Detalles del proyecto

Tarea III: Algoritmo de Booth - Multiplicador con signo **Integrantes del proyecto:** Felipe Sánchez Segura

1. Introducción

El diseño de sistemas digitales modernos requiere no solo conocimientos teóricos de lógica digital, sino también habilidades prácticas en la implementación de algoritmos mediante circuitos secuenciales y estructuras sincronizadas. Uno de los algoritmos clásicos utilizados en la aritmética binaria es el algoritmo de Booth, empleado para realizar multiplicaciones entre números con signo de forma eficiente.

Este proyecto tiene como propósito implementar un sistema completo de multiplicación con signo utilizando el algoritmo de Booth, desarrollado para una FPGA TangNano 9K. La solución integra múltiples subsistemas, incluyendo captura de datos desde teclado hexadecimal, conversión de datos, ejecución del algoritmo iterativo, y visualización del resultado en displays de 7 segmentos.

Durante el desarrollo se emplearon técnicas de control secuencial, sincronización de señales externas, conversión entre representaciones numéricas y despliegue de datos en hardware. El diseño fue estructurado en módulos independientes que interactúan por medio de señales de control, siguiendo buenas prácticas de modularidad, sincronismo y verificación.

1.1 Objetivo General

Implementar un sistema digital completo basado en el algoritmo de Booth para realizar la multiplicación de números enteros con signo en una FPGA, empleando máquinas de estados finitos y protocolos de comunicación entre bloques.

1.2 Objetivos Específicos

- Diseñar e implementar un sistema digital de multiplicación con signo en una FPGA TangNano.
- Desarrollar una máquina de estados finita (FSM) para controlar la entrada de datos desde un teclado hexadecimal.
- Implementar el algoritmo de Booth para realizar multiplicaciones iterativas con signo.
- Diseñar un sistema de conversión de binario a BCD para la visualización del resultado.
- Desplegar el resultado final en displays de 7 segmentos, incluyendo el signo.
- Verificar el sistema mediante simulaciones funcionales y de temporización.
- Analizar el consumo de recursos, potencia y velocidad de operación del diseño sintetizado.
- Promover el trabajo colaborativo y el uso de herramientas de control de versiones.

2. Descripción general del sistema

El sistema diseñado permite realizar la multiplicación de dos números decimales con signo, ingresados mediante un teclado hexadecimal, y mostrar el resultado en formato decimal utilizando displays de 7 segmentos. Para ello, el diseño se estructura en cuatro subsistemas principales conectados entre sí mediante señales de control.

Cada subsistema tiene una función específica dentro del proceso general, desde la captura de datos del usuario hasta la visualización del resultado. Todo el sistema está controlado por máquinas de estados que aseguran el flujo correcto de información entre bloques, respetando la lógica secuencial del algoritmo de Booth.

La arquitectura general del sistema se compone de:

1. Subsistema de lectura de datos
2. Subsistema de cálculo de multiplicación
3. Subsistema de despliegue

2.1 Subsistema de lectura de datos

Este módulo se encarga de recibir los dos operandos desde un teclado hexadecimal. Los operandos son números decimales de dos dígitos, con signo. El subsistema incluye lógica de sincronización para eliminar rebotes mecánicos y una FSM encargada de controlar el ingreso secuencial de los dígitos, similar al comportamiento de una calculadora.

Mientras el usuario ingresa los valores, estos se muestran parcialmente en los displays de 7 segmentos. Una vez ingresados ambos operandos, se activa una señal de validación (`rdy`) que comunica al siguiente módulo que los datos están listos para ser procesados.

2.2 Subsistema de cálculo de multiplicación

Este módulo implementa el algoritmo de Booth para multiplicación con signo. Recibe los operandos en formato binario y, mediante una FSM, ejecuta de forma iterativa las operaciones necesarias para obtener el resultado. El algoritmo reduce el uso de sumadores combinacionales al realizar desplazamientos y sumas/restas controladas en múltiples ciclos de reloj.

El subsistema genera una señal de salida (`done`) cuando el resultado de la multiplicación está listo para ser convertido y desplegado.

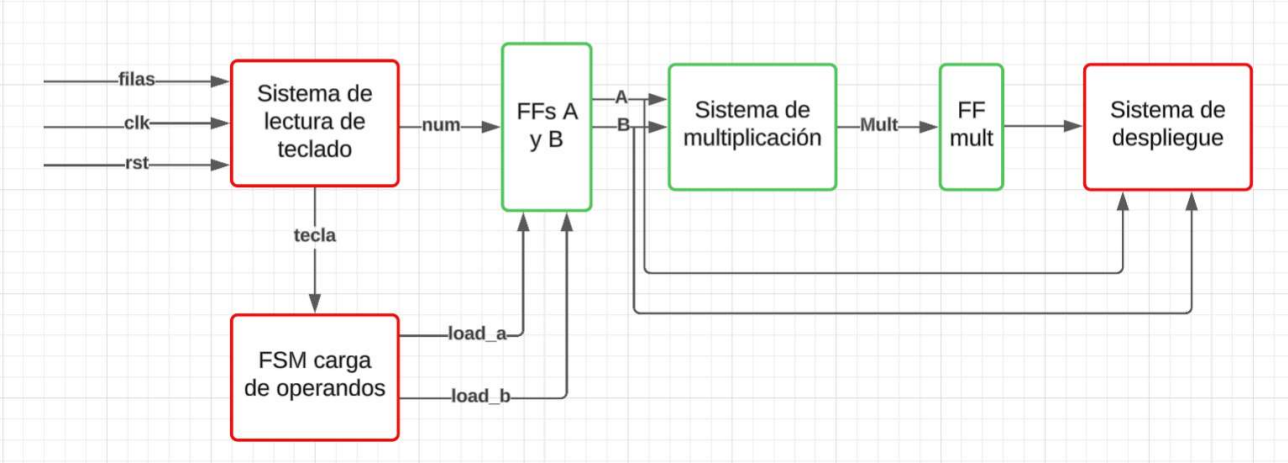
2.3 Subsistema de despliegue

Este subsistema toma el resultado en formato binario y lo decodifica en formato BCD para ser desplegado en el display de 7 segmentos. Se utiliza un contador de anillo que elige que ánodo encender dependiendo del estado en el que se encuentre.

3. Diagramas de bloques

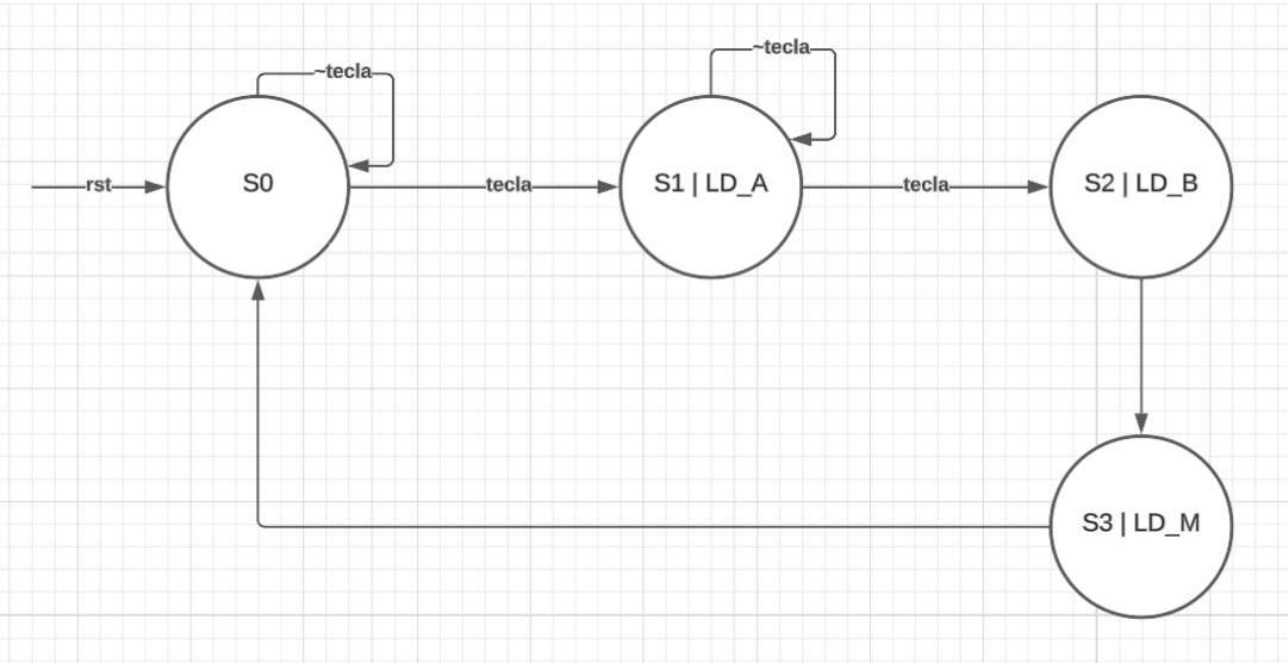
3.1. Diagrama general del sistema

En el presente diagrama se muestra un esquema general de las conexiones del sistema. Se tiene como entrada las filas, el reloj y un botón de reset que reinicia el sistema. Como salida se tiene el resultado de la multiplicación de los números y su presentación en el sistema de despliegue.



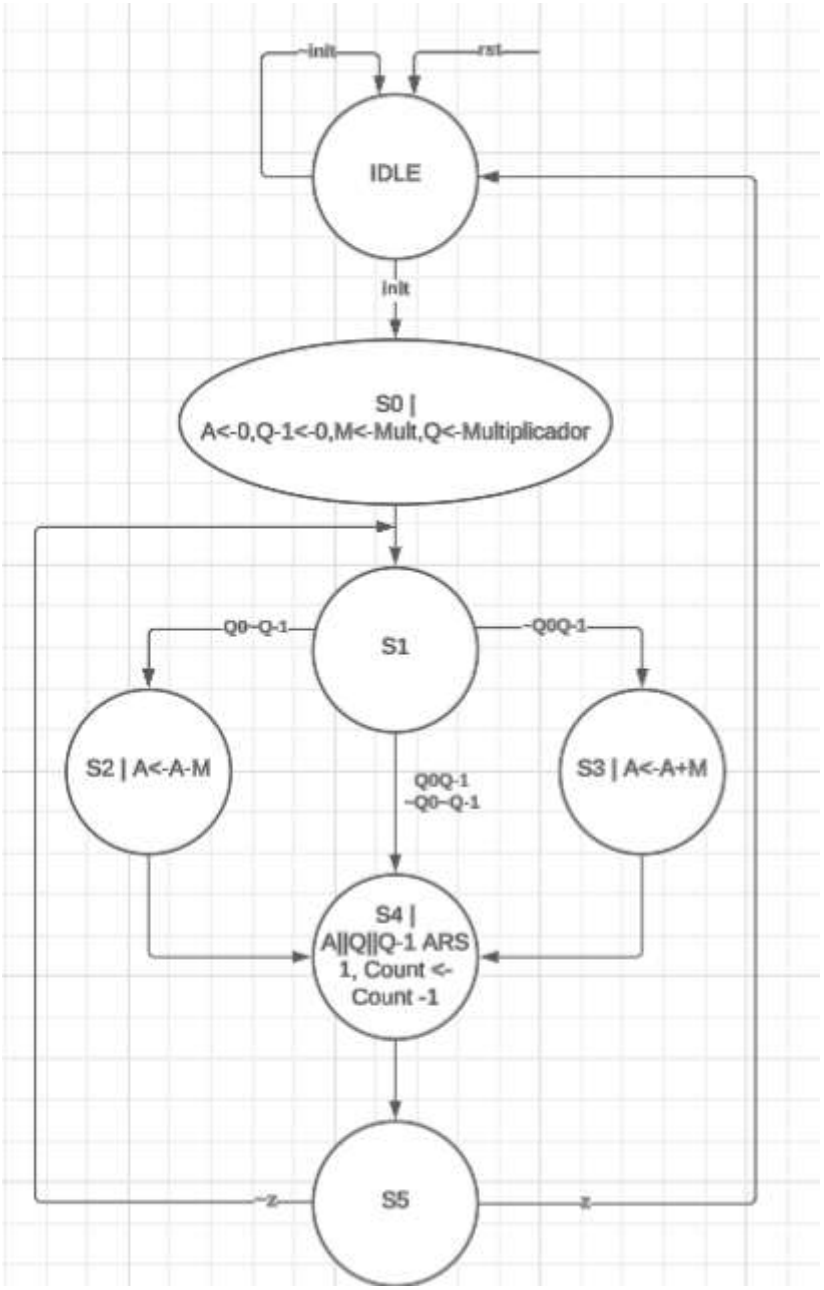
3.2. FSM carga de dígitos

Esta máquina de estados es la encargada de cargar los números en función de una señal de control llamada `tecla`. Este pulso proviene del sistema de lectura y es el encargado de señalar si se ha presionado una tecla o no. Cuando ya se han cargado todos los dígitos, esta máquina da como salida una señal `load_m`, la cual será una señal de control en la siguiente máquina de estados.



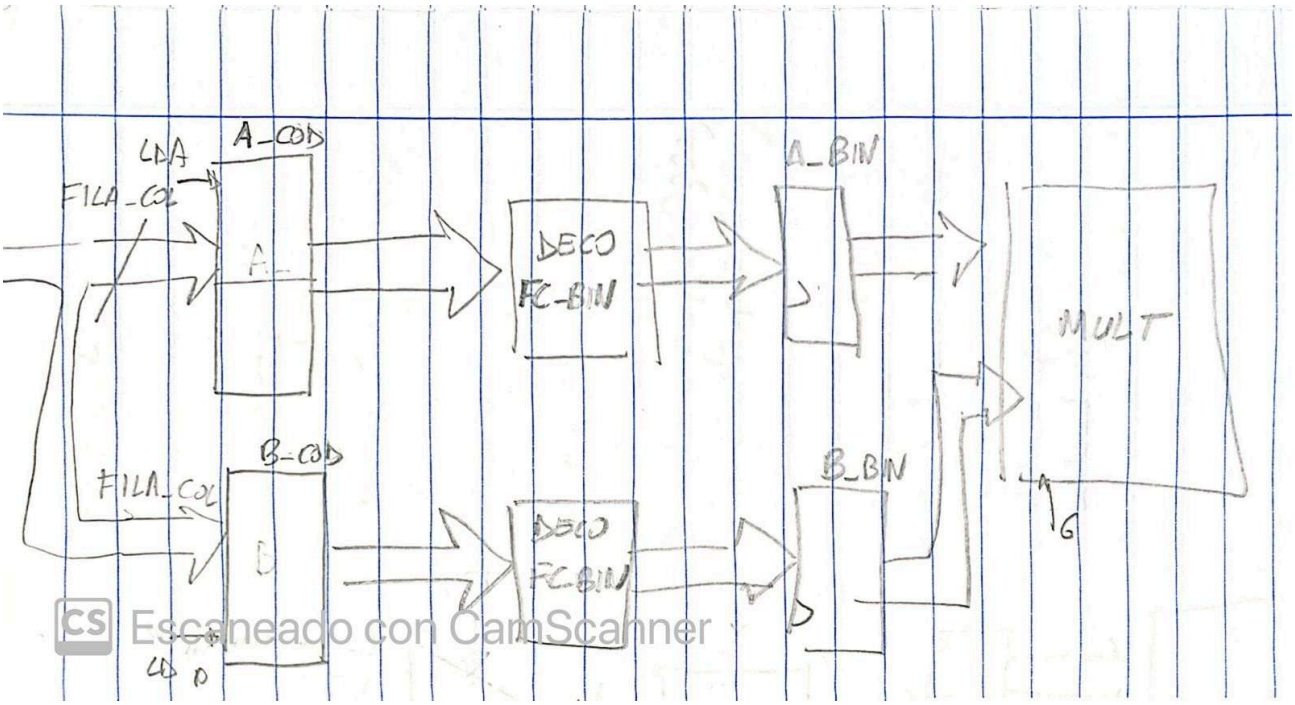
3.3 FSM Booth

Esta FSM es la encargada de controlar el multiplicador de Booth proporcionado por el docente. Está basado en el ASM que se encuentra en el instructivo de la tarea y controla al multiplicador mediante señales de control `add_sub`, `shift`, `load_add`, etc. En resumen, cuando recibe una señal `init` comienza su funcionamiento. En `S0` reinicia el acumulador, establece el bit menos significativo del registro Q en 0, Almacena el multiplicando en el registro M y guarda el multiplicador en los bits restantes de Q. En `S1` compara los dos bits menos significativos de Q y decide si ir a `S2`, `S3` o `S4`. Cabe resaltar que independientemente del camino que tome la FSM, esta siempre debe llegar a `S4`. Por último, en `S5` si el contador es cero entonces la multiplicación termina, si no, vuelve al estado `S1`.



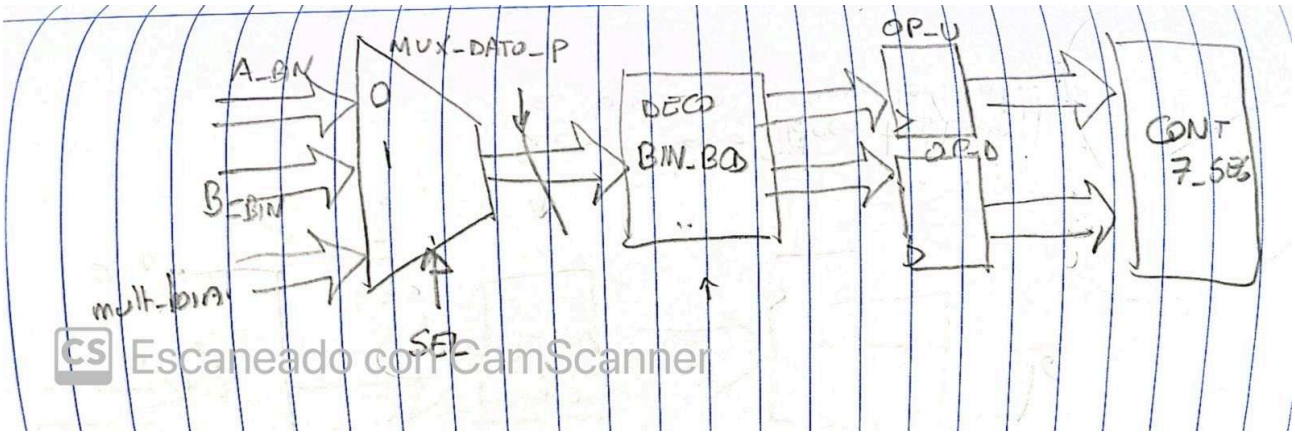
3.4. Sistema de lectura de teclado

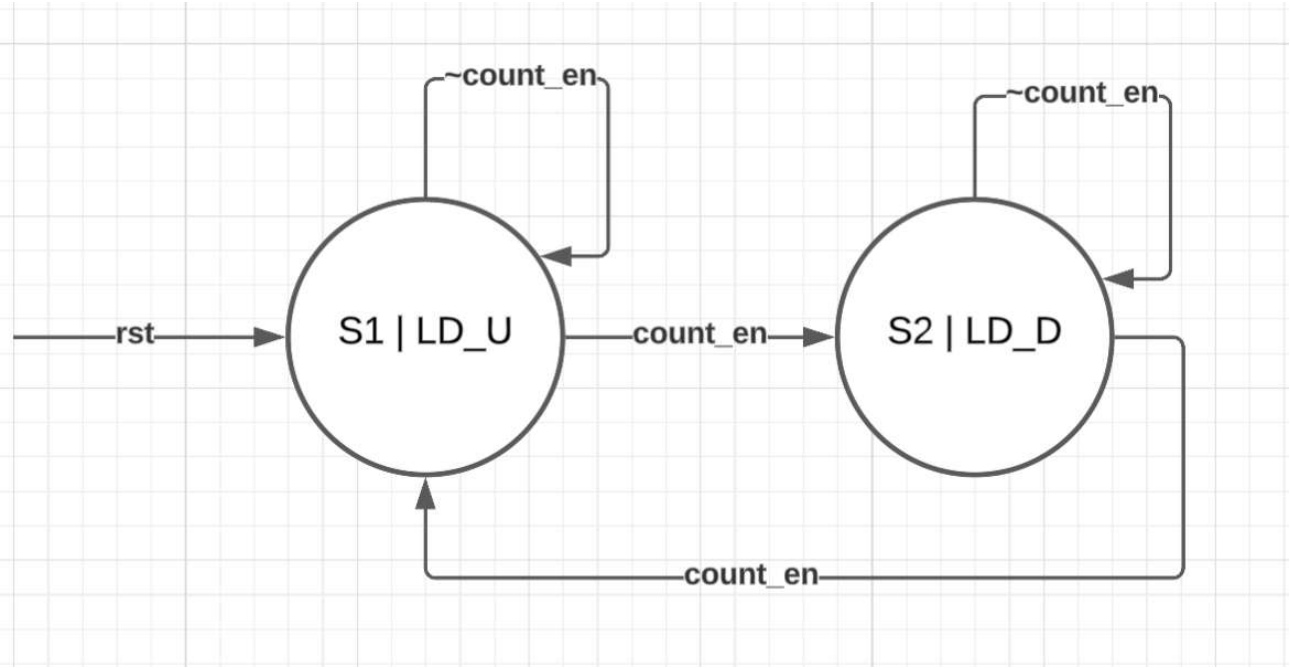
Este subsistema permite capturar los dígitos ingresados desde un teclado hexadecimal matricial. Utiliza un contador de 2 bits para generar el barrido lógico de columnas, mientras se monitorean las filas para detectar pulsaciones. Las señales pasan por un debouncer que elimina rebotes mecánicos y generan un pulso limpio (`tecla`). Luego, un decodificador identifica la tecla presionada a partir de la combinación fila-columna. La FSM de carga controla la secuencia de registro de los números ingresados para ser procesados por el multiplicador.



3.5. Sistema de despliegue

Este sistema muestra los operandos A, B y la multiplicación mediante un selector y un controlador de siete segmentos. La señal `SEL` viene codificada de la salida de la FSM de operandos. Dependiendo de la selección de dígitos, estos pasan por un decodificador de binario a BCD para pasar al controlador de 7 segmentos. Este controlador es controlado (valga la redundancia) por un contador de anillo que enciende los ánodos cuando un contador llega a `count_en` .





4. Simulaciones y consumo de recursos

En esta sección se muestra el tb ejecutado para comprobar el funcionamiento del sistema, así también como el consumo de recursos.

4.1. Testbench

```
VCD info: dumpfile module_top_tb.vcd opened for output.
filas presionadas: 0100, | Salida: Xx
../sim/module_top_tb.sv:59: $finish called at 1300 (1ns)
```

4.2. Consumo de recursos

```
=== module_top ===

   Number of wires:                187
   Number of wire bits:            358
   Number of public wires:         187
   Number of public wire bits:     358
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                156
     ALU                            14
     DFF                            11
     DFFR                            3
     DFFRE                          29
```

DFFSE	1
GND	1
IBUF	6
LUT1	14
LUT2	12
LUT3	15
LUT4	21
MUX2_LUT5	16
MUX2_LUT6	5
OBUF	7
VCC	1

5. Problemas encontrados durante el proyecto

Durante el desarrollo del proyecto, se identificaron los siguientes desafíos:

1. **Manejo del tiempo y coordinación en el equipo**
   Uno de los principales inconvenientes fue la falta de una planificación estructurada desde el inicio del proyecto. Aunque se trabajó de forma individual, no se definieron fechas límite internas para la finalización de módulos clave ni se organizaron sesiones de revisión. Esta falta de cronograma provocó retrasos en la integración de los subsistemas y en la verificación completa del sistema.

2. **Sincronización entre subsistemas** La comunicación entre los subsistemas de lectura, cálculo, conversión y despliegue requirió una coordinación precisa mediante señales de control (`valid` , `done` , `reset` , etc.). En las primeras versiones del sistema, se observaban condiciones de carrera y transiciones inestables debido a señales no sincronizadas entre módulos que operaban a diferentes velocidades. Este problema persistió hasta la versión final.

3. **Transición del diseño RTL a la implementación física**
   Al pasar de simulación a implementación en la FPGA, surgieron problemas relacionados con tiempos de respuesta, incompatibilidades de conexión y frecuencia de operación. Algunos módulos que funcionaban en simulación no reaccionaban correctamente en hardware real. Esto obligó a depurar señales internas, revisar restricciones de pines y ajustar los divisores de frecuencia para el teclado y los displays. A pesar de los esfuerzos, no se pudo realizar la implementación física apropiada del diseño en HDL.

6. Referencias

[1] David Harris y Sarah Harris. *Digital Design and Computer Architecture. RISC-V Edition*. Morgan Kaufmann, 2022. ISBN: 978-0-12-820064-3 [2] Andrew House. Hex Keypad Explanation. Nov. de 2009. url: https://www-ug.eecg.toronto.edu/msl/nios_devices/datasheets/hex_expl.pdf.