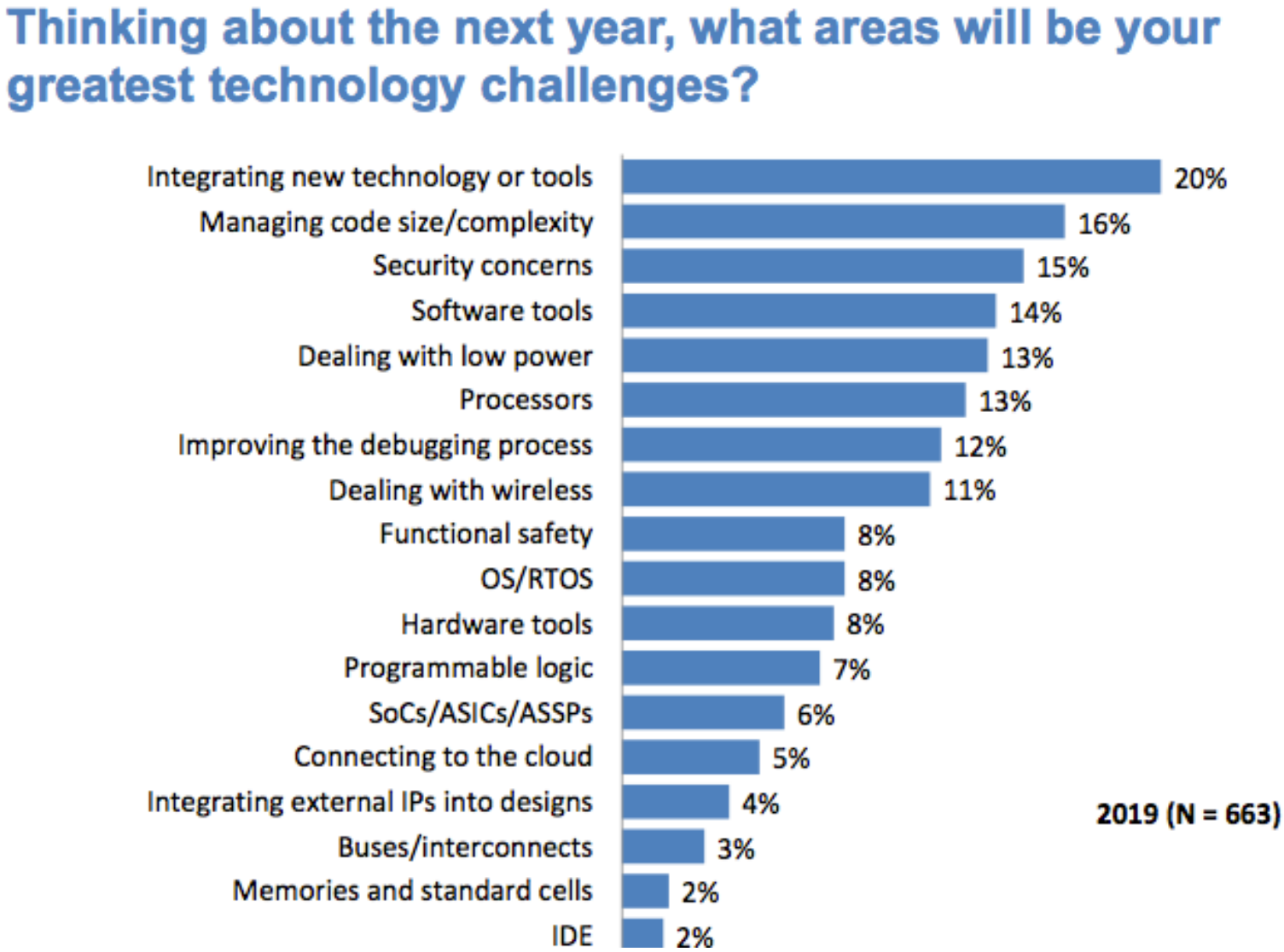
Niveles de Diseño de un SE



Técnicas de Diseño y Validación de SE

- ¿Cómo facilitar que los conceptos de diseño sean implementados?
- Técnicas de Diseño y Validación de SE
  - Compilación/Síntesis: Generación automática de la especificación de un nivel a partir de la especificación de un nivel superior.
  - Librerías/IP cores: Incorporación al diseño de alto nivel de módulos pre-diseñados y completamente implementables a nivel de detalle.
  - Test/Validación: Asegurar el correcto funcionamiento de cada nivel, reduciendo el número de iteraciones entre niveles.

Técnicas de Diseño y Validación de SE

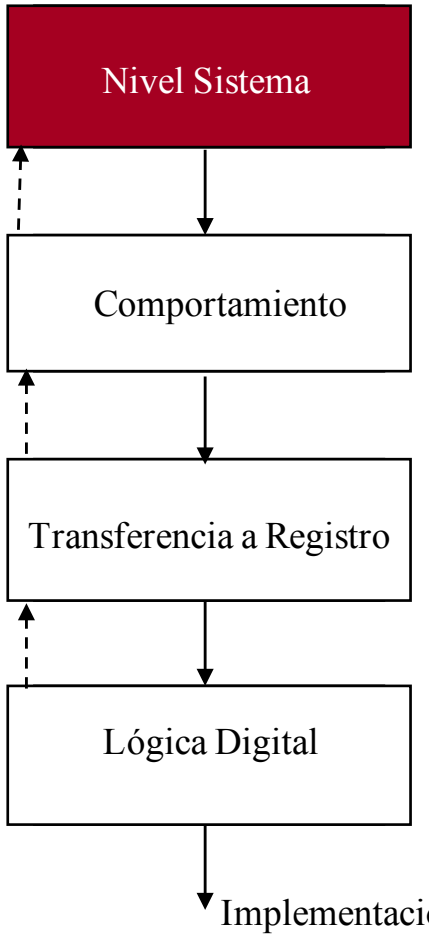


Técnicas de Diseño y Validación de SE

- **Compilación/Síntesis:**
- Librerías/IP cores:
- Test/Validación:

Técnicas de Diseño y Validación de SE

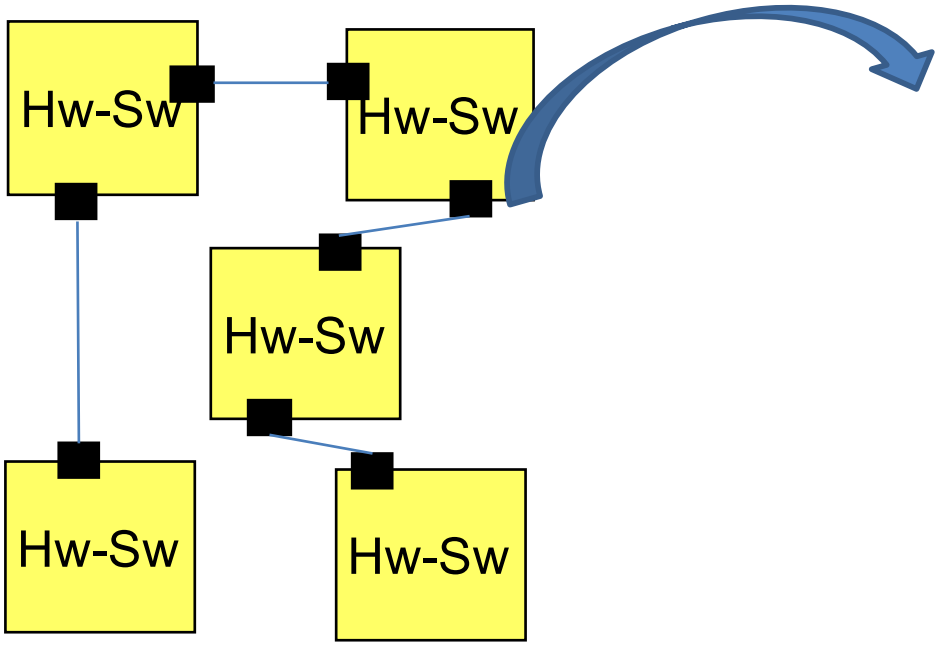
- Compilación/**Síntesis HW**:



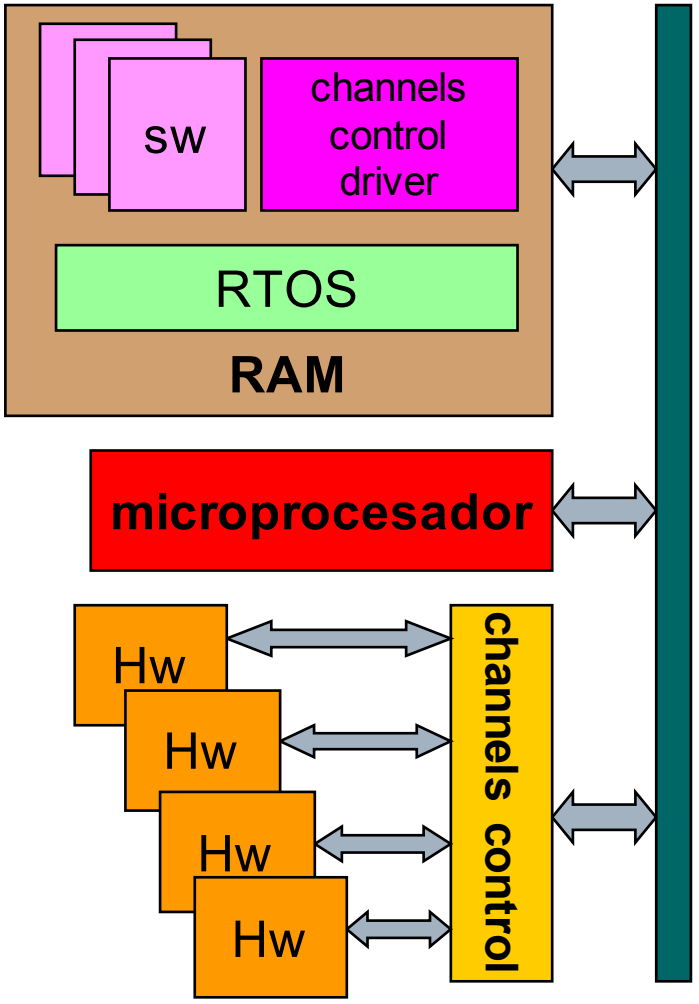
Especificación Sistema ->	**Partición en Componentes HW y SW**
Especificación Secuencial ->	Máquina de Estados Finita
Máquina de Estados Finita->	Registros Ruta de Datos (Data Path) Controlador (Exp Bool)
Expresiones Booleanas->	Puertas Lógicas (NetList)

Técnicas de Diseño y Validación de SE

- Síntesis Sistema:
  - Especificación Sistema -> Particionado Automático

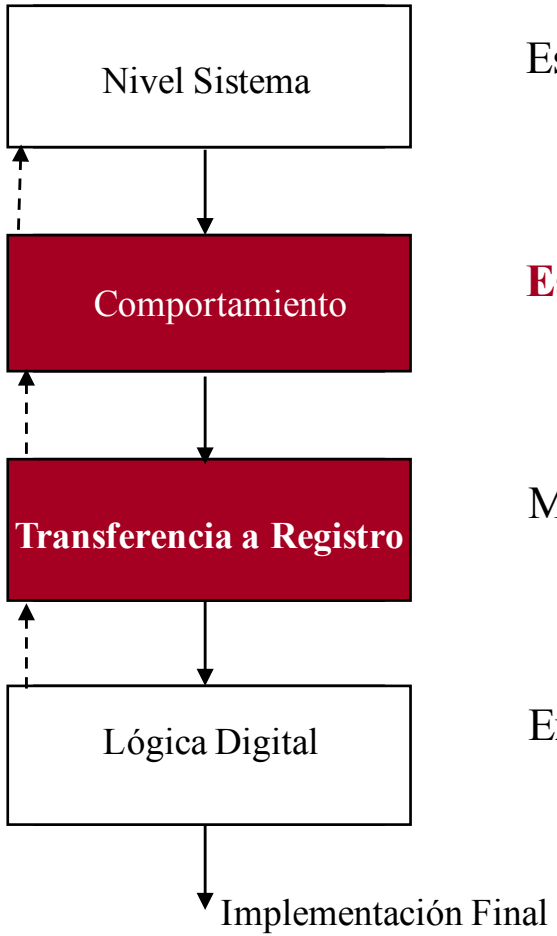


Componentes Sistema



Técnicas de Diseño y Validación de SE

- Compilación/**Síntesis HW**:



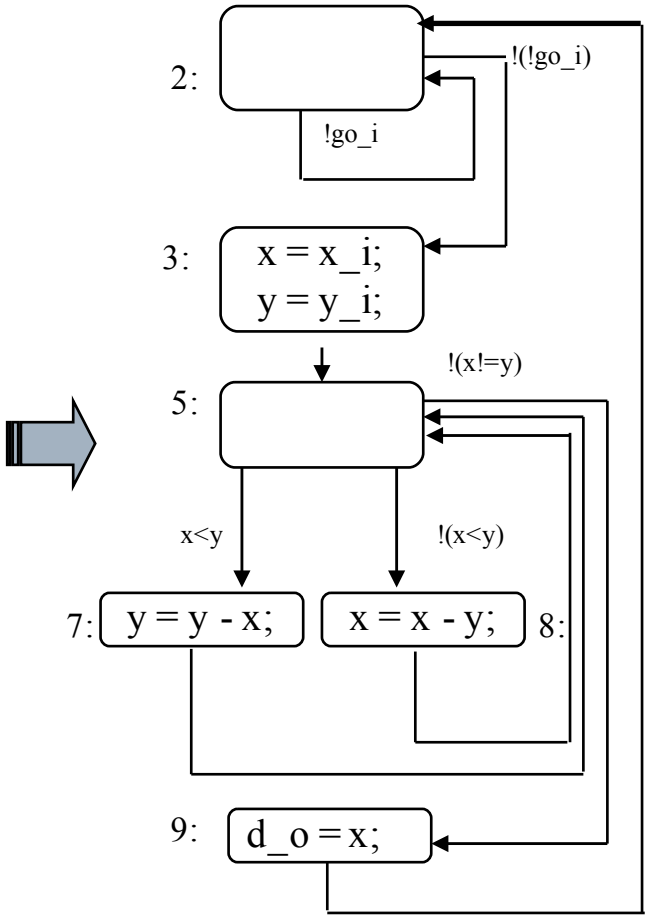
Especificación Sistema ->	Partición en Componentes HW y SW
Especificación Secuencial ->	Máquina de Estados Finita
Máquina de Estados Finita->	**Registros** **Ruta de Datos (Data Path)** **Controlador (Exp Bool)**
Expresiones Booleanas->	Puertas Lógicas (NetList)

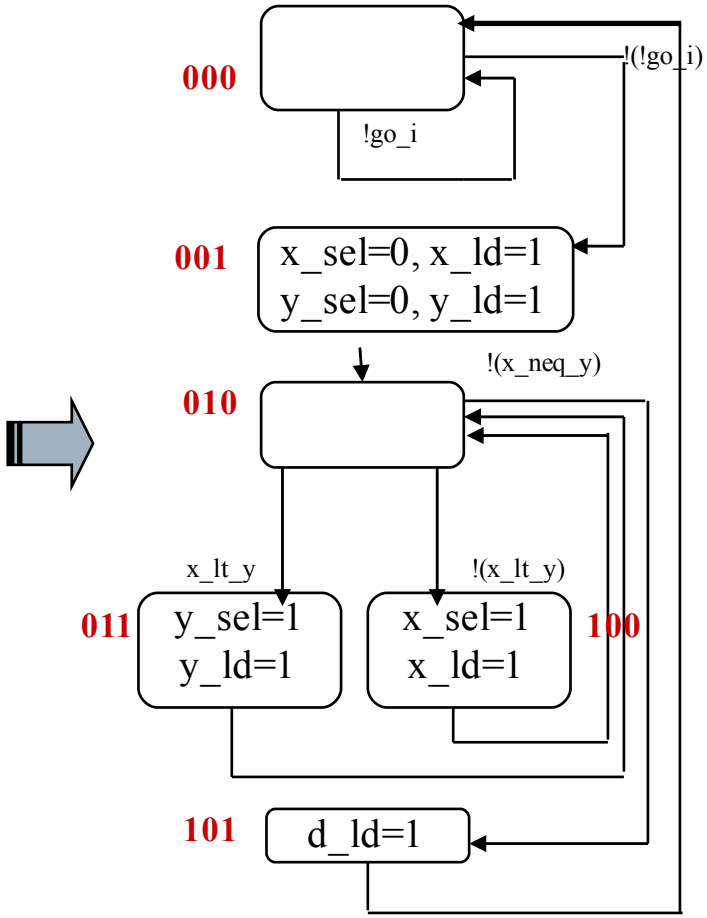
Técnicas de Diseño y Validación de SE

Síntesis Comportamiento +Transferencia Registro

ALGORITMO

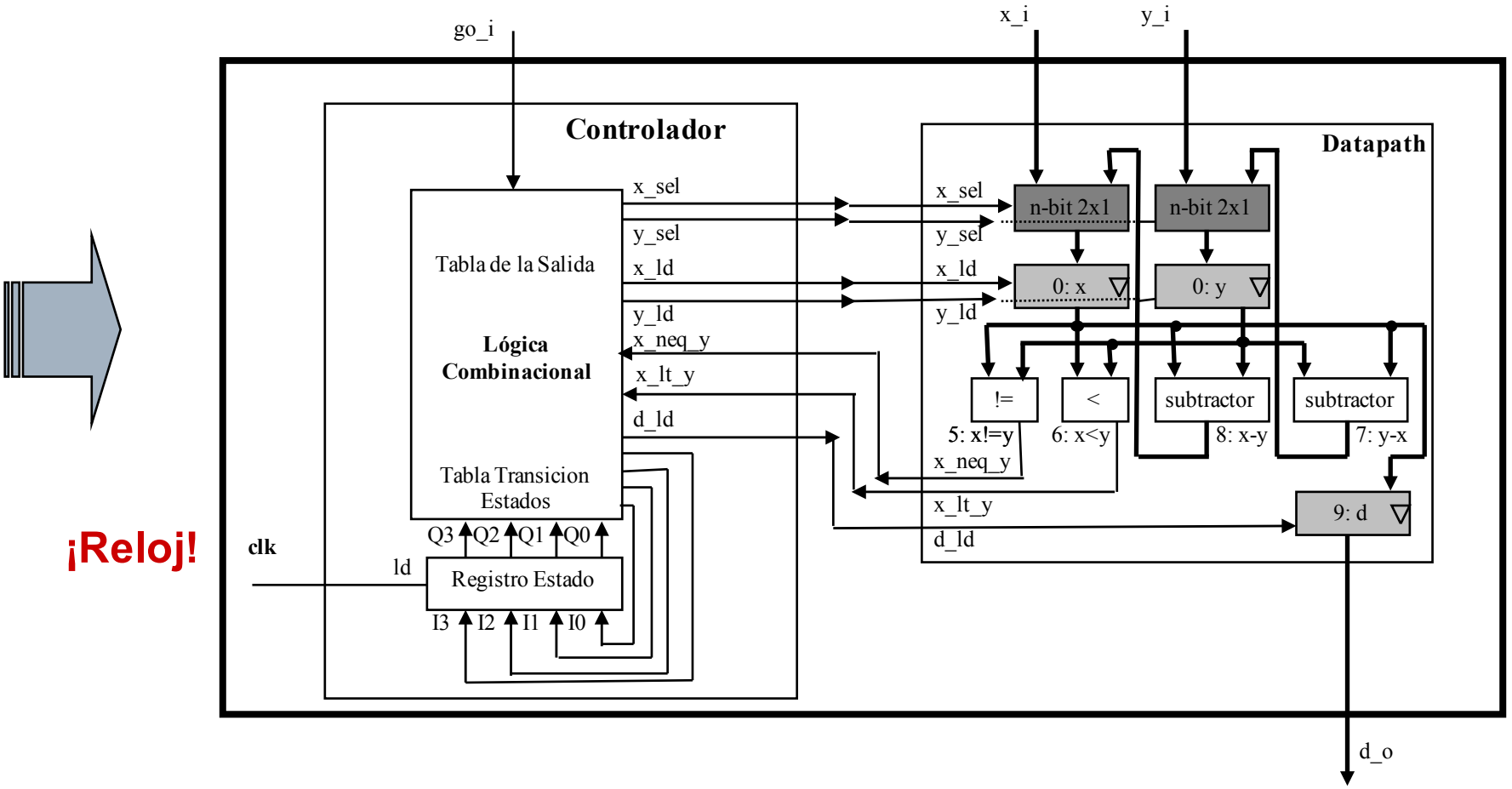
```
0: int  x, y;
1: while (1) {
2:   while (!go_i);
3:   x = x_i;
4:   y = y_i;
5:   while  (x != y)  {
6:      if  (x < y)
7:         y = y - x;
       else
8:         x = x - y;
     }
9:   d_o = x;
   }
```





Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento +Transferencia Registro



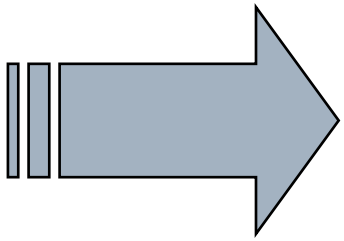
Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento +Transferencia Registro

ALGORITMO SystemC

```
void circuit::do_gcd()
{
   sc_uint<32> xreg,yreg;
   while (1) {
    do {
                    wait();
    } while (!start.read());
    xreg = in1; yreg = in2;
    wait ();
    while (xreg != yreg) {
                    if (xreg > yreg) {
                      xreg -= yreg;
                    } else {
                      yreg -= xreg;
                    }
                    wait ();
    }
    result = xreg; finish = 1; wait ();
    finish = 0; wait ();
   }
}
```

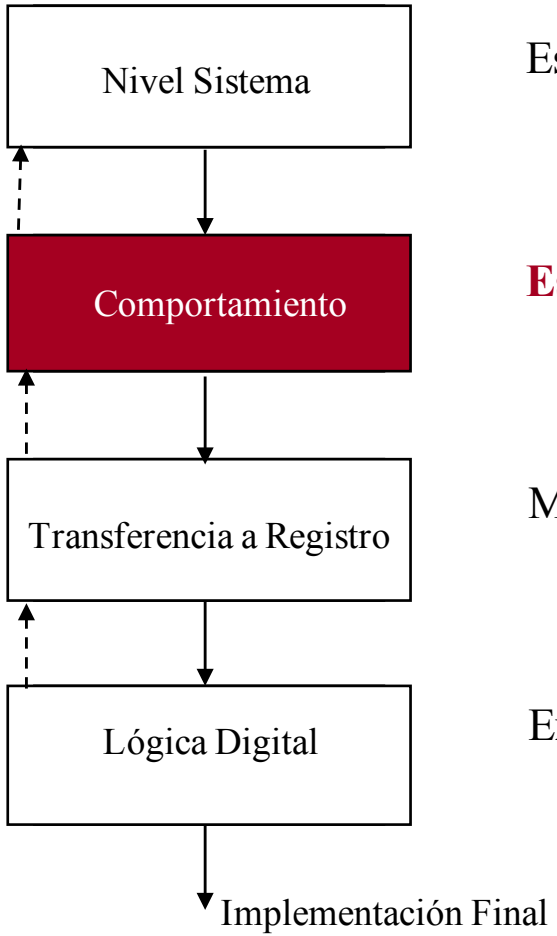
¡¡ SYSTEM CRAFTER !!!



VHDL GATE LEVEL

Técnicas de Diseño y Validación de SE

- Compilación/**Síntesis HW**:

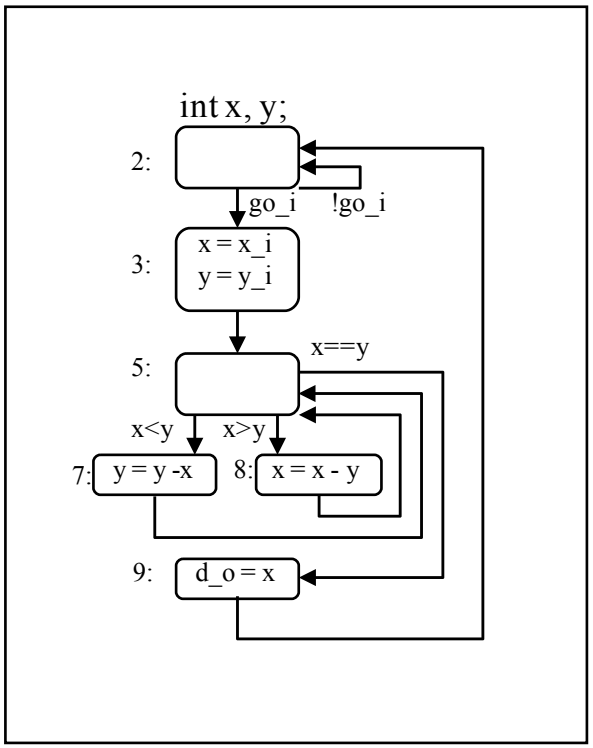


Especificación Sistema ->	Partición en Componentes HW y SW
Especificación Secuencial ->	**Máquina de Estados Finita**
Máquina de Estados Finita->	Registros Ruta de Datos (Data Path) Controlador (Exp Bool)
Expresiones Booleanas->	Puertas Lógicas (NetList)

Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento:
  - Especificación Secuencial-> Máquina de Estados con Datos (FSMD)

```
0: int  x, y;
1: while (1) {
2:   while (!go_i);
3:   x = x_i;
4:   y = y_i;
5:   while  (x != y)  {
6:      if  (x < y)
7:         y = y - x;
       else
8:         x = x - y;
     }
9:   d_o = x;
   }
```

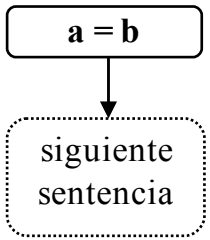


Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento:
  - 1ª Etapa: Aplicación de patrones de Síntesis.

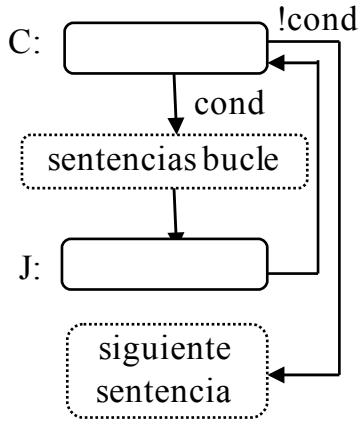
Asignación

a = b;



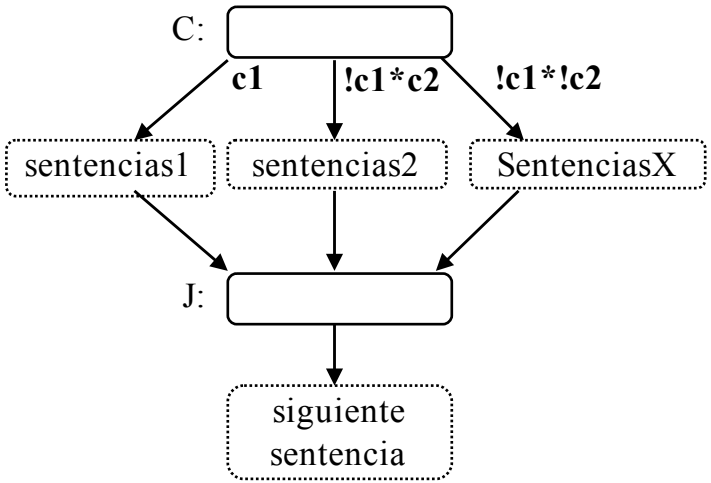
Bucle while

while (cond) {
sentencias bucle
}



if else if else...

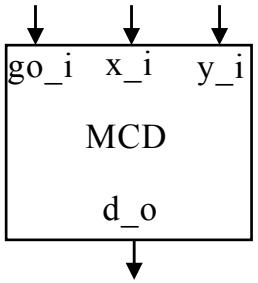
if (c1) sentencias1;
else if (c2) sentencias2;
else sentenciasX



Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento.
  - Ejemplo de aplicación de patrones de Síntesis:

MAXIMO COMÚN DIVISOR



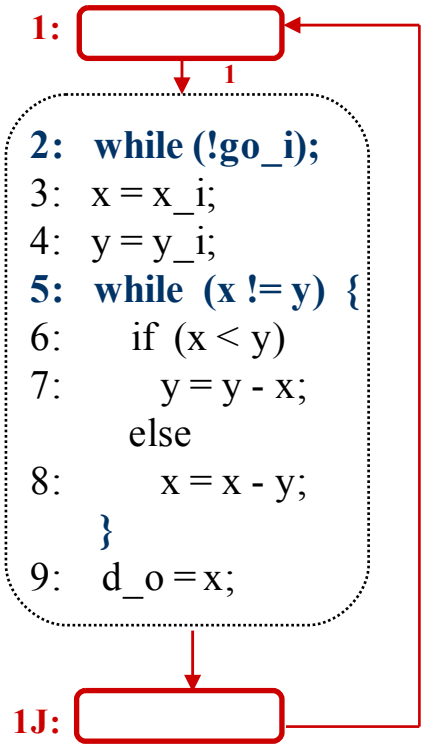
ALGORITMO

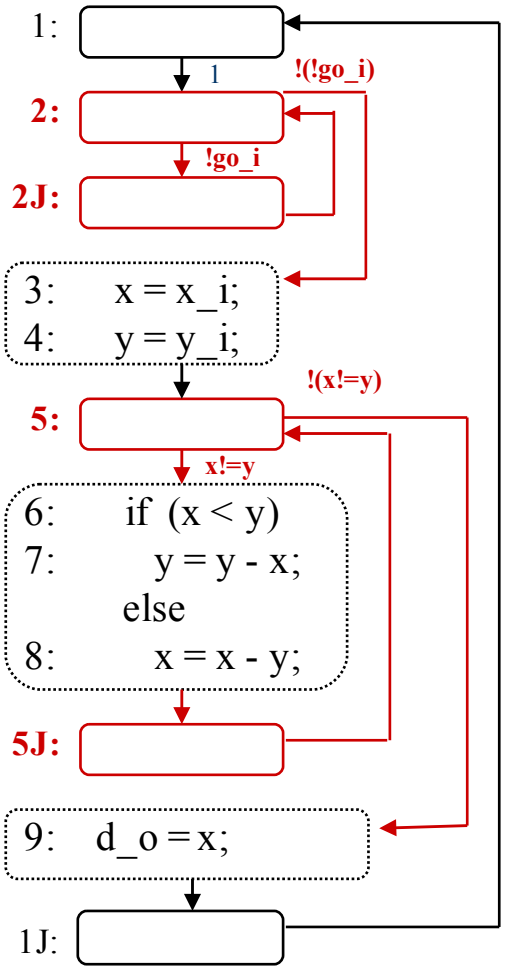
```
0: int  x, y;
1: while (1) {
2:   while (!go_i);
3:   x = x_i;
4:   y = y_i;
5:   while  (x != y)  {
6:       if  (x < y)
7:           y = y - x;
         else
8:           x = x - y;
     }
9:   d_o = x;
   }
```

Síntesis Comportamiento.

- Ejemplo de aplicación de patrones de síntesis:

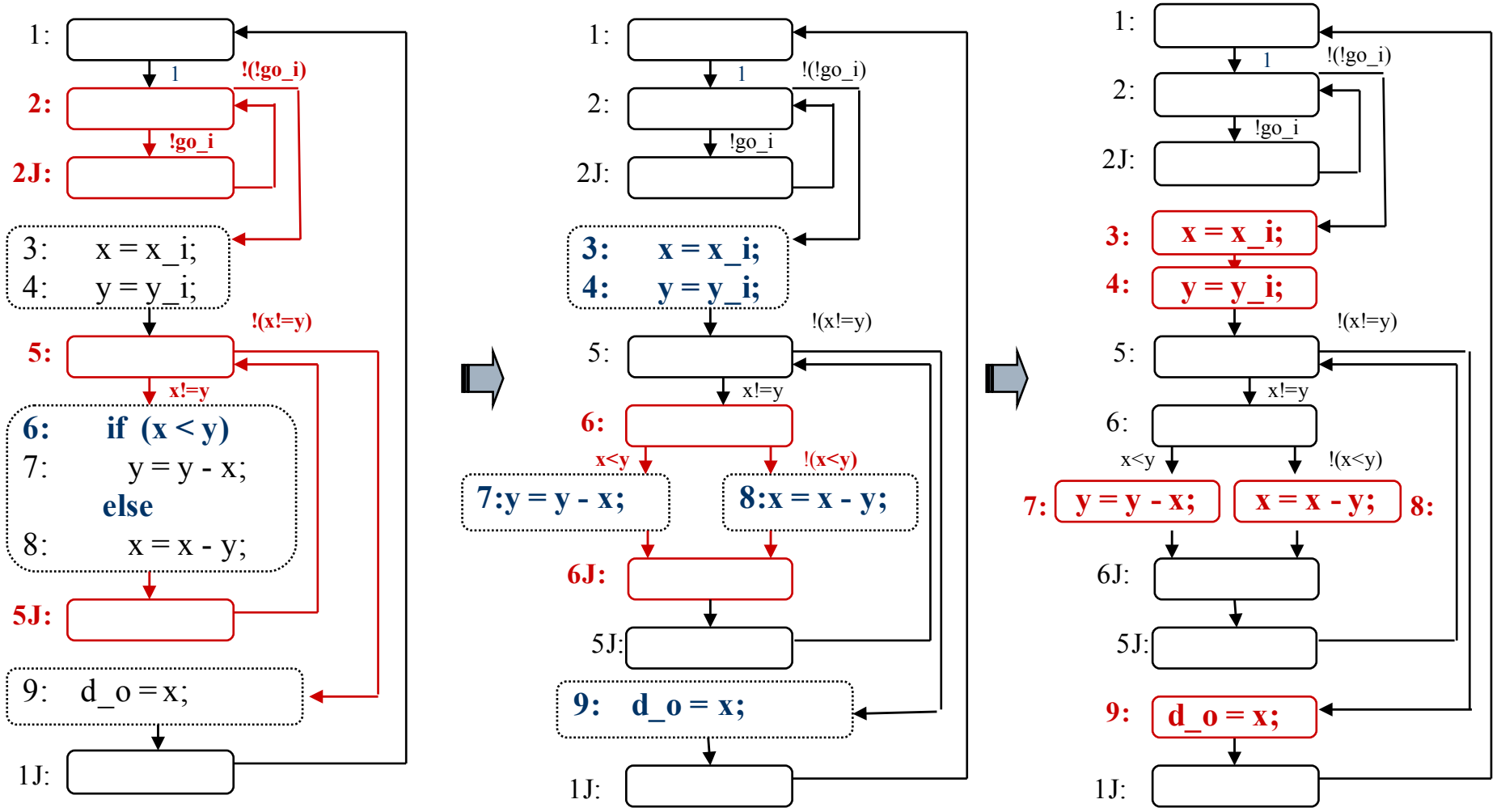
```
0: int  x, y;
1: while (1) {
2:   while (!go_i);
3:   x = x_i;
4:   y = y_i;
5:   while  (x != y)  {
6:       if  (x < y)
7:           y = y - x;
         else
8:           x = x - y;
     }
9:   d_o = x;
   }
```





Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento.
  - Ejemplo de aplicación de patrones de síntesis:

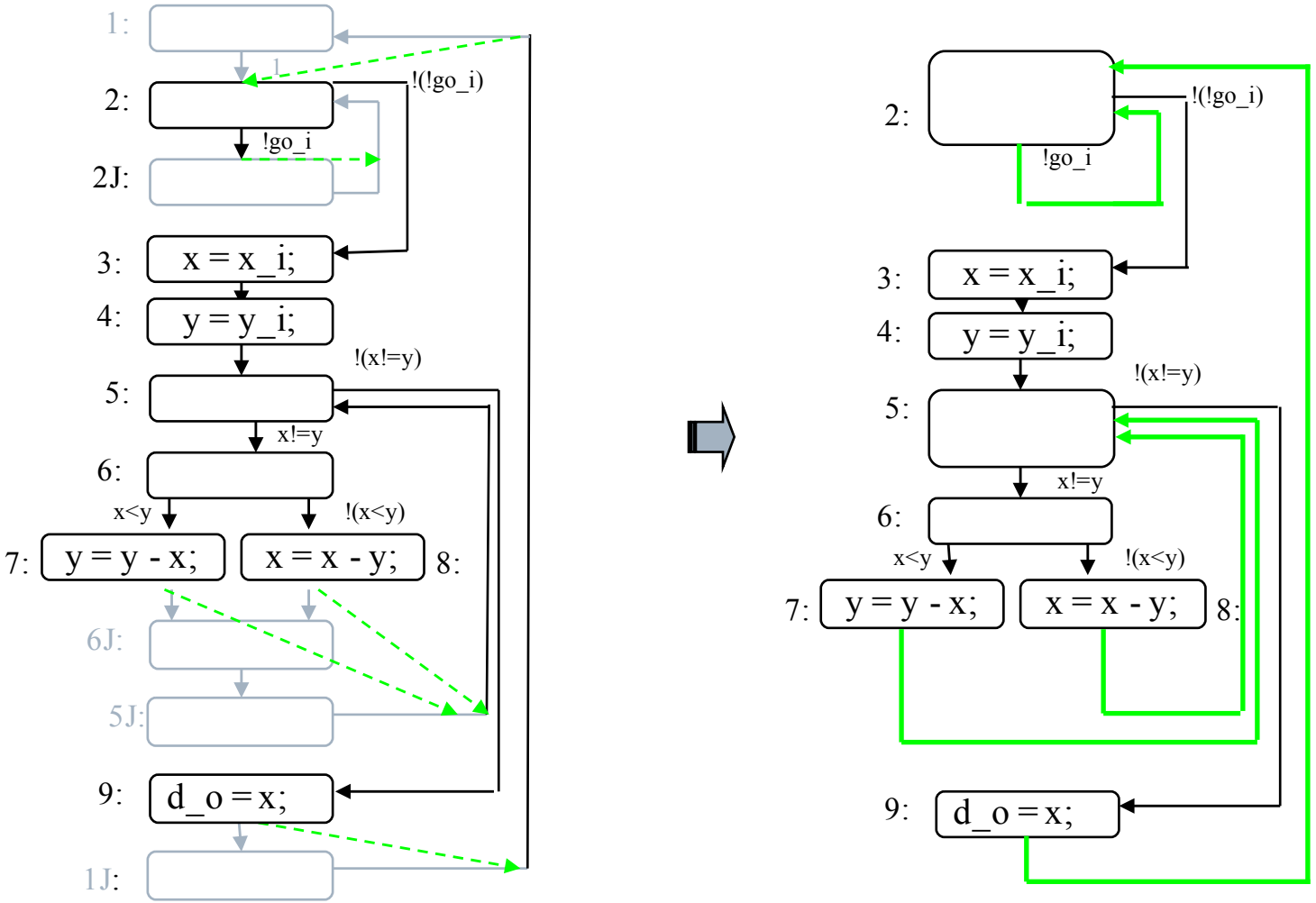


Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento:
  - 2ª Etapa: Reducción de la máquina de estados

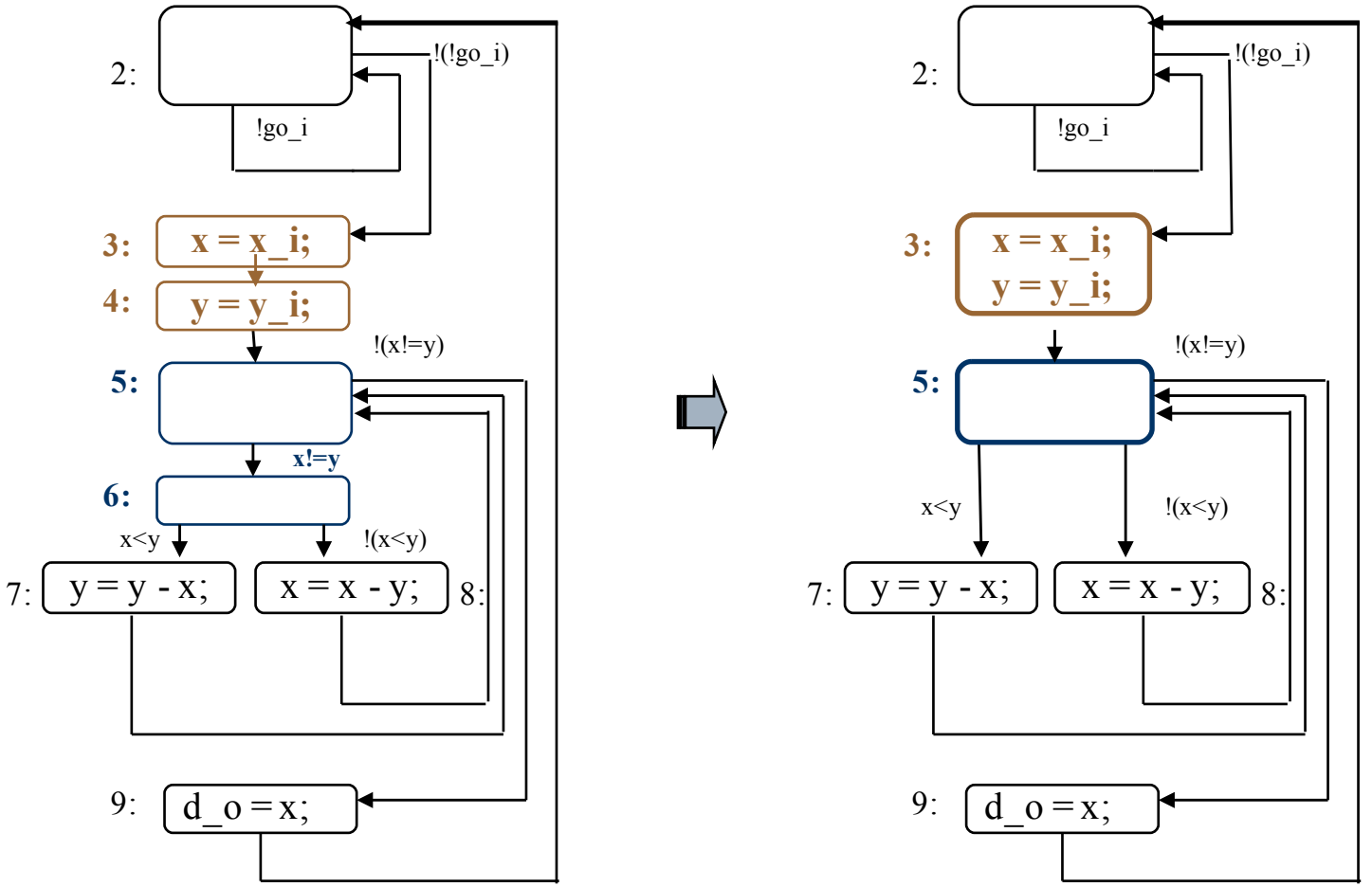
  - **Eliminación de estados**
    - Estados con transiciones de salida con condiciones constantes pueden ser eliminados
  - **Combinación de estados**
    - Estados adyacentes con operaciones independientes pueden ser combinados en un sólo estado.

- Síntesis Comportamiento.
  - Ejemplo de eliminación de estados:



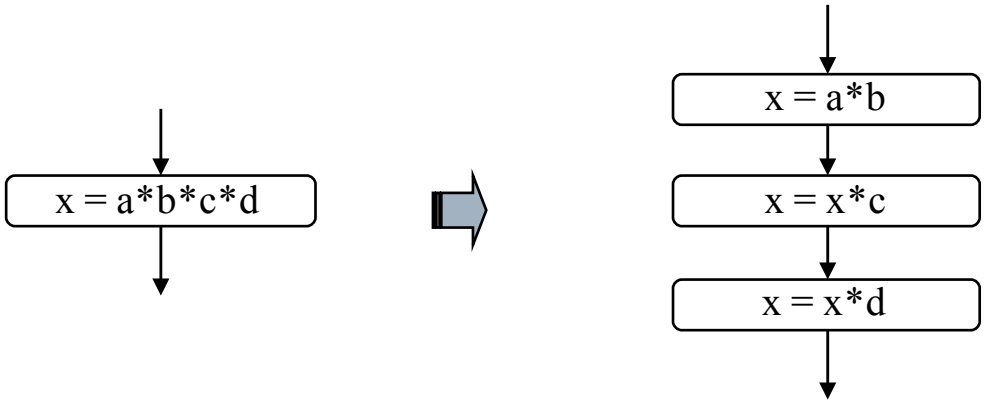
Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento.
  - Ejemplo de combinación de estados:



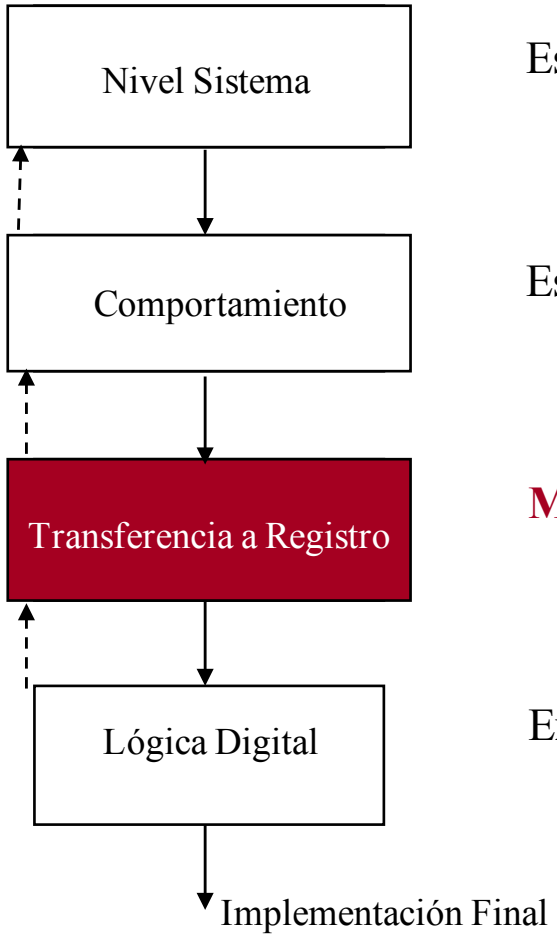
Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento:
  - Optimización
    - **Separación estados**
      - Estados que requieren operaciones complejas (a*b*c*d) pueden ser separados en varios estados con operaciones más simples con el fin de simplificar el hw que requiere su implementación.



Técnicas de Diseño y Validación de SE

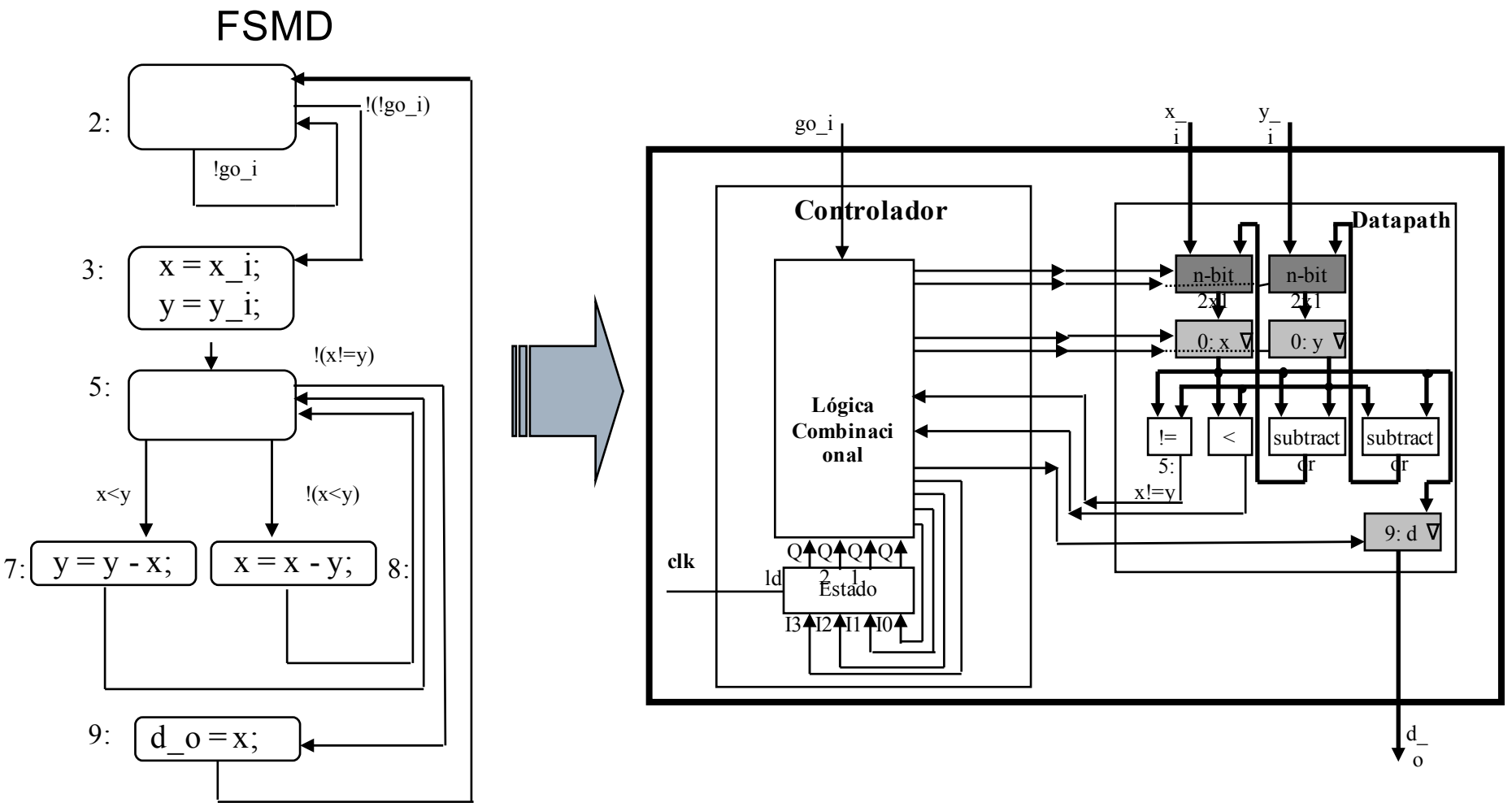
- Compilación/**Síntesis HW**:



Especificación Sistema ->	Partición en Componentes HW y SW
Especificación Secuencial ->	Máquina de Estados Finita
Máquina de Estados Finita->	**Registros** **Ruta de Datos (Data Path)** **Controlador (Exp Bool)**
Expresiones Booleanas->	Puertas Lógicas (NetList)

Técnicas de Diseño y Validación de SE

- Síntesis Comportamiento +Transferencia Registro



Técnicas de Diseño y Validación de SE

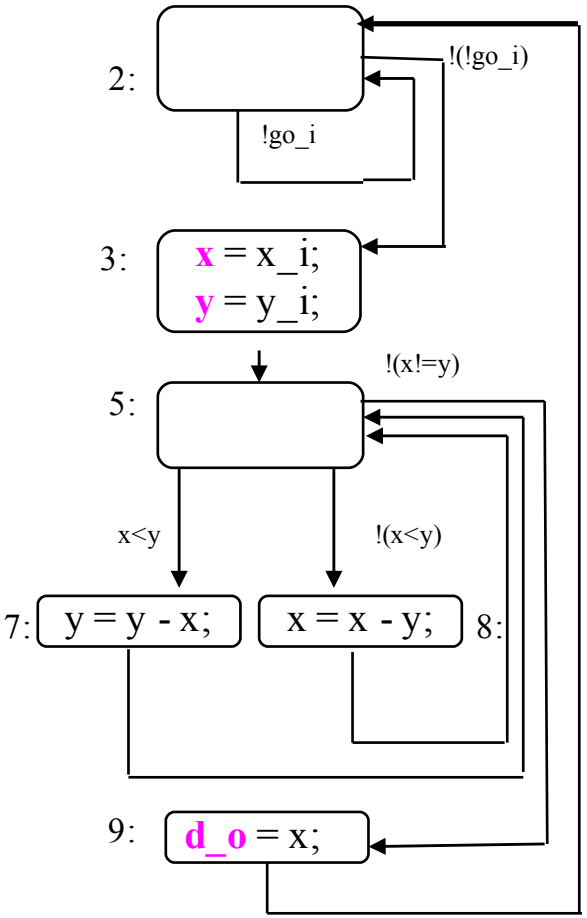
- Síntesis Transferencia a Registro:
  - 1ª Etapa: Crear el DataPath
  - 2ª Etapa: FSMD -> FSM
  - 3ª Etapa: Codificación de Estados
  - 4ª Etapa: Especificación del Controlador
    - Tabla de Transición de Estados
    - Tabla de la Salida

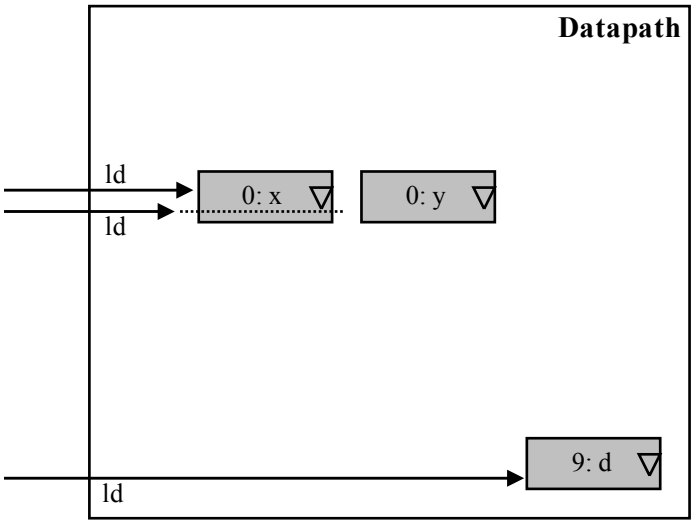
Técnicas de Diseño y Validación de SE

- Síntesis Transferencia a Registro:
  - 1ª Etapa: Crear el DataPath
    - Añadir un **registro** por cada **variable**
    - Añadir una **unidad funcional** por cada **operación aritmética o lógica**
    - Conectar los registros, puertos y unidades funcionales añadiendo **multiplexores** cuando haya **varias fuentes**
    - Asignar un identificador único a cada entrada y salida del **DataPath**
      - **Entradas = señales de control** de los registros, multiplexores y unidades funcionales
      - Salidas = **condiciones de disparo** de las **transiciones**

Técnicas de Diseño y Validación de SE

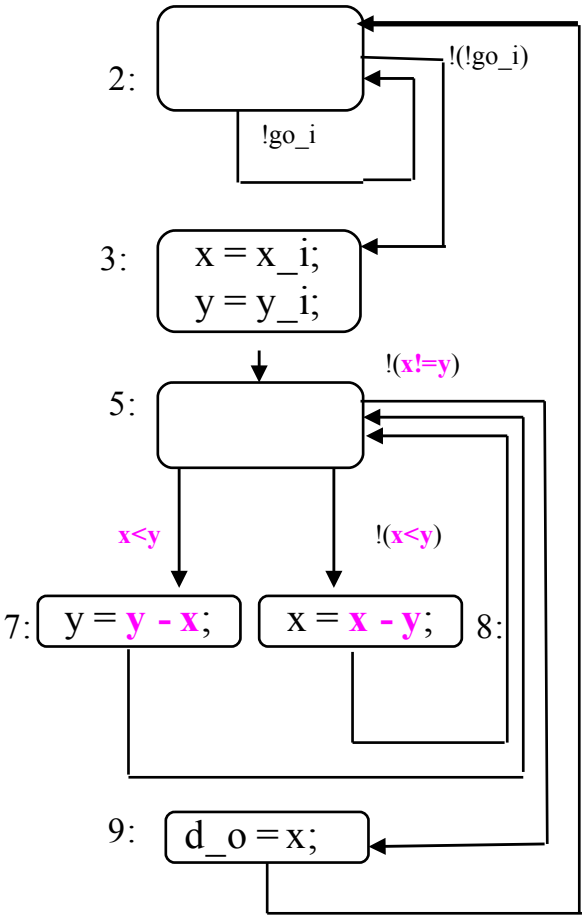
- Síntesis Transferencia a Registro:
  - 1ª Etapa:Crear el DataPath
    - Añadir un registro por cada variable local

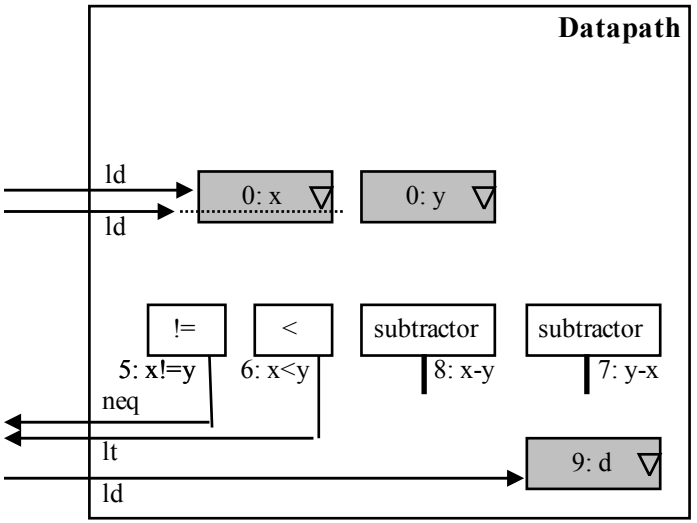




Técnicas de Diseño y Validación de SE

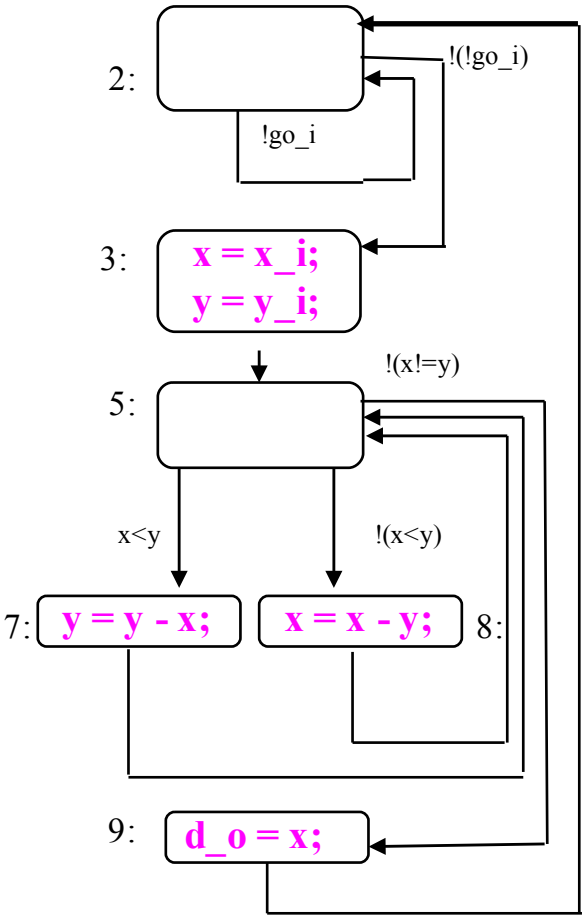
- Síntesis Transferencia a Registro:
  - 1ª Etapa:Crear el DataPath
    - Añadir unidades funcionales

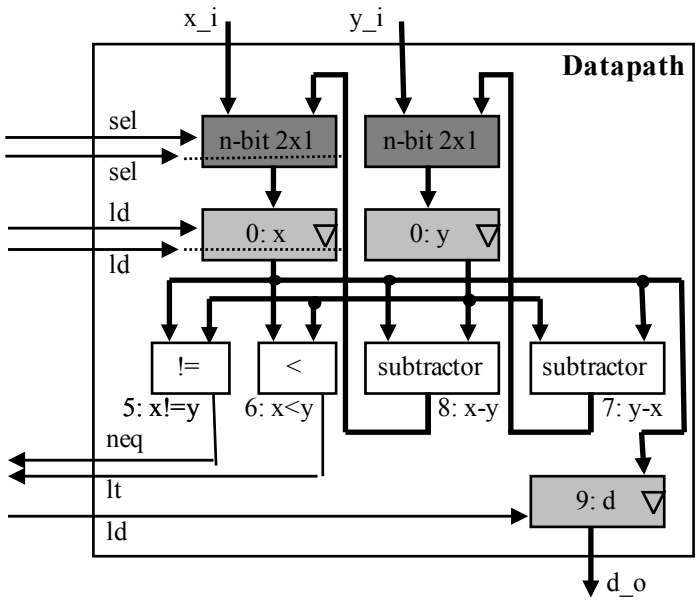




Técnicas de Diseño y Validación de SE

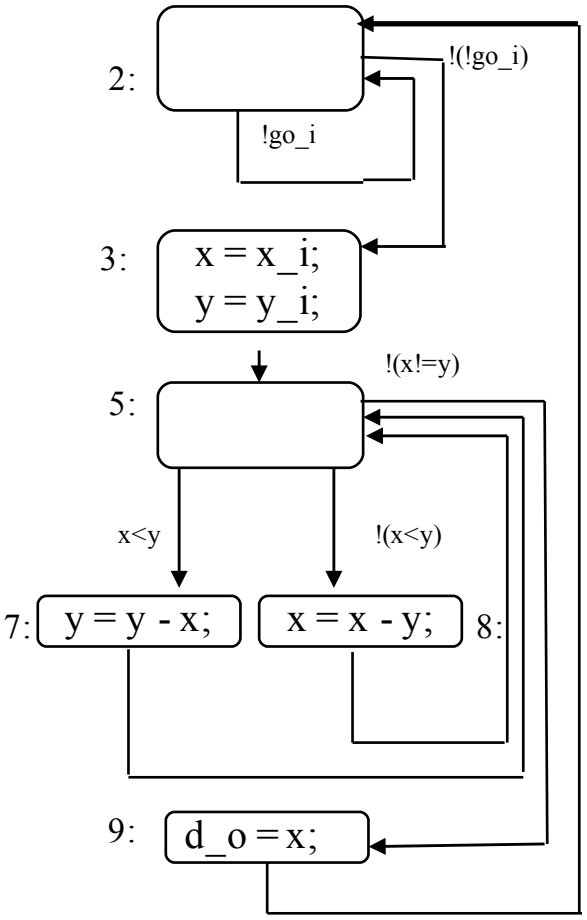
- Síntesis Transferencia a Registro:
  - 1ª Etapa:Crear el DataPath
    - Conectar y añadir multiplexores

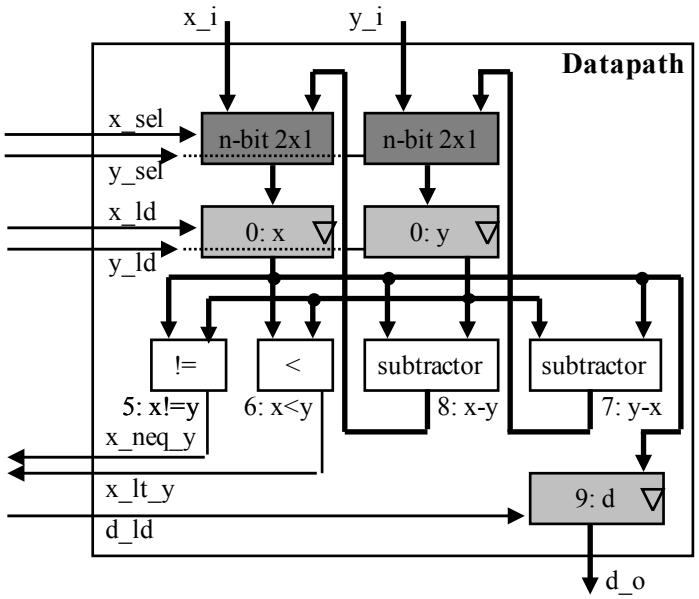




Técnicas de Diseño y Validación de SE

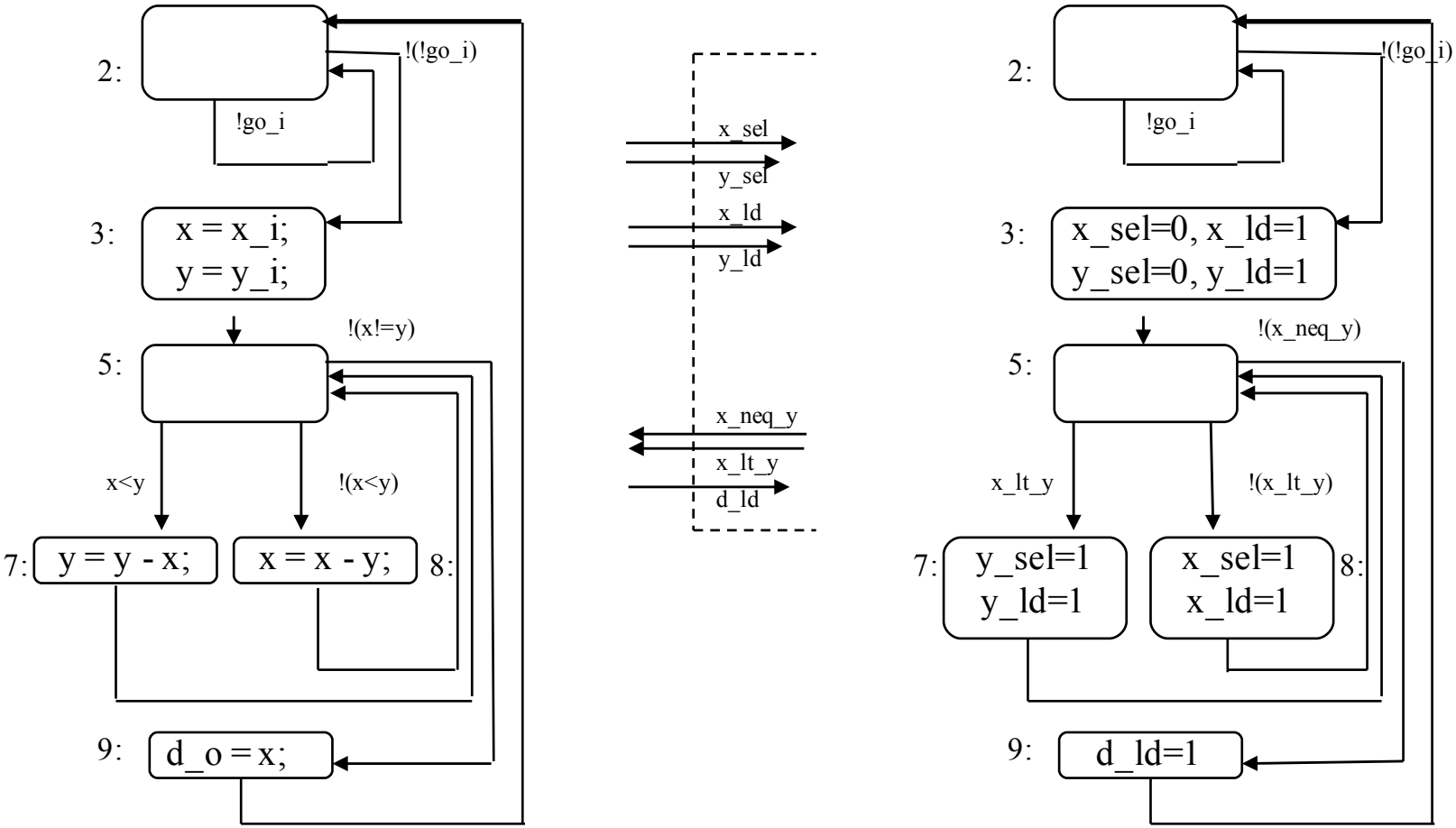
- Síntesis Transferencia a Registro:
  - 1ª Etapa:Crear el DataPath
    - Asignar identificadores





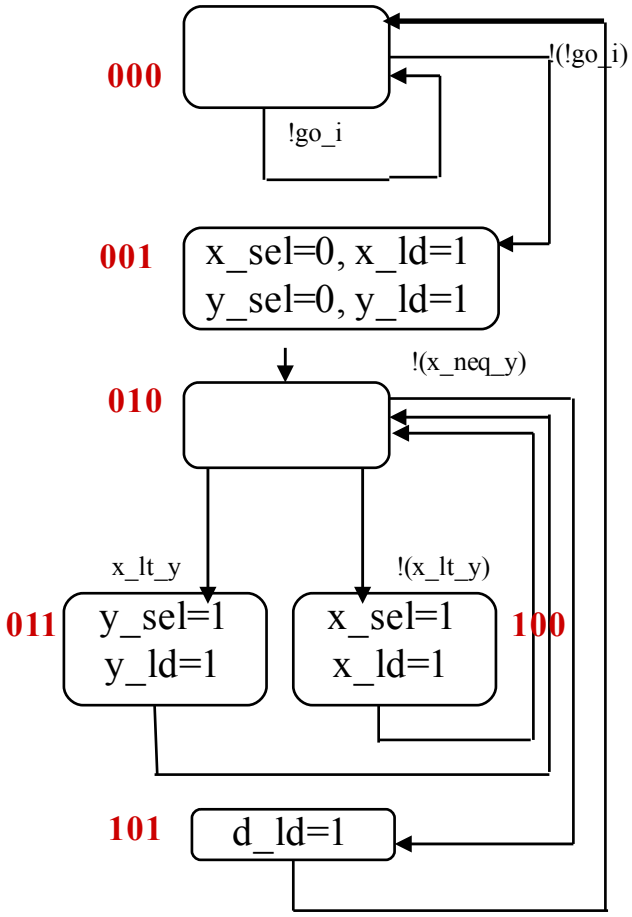
Técnicas de Diseño y Validación de SE

- Síntesis Transferencia a Registro:
  - 2ª Etapa:FSMD -> FSM



Técnicas de Diseño y Validación de SE

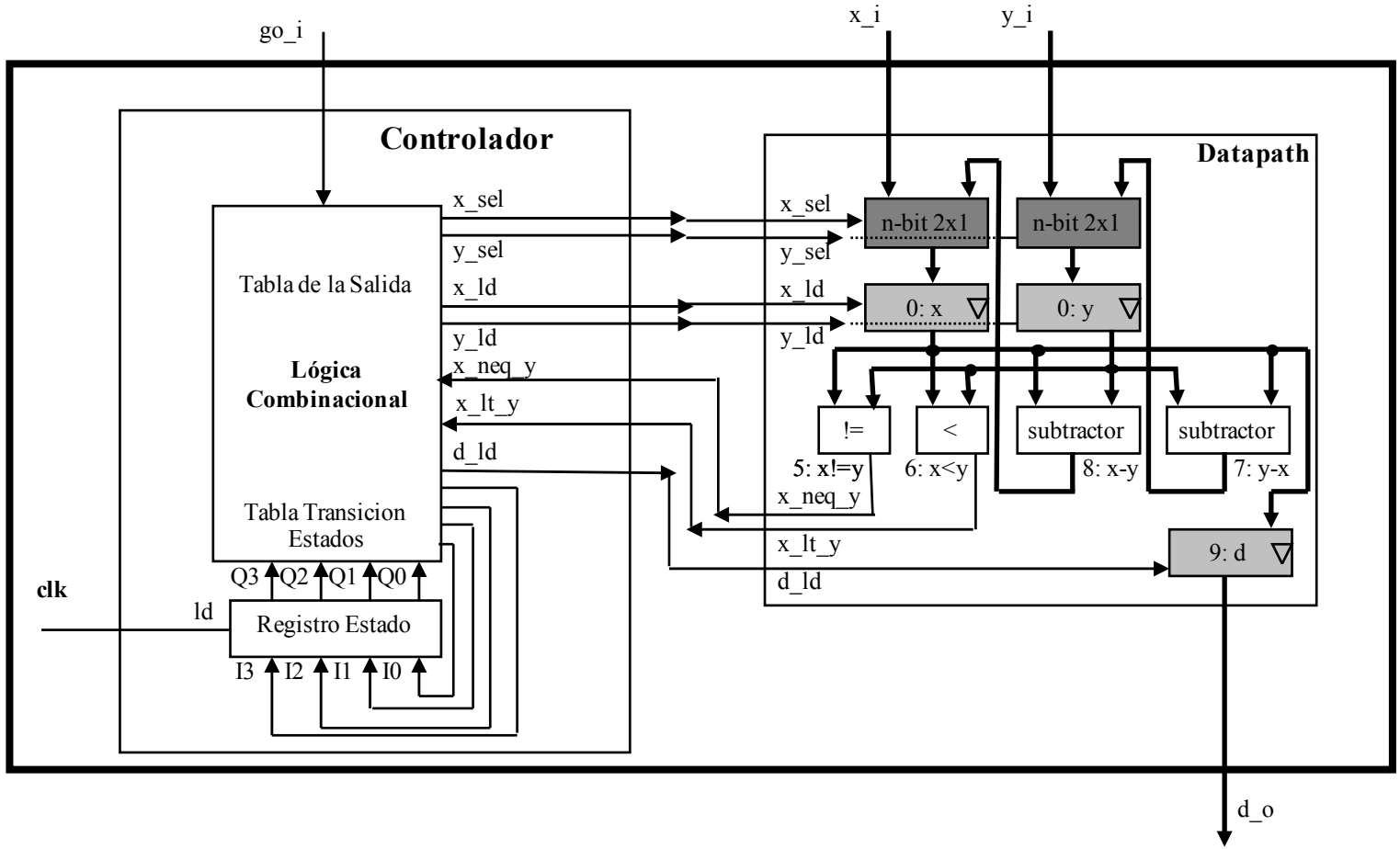
- Síntesis Transferencia a Registro:
  - 3ª Etapa:Codificación de Estados



Técnicas de Diseño y Validación de SE

- Síntesis Transferencia a Registro:
  - 4ª Etapa:Especificación Controlador

¡Reloj!



Técnicas de Diseño y Validación de SE

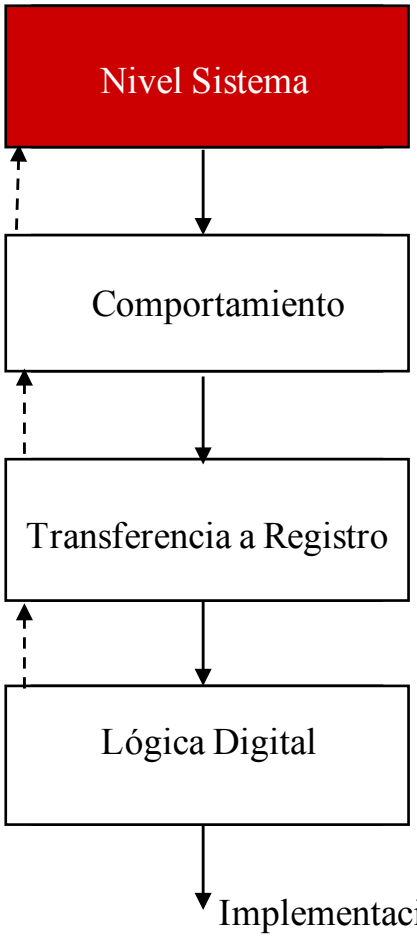
- Síntesis Transferencia a Registro:
  - 4ª Etapa:Especificación Controlador
    - Tabla de Transición de Estados
    - Tabla de la Salida

Inputs						Outputs							
Q2	Q1	Q0	x_neq_y	x_lt_y	go_i	I2	I1	I0	x_sel	y_sel	x_ld	y_ld	d_ld
0	0	0	*	*	0	0	0	0	X	X	0	0	0
0	0	0	*	*	1	0	0	1	X	X	0	0	0
0	0	1	*	*	*	0	1	0	0	0	1	1	0
0	1	0	0	*	*	1	0	1	X	X	0	0	0
0	1	0	1	0	*	1	0	0	X	X	0	0	0
0	1	0	1	1	*	0	1	1	X	X	0	0	0
0	1	1	*	*	*	0	1	0	X	1	0	1	0
1	0	0	*	*	*	0	1	0	1	X	1	0	0
1	0	1	*	*	*	0	0	0	X	X	0	0	1
1	1	0	*	*	*	0	0	0	X	X	0	0	0
1	1	1	*	*	*	0	0	0	X	X	0	0	0

Expresiones Booleanas

Técnicas de Diseño y Validación de SE

- Compilación/Síntesis **SW**:



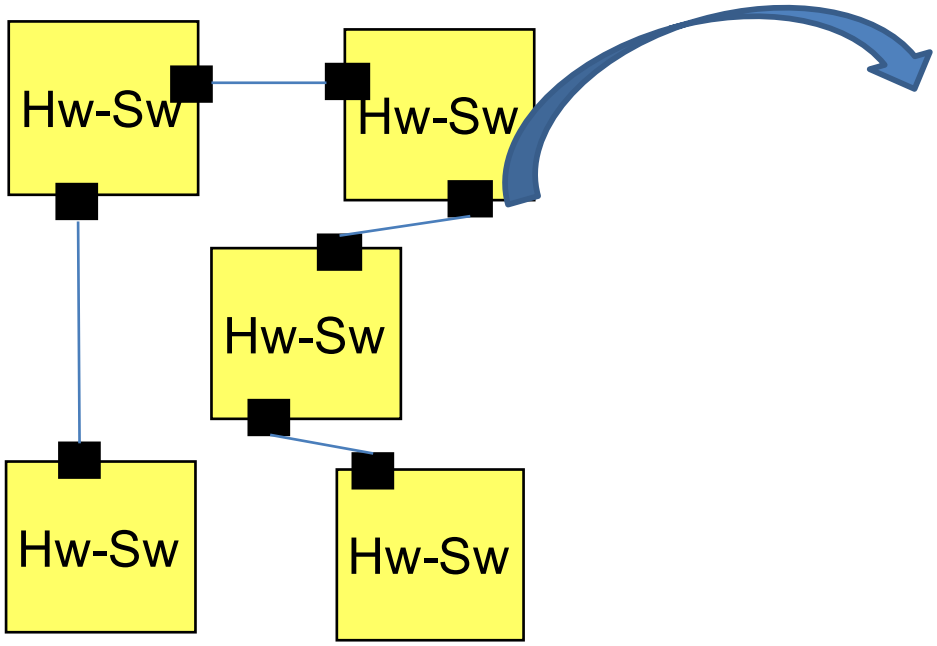
Especificación Sistema -> **Partición en Componentes HW y SW**

Especificación secuencial -> Código Fuente

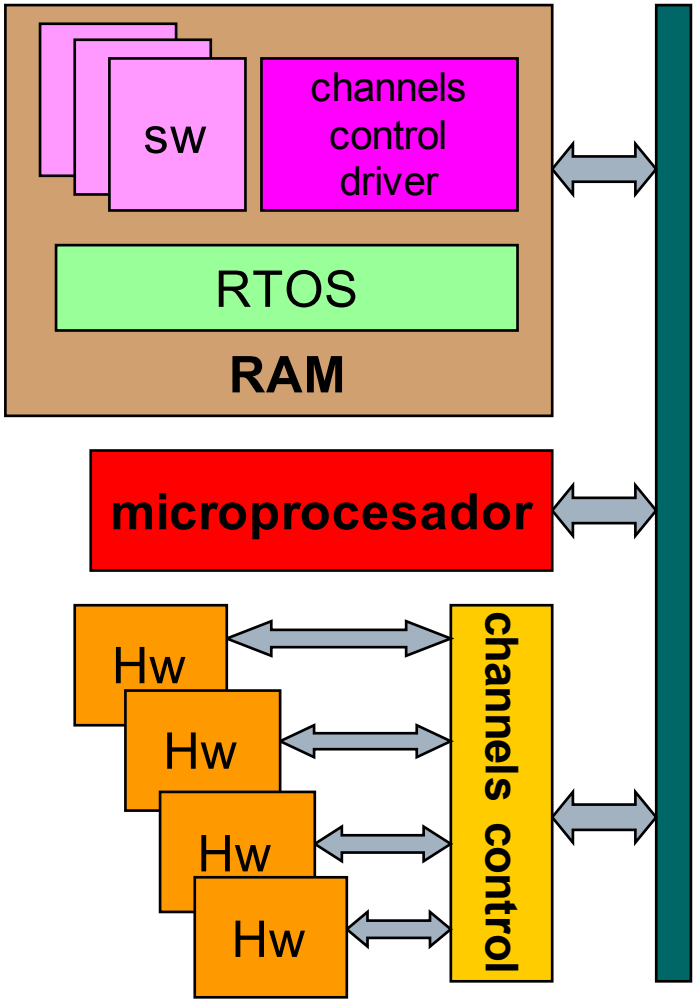
Código Fuente -> Código Ensamblador

Técnicas de Diseño y Validación de SE

- Síntesis Sistema:
  - Especificación Sistema-> Partición Componentes

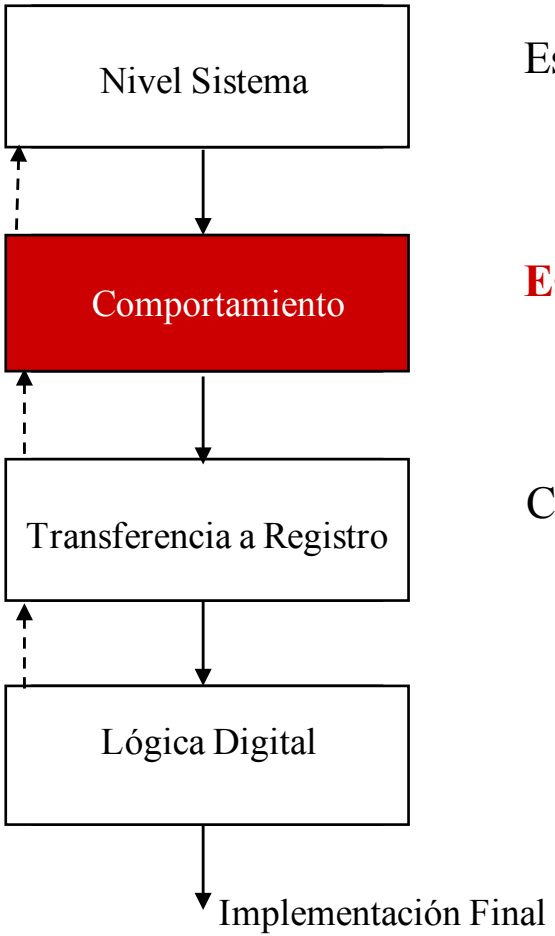


Componentes Sistema



Técnicas de Diseño y Validación de SE

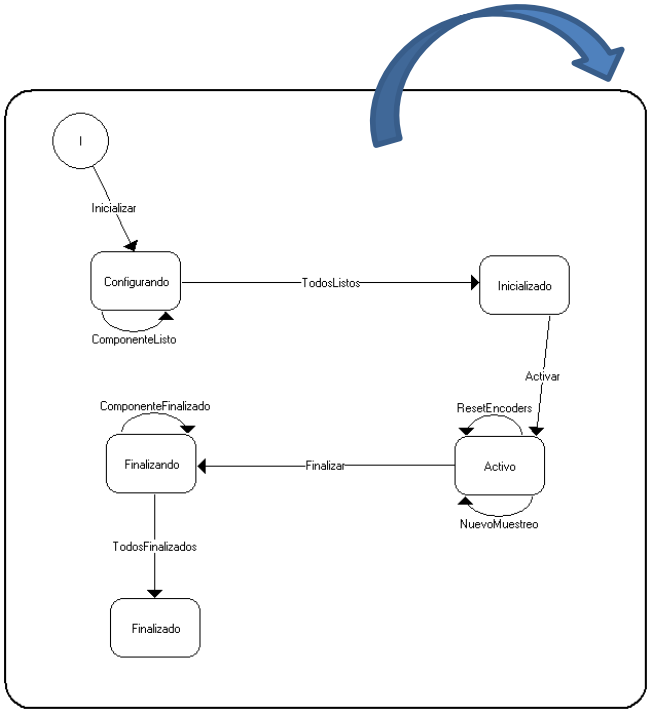
- Compilación/Síntesis **SW**:



Especificación Sistema ->	Partición en Componentes HW y SW
Especificación secuencial ->	**Código Fuente**
Código Fuente ->	Código Ensamblador

Técnicas de Diseño y Validación de SE

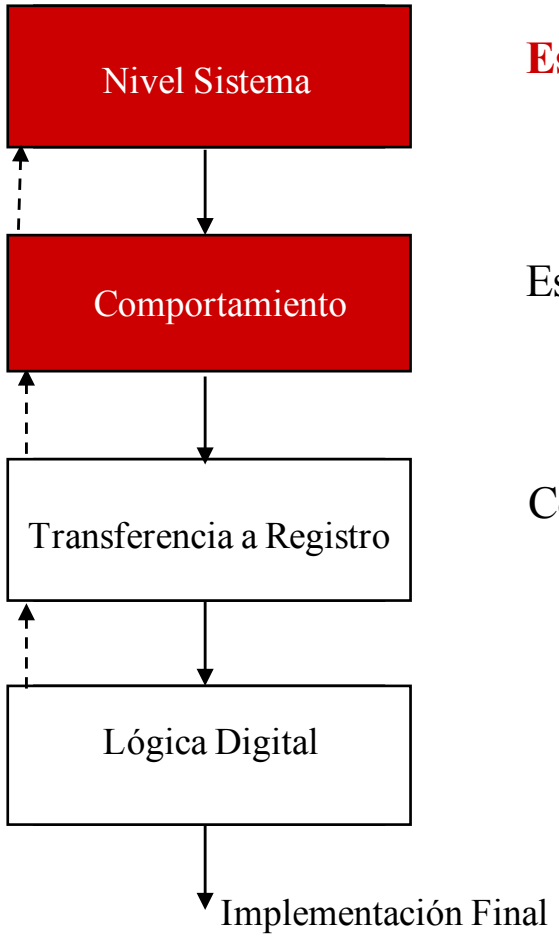
- Síntesis Comportamiento:
  - Especificación Secuencial-> Código fuente




```
TEDROOMTransId edroomCurrentTrans;
edroomCurrentTrans = EDROOMIllegada();
do{
    switch(edroomCurrentTrans.localId){
            case(Inicializar):
                    FInitTimeStamp();
                    edroomNextState = Configurado;
                     break;
            case(TodosListos):
                    FTimer1_2Sec();
                    edroomNextState = Inicializado;
…
    }
     switch(edroomNextState){
             case(I):
                 edroomCurrentTrans=EDROOMIllegada(); break;
             case(Configurado):
                    edroomCurrentTrans=EDROOMConfiguradollegada();
                                                          break;
…
    }
```

Técnicas de Diseño y Validación de SE

- Compilación/Síntesis **SW**:



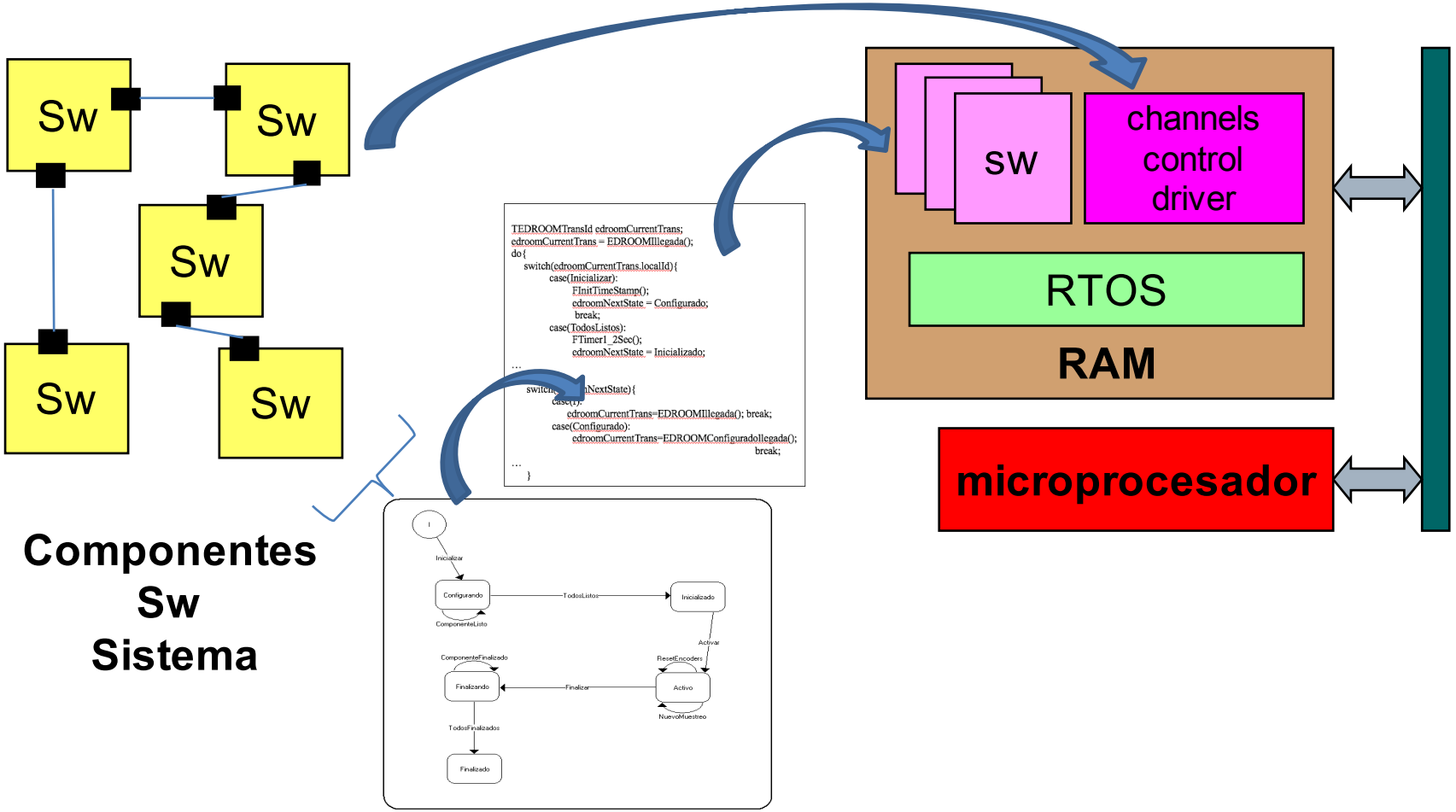
Especificación Sistema -> Partición SW

Especificación secuencial -> **Código Fuente**

Código Fuente -> Código Ensamblador

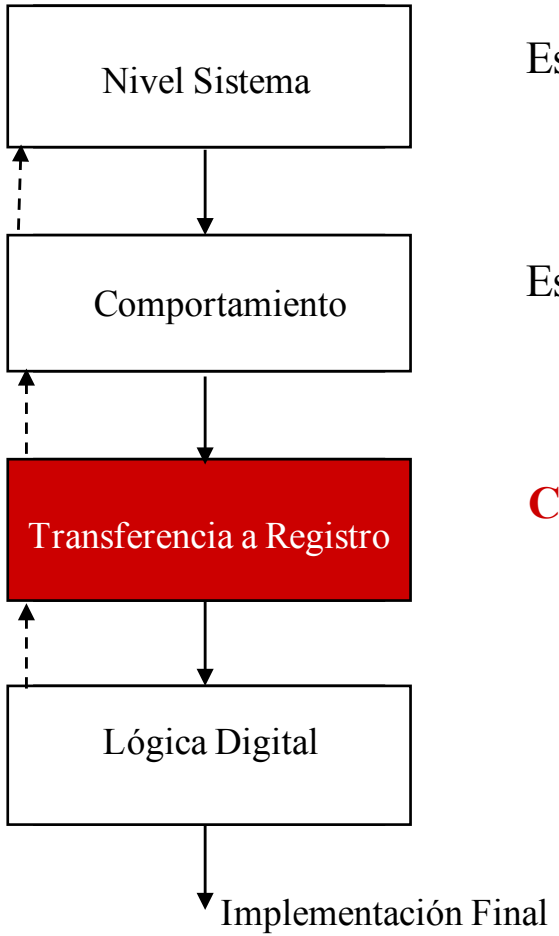
Técnicas de Diseño y Validación de SE

- Síntesis Sistema:
  - Especificación Sistema-> Partición Componentes



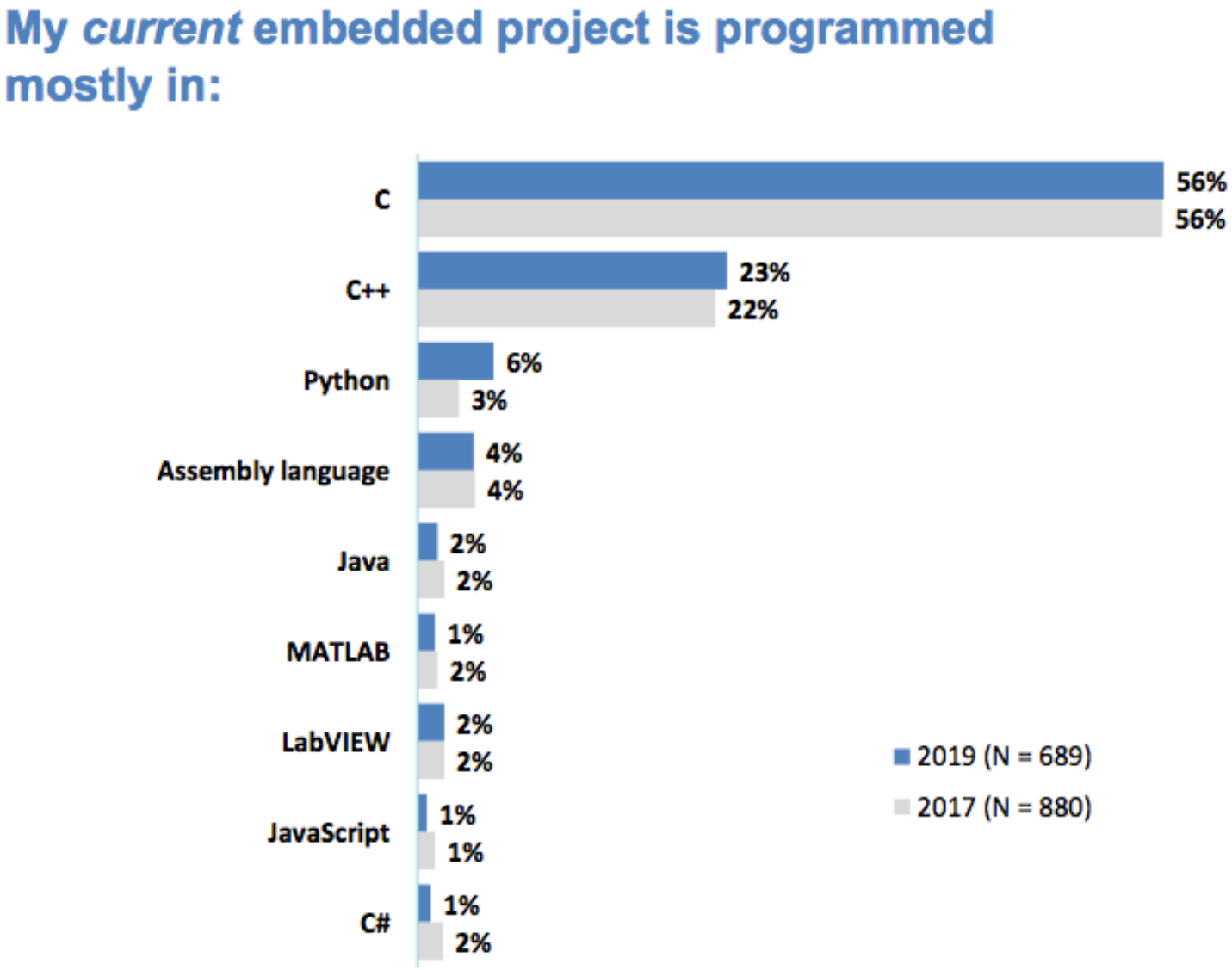
Técnicas de Diseño y Validación de SE

- **Compilación**/Síntesis **SW**:



Especificación Sistema ->	Partición en Componentes HW y SW
Especificación secuencial ->	Código Fuente
Código Fuente ->	**Código Ensamblador**

Técnicas de Diseño y Validación de SE

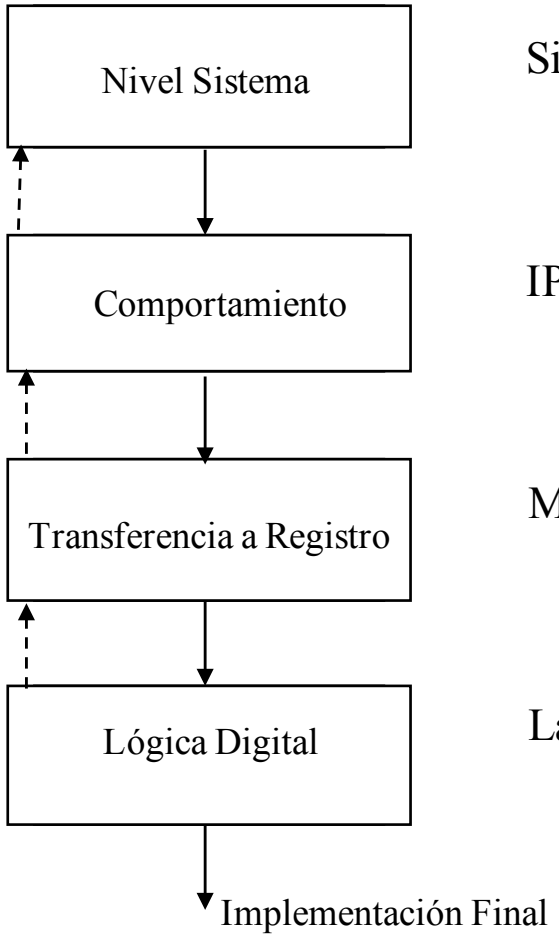


Técnicas de Diseño y Validación de SE

- Compilación/Síntesis:
- **Librerías/IP cores:**
- Test/Validación:

Técnicas de Diseño y Validación de SE

- Librerías/**IP Cores HW**:



Sistema Sintetizable, SOC (System On a Chip)

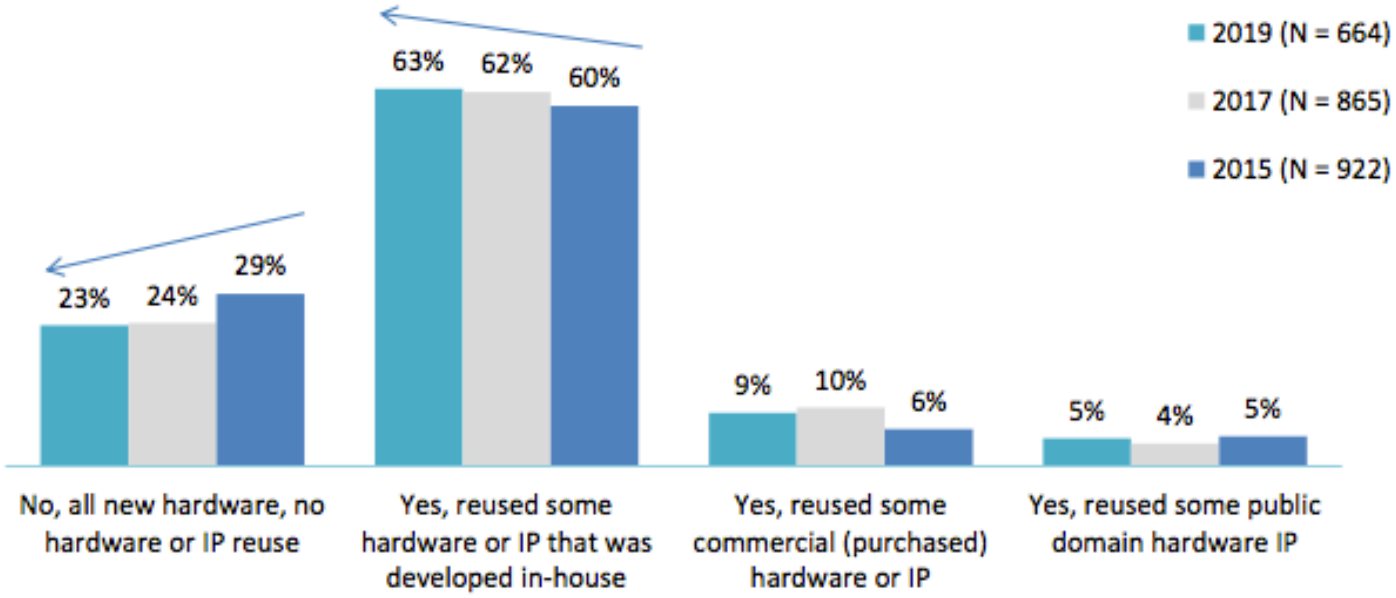
IPCores: Interfaces Bus, Controladores, Procesadores

Módulos MSI

Layout Puertas Lógicas

Técnicas de Diseño y Validación de SE

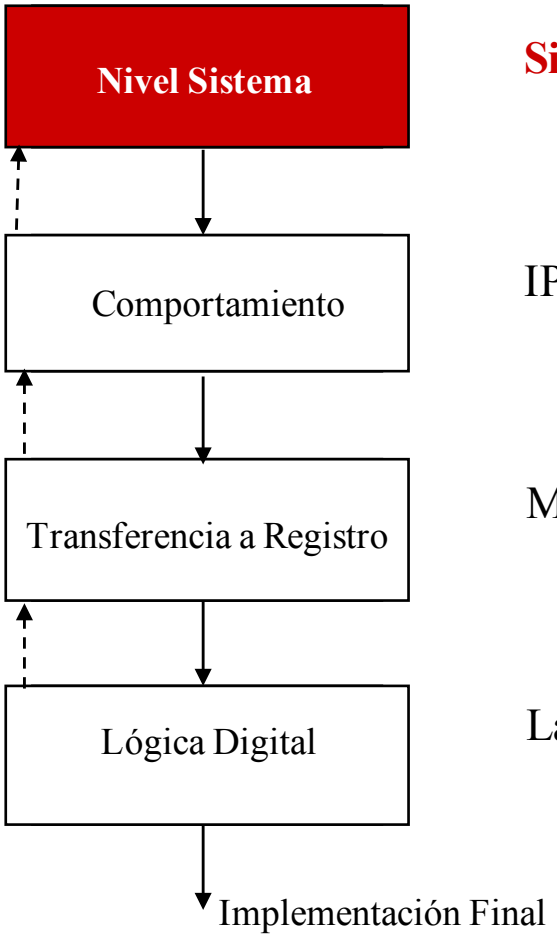
Does your current embedded project reuse hardware or hardware IP from a previous project?



Over three quarters **(77%)** of embedded developers reuse hardware or hardware IP. **63%** reuse hardware or hardware IP that was developed **in house**. Possibly a **slight trend** towards using more in-house hardware or hardware IP in future designs.

Técnicas de Diseño y Validación de SE

- Librerías/**IP Cores HW**:



Sistema Sintetizable, SOC (System On a Chip)

IPCores: Interfaces Bus, Controladores, Procesadores

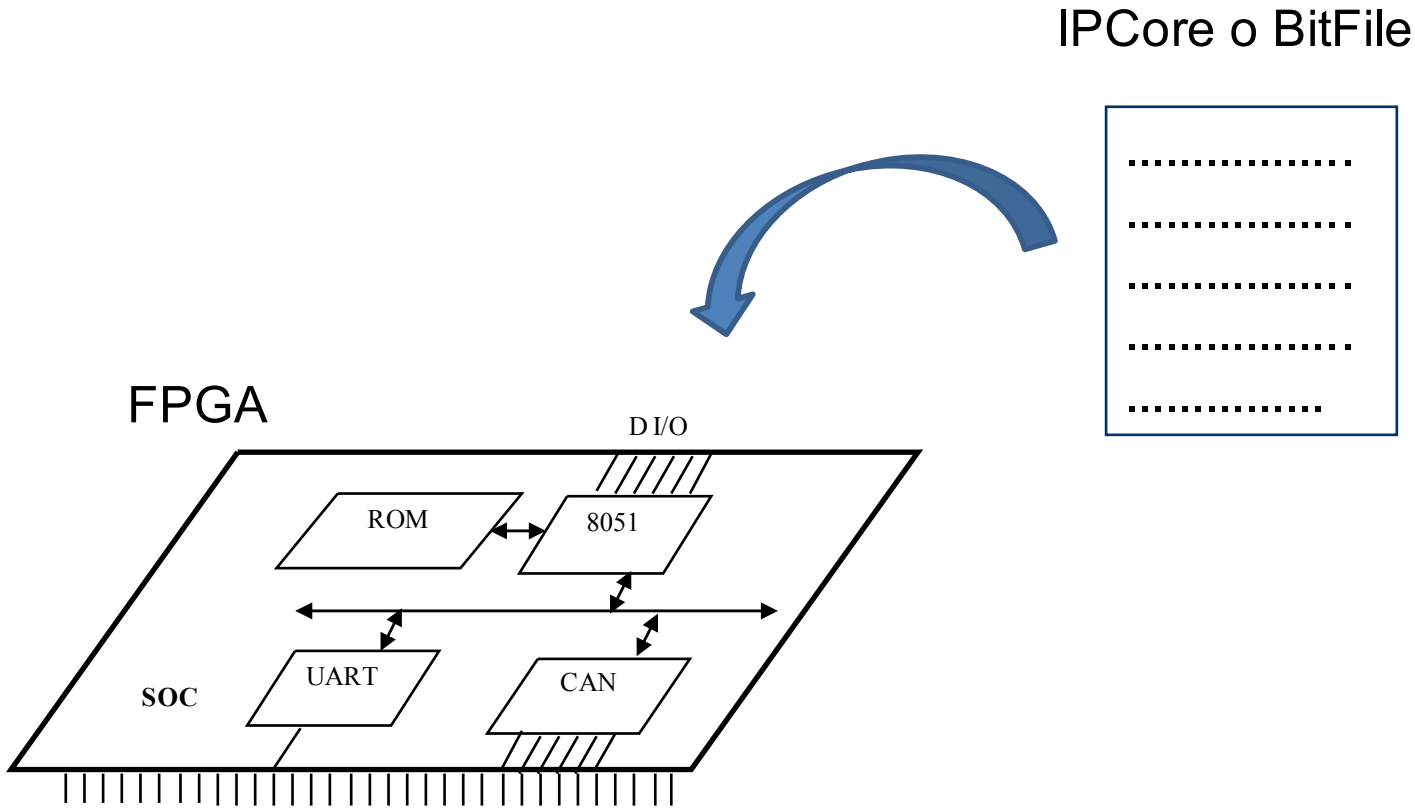
Módulos MSI

Layout Puertas Lógicas

Técnicas de Diseño y Validación de SE

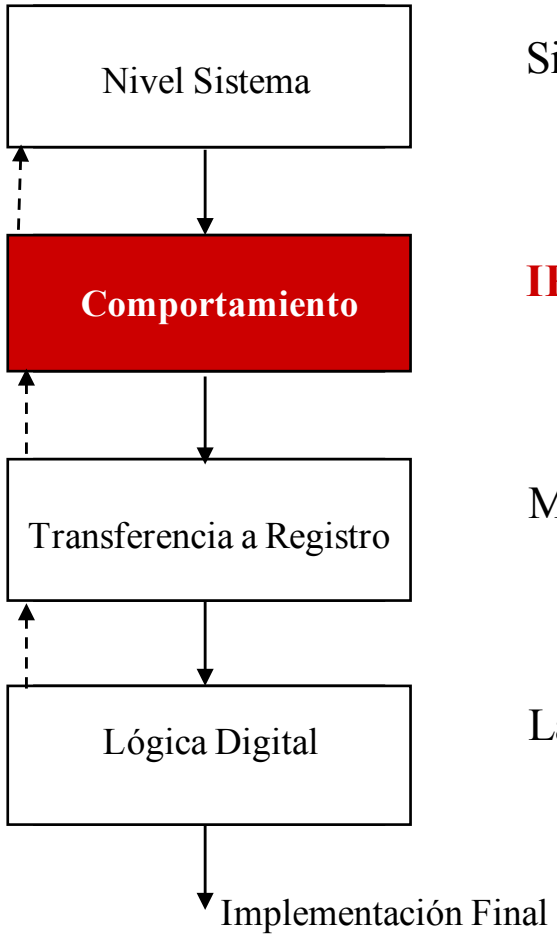
IP Core Sistema HW

- Sistema Sintetizable, SOC (System On a Chip)



Técnicas de Diseño y Validación de SE

- Librerías/**IP Cores HW**:



Sistema Sintetizable(Bit File) SOC

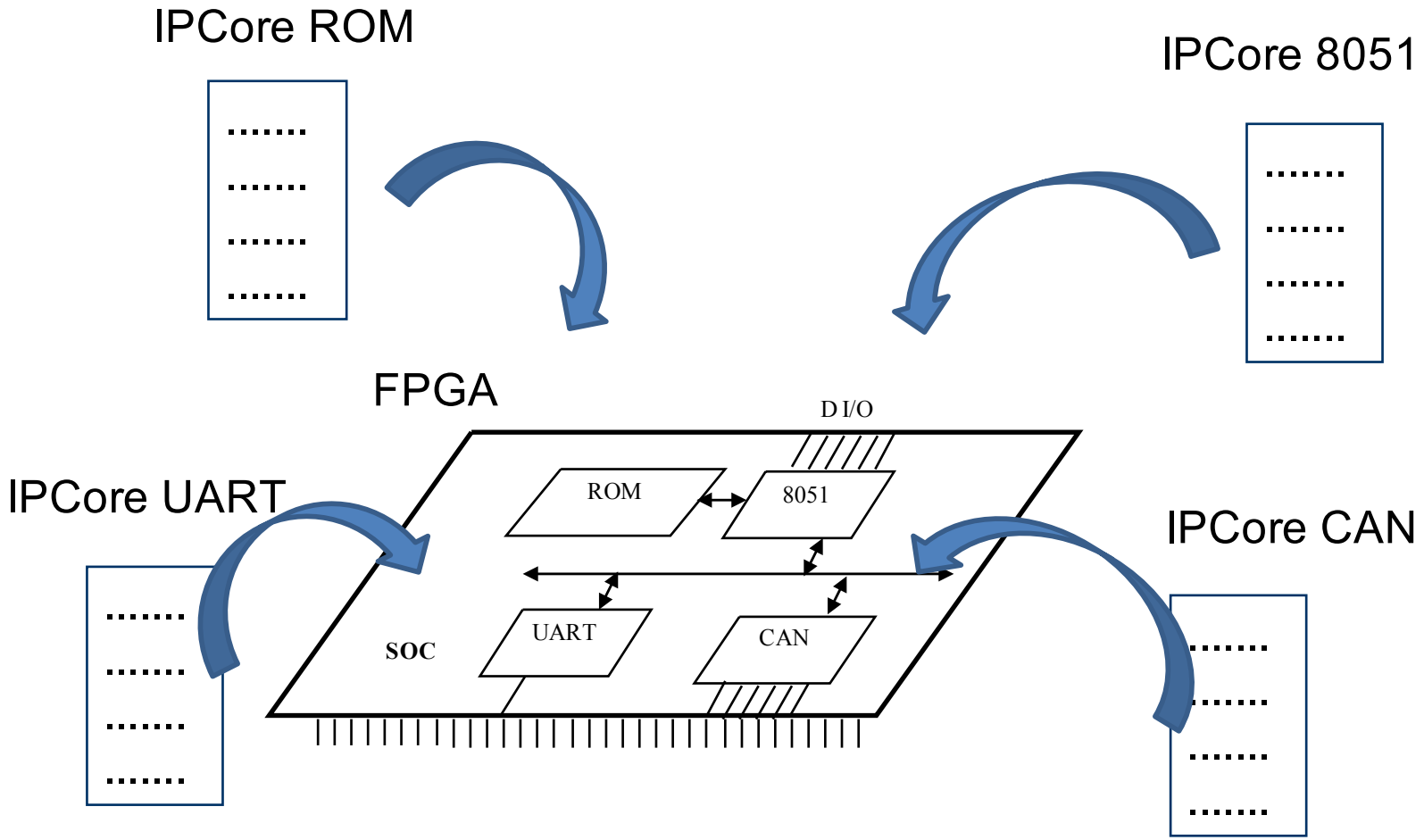
IPCores: Interfaces Bus, Controladores, Procesadores

Módulos MSI

Layout Puertas Lógicas

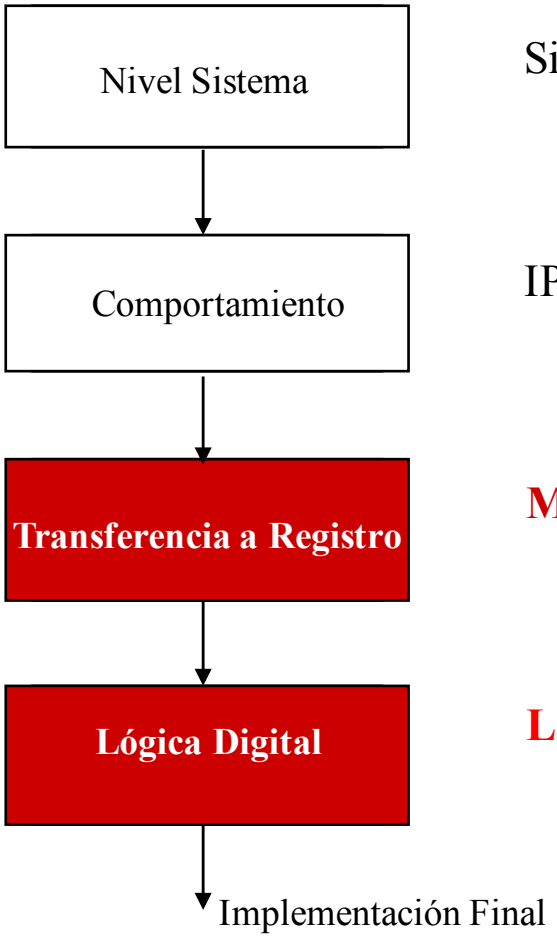
Técnicas de Diseño y Validación de SE

- **IP Cores HW**:
  - IPCores: Interfaces Bus, Controladores, Procesadores



Técnicas de Diseño y Validación de SE

- Librerías/**IP Cores HW**:



Sistema Sintetizable(Bit File) SOC

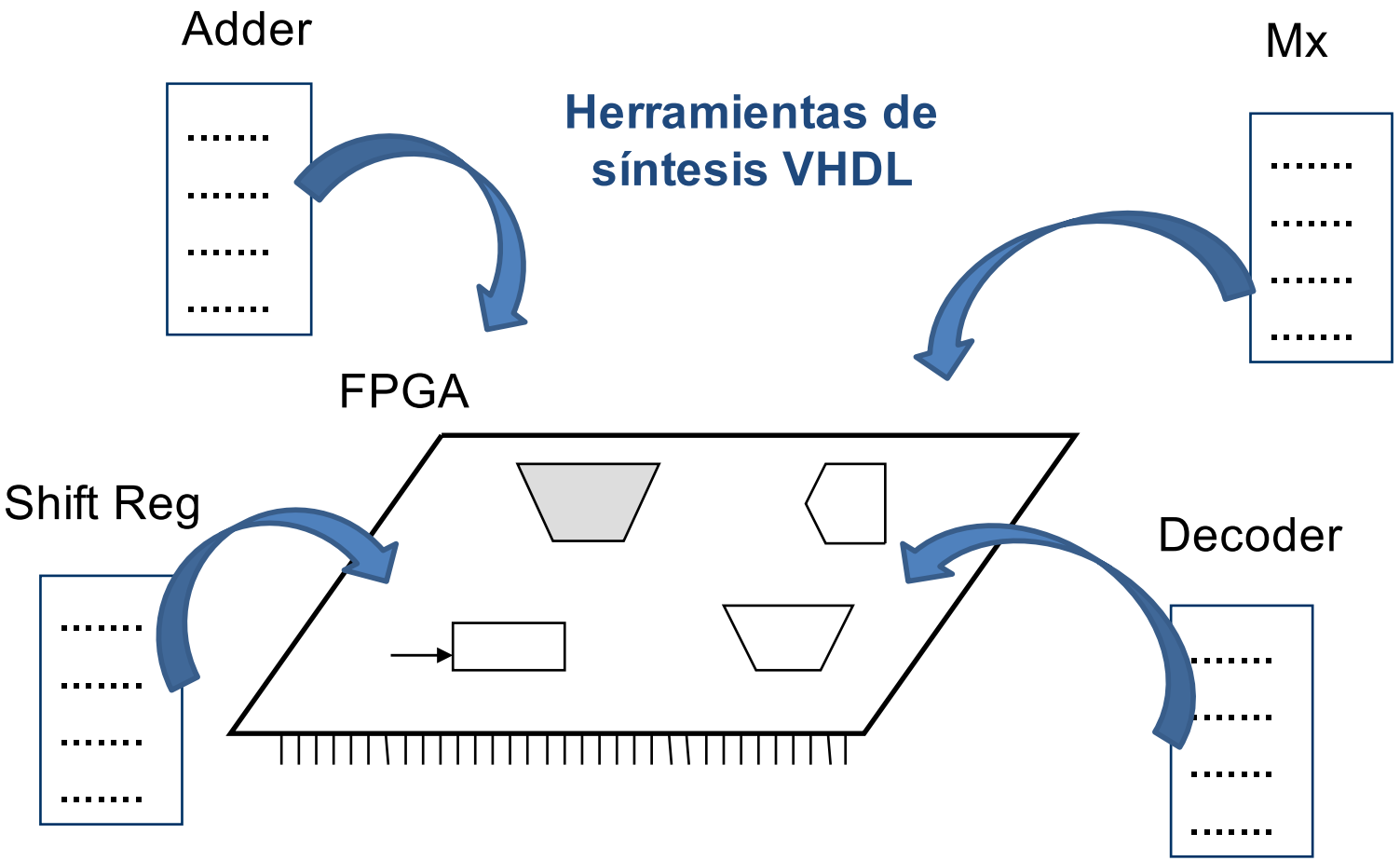
IPCores: Interfaces Bus, Controladores, Procesadores

Módulos MSI

Layout Puertas Lógicas

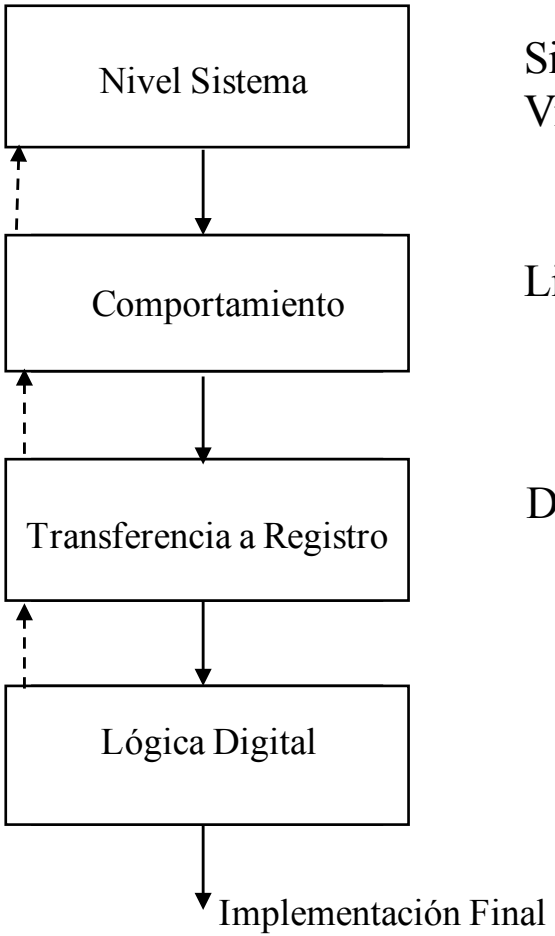
Técnicas de Diseño y Validación de SE

- **IP Cores HW**:
  - Modulos MSI



Técnicas de Diseño y Validación de SE

Librerías/IP Cores SW:



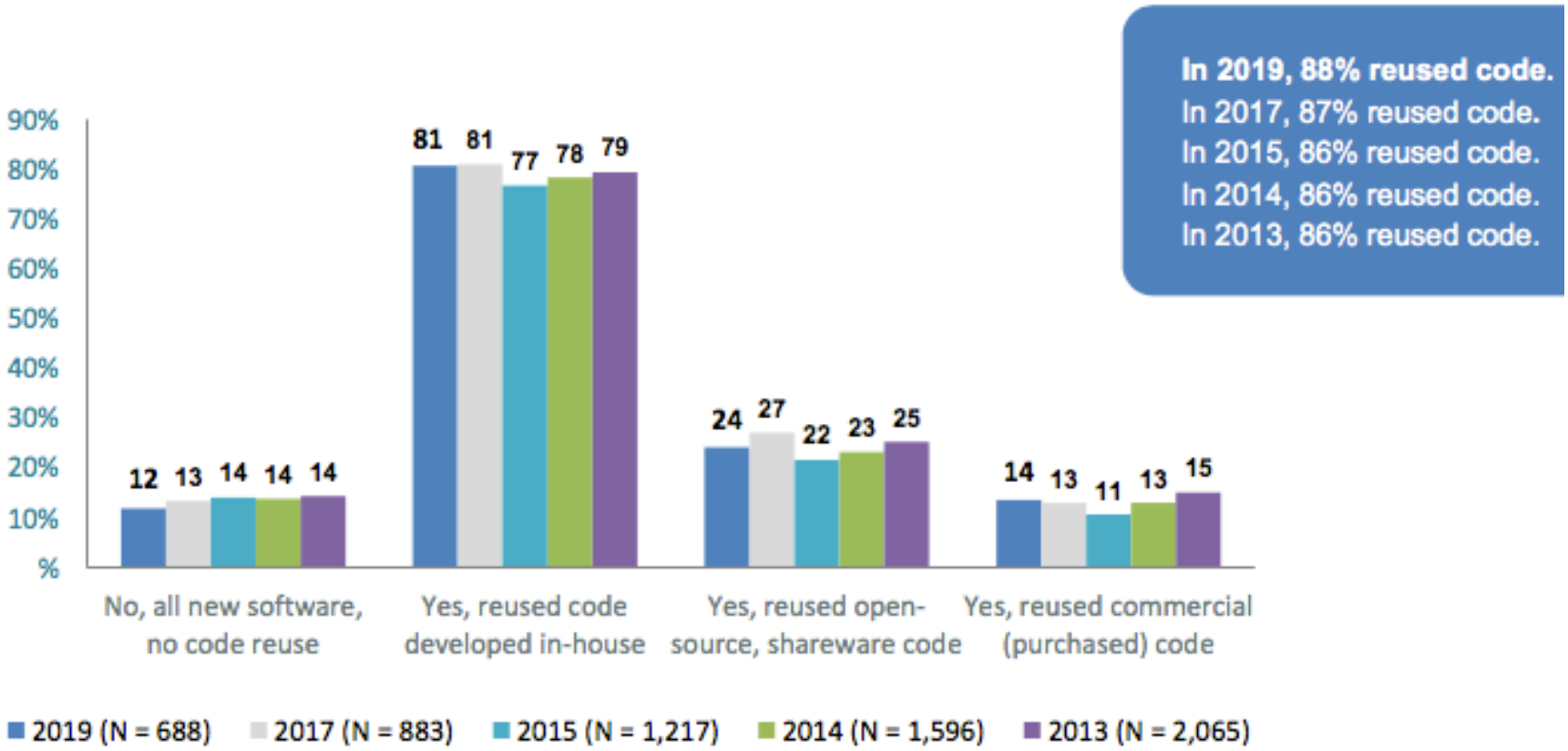
Sistema Software Compatible, Sistemas Operativos, Máquinas Virtuales

Librerías de Servicios, Componentes SW, Clases

Drivers

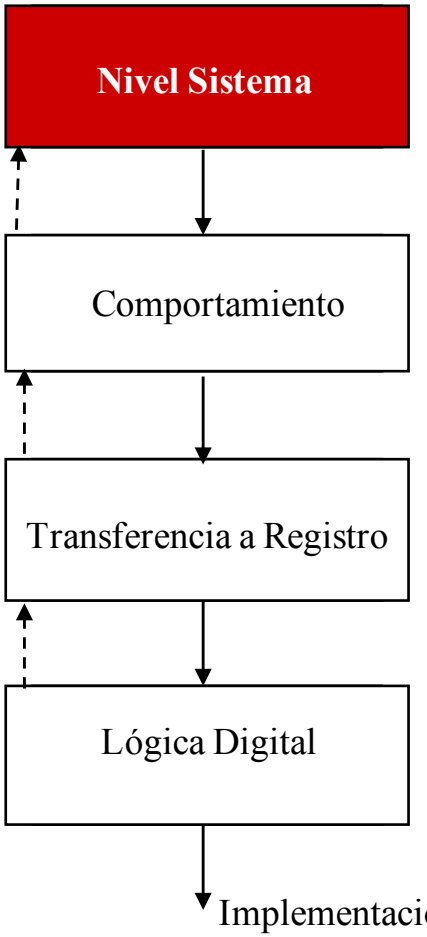
Reutilización SW:

Does your current project reuse code from a previous embedded project?



Técnicas de Diseño y Validación de SE

- Librerías/IP Cores SW:

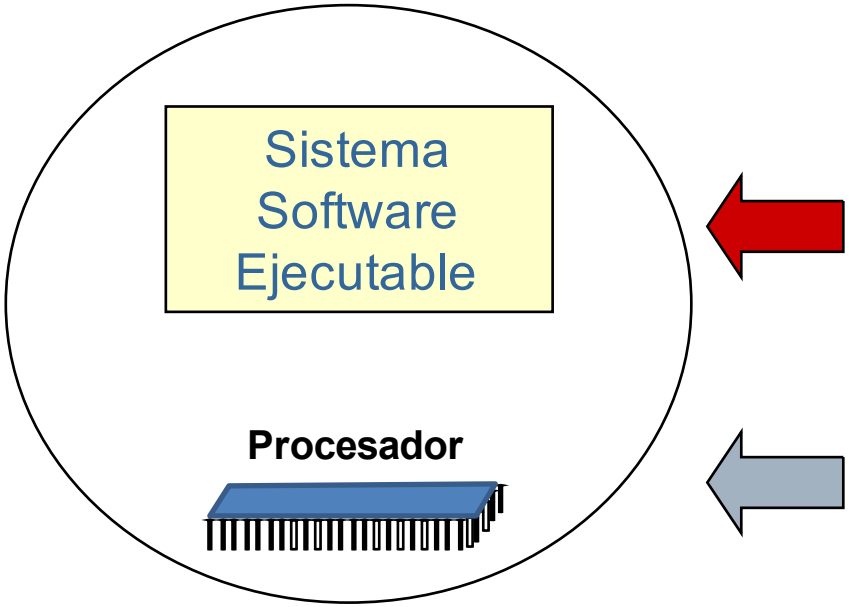


Sistema Software Compatible, Sistemas Operativos, Máquinas Virtuales

Librerías de Servicios, Componentes SW, Clases

Drivers

- **Librería SW**:
  - Sistema Software Compatible, SO, MV

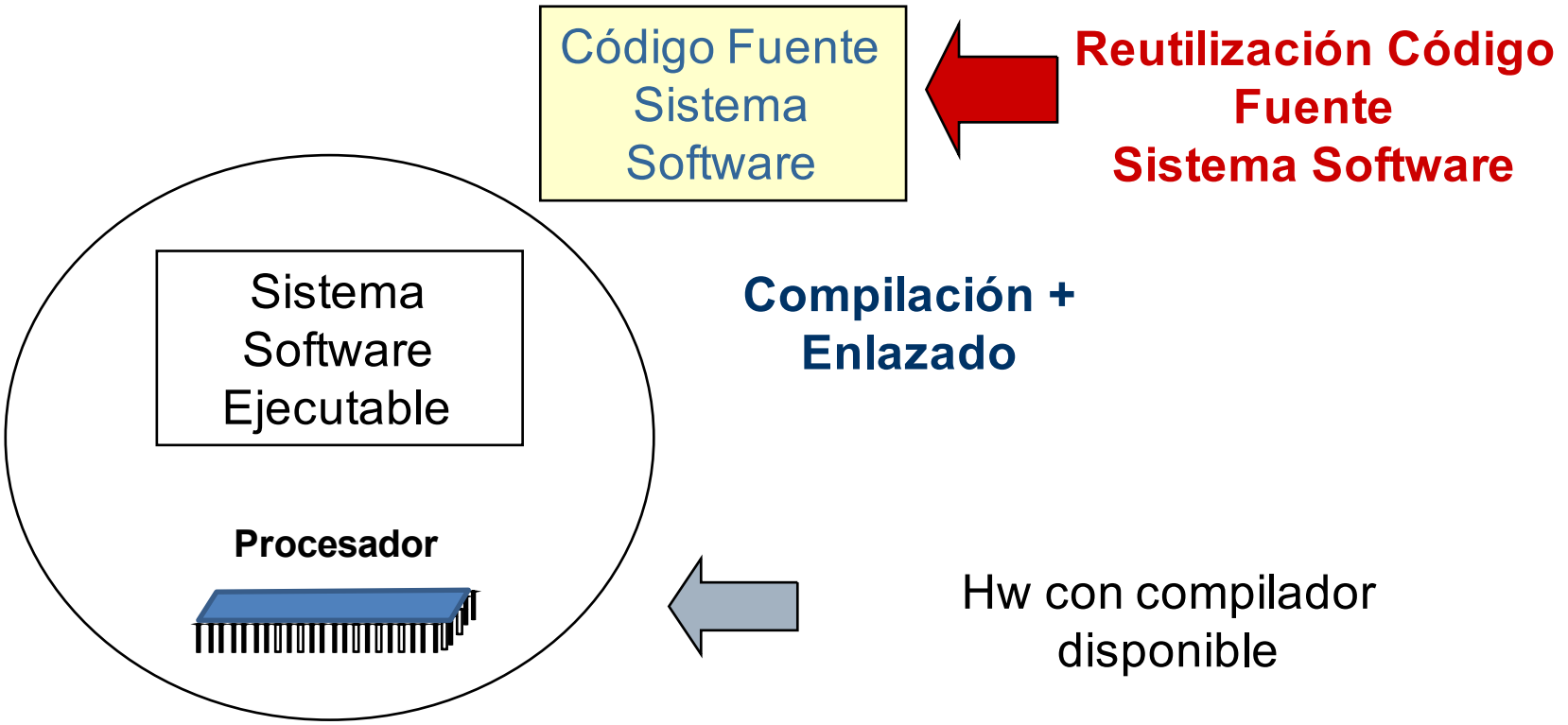


Reutilización del Sistema Software

Hw compatible con el conjunto de instrucciones

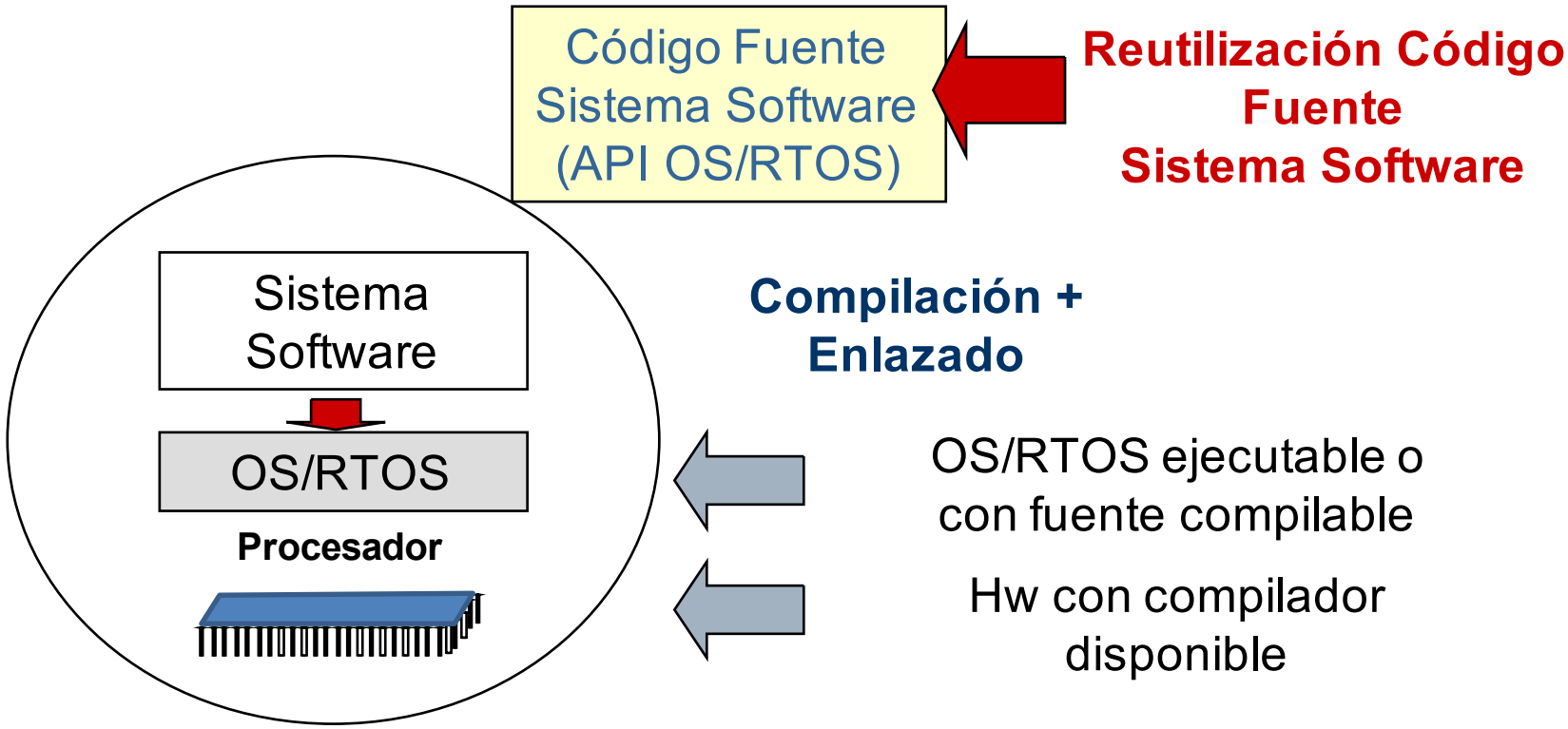
Técnicas de Diseño y Validación de SE

- **Librería SW**:
  - Sistema Software Compatible, SO, MV



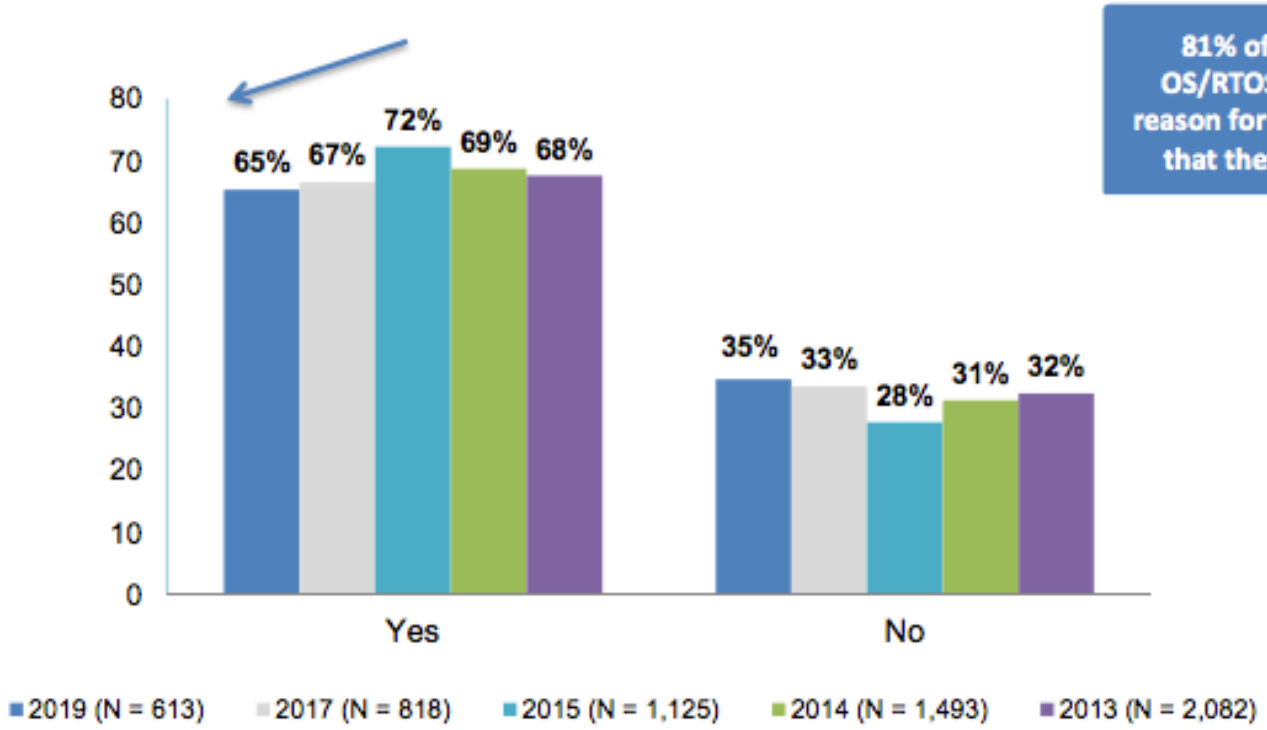
Técnicas de Diseño y Validación de SE

- **Librería SW**:
  - Sistema Software Compatible, SO, MV

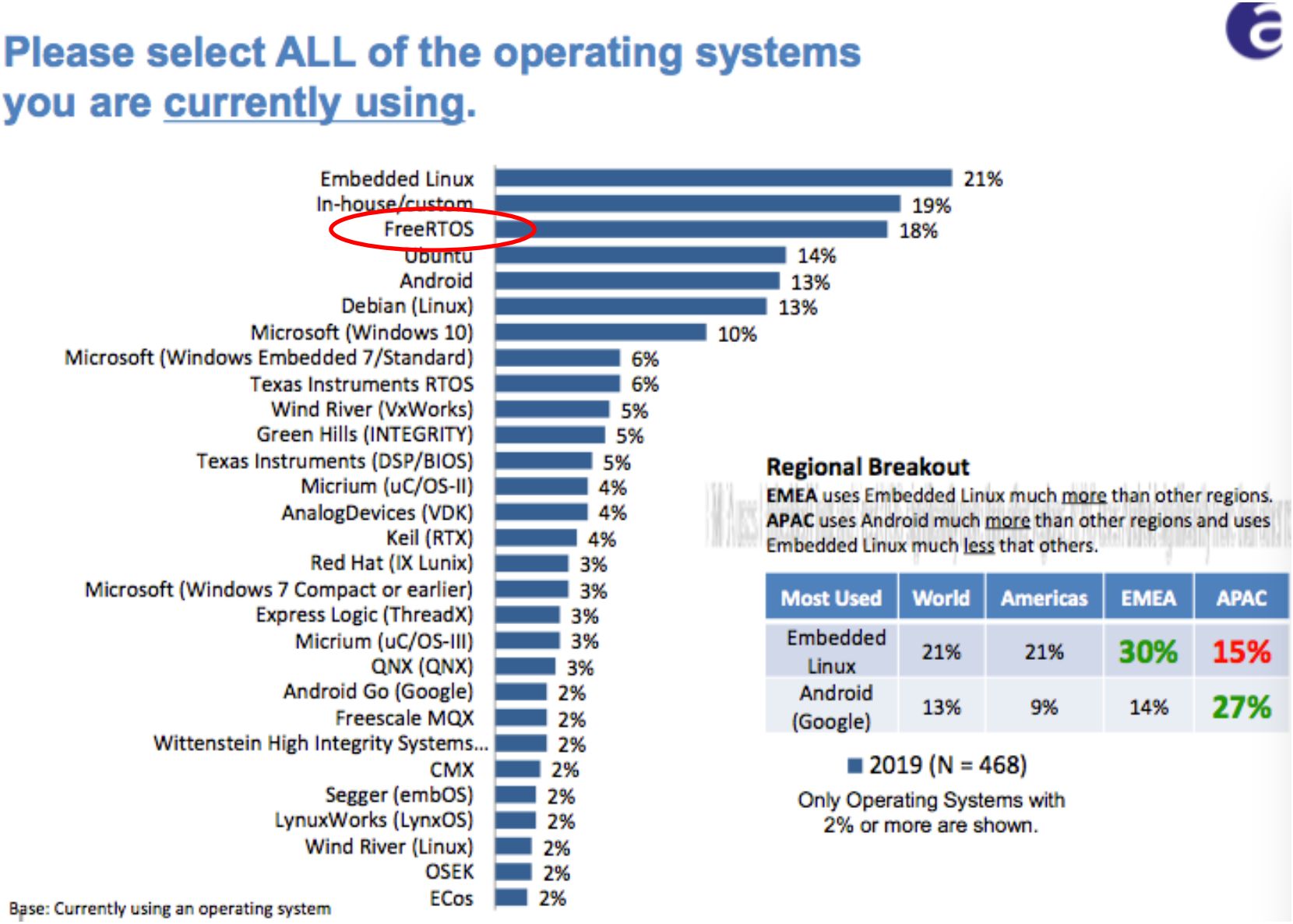


Técnicas de Diseño y Validación de SE

Does your current embedded project use an operating system, RTOS, kernel, software executive, or scheduler of any kind?

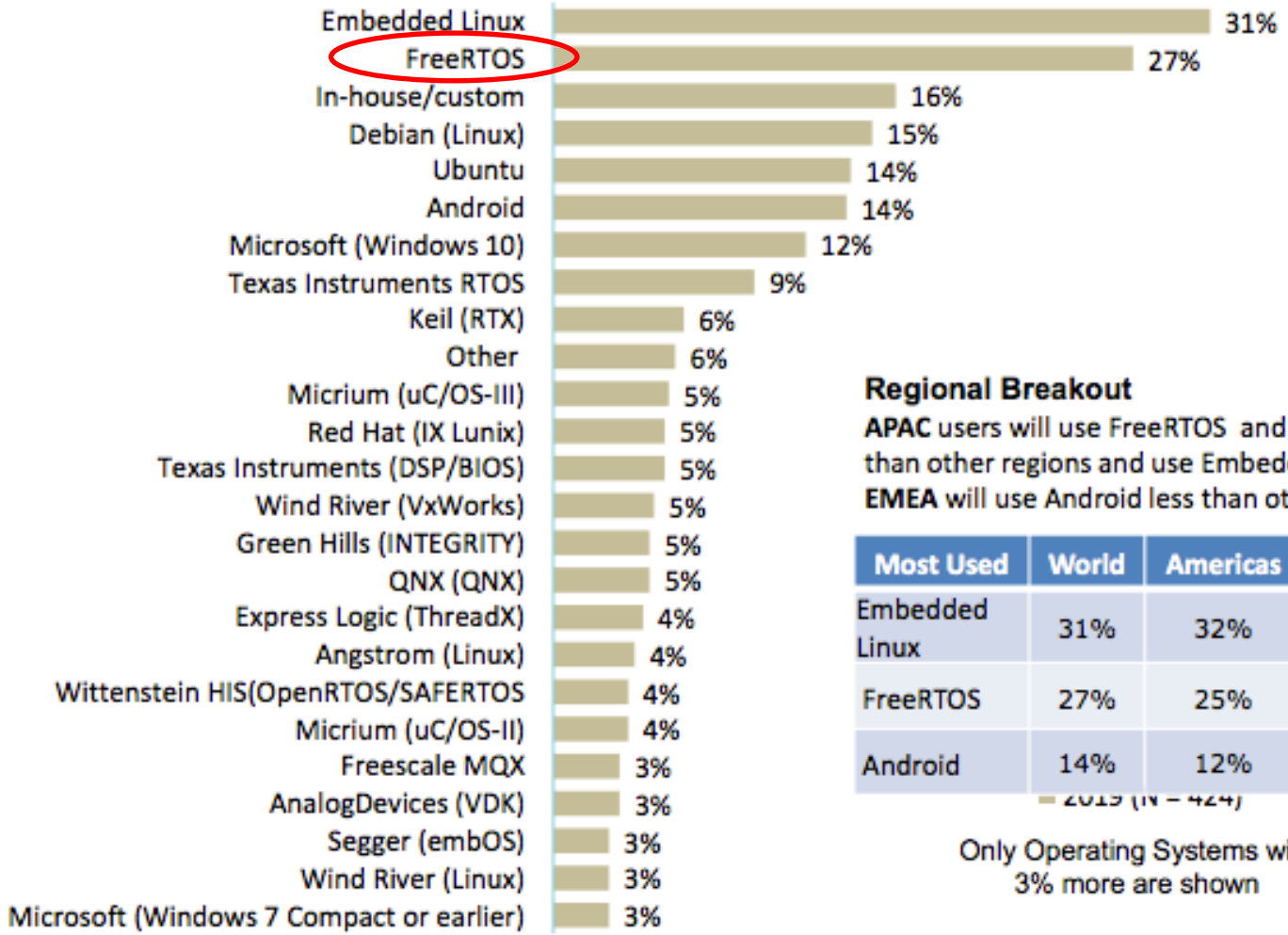


Técnicas de Diseño y Validación de SE



Técnicas de Diseño y Validación de SE

Please select ALL of the operating systems you are considering using in the next 12 months.



Regional Breakout

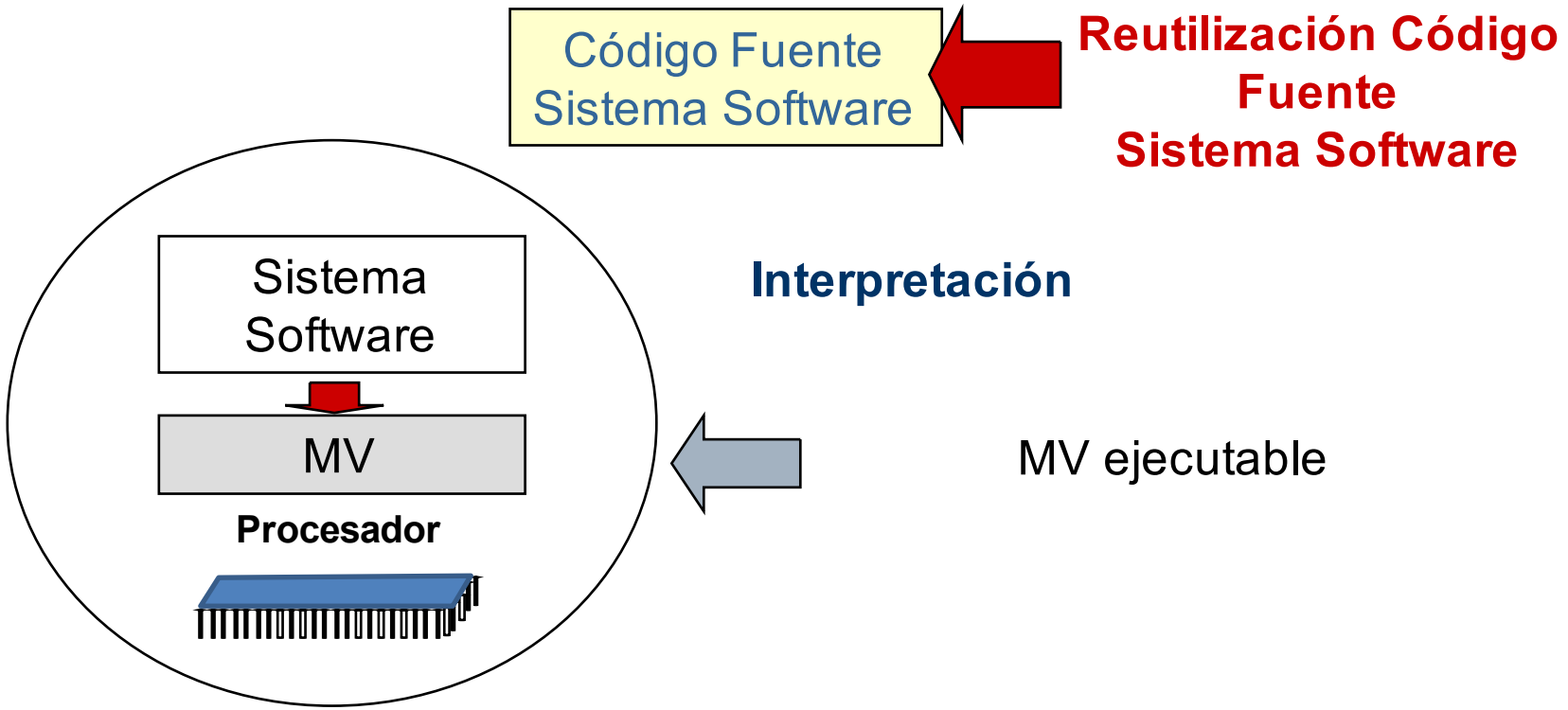
APAC users will use FreeRTOS and Android much more than other regions and use Embedded Linux much less.
EMEA will use Android less than other regions.

Most Used	World	Americas	EMEA	APAC
Embedded Linux	31%	32%	31%	**26%**
FreeRTOS	27%	25%	24%	**37%**
Android	14%	12%	**10%**	**26%**

2019 (N = 424)

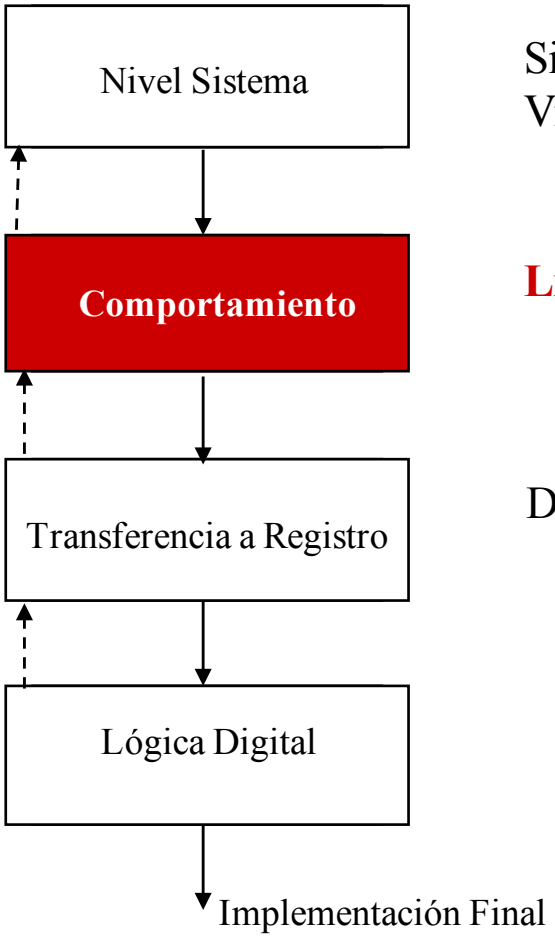
Only Operating Systems with 3% more are shown

- **Librería SW**:
  - Sistema Software Compatible, SO, MV



Técnicas de Diseño y Validación de SE

Librerías/IP Cores SW:

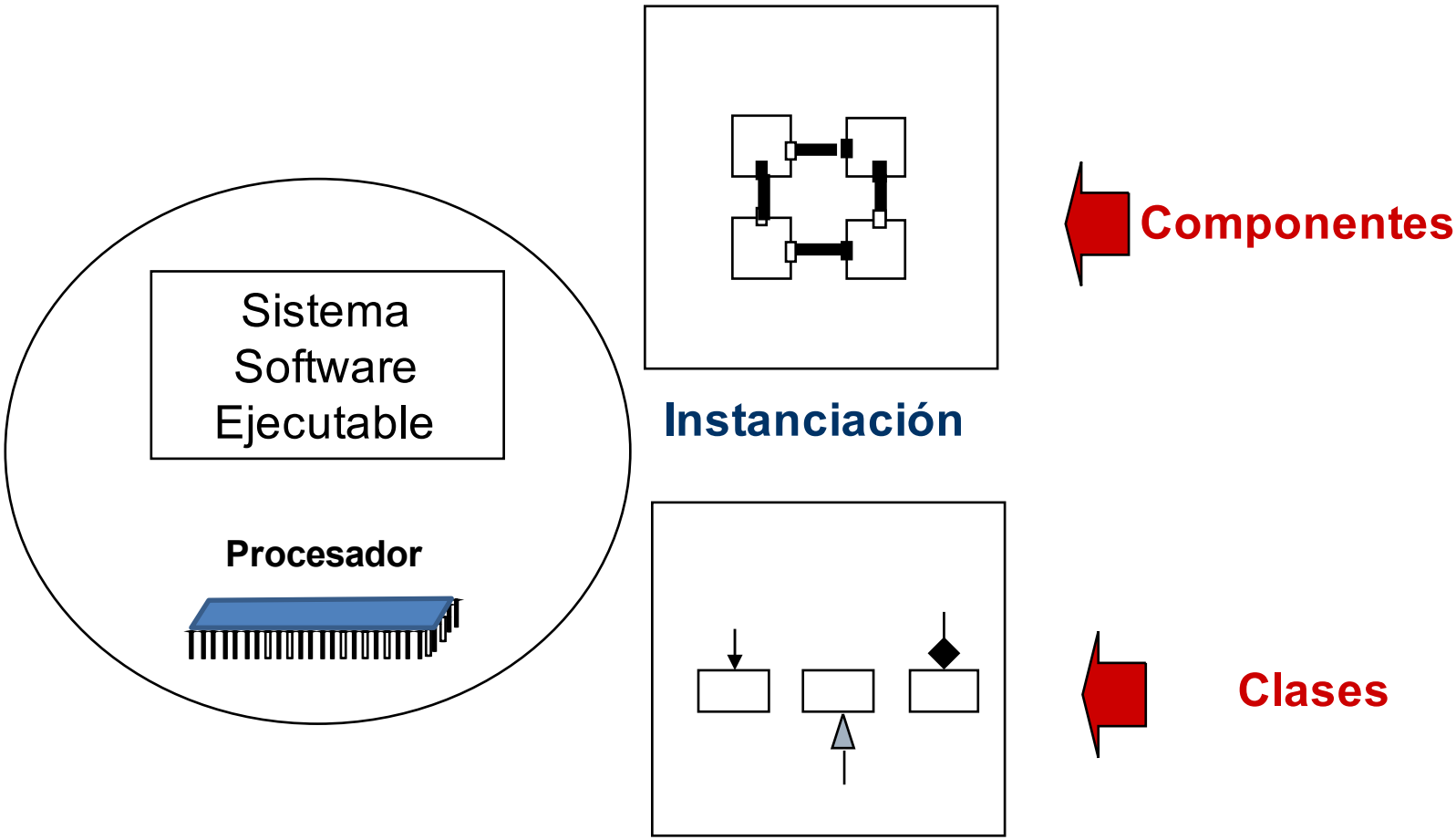


Sistema Software Compatible, Sistemas Operativos, Máquinas Virtuales

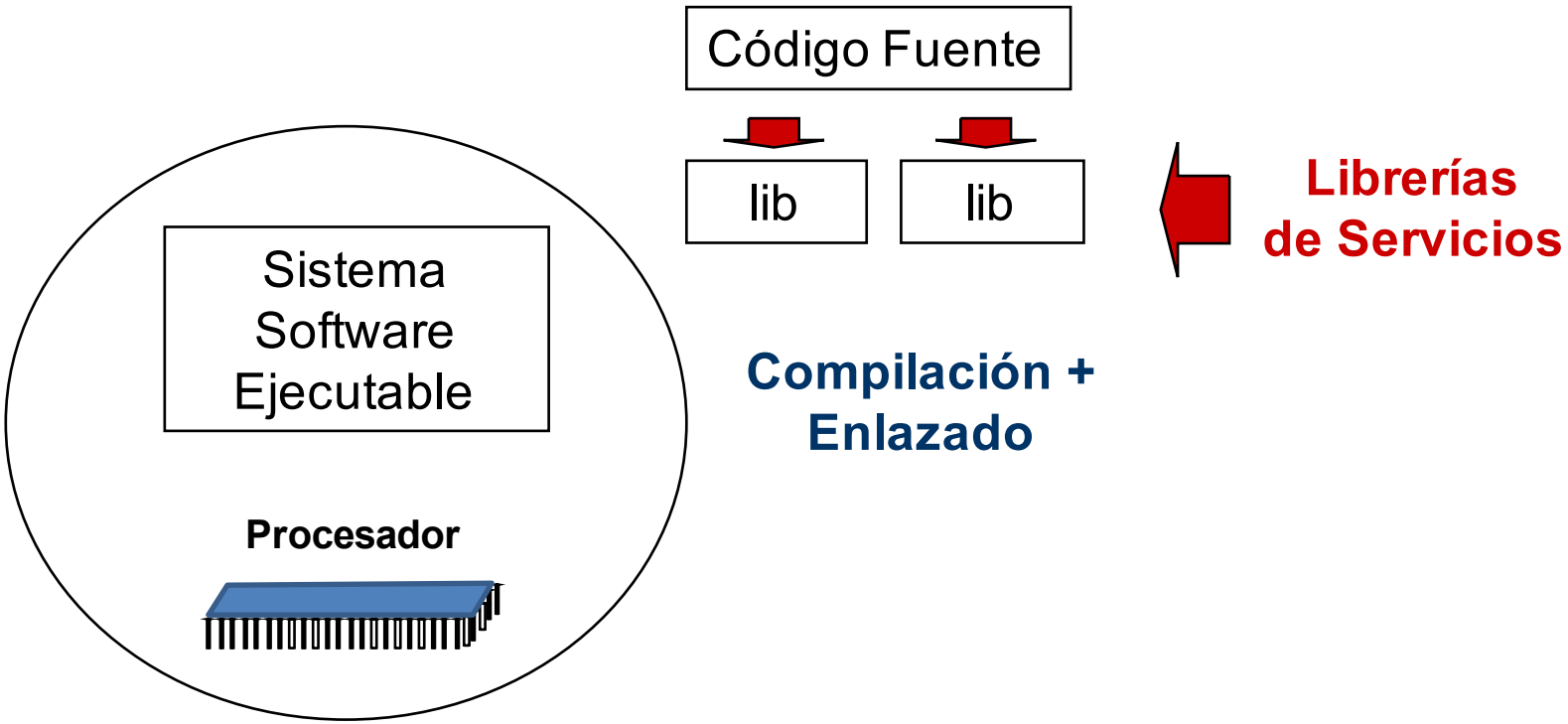
Librerías de Servicios, Componentes SW, Clases

Drivers

- **Librería SW**:
  - Librerías de Servicios, Componentes SW, Clases

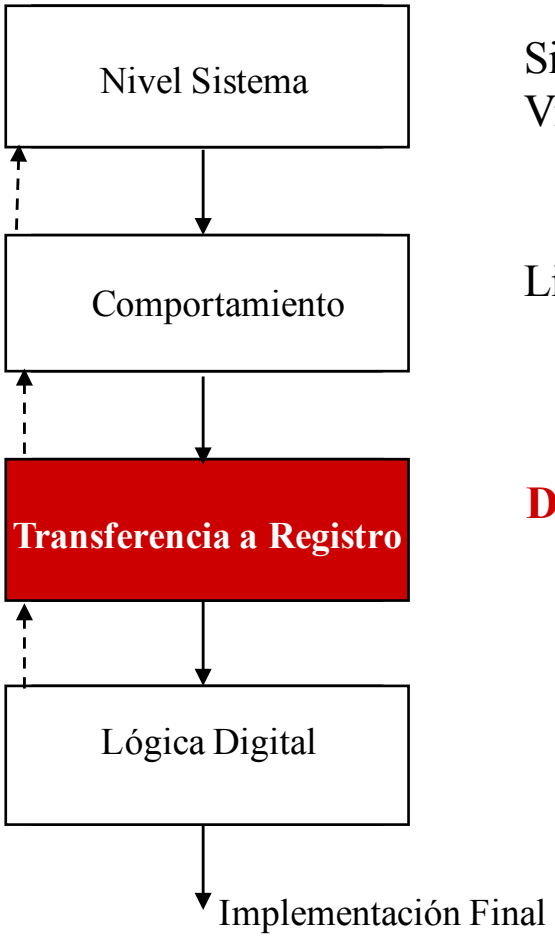


- **Librería SW**:
  - Librerías de Servicios, Componentes SW, Clases



Técnicas de Diseño y Validación de SE

Librerías/IP Cores SW:



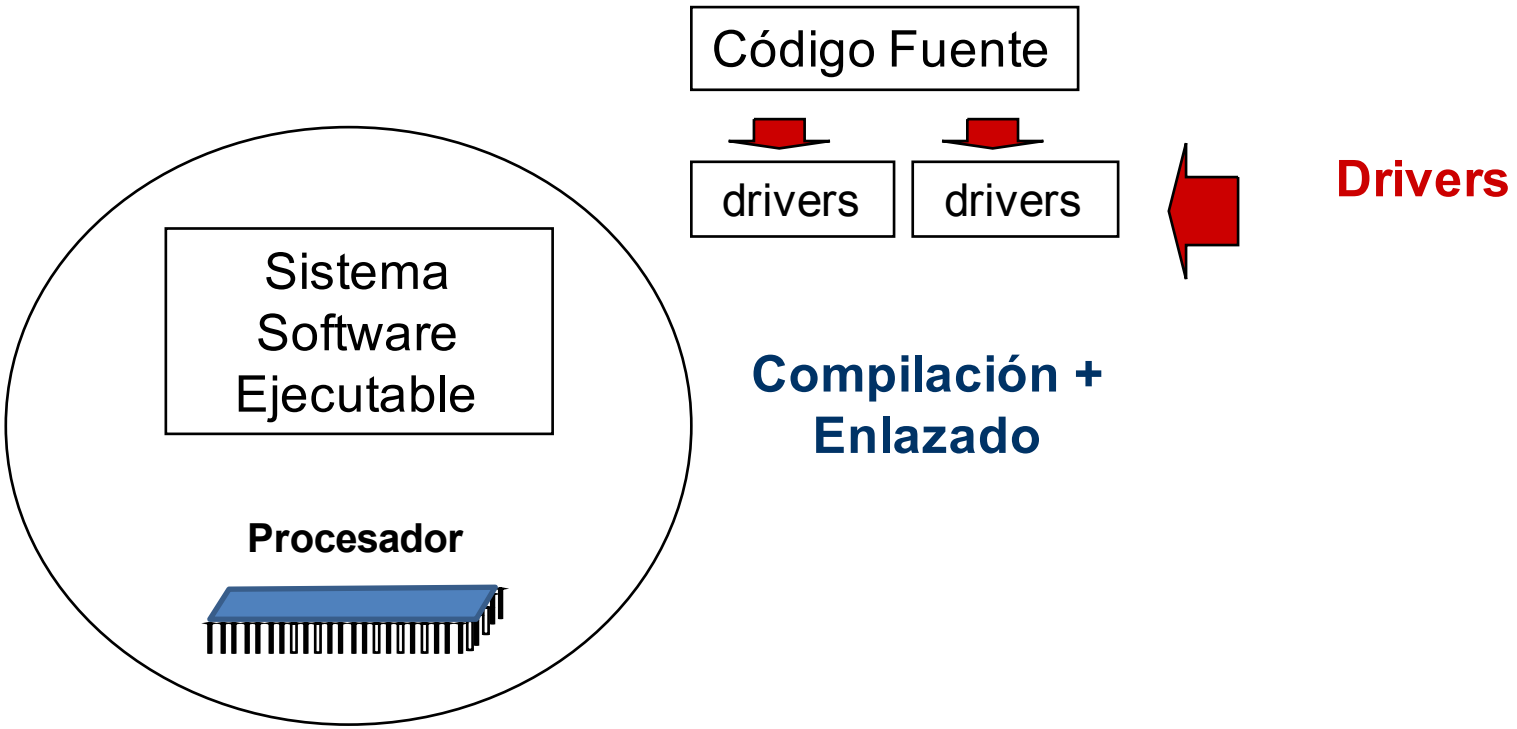
Sistema Software Compatible, Sistemas Operativos, Máquinas Virtuales

Librerías de Servicios, Componentes SW, Clases

Drivers

Técnicas de Diseño y Validación de SE

- **Librería SW**:
  - Drivers

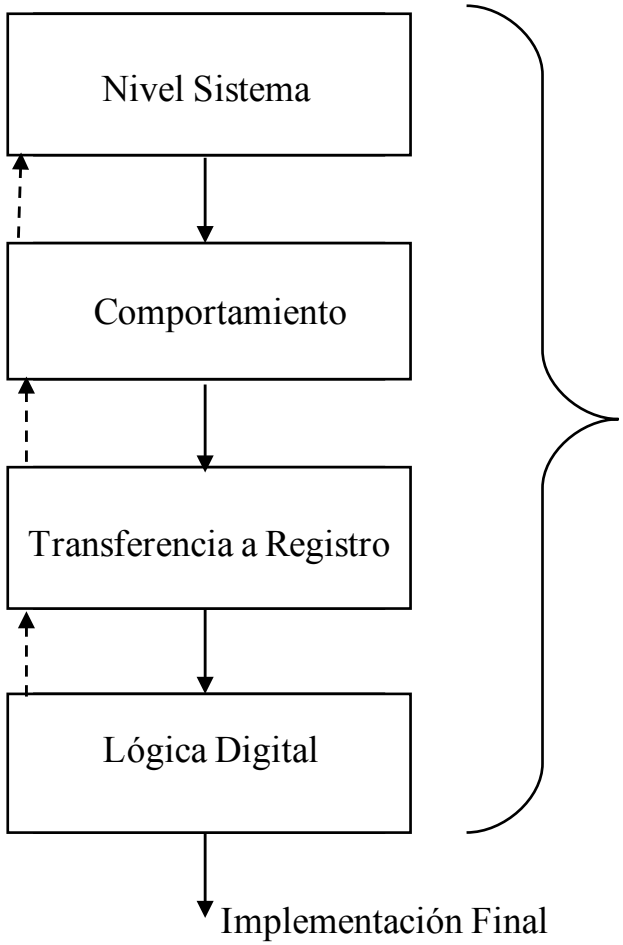


Técnicas de Diseño y Validación de SE

- Compilación/Síntesis:
- Librerías/IP cores:
- **Test/Validación:**

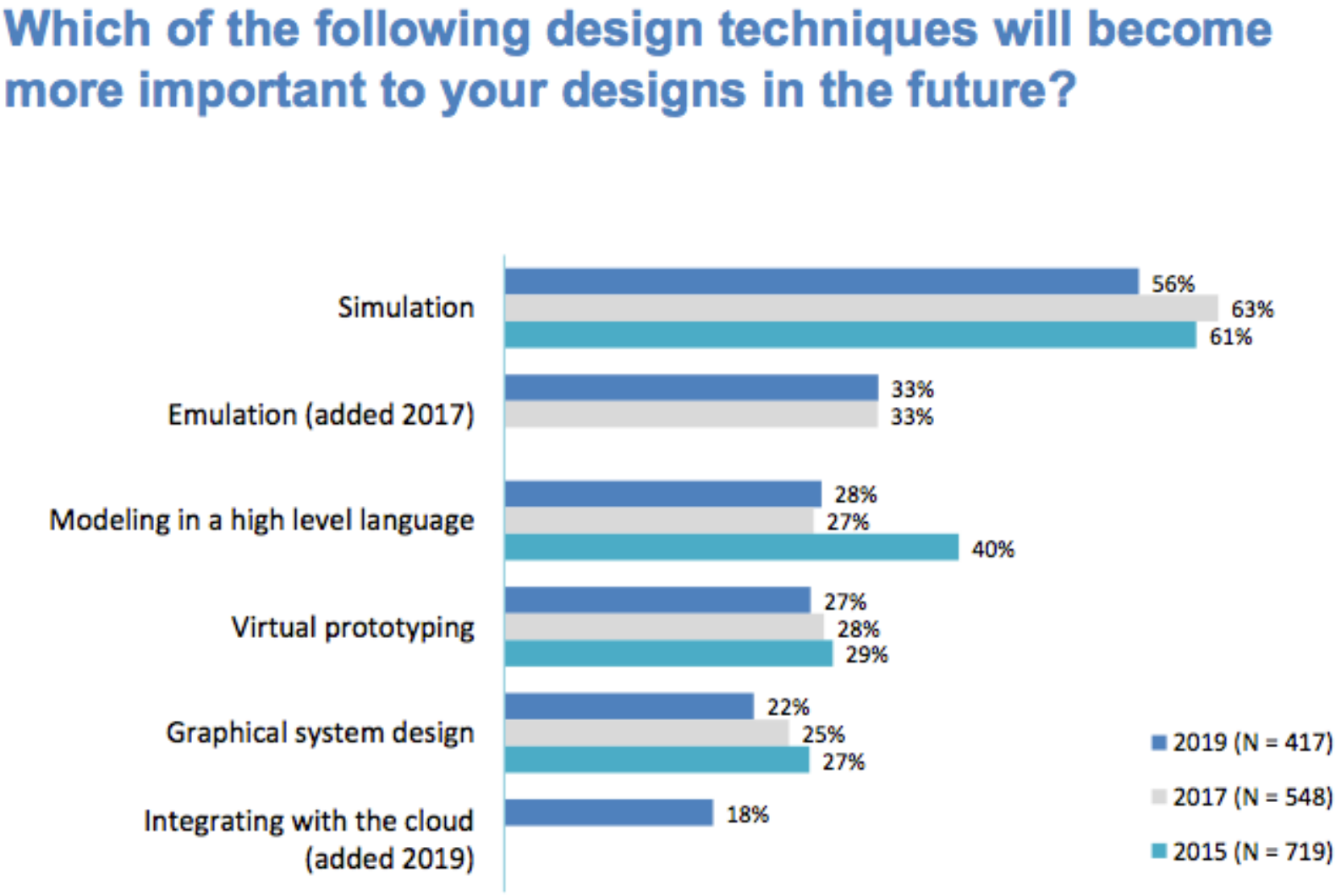
Técnicas de Diseño y Validación de SE

Test/Validación HW y SW



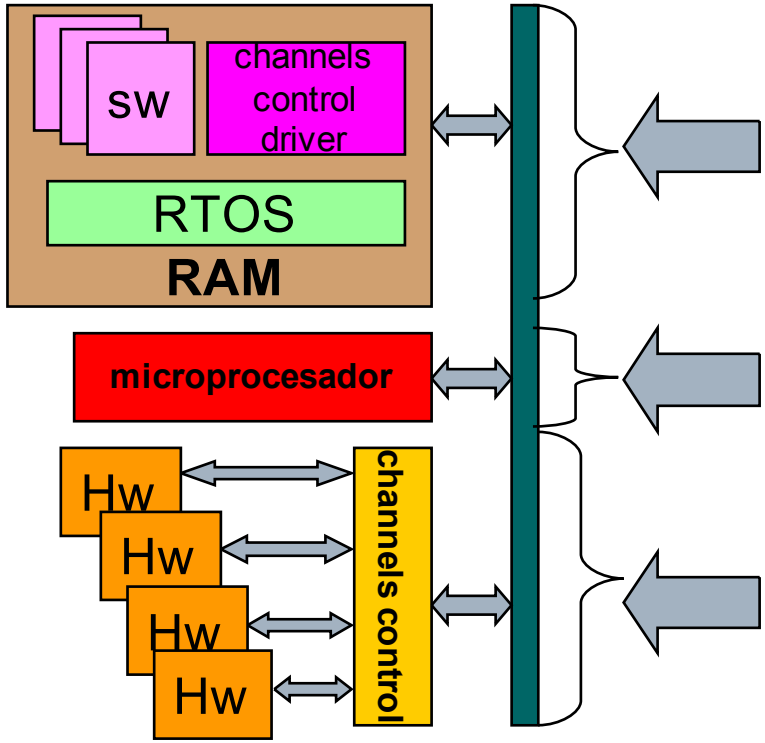
Simulación
Cosimulación Hw-Sw
Depuración
Trazado

Técnicas de Diseño y Validación de SE



Técnicas de Diseño y Validación de SE

- **Test/Validación HW y SW**:
  - Cosimulación HW - SW
    - Simulación en SystemC, del modelo completo que incluye entidades hardware y software.



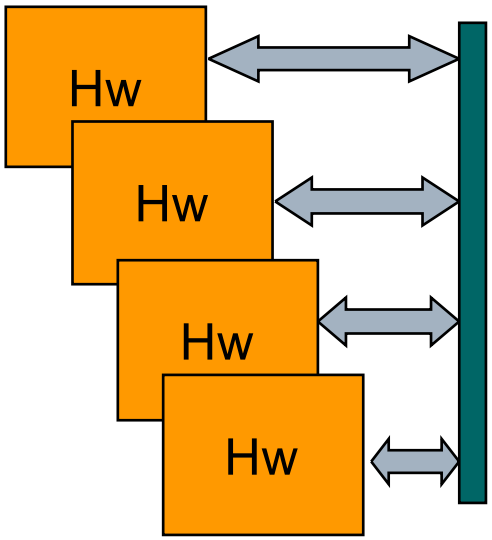
Memoria inicializada (entidades Sw, RTOS y drivers compilados enlazados y formateados en hex/bin)

System C Wrapper del ISS o IPCore VHDL -> SystemC

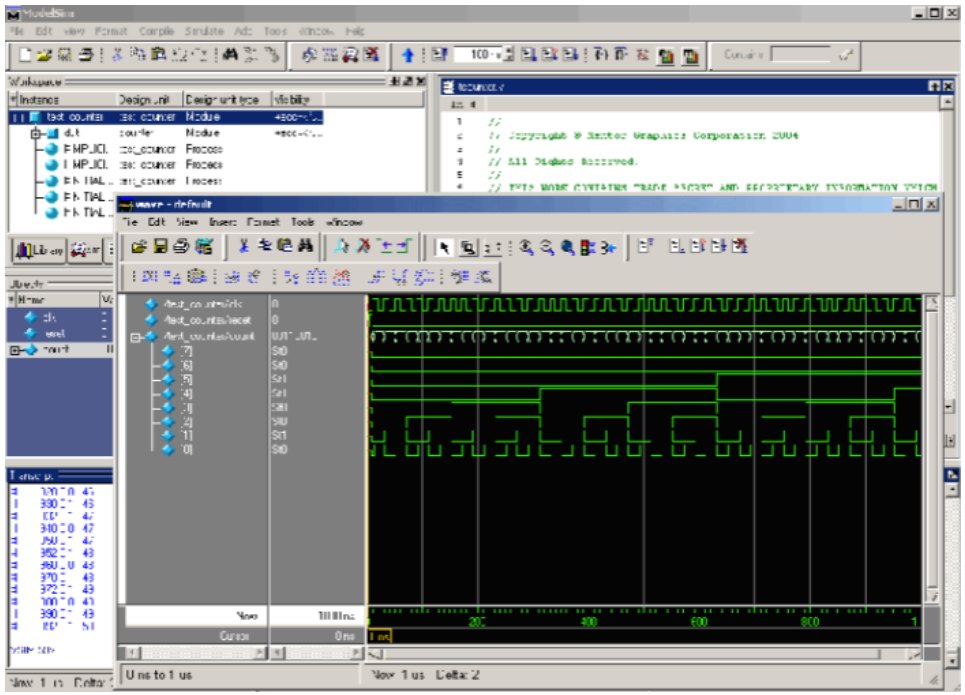
Definidos en System C o VHDL->SystemC

Técnicas de Diseño y Validación de SE

- **Test/Validación HW** y SW:
  - Simulación Funcional Modelo HW
    - Simulación en HDL (VHDL, Verilog, SystemC, ...) de la respuesta funcional de las entidades hardware.

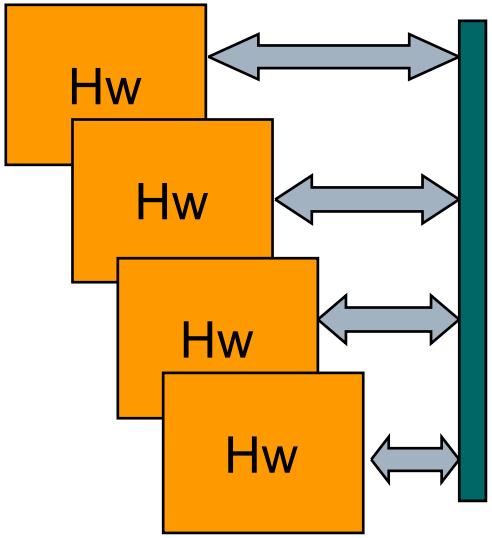


¡Especificación en HDL!

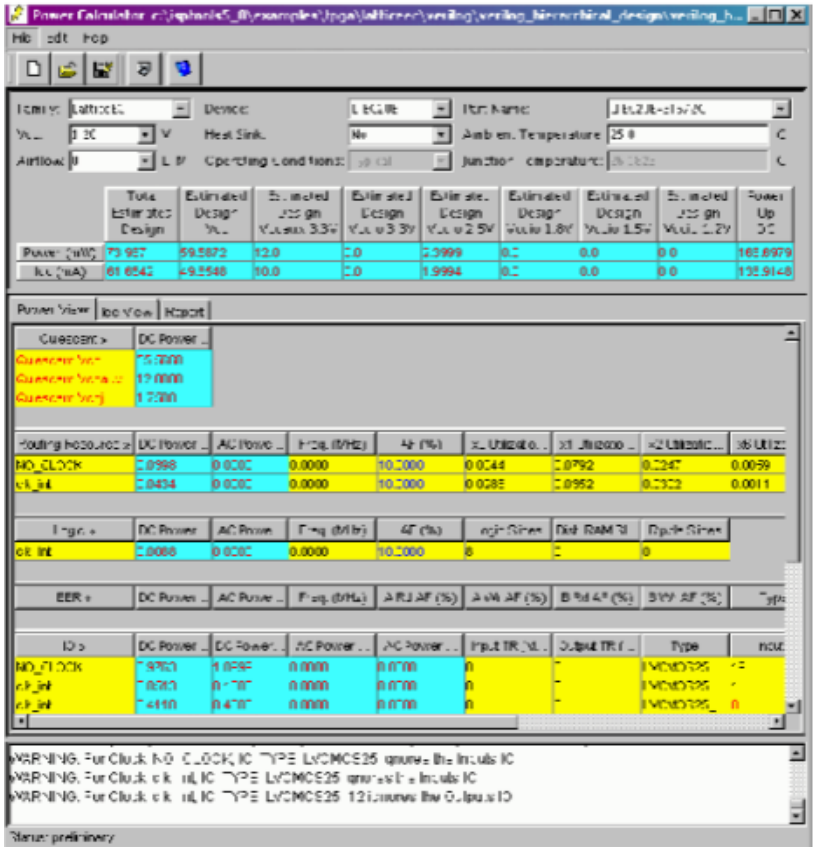


Técnicas de Diseño y Validación de SE

- **Test/Validación HW** y SW:
  - Simulación Consumo Potencia Modelo HW
    - Simulación del consumo de potencia
      - Parametrización de temperatura, condiciones de operación, ventilación, disipación y frecuencia de trabajo

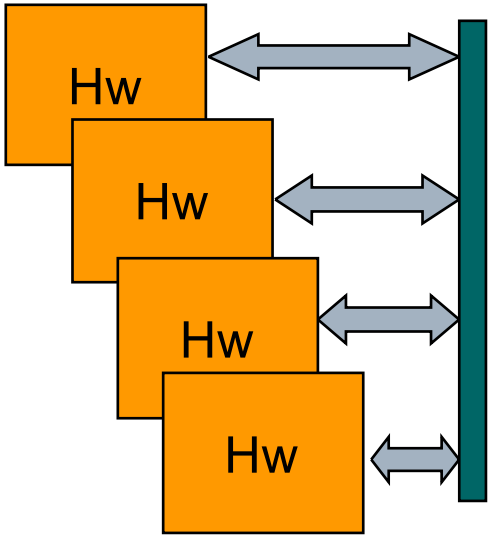


¡Especificación en HDL!

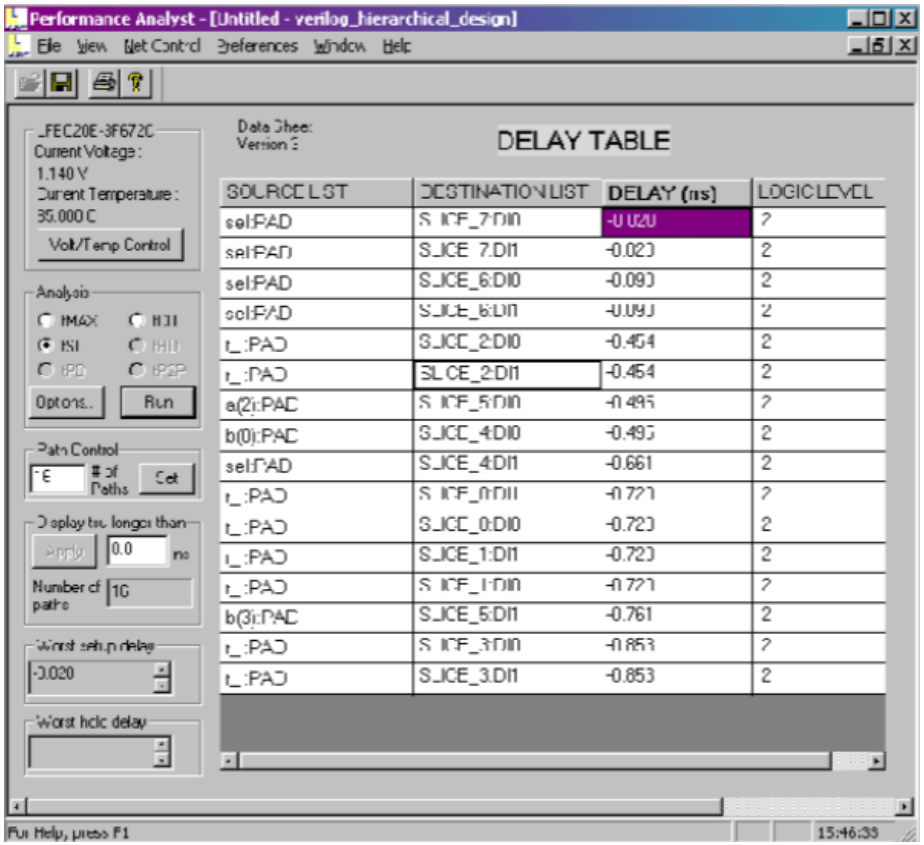


Técnicas de Diseño y Validación de SE

- **Test/Validación**, **HW** y SW:
  - Simulación Rendimiento Modelo HW
    - Simulación del retardo en el peor caso
      - Detección de caminos críticos
      - Detección de cuellos de botellla

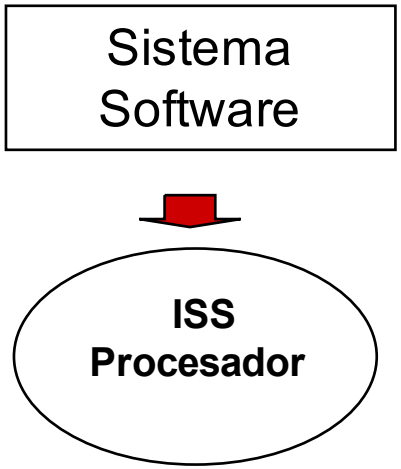


¡Especificación en HDL!



Técnicas de Diseño y Validación de SE

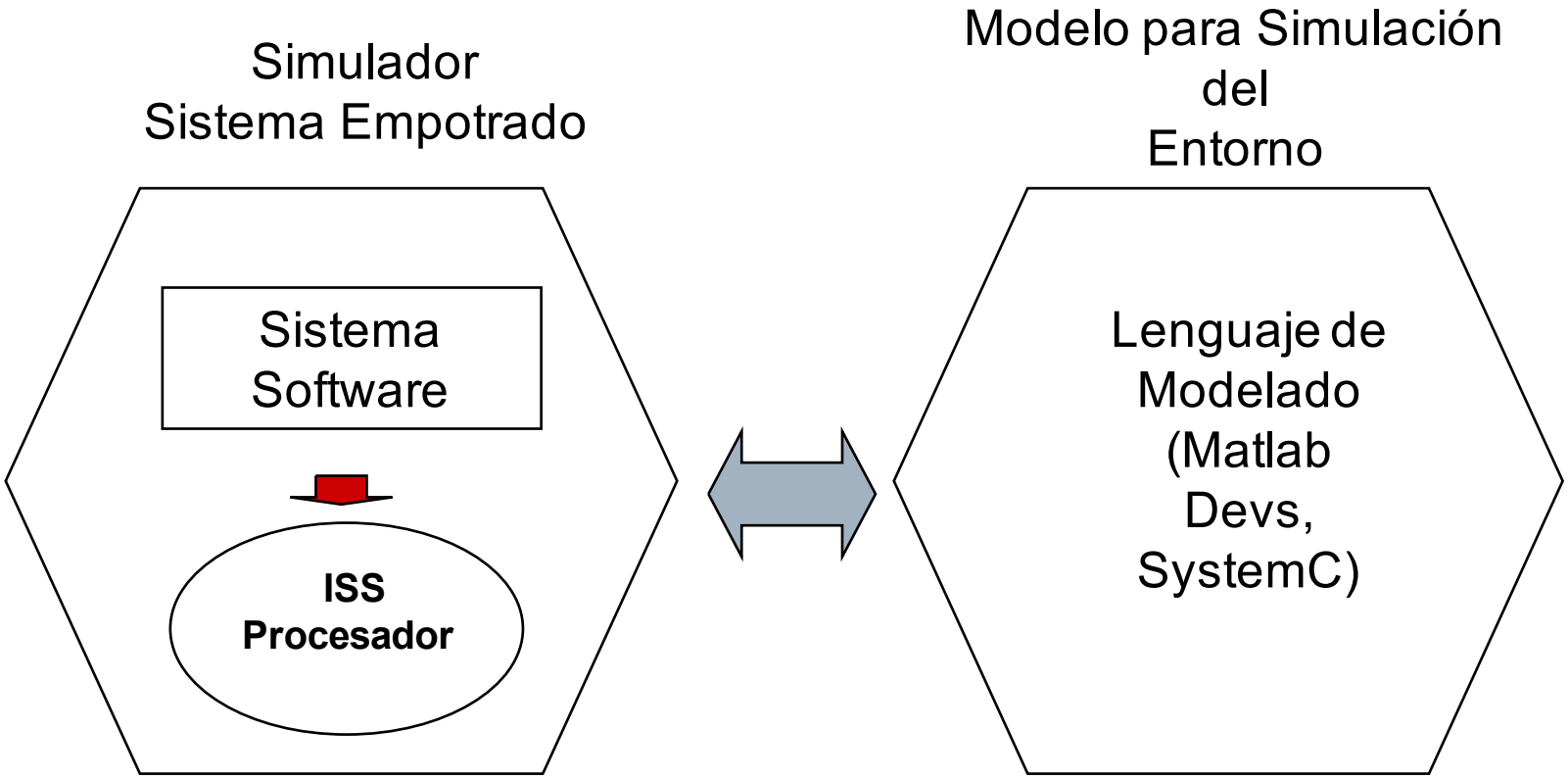
- **Test/Validación HW** y SW:
  - Simulación Sistema SW
    - Simulación funcional



ISS
Instruction Set
Simulator

Técnicas de Diseño y Validación de SE

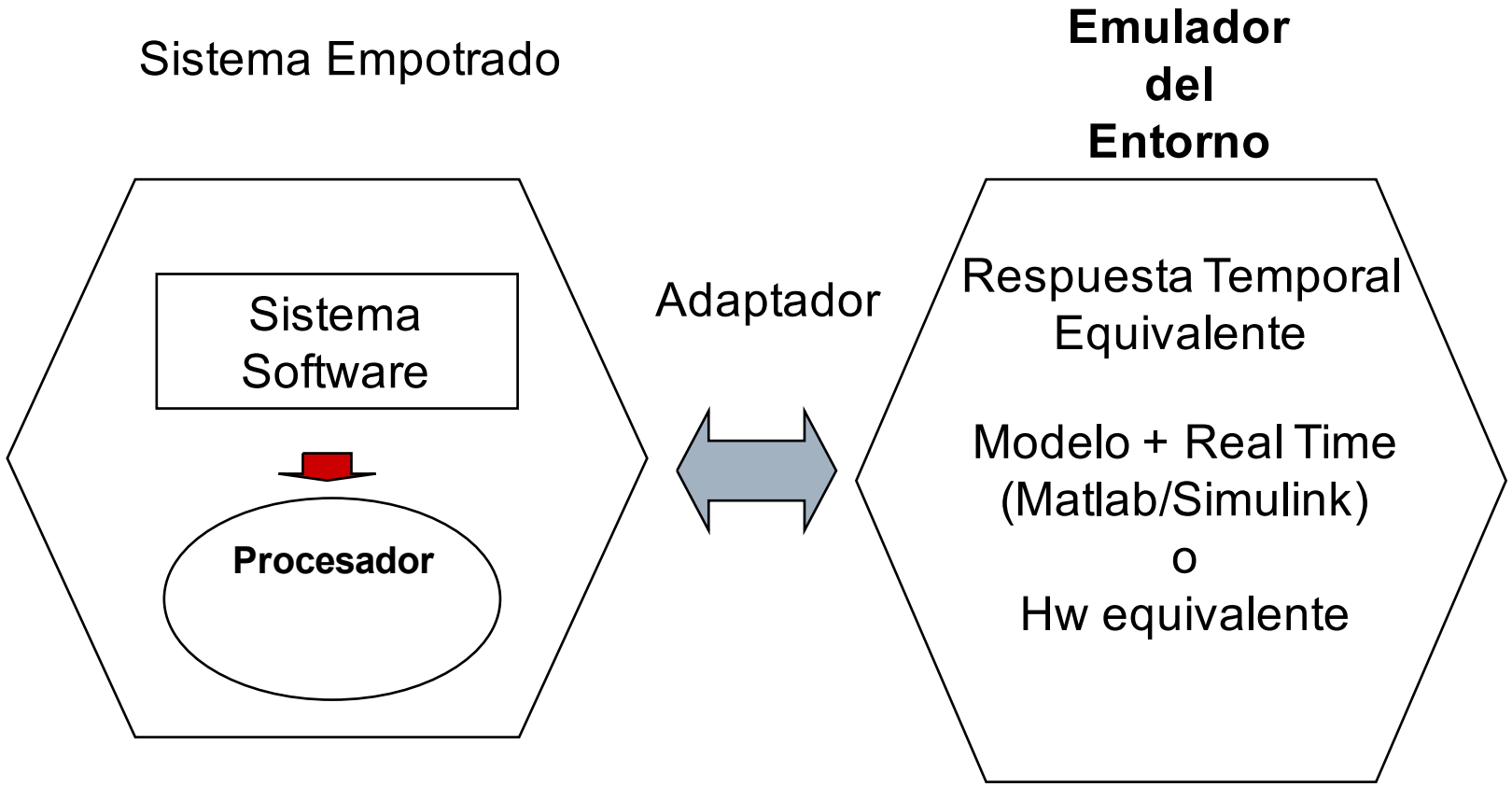
- **Test/Validación HW y SW**:
  - Simulación Sistema + Interacción con el Entorno



¡Comparten Tiempo Simulación!

Técnicas de Diseño y Validación de SE

- **Test/Validación HW y SW**:
  - Sistema (Hw en el lazo) + Interacción con el Entorno



Técnicas de Diseño y Validación de SE

- **Test/Validación HW y SW**:
  - Sistema (Hw en el lazo) + Interacción con el Entorno

Sistema Empotrado

Emulador del Entorno

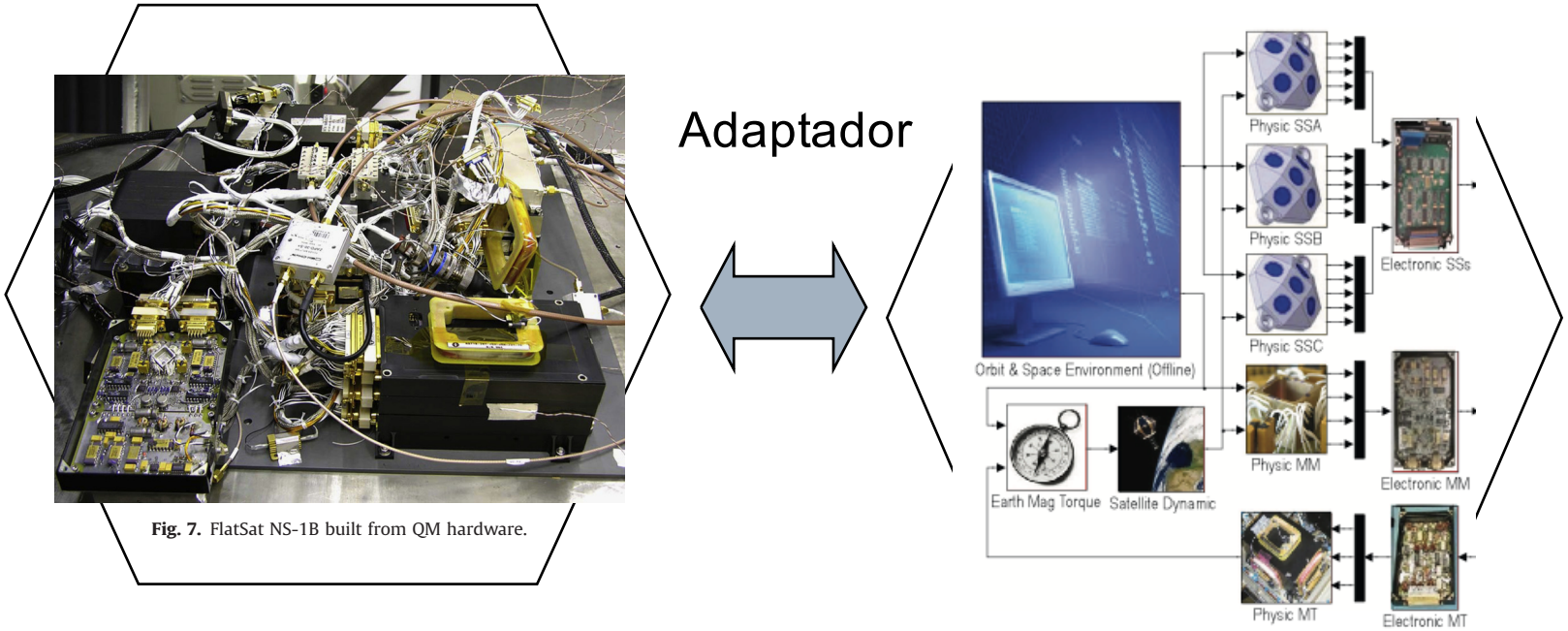


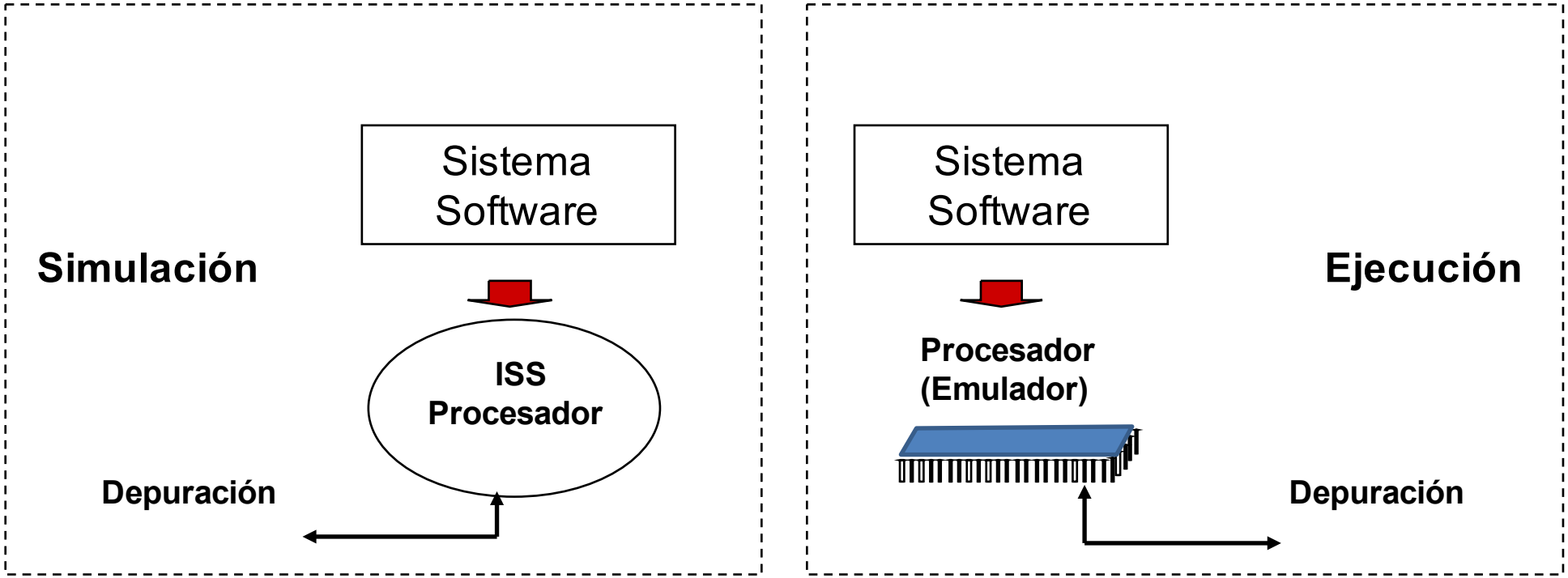
Fig. 7. FlatSat NS-1B built from QM hardware.

Nanosat-1B HW + SW

Emulan Dinámica Satélite + Órbita + Sol + Campo Mag + Sensores + Actuadores.

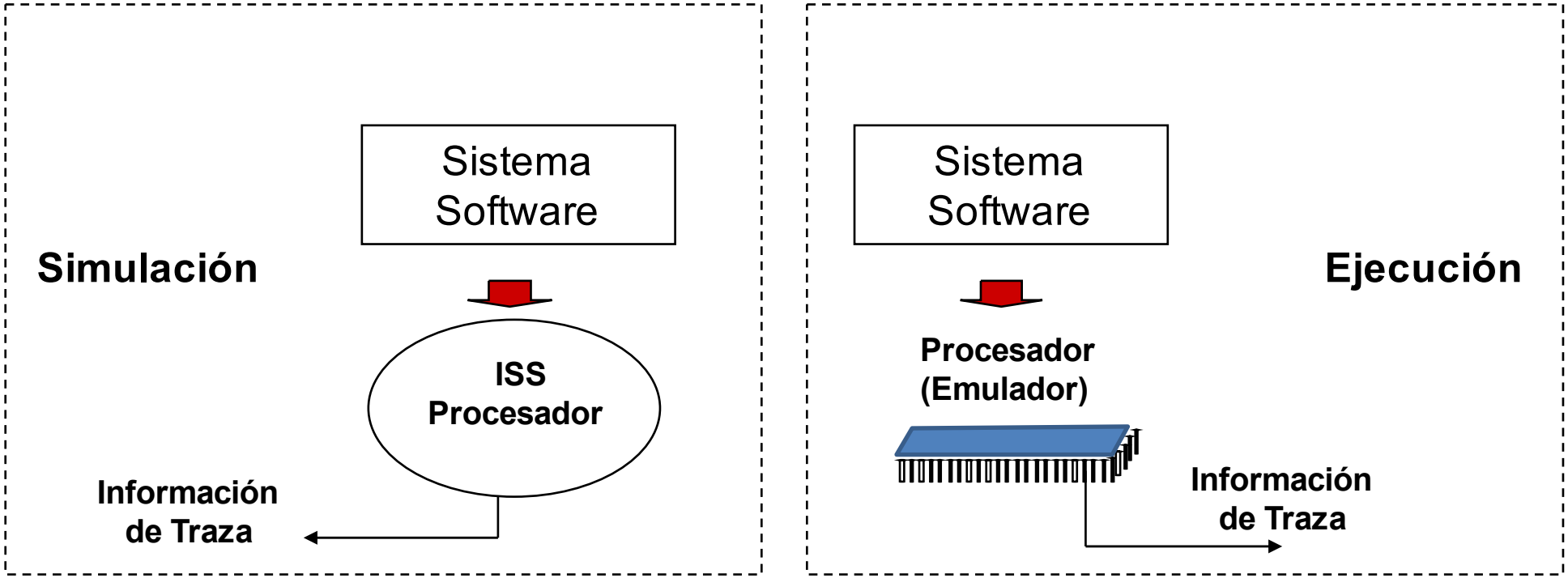
Técnicas de Diseño y Validación de SE

- **Test/Validación** HW y **SW**:
  - Depuración Sistema SW
    - Depuracion Errores
      - La ejecución se detiene en los puntos de ruptura.

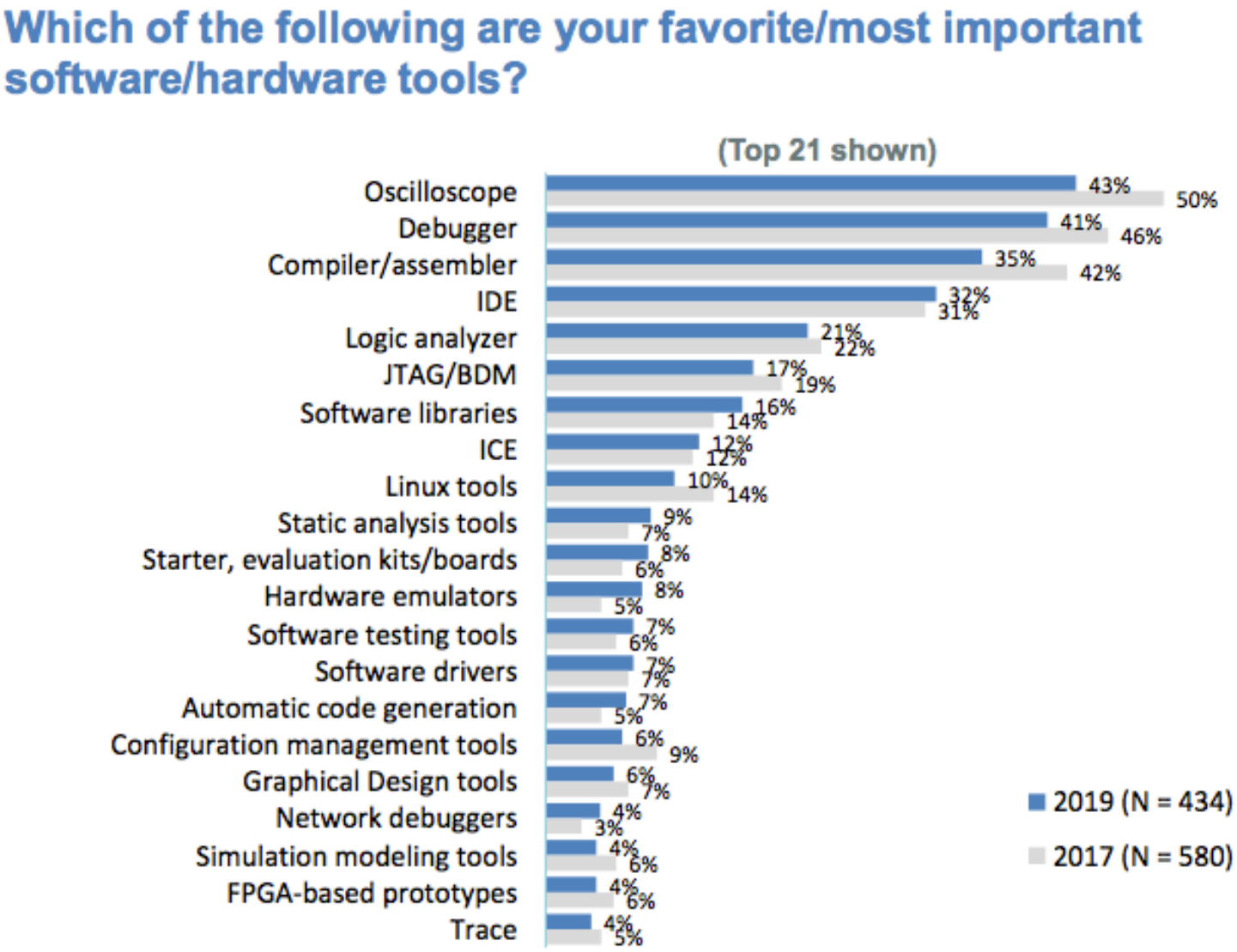


Técnicas de Diseño y Validación de SE

- **Test/Validación** HW y **SW**:
  - Trazado Sistema SW
    - Validación Escenarios
    - Cobertura
    - Profiling
    - La ejecución **NO se detiene. El análisis es off-line**

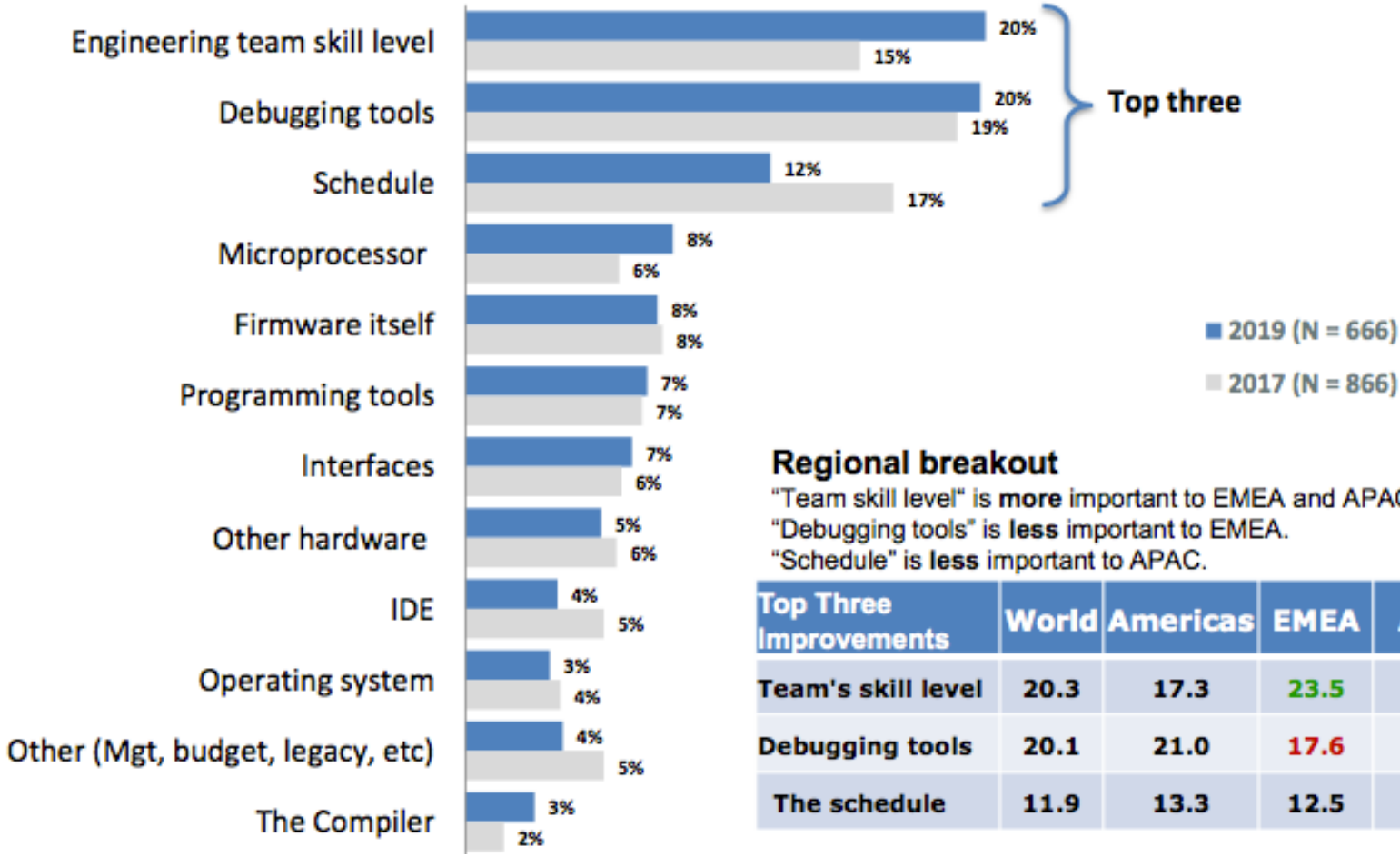


Técnicas de Diseño y Validación de SE



Técnicas de Diseño y Validación de SE

If you could improve one thing about your embedded design activities, what would it be?



Regional breakout

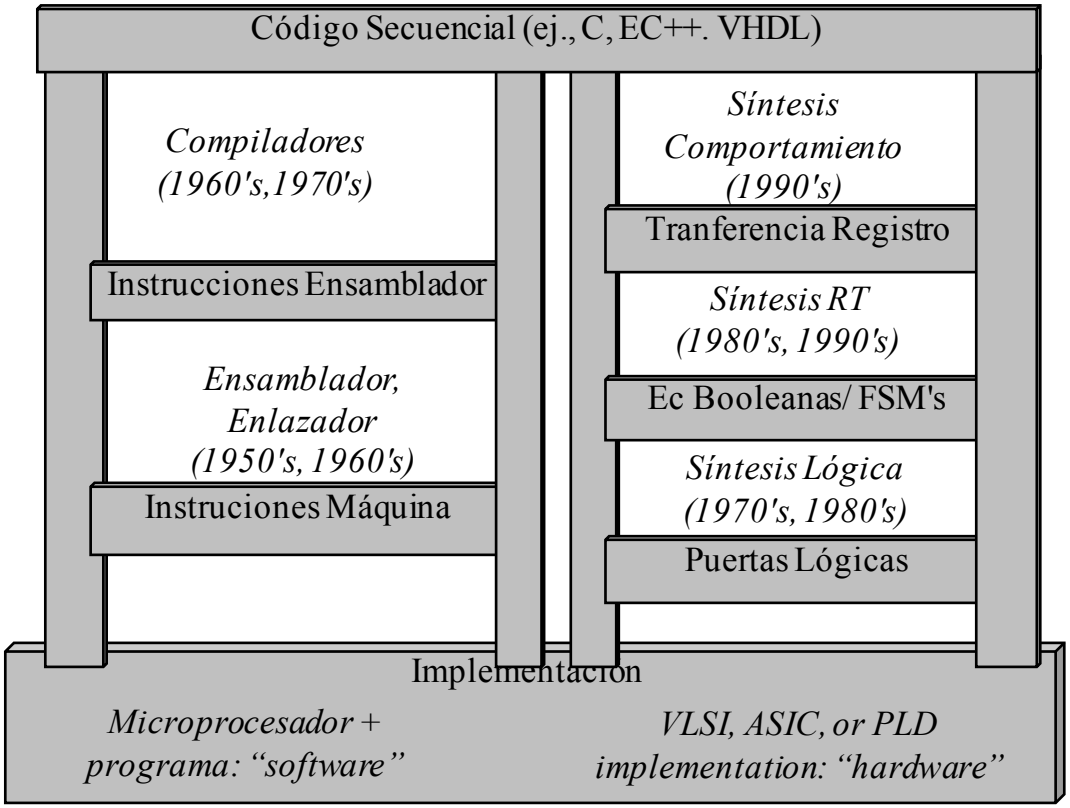
"Team skill level" is **more** important to EMEA and APAC.
"Debugging tools" is **less** important to EMEA.
"Schedule" is **less** important to APAC.

Top Three Improvements	World	Americas	EMEA	APAC
Team's skill level	20.3	17.3	23.5	26.2
Debugging tools	20.1	21.0	17.6	20.0
The schedule	11.9	13.3	12.5	6.9

Técnicas de Diseño y Validación de SE

Técnicas Hw vs Técnicas Sw

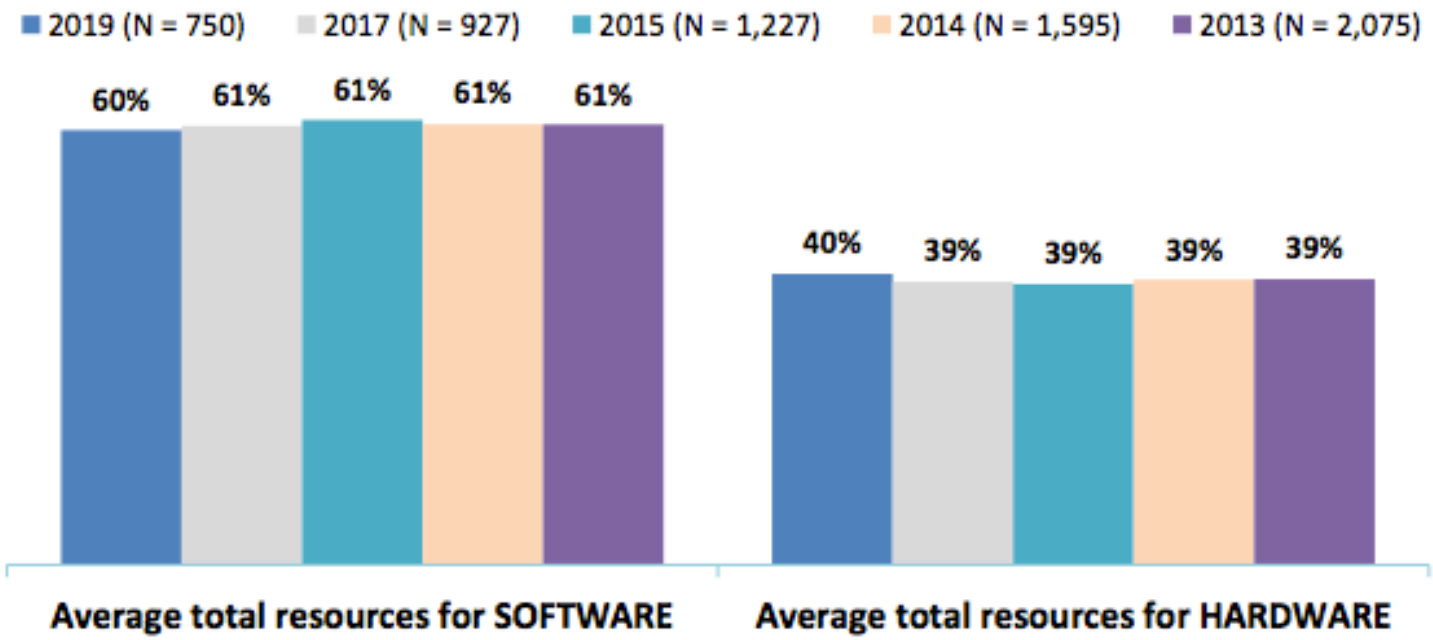
- En el pasado HW y SW empleaban técnicas de diseño muy diferentes
- Las técnicas de síntesis actuales han unificado ambas disciplinas
- => **Codiseño Hw/Sw**



La decisión sobre qué debe implementarse mediante Hw o Sw se basa en los resultados esperados en las métricas de diseño: rendimiento, consumo, tamaño, coste NRE, flexibilidad, ...;
No hay diferencia entre lo que el hardware y el software pueden implementar.

Técnicas de Diseño y Validación de SE

What is your development team's ratio of total resources (including time/dollars/manpower) spent on software vs. hardware for your embedded projects?



In 2019, respondents averaged working on 2.1 projects at the same time.
In 2017, respondents averaged working on 2.1 projects at the same time.
In 2015, respondents averaged working on 2.1 projects at the same time.
In 2014, respondents averaged working on 2.0 projects at the same time.