Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

Programa de Licenciatura en Ingeniería en Computadores

Curso: Introducción a los sistemas embebidos

Examen

Emanuel Esquivel **2016133597**
Roger Valderrama **2017113167**

Profesor:

Luis Barboza Artavia

Fecha: Junio 10, 2021

Respuesta corta

1. **Explique 3 características de los sistemas embebidos. (3 pts)**

    - Bajo consumo de potencia ya que estos están prediseñados para la gestión de aplicaciones específicas lo que en su interior se componen de partes independientes.
    - Bajo costo, ya que muchos de los sistemas embebidos están hechos a la medida por lo que sus partes internas están justas y necesarias
    - La principal característica es que están diseñados para realizar una función específica, por lo que por eso se comparan con las características anteriores.

2. **Explique la utilidad de un toolchain. Un toolchain muy conocido es el GNU, describa tres componentes contenidos en GNU Toolchain. (3 pts)**

    Un toolchain es un conjunto o paquete de herramientas informáticas las cuales se utilizan para la construcción de programas informáticos.

    Tres componentes los cuales están presentes en GNU toolchain son los siguientes:

    1. GNU Make: Este componente nos brinda la facilidad de automatización de la estructura y compilación de programas, generalmente C/C++
    2. GNU Compiler: Principal colección de compiladores para varios lenguajes (GCC)
    3. GNU Debugger: Es un depurador que funciona de manera interactiva (GDB)

3. **Explique dos diferencias entre un microcontrolador y un microprocesador. (2 pts)**

    Los microcontroladores son los encargados de cumplir tareas puntuales, como solo controlar I/O, las cuales se pueden encontrar en interfaces conectadas a un ordenador, a diferencia los microprocesadores los cuales si se requiere que tenga una alta capacidad de cómputo ya que normalmente este forma parte de sistemas que controlar distintos periféricos, por lo que tiene que procesar mucha información.

Del mismo modo, los microprocesadores requieren de memorias RAM y ROM externas pero los microcontroladores las incluyen en un solo circuito integrado. Esto permite que los microprocesadores son circuitos más “abiertos” a ser adaptados y de ser necesario escalados con más memoria siempre que el bus lo permita, sin embargo, los microcontroladores no ofrecen esa ventaja, por lo que, destacan en tareas específicas con un costo económico mucho menor a los microprocesadores.

4. **Explique de manera detallada la diferencia entre un emulador y un simulador. (2 pts)**

Un emulador es un *software* que permite ejecutar programas o videojuegos en una plataforma (sea una arquitectura de hardware o un sistema operativo) diferente de aquella para la cual fueron escritos originalmente. A diferencia de un simulador, que solo trata de reproducir el comportamiento del programa, un emulador trata de modelar de forma precisa el dispositivo de manera que este funcione como si estuviera siendo usado en el aparato original. A diferencia de la simulación y su enfoque con la apariencia de la funcionalidad, la emulación al tener un enfoque más real, siempre retornara la misma salida que el sistema en que se basa con una alta probabilidad a que sea en tiempo real por lo que su uso se centra en sustituir, sin embargo, en las simulaciones pueden operar en la escala de tiempo que sea y retornan el resultado del comportamiento no del sistema, por lo que, destacan para realizar análisis y estudios.

5. **Describa qué es un JTAG y por qué es importante tomarlo en consideración en el diseño de un sistema. (2 pts)**

JTAG es una interfaz de bus estándar de la industria para pruebas de fabricación, como en la prueba de conexión de prueba. Lo que JTAG 'significa' para un desarrollador de software integrado es la interfaz de depuración en el SoC / microprocesador para la depuración externa del software integrado que se ejecuta en el chip. Casi todas las CPU / SoC en el mercado usan la interfaz JTAG no solo para pruebas de fabricación, sino también para soporte de depuración de software. Encontrará una interfaz JTAG en todo, desde CPU de clase de servidor x86 hasta enrutadores WiFi domésticos. Con el hardware de interfaz correcto y el software

adecuado, estas interfaces le otorgarán la capacidad de leer / escribir memoria, establecer puntos de interrupción y código de un solo paso.

La clave, por supuesto, es obtener el hardware y el software correctos. Aunque la interfaz JTAG básica está estandarizada, es posible que las funciones de depuración de software específicas disponibles en un dispositivo no lo estén. JTAG es como TCP / IP. Existe en el medio de la pila de aplicaciones. Necesita la capa física correcta para comunicarse con el equipo, Y necesita el software de aplicación correcto encima. Sin embargo, la fortaleza del ecosistema ARM ha llevado a cierto grado de estandarización allí como por ejemplo OpenOCD.

6. **Al tratar de compilar un código fuente, Fabián tiene el siguiente error** `undefined` **reference to** `'sqrt'`**. Fabian dice que ya incluyó** `math.h`**, pero el error no se corrige. ¿Cómo solucionaría el problema que tiene Fabián? (1 pts).**

Fabian debe de colocar la bandera `-lm`, ya que esta bandera se encarga de enlazar la biblioteca dinámica de `math` la cual tiene lo necesario para compilar el código y utilizar `sqrt`

7. **Explique de manera detallada las formas en que se puede modelar un diseño. (3 pts)**

Un modelo Y-Chart consiste de 3 formas cada una con 4 niveles de abstracción denominados sistema, procesador, lógica y circuito. Para crear el modelo de una forma partiendo de otra se realiza por medio de una síntesis.

- **Comportamiento**: En esta forma el diseño se describe como una caja negra, por lo que las salidas se diseñan a partir de sus entradas. En el nivel de procesador cada elemento será un elemento de procesamiento (PE). Cada PE ejecuta funciones específicas o estándar, los cuales se describen como máquinas de estado finito (FSM), FSM con datos (FSMD), grafos de control y datos (CDFG) y flujo de set de instrucciones (ISF). Por otra parte, a nivel de sistema, los modelos de comportamiento (CDFG) no son ideales, ya que se

deben considerar múltiples procesos, interacción entre SW/HW, concurrencia por lo que se crea una máquina de estados de procesos (PSM).

- **Estructura**: Describe el diseño como un grupo de componentes y conexiones, por lo que, el comportamiento de la caja negra viene dado por dichos componentes y su interacción. A nivel de procesador cada PE se realiza por medio de componentes de RTL, el cual posee algún datapath, controlador y se maneja por ciclo de reloj por lo que se considera la parte temporal. Por otra parte, a nivel de sistema, se tiene un diagrama de bloques de los componentes del sistema creado en base de la descripción del comportamiento. Dicho diagrama contiene PE, elementos de almacenamiento, elementos de comunicación, interfaces y módulos de propiedad intelectual.
- **Diseño físico**: Añade dimensionalidad a la estructura, por lo que se especifica el tamaño de cada componente, puerto, conexión, etc. En esta forma se muestra con mayor detalle la funcionalidad de los componentes.

8. Realice una comparación detallada del diseño de sistemas bottom-up y up-down. Describa por lo menos dos diferencias. (2 pts)

En la metodología bottom-up se inicia desde el nivel más bajo, es decir, a nivel de circuitos básicos de nivel de lógica, para de ahí crear componentes RTL y seguir hasta modelar todo a nivel de sistema. Esta forma posee la gran ventaja de brindar la capacidad de estimar con alta precisión las métricas en cada nivel, además que facilita la distribución de tareas para un desarrollo lo más paralelo posible. No obstante, al poseer todos los componentes en cada nivel es difícil de diseñar un modelo óptimo y crear un "layout" en cada nivel crea más trabajo.

Por otra parte, con la metodología top-bottom, se tiene un modelo de más alto nivel de forma inicial, y se va descomponiendo al bajar cada nivel y llegar al modelo de transistores el cual se hace solo al final. Su fortaleza consiste en poder realizar un diseño más óptimo en cada nivel de abstracción, sin embargo, en esta metodología no se pueden quitar o agregar componentes con facilidad debido a que se modifica en la capa superior y se debe ir descomponiendo en las siguientes capas. Del mismo modo, la estimación de métricas se dificulta en las capas superiores, ya que, el 'layout' se sabe hasta el final. Por lo que, el impacto en las decisiones de diseño no son claras y

se deben realizar anotaciones de métricas “closure” desde los niveles bajos hacia lo altos durante cada iteración del diseño.

9. **Explique cómo el DVFS ayuda en las tareas de encriptación de información y potencia. (2 pts)**

Cuando el procesador no tiene DVFS, durante el tiempo de ejecución genera patrones de energía que pueden ser obtenidos por personas externas malintencionadas, Estos patrones indican a forasteros malintencionados el tipo de procesamiento que se está produciendo en la CPU. Este tipo de ataques se denominan ataques de canal lateral. En este sentido, cuando el procesador está ejecutando algoritmos de cifrado, estos forasteros pueden analizar estos patrones para identificar el algoritmo de cifrado que se pretendía que fuera secreto. Entonces, con el uso de DVFS es posible ocultar estos patrones dados. El controlador DVFS varía el voltaje y las frecuencias según lo requiera el procesador, o de forma aleatoria en algunos casos, lo que hace imposible interceptar estas señales de potencia e identificar el algoritmo de cifrado utilizado.

10. **¿Cual es la diferencia principal entre una biblioteca estática y una dinámica? (1 pt)**

Las bibliotecas estáticas, aunque son reutilizables en varios programas, se bloquean en un programa en tiempo de compilación. En cambio, las bibliotecas dinámicas o compartidas existen como archivos separados fuera del archivo ejecutable.

La desventaja de utilizar una biblioteca estática es que su código está bloqueado en el archivo ejecutable final y no puede modificarse sin volver a compilar. En cambio, una biblioteca dinámica puede modificarse sin necesidad de volver a compilar.

Como las bibliotecas dinámicas viven fuera del archivo ejecutable, el programa solo necesita hacer una copia de los archivos de la biblioteca en tiempo de compilación. Mientras que el uso de una biblioteca estática significa que cada archivo en su programa debe tener su propia copia de los archivos de la biblioteca en tiempo de compilación.

La desventaja de usar una biblioteca dinámica es que un programa es mucho más susceptible de romperse. Si una biblioteca dinámica, por ejemplo, se corrompe, el archivo ejecutable puede dejar de funcionar. Una biblioteca eśtatica, sin embargo, es intocable porque vive dentro del archivo ejecutable.

La ventaja de utilizar una biblioteca dinámica es que varias aplicaciones en ejecución pueden utilizar la misma biblioteca sin necesidad de que cada una tenga su propia copia. Otra ventaja de utilizar bibliotecas estáticas es la velocidad de ejecución en tiempo real. Dado que su código objeto (binario) ya está incluido en el archivo ejecutable, las múltiples llamadas a las funciones pueden ser manejadas mucho más rápidamente que el código de una biblioteca dinámica, que necesita ser llamado desde archivos fuera del ejecutable.

11. **Lucia trata de portear el paquete opencv para raspberry pi zero, utilizando Yocto. Al realizar el comando bitbake rpi basic image, tiene el siguiente error: "Nothing provides opencv". ¿Cómo podría solucionar el error a Lucıa? (1 pt)**

Lucia primero ocupa verificar si existe o no el `layer`. Se debe de abrir el archivo `bblayers.conf`, si este archivo no tiene es necesario ejecutar el siguiente comando.

```
bitbake-layers create-layer
```

Con esto nos aseguramos de la creación del archivo.
Si es el caso de que este archivo exista se puede ejecutar el siguiente comando.

```
bitbake-layers add-layer
```

12. **En el comando** `$(CC) -o thisiswrong.c -I../include -L../lib -lsomething` **¿Cual es el nombre de archivo completo de la biblioteca dinámica enlazada? (1 pt)**

El nombre de esta biblioteca enlazada es `libsomething` esto debido a que el enlazador es `-lsomething`.

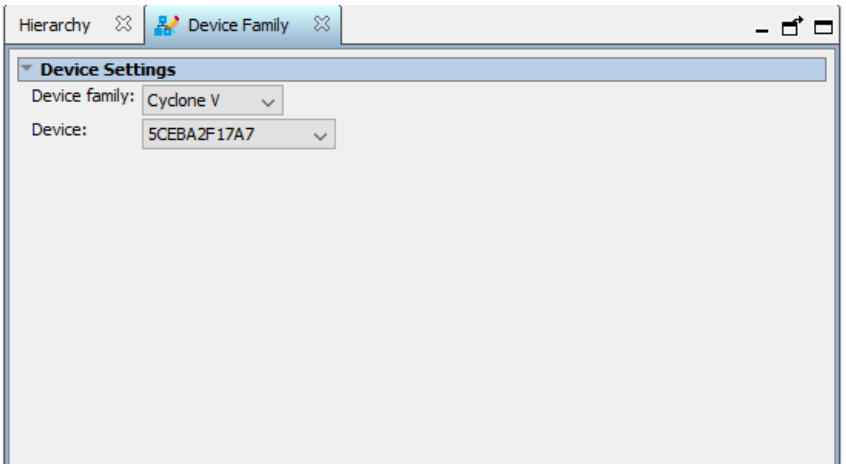
Desarrollo

1. Modeling
   Diagrama de procesos Khan:

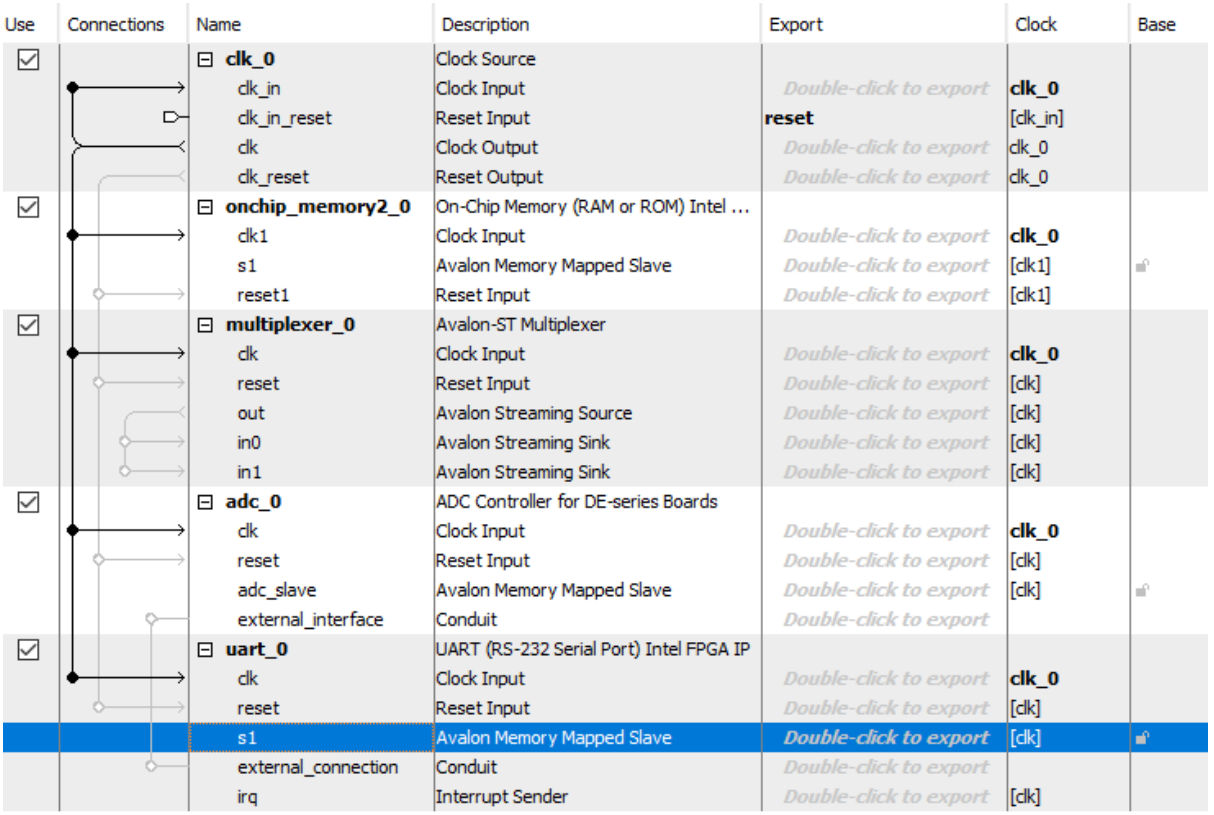
   Bloques necesarios:
   1. ADC
   2. Clk
   3. Memoria al menos de 4kb mínimo
   4. Procesador

   Luego se procede a seleccionar el dispositivo y agregar módulos UART para el manejo de las interrupciones

   Selección de dispositivos:

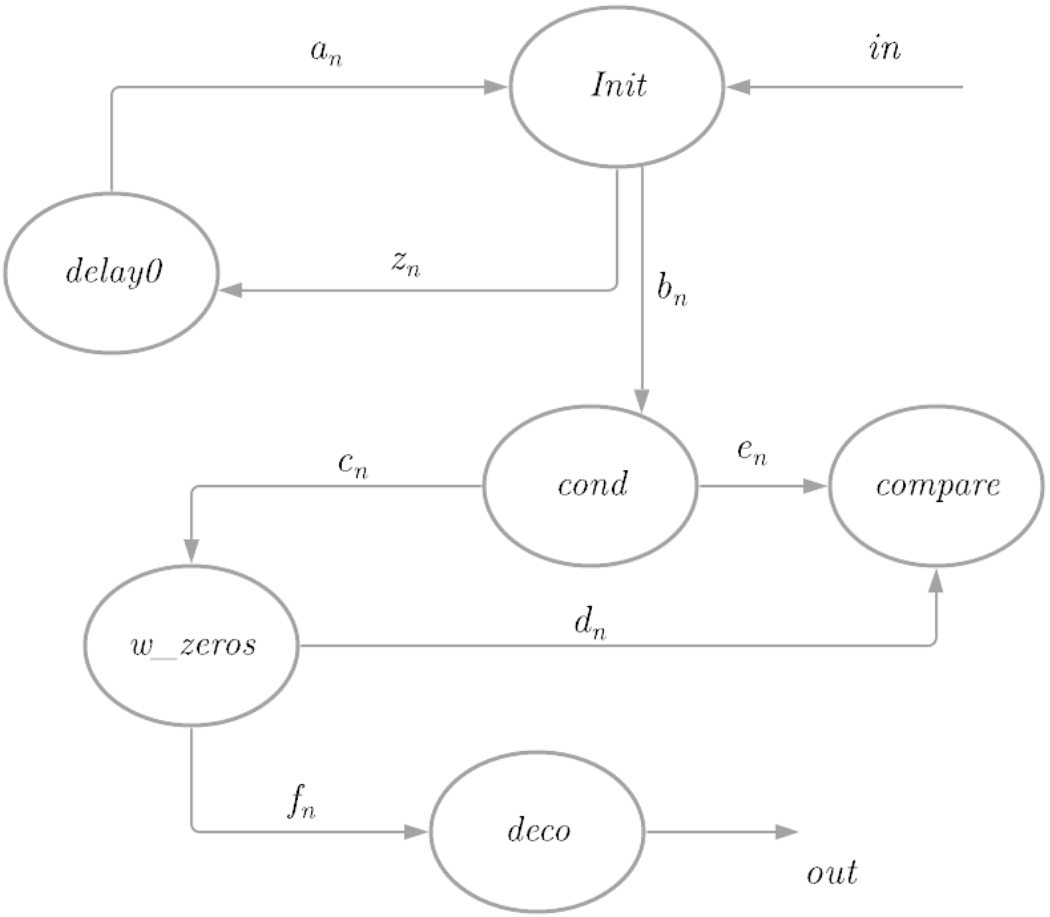


   Selección de componentes:



Con esto sería lo necesario para la generación del estructural del código de la figura 1

2. Diagrama del modelo KPN para RZE



Para el diagrama anterior tenemos lo siguiente:

- **an**: controla los índices del arreglo de entrada
- **zn**: controla los datos de entrada.
- **bn**: maneja la cola de datos a operar
- **cn**: controla la cola de ceros que están seguidos en el arreglo.
- **dn**: indica cual número viene después del cero
- **en**: maneja las entradas diferentes de cero
- **fn**: contiene un valor con la cantidad de ceros seguidos
- **gn**: es la cantidad de ceros que se escriben.

Los procesos se hacen mediante los siguientes estados:

Init: Es el proceso donde se inicializa el RZE

delay0: actualiza el contador, lo que se usa para recorrer el array de entrada, y es **delay0** debido a que los índices en python empiezan desde 0.

cond: este proceso realiza el control de los 0, si aca llega un cero este aumenta la variable y cada vez que llegue otro número diferente de 0 se activa la señal para escribir ceros **w_zeros**.

w_zeros: recibe la cantidad de ceros seguidos, lo que genera una cola para su escritura en el array.

compare: determina qué número tiene seguido es o no cero, si seguid vienen números diferentes de 0 el debe de colocar cero por medio en el nuevo array.

deco: se encarga de escribir en la salida un nuevo array, por lo que le llegan las colas del compare y del w_file.

El archivo se encuentra en la carpeta, el codigo es llamado pregunta2.py