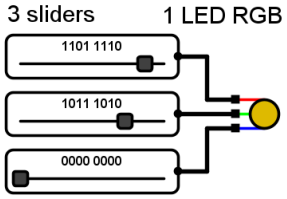
CC4301 Arquitectura de Computadores – Examen – Primavera 2025 – Prof.: Luis Mateu

Pregunta 1



Parte a.- Suponga que Logisim dispone de 2 nuevos dispositivos: el *slider* y el *led rgb*. La figura muestra 3 sliders conectados al led rgb. Un slider genera una salida de 8 bits. El usuario hacer variar el valor de la salida moviendo la manilla con la mano de Logisim, desde 0 (manilla completamente a la izquierda) hasta 255 (completamente a la derecha). El led rgb posee 3 entradas de 8 bits cada una, que indican la intensidad del rojo (arriba), verde (al centro) y azul (abajo). En el ejemplo de la figura el usuario puede mostrar cualquier color moviendo las 3 manillas.

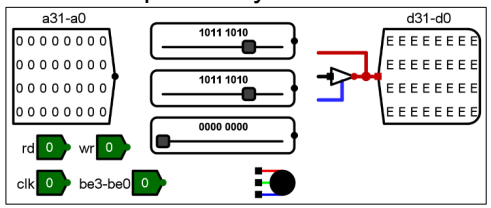
Diseñe una interfaz de entrada y salida para LRV32IM que permita a un programa leer el estado de los 3 sliders y especificar el color del led rgb invocando las funciones:

```
typedef unsigned char uchar;
void getSliders(uchar *pr, uchar *pg, uchar *pg);
void setRgbLed(uchar red, uchar green, uchar blue);
```

Ejemplo de uso:

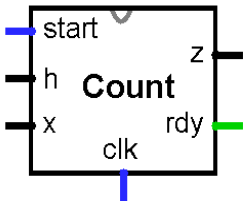
```
uchar r, g, b;
getSliders(&r, &g, &b);
setRgbLed(r, g, b);
```

El diseño es libre: Ud. define el número de puertos y sus direcciones (puede usar un solo puerto de 32 bits o 3 puertos de 8). La figura de la derecha muestra las entradas y salidas que debe considerar en su interfaz de los dispositivos con los buses de direcciones, datos y control.

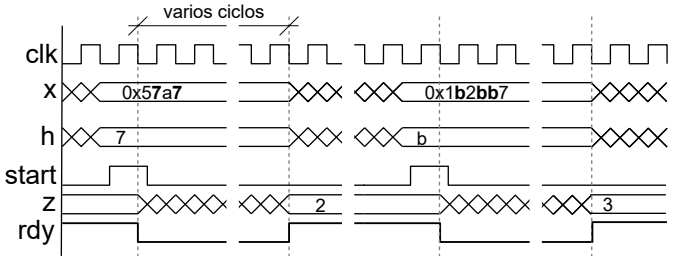


Parte b.- **Programe** las funciones getSliders y setRgbLed.

Pregunta 2



I) Implemente el circuito *Count* de la figura. Recibe como entradas un entero *x* de 8 cifras hexadecimales (32 bits en total) y una cifra *h* hexadecimal distinta de 0. Debe entregar en la salida *z* (de 4 bits) la cantidad de cifras *h* presentes en *x*. Por ejemplo si *x* es 0x5**7**a**7** y *h* es 7, z debe ser 2 porque 7 aparece 2 veces. El cálculo comienza cuando la entrada *start* se pone en 1 (por un solo ciclo del reloj) y termina cuando la salida *rdy* es 1 arrojando el resultado en *z*. Mientras se realiza el cálculo *rdy* debe permanecer en 0 y el valor de la salida *z* es irrelevante. Las entradas *x* y *p* se mantienen constantes hasta que *rdy* se ponga en 1. Debe mantener las salidas *rdy* en 1 y *z* constantes hasta que *start* vuelva a 1. El diagrama de tiempo muestra el funcionamiento solicitado.



II) Escriba en systemverilog una versión sintetizable del mismo módulo *Count* de la parte I.

Pregunta 3

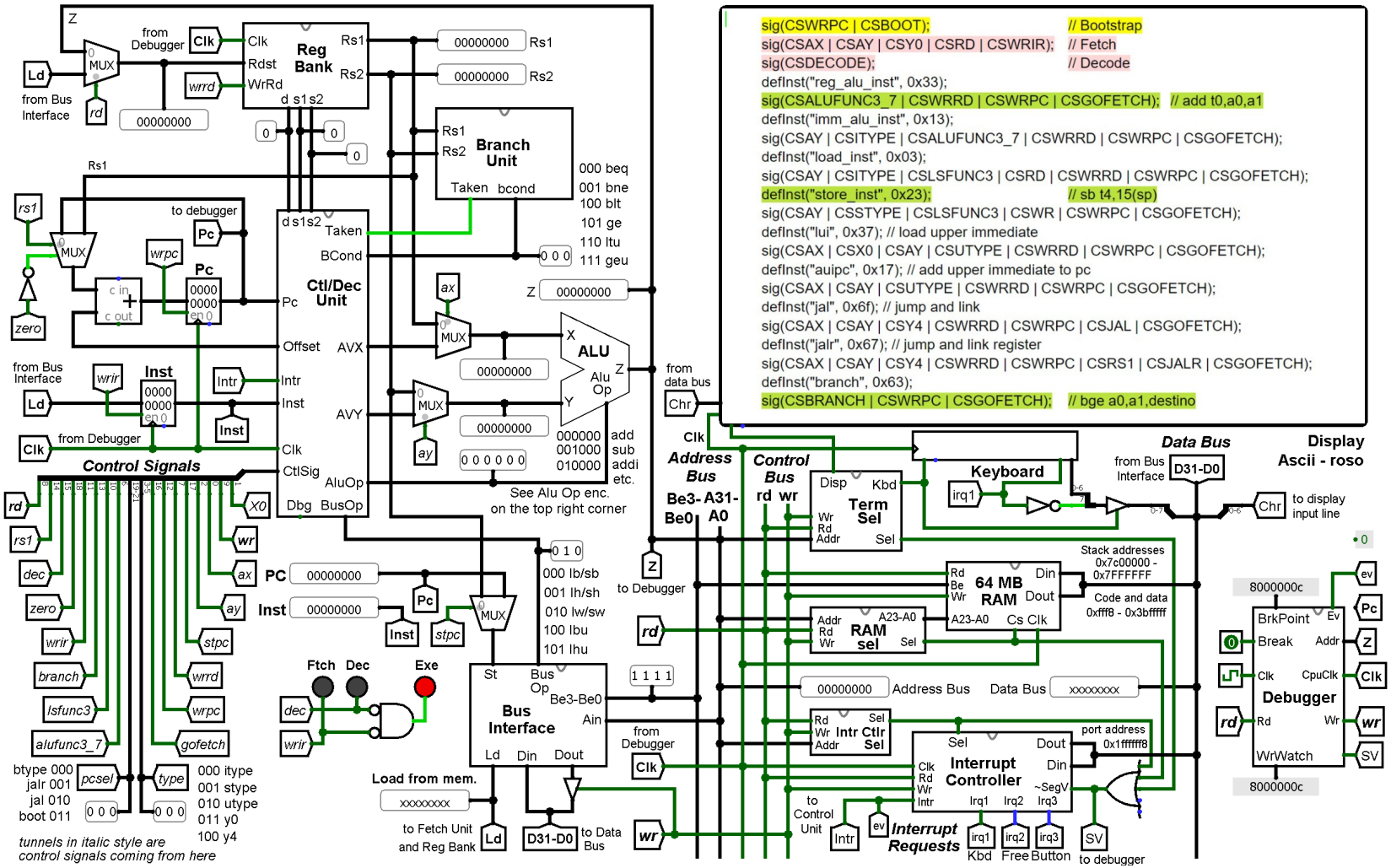
a) PC ← T2-14
b) S4 ← T2
c) T2 ← S4-14
d) Inst ← Mem[T2-14]
e) S4 ← PC-14

A) Considere el diseño de LRV32IM microprogramado que usó en sus tareas. Al reverso del enunciado está el diagrama de bloques. El registro de instrucción *Inst* tiene cargada la instrucción *andi s4,t2,-14* (*s4* es el registro de destino). **Evalúe** de manera independiente si cada una de las 5 transferencias entre registros del cuadro de la derecha se puede realizar en *un solo ciclo del reloj*. Si la transferencia se puede realizar indique **cuáles son la señales de control** requeridas para llevarla a cabo. Si no se puede realizar, explique por qué.

B) La figura muestra un extracto del estado actual de un *caché* de 8 KB (2^{13} bytes) de 2 grados de asociatividad, cada uno de los 2 bancos con 256 líneas de 16 bytes. El caché no está vacío. Por ejemplo en la línea 2a del caché (en hexadecimal) se almacenan las líneas de memoria que tienen como etiqueta 92a y f2a (es decir, la línea que va de la dirección f2a0 a la dirección f2af). Un programa accede a las siguientes direcciones de memoria (en hexadecimal): 6b58, d789, f2ac, a2a0, 2b54, 1788, 92a0, eb5c y 32a0. **Indique** qué accesos a la memoria son aciertos en el caché, cuáles son desaciertos y **rehaga la figura** mostrando el estado final del caché. Por ejemplo el acceso 6b58 es un acierto.

línea cache	*etiqueta*	*contenido*	*etiqueta*	*contenido*
b5	2b5		6b5	
78	d78		478	
2a	92a		f2a	

CC4301 Arquitectura de Computadores – Examen – Primavera 2025 – Prof.: Luis Mateu




```
sig(CSWRPC | CSBOOT);                          // Bootstrap
sig(CSAX | CSAY | CSY0 | CSRD | CSWRIR);       // Fetch
sig(CSDECODE);                                 // Decode
defInst("reg_alu_inst", 0x33);
sig(CSALUFUNC3_7 | CSWRRD | CSWRPC | CSGOFETCH);  // add t0,a0,a1
defInst("imm_alu_inst", 0x13);
sig(CSAY | CSITYPE | CSALUFUNC3_7 | CSWRRD | CSWRPC | CSGOFETCH);
defInst("load_inst", 0x03);
sig(CSAY | CSITYPE | CSLSFUNC3 | CSRD | CSWRRD | CSWRPC | CSGOFETCH);
defInst("store_inst", 0x23);                   // sb t4,15(sp)
sig(CSAY | CSSTYPE | CSLSFUNC3 | CSWR | CSWRPC | CSGOFETCH);
defInst("lui", 0x37); // load upper immediate
sig(CSAX | CSX0 | CSAY | CSUTYPE | CSWRRD | CSWRPC | CSGOFETCH);
defInst("auipc", 0x17); // add upper immediate to pc
sig(CSAX | CSAY | CSUTYPE | CSWRRD | CSWRPC | CSGOFETCH);
defInst("jal", 0x6f); // jump and link
sig(CSAX | CSAY | CSY4 | CSWRRD | CSWRPC | CSJAL | CSGOFETCH);
defInst("jalr", 0x67); // jump and link register
sig(CSAX | CSAY | CSY4 | CSWRRD | CSWRPC | CSRS1 | CSJALR | CSGOFETCH);
defInst("branch", 0x63);
sig(CSBRANCH | CSWRPC | CSGOFETCH);            // bge a0,a1,destino
```

Alu Op encoding

bit 5 (left): mul/div operation
bit 4: immediate format
bit 3: substract / sra
bit 2-0 (right): **func3**

Alu Op func3 encoding

000 add/sub	100 xor
001 sll	101 srl/sra
010 slt	110 or
011 sltu	111 and

LRV32IM CPU

Risc-V 32 bit integer base instruction set with multiply/divide extension

Simple microprogrammed physical architecture

64 MB RAM

No Memory Management Unit

By Luis Mateu

Previous work by leijurv

GPL Licence