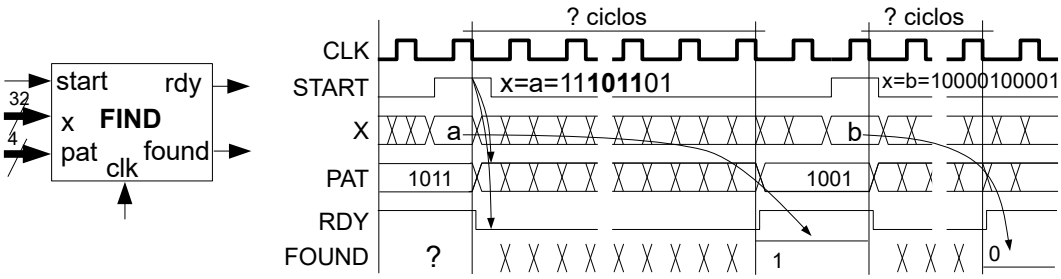
CC4301 Arquitectura de Computadores – Control
Otoño 2021 – Prof.: Luis Mateu

Pregunta 1



Use diseño modular para implementar el circuito *FIND* de la figura. Este circuito determina si la secuencia de bits *PAT* de 4 bits está presente en el número *X* de 32 bits. La búsqueda comienza cuando *START* es 1. Mientras dura, *RDY* debe estar en 0. Cuando concluye la búsqueda, *RDY* se debe colocar en 1, y si *PAT* está presente en *X*, *FOUND* debe ser 1. En caso contrario *FOUND* es 0. Por ejemplo, el diagrama muestra que cuando X es 0...0011**1011**01, 1011 sí está presente y cuando X es 0...0010000100001, 1001 no está presente. En la búsqueda se deben considerar los 32 bits de X, incluso si los bits más significativos están en 0.

Resuelva el problema de la siguiente manera: Cuando *START* es 1, almacene *X* en el registro *RX* y *PAT* en el registro *RPAT*. A continuación, en cada ciclo compare los 4 bits menos significativos de *RX* con *RPAT*. Desplace *RX* en 1 bit hacia la derecha para el próximo el ciclo. Termine la búsqueda cuando *RX* coicide con *RPAT,* o *RX* es 0. En su solución puede usar cualquier componente de Logisim. En el archivo adjunto *find-plantilla.circ* están las componentes usadas en la solución del profesor. En ella se usó la entrada *ENABLE* de los registros para hacer que *RX* y *RPAT* permanezcan constantes una vez concluida la búsqueda, forzando a que *RDY* y *FOUND* no cambien hasta que *START* se ponga en 1 nuevamente. Puede usar otras componentes de Logisim si lo requiere.

Pregunta 2

Parte a.- (2 puntos) Simplifique la siguiente fórmula booleana a su mínima expresión como suma de productos. Justifique su resultado usando álgebra de boole, recurriendo específicamente a las 2 reglas de simplificación de fórmulas vistas en el minuto 37:12 del video de la clase del 19 de marzo.

$$\overline{x}\,\overline{y}\,z\,w + x\,y\,\overline{z}\,w + \overline{x}\,y\,\overline{z}\,\overline{w} + x\,\overline{y}\,z\,w + \overline{x}\,\overline{y}\,z\,\overline{w} + x\,\overline{y}\,z\,\overline{w} + x\,y\,z\,w$$

Ayuda: use mapas de Karnaugh para encontrar las simplificaciones convenientes.

Parte b.- (4 puntos) El siguiente es un programa en assembler Risc-V. Escriba el programa *equivalente* en C sin usar la instrucción **goto** de C. Preocúpese de *reproducir* en C todos los aspectos del programa original en assembler, en particular el valor retornado.

```
incognito: # int incognito(...);
        mv      a5,a0
        beq     a0,zero,.L4
        li      a0,0
.L3:
        lw      a4,0(a5)
        add     a0,a0,a4
        lw      a5,4(a5)
        bne     a5,zero,.L3
        ret
.L4:
        li      a0,0
        ret
```

Ayuda: La función *incognito* usa un *struct*. Compile su programa con gcc -S -O -std=c99 -Wall -pedantic para verificar que su solución al menos no tiene errores sintácticos. Si copia su solución al directorio Ejemplos-RiscV-3 en el archivo *incognito.c*, incluso puede ver el assembler Risc-V generado con "*make incognito.s*". Pero considere que el assembler generado no tiene por qué coincidir con el del enunciado.