***Laboratorio 2 – Parpadeo de LED***

Objetivos del laboratorio son conocer:

* Entender el uso de señales y constantes

*Creación del Proyecto de LEDs*

1 - Abrir Vivado en la pantalla principal y presionar en abrir proyecto

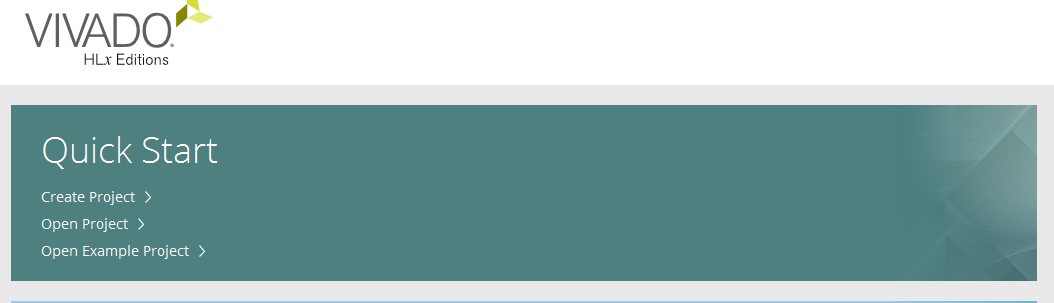
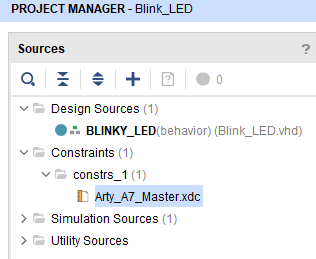


Figura 1.1 – Creación del proyecto en Vivado.

2 – Del proyecto suministrado abra el Project manager, deberá haber tenido una pantalla similar a esta



*Figura 1.2 – Archivos de trabajo de proyecto*

*Señales y Constantes*

Las constantes en VHDL no son tipos de datos.

Se pueden declarar de diferentes maneras como INTEGER, STD\_LOGIC, STD\_LOGIC\_VECTOR tanto para constante como para señales.

Las constantes se utilizan para definir mejor el código en VHDL. Asumamos tenemos por todo el código un número que define el tamaño de la señal HDMI a 720p, podemos entonces en vez de cambiar cada línea de los 720 a una constante en la parte superior de nuestro archivo VHD que al cambiar, cambie todas las líneas para que un solo cambio afecte todos. En C sería ídem al:

#define <NombreDeConstante> <ValorDeConstante>

VHDL Sintaxis

Constant <constan\_name> : <constant\_type> := <constant\_value>

Las señales son las conexiones intermedias en nuestro circuito. En el mundo real, al utilizar un “breadboard” o también llamado “protoboard”, para conectar integrados tenemos cables; en VHDL estos cables se llaman señales. Las señales pueden conectar la salida de una entidad a la entrada de la siguiente entidad, conexiones internas de la entidad y así sucesivamente.

En resumen es una manera de conectar data dinámica, debido a que esta señal es cambiante en el tiempo. Una señal tiene asociado el tipo de dato, valor y tiempo.



*Figura 1.3 – Ejemplo de una señal que sale de un bloque lógico y entra a otro bloque lógico. Puede ver todo el sistema también como una gran entidad y imaginarse que las señales son internas.*



*Figura 1.4 – Ejemplo de una constante.*

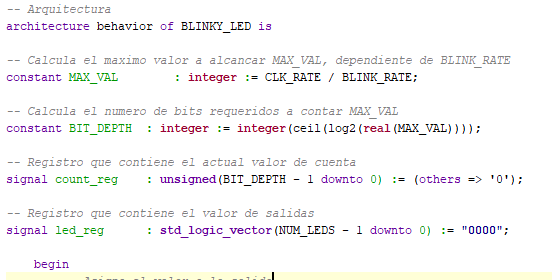
3 - Las señales y constantes se definen antes de begin y después de architecture. Para el archivo abierto cree las siguientes señales y constantes.

MAX\_VAL será el valor máximo del contador en el cual causaremos un cambio de estado en el LED.

BIT\_DEPTH calcula la cantidad de bits, en este caso dependiendo del valor máximo. Así, si el valor máximo es 30000, esto automáticamente calculará que necesitamos 8 bits.

count\_reg es una señal que almacena la cuenta que varía de 0 a MAX\_VAL

led\_reg es una señal que pasa este valor a la salida de leds, más adelante se verá esto.

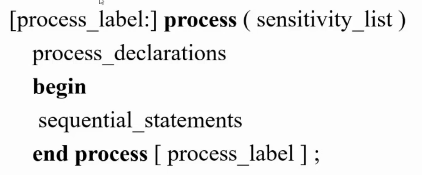


*Figura 1.5 – Ejemplo de señales y constantes*

*Procesos*

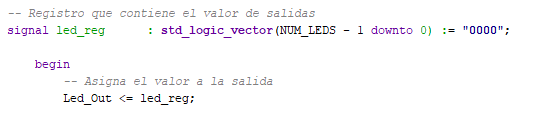
Dentro de VHDL existen sentencias que se realizan instanteamente, por ejemplo cuando configuramos una OR, las relaciones entre salidas y entradas son evaluadas instantemente.

Si deseamos en VHDL algo QUE OCURRA EN UN TIEMPO ESPECÍFICO, entonces necesitamos declararlo como proceso. En otras palabras, ud. desea ejecutar y evaluar esta parte de VHDL para condiciones específicas.



*Figura 1.6 – Ejemplo de process*

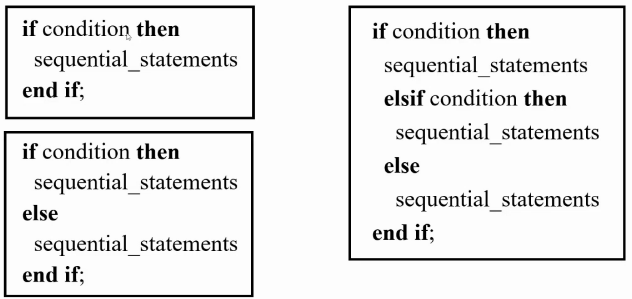
4 - Un proceso tiene, como en la figura anterior, una etiqueta (opcional), la palabra process y una lista de sensitividad. Esta lista puede contener (en este caso el reloj) señales que se deseen evaluar. Caso general es procesar el reloj para que en un flanco de subida o bajada ocurran acciones específicas. La lista de sensitividad puede tener cualquier señal que deseemos evaluar.



*Figura 1.7 – Asigna la señal de led\_reg directamente a la salida.*

*If-else*

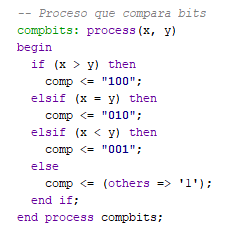
En VHDL hay que tener cuidado al usar los IF statements. Si bien es cierto la manera de programación en VHDL es concurrente, los el IF superior se ejecutará primero que la sentencia ELSE o ELSIF.



*Figura 1.8 – Sintaxis IF ELSE ELSIF END IF*

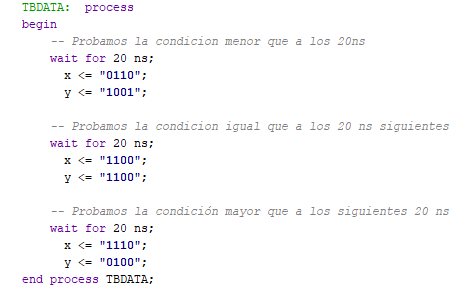
4.a - Para probar un momento la sintaxis if-elsif-else-endif guarde y cierre un momento el proyecto y abra el proyecto Comparator.

Observe que en la sección crítica tenemos comparaciones bit a bit de los elementos ‘x’ y ‘y’.



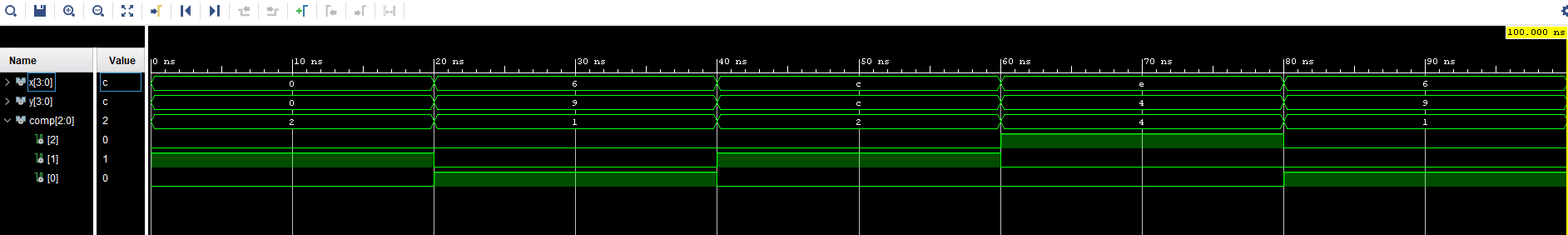
*Figura 1.9 – Proceso de comparación y uso del if.*

4.b - Adicionalmente observe que hemos creado un archivo llamado ComparatorTB y que definimos los casos que queremos evaluar



*Figura 1.10 – Archivo ComparatorTB. Comparamos los tres casos y terminamos la simulación.*

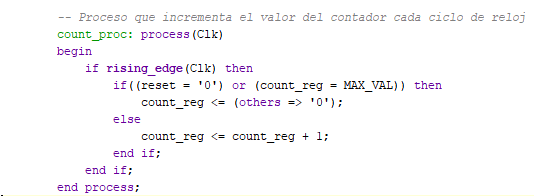
4.c - Ejecute la simulación y observe el comportamiento del comparador. Verifique que cumple su cometido.



*Figura 1.11 – Simulación del comparador. Tenemos entonces que para cada condición cumple su cometido.*

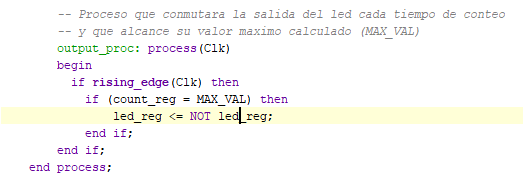
5 - Cierre el proyecto del comparador y vuelva a abrir el proyecto de botones y led.

Ahora crearemos un proceso de conteo que incremente el contador cada vez que un flanco de subida sea detectado.



*Figura 1.12 – Asigna la señal de led\_reg directamente a la salida.*

6 - Finalmente crearemos un proceso que pase el valor actual del estado de los led y lo conmute (1’s x 0’s y 0’s por 1’s)



*Figura 1.13 – Asigna la señal de led\_reg directamente a la salida.*

7 – Especificar los LEDs y Botones a utilizar en la tarjeta Arty A7, en nuestro caso.

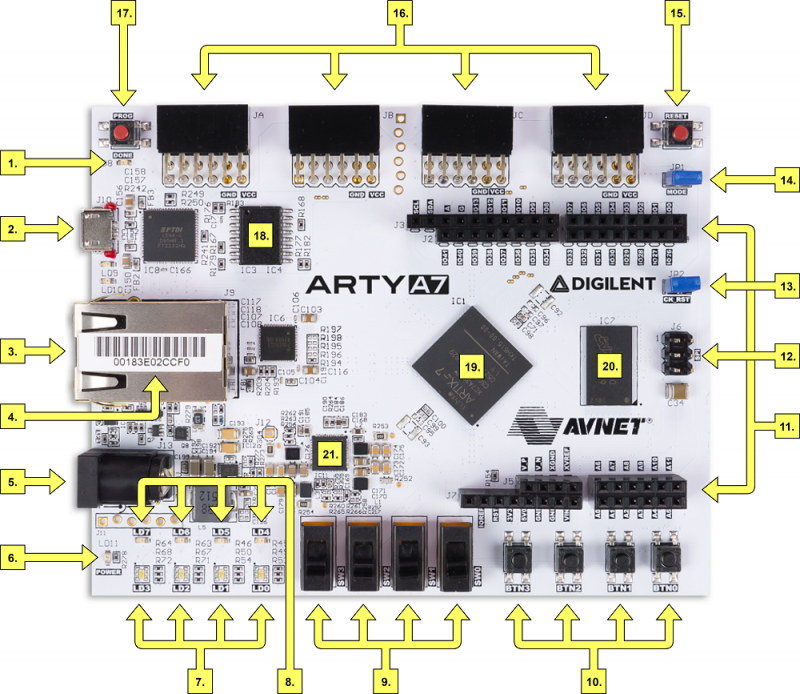
Primeramente mostramos los componentes que posee nuestra tarjeta y donde están ubicados.

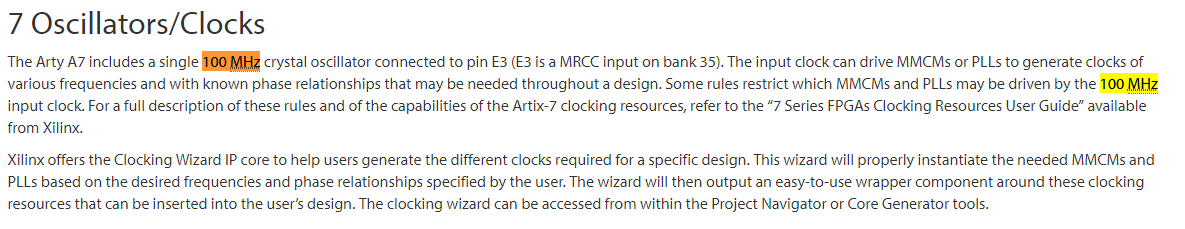
Figura 1.14 – Componentes de Tarjeta Arty A7

Tabla 1.1 – Documentación de Componentes de Arty-A7

| **Callout** | **Description** | **Callout** | **Description** | **Callout** | **Description** |
| --- | --- | --- | --- | --- | --- |
| 1 | FPGA programming DONE LED | 8 | User RGB LEDs | 15 | chipKIT processor reset |
| 2 | Shared USB JTAG / UART port | 9 | User slide switches | 16 | Pmod connectors |
| 3 | Ethernet connector | 10 | User push buttons | 17 | FPGA programming reset button |
| 4 | MAC address sticker | 11 | Arduino/chipKIT shield connectors | 18 | SPI flash memory |
| 5 | Power jack for optional external supply | 12 | Arduino/chipKIT shield SPI connector | 19 | Artix FPGA |
| 6 | Power good LED | 13 | chipKIT processor reset jumper | 20 | Micron DDR3 memory |
| 7 | User LEDs | 14 | FPGA programming mode | 21 | Dialog Semiconductor DA9062 power supply |

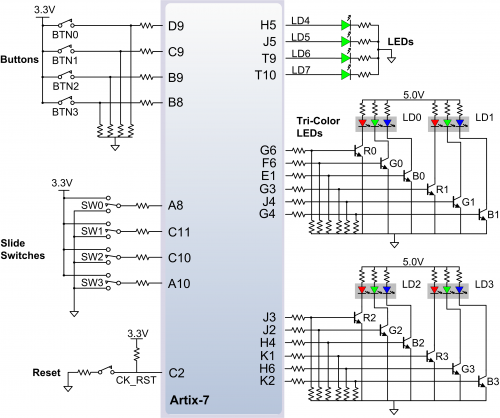
*Reloj / Oscilador:*

Según la documentación de la Arty, posee un cristal que corre alrededor de 100MHz y esto es de considerar al momento de realizar el ingreso de MAX\_VALUE



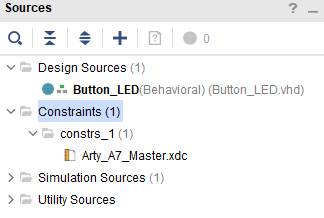
*Figura 1.15. En la documentación de la Arty A7 detalla que el reloj es de 100 MHz.*

8 - Dados los componentes de la tarjeta ahora notaremos en referencia cuales son las entradas y salidas.



*Figura 1.16 – Muestra de ubicación de los botones y LEDs en la Arty A7*

9 - Verifique la ventana de Constraints y observe que se añadió al proyecto

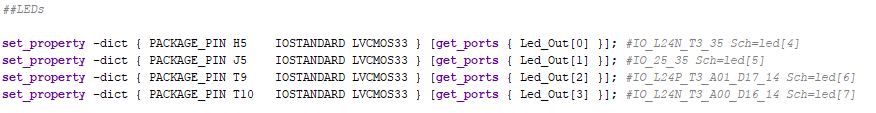


*Figura 1.17 – Verificación del archivo de definiciones/limitantes (constraints).*

10 - Abrir el archivo XDC y habilitar los LEDs y asigne las entradas y salidas como las declaró en su componente:

Tenemos leds que asignamos, en total la Arty A7 serían 4 LEDs, 1 switch y la señal de reloj





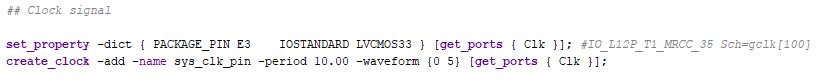


Figura 1.18 – Archivo de definiciones de Arty. Para la Arty A7 y este proyecto, este será el archivo de definiciones.

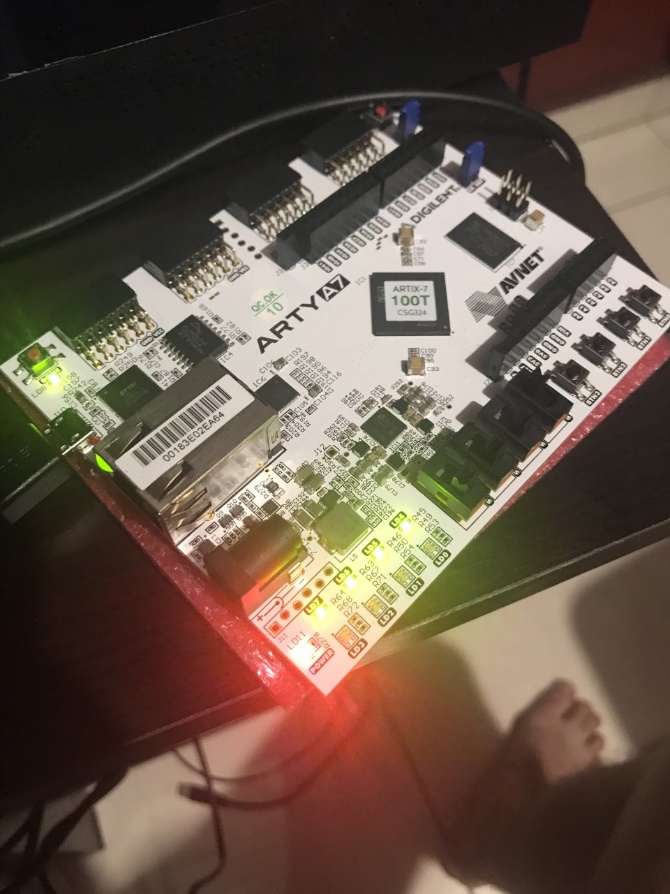
Note lo siguiente. Para la señal de reloj se puede especificar el período (en este caso 10ns), que es una forma de onda de ciclo de trabajo de 50%

11 – Genere el archivo de salida para programación del FPGA

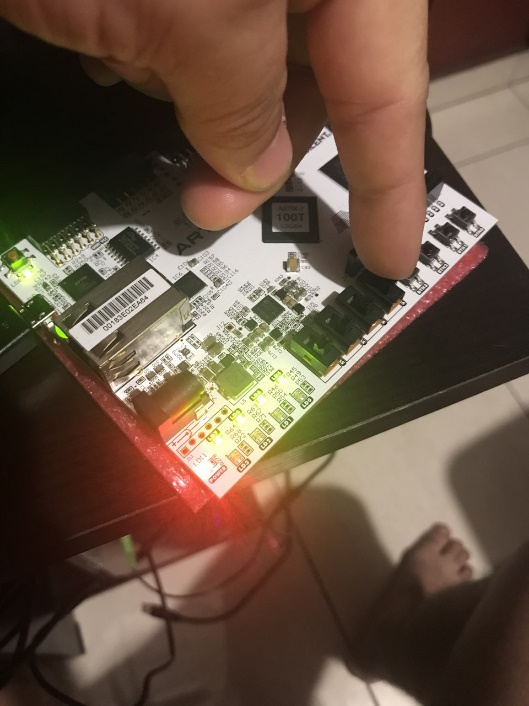
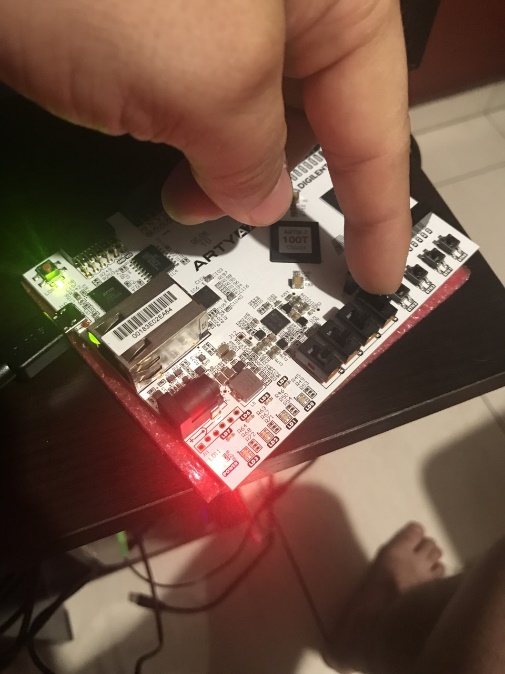
12 – Descargue el archivo y observe el funcionamiento.

13 – Pruebe la tarjeta:

* SW0 en su posición normal es el enable
* LEDs LED0 a LED3 se encenderán y apagaran si el Enable está habilitado
* Los leds mantendrán su etado si el enable está inhabilitado.



*Figura 1.19 – Enable debe estar activado (SW0) para que se vea el funcionamiento. Conmuta a la frecuencia configurada.*

(a) (b)

*Figura 1.20 – En cualquier caso, si inhabilitamos el enable estando los leds encendidos permanecerán encendidos, si está apagado e inhabilitamos la cuenta, permanecerá apagado.*

Evaluación:

50% - Completar todos los pasos anteriores:

* Ejecución del programa de botones y LED en un FPGA.

50% - Realizar los siguientes cambios

* Cumplir con lo siguiente
  + Escriba el Código en VHDL
  + Simule
  + Implemente en el FPGA

Debe realizar un programa que al cambiar un interruptor solamente se conmute el LED asociado a la tabla luego de excedida la cuenta del valor máximo. Rata de cambio de 0.5 Hz.

Por ejemplo, para el caso de la tabla inferior (caso 1).

SW0 conmutará LED1 excedida la cuenta del contador, dependiente del Enable

SW1 conmutará LED1 excedida la cuenta del contador, dependiente del Enable

SW2 conmutará LED1 excedida la cuenta del contador, dependiente del Enable

SW3 conmutará LED1 excedida la cuenta del contador, dependiente del Enable

El botón de Enable es BTN0

Ver tabla de siguiente página

*Tabla 1.2 – Listado de Asignación*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | SW0 | SW1 | SW2 | SW3 | Enable (BTN) |
| 1 |  | LED | 1 | 3 | 1 | 2 | 0 |
| 2 |  | LED | 3 | 1 | 2 | 3 | 0 |
| 3 |  | LED | 0 | 3 | 0 | 1 | 3 |
| 4 |  | LED | 2 | 3 | 2 | 3 | 0 |
| 5 |  | LED | 2 | 1 | 2 | 0 | 2 |
| 6 |  | LED | 2 | 2 | 3 | 1 | 1 |
| 7 |  | LED | 3 | 2 | 3 | 1 | 1 |
| 8 |  | LED | 0 | 2 | 3 | 1 | 3 |
| 9 |  | LED | 0 | 1 | 3 | 1 | 1 |
| 10 |  | LED | 2 | 3 | 2 | 3 | 1 |
| 11 |  | LED | 0 | 0 | 1 | 1 | 3 |
| 12 |  | LED | 1 | 2 | 3 | 3 | 2 |
| 13 |  | LED | 1 | 3 | 0 | 1 | 0 |
| 14 |  | LED | 3 | 2 | 3 | 2 | 3 |
| 15 |  | LED | 1 | 1 | 0 | 3 | 1 |
| 16 |  | LED | 1 | 1 | 3 | 0 | 2 |
| 17 |  | LED | 0 | 0 | 3 | 0 | 2 |