ESCUELA SUPERIOR POLITECNICA DEL LITORAL
FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACIÓN

SISTEMAS DIGITALES II

SEGUNDA EVALUACIÓN SEGUNDO TÉRMINO 2011-2012 1 de Febrero del 2012

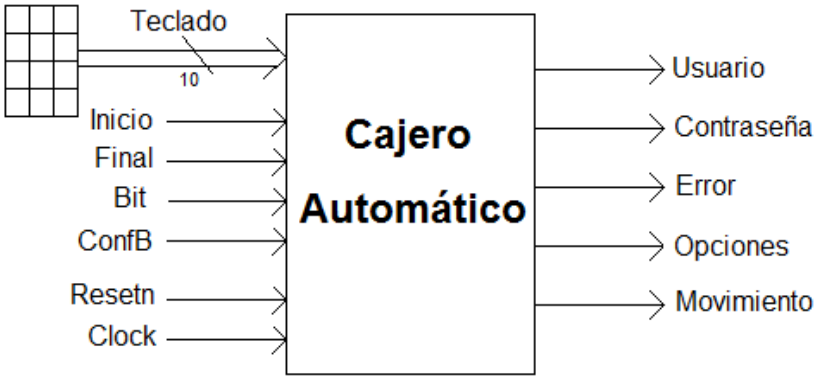
NOMBRE : _____ **PARALELO :** ___

PROBLEMA # 1 (20 p)

Diseñe un pequeño Sistema Digital prototipo para manejo de un Cajero Automático. Al ser un sistema prototipo, sólo se contará con dos usuarios, algunas opciones se ingresaran por teclado y otras mediante un botón (contraseña). Para cada usuario el sistema deberá almacenar 3 bits para la contraseña binaria y 5 bits para el saldo de la cuenta (en binario de 0 a 31 dólares). Estos 8 bits inicialmente deben ser cero.
Para que el sistema arranque debe activarse la entrada **Inicio**, luego de lo cual irá a un estado de activación. Si desde este estado se activa la señal **Final**, regresará al estado inicial.
En el estado de activación debe activarse la salida **Usuario**, la misma que se permanece activa hasta que se presionen "0" o "1" del teclado decimal, lo que representa con cual usuario se desea trabajar (usuario 0 o usuario 1). Después el sistema activa la salida **Contraseña** la misma que continua activada hasta que inicie el proceso de ingreso de la contraseña. La contraseña es un número binario de 3 bits que debe ingresarse bit por bit, para este fin el usuario debe colocar el valor requerido del primer bit (MSB) en la entrada **Bit** y luego presionar y soltar el botón **ConfB**. Este proceso debe repetirse para los siguientes dos bits (uno por uno).
Ahora el sistema debe verificar que la contraseña ingresada sea igual a la almacenada, si la contraseña no es correcta deberá activar la Salida **Error** y regresar a pedir una nueva contraseña (no hay límite de intentos). Si la contraseña es correcta ahora debe activar la salida **Opciones** y esperar a que en el teclado se presione y suelte la correspondiente opción:

- Opción 1: Cambio de contraseña: Se debe activar la salida **Contraseña** y luego ingresar bit por bit la nueva contraseña, al igual que antes.
- Opción 2: Depósito: Se debe activar la salida **Movimiento** y luego presionar y soltar la tecla correspondiente al valor del depósito (máximo $9) que se debe agregar al saldo.
- Opción 2: Retiro: Se debe activar la salida **Movimiento** y luego presionar y soltar la tecla correspondiente al valor del retiro (máximo $9) que se debe restar del saldo inicial.

En cualquier caso y por ser un prototipo simple, no es necesario verificar que el saldo esté entre $0 y $31. El sistema deberá actualizar los valores de saldo y contraseña y luego regresar al estado de activación a esperar un nuevo usuario (es decir solo ejecuta una opción por ingreso).



Presente:

1. **Partición funcional del Cajero automático indicando claramente las señales y componentes utilizados**
2. **Diagrama ASM del Controlador.**

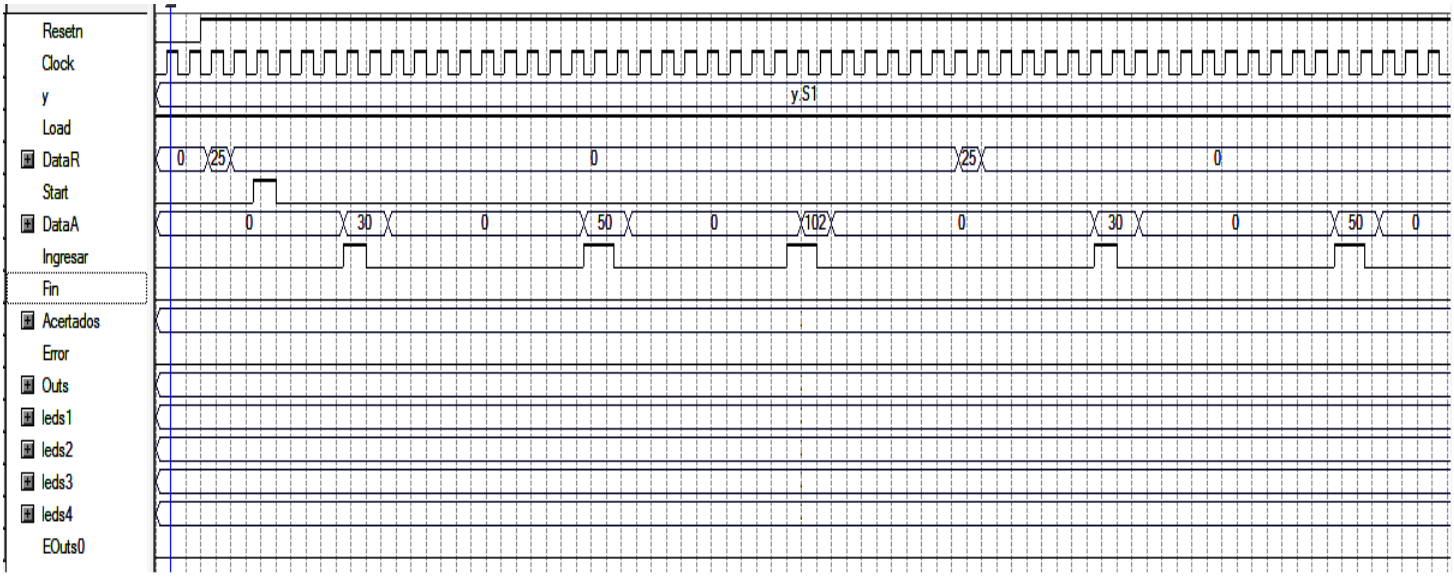
PROBLEMA # 2 (23 p)

Dada la siguiente descripción en VHDL del funcionamiento de un Sistema Digital:
Presentar:

1. **Partición Funcional del Sistema Digital.**
2. **Diagrama ASM del circuito Controlador del Sistema Digital, indicando claramente todas las salidas que deben ser generadas.**
3. **Diagramas de Tiempo del circuito Controlador asumiendo las condiciones de entrada dadas. Indique claramente los nombres y la duración de cada estado (y).**

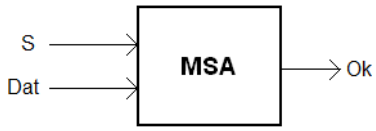
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY Pseudo IS
        PORT(           Clock, Resetn :             IN STD_LOGIC ;
                        Load, Ingresar, Start,Fin:          IN STD_LOGIC ;
                        DataA, DataR :              IN STD_LOGIC_VECTOR(6 DOWNTO 0) ;
                        Acertados:                  BUFFER STD_LOGIC_VECTOR(4 DOWNTO 0);
                        Outs:                       BUFFER STD_LOGIC_VECTOR(6 DOWNTO 0);
                        leds1, leds2, leds3, leds4:         OUT STD_LOGIC_VECTOR(1 TO 7);
                        Error, EOuts0:                      OUT STD_LOGIC ) ;
END Pseudo ;

ARCHITECTURE mixta OF Pseudo IS
        TYPE State_type IS ( S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14); SIGNAL y : State_type ;
        SIGNAL QR, QA, Nseudo : STD_LOGIC_VECTOR(6 DOWNTO 0) ;
        SIGNAL Aciertos, RC:    INTEGER RANGE 0 TO 31;
        SIGNAL AdRAM: STD_LOGIC_VECTOR(4 DOWNTO 0);
        SIGNAL Menora100, Datos32, DataA_OK, AesB, DatoMayor, WR: STD_LOGIC ;
        SIGNAL LR, ER, EA, EC, WE, EDA, LC, EOuts : STD_LOGIC ;
Component shiftrne
        GENERIC (N: INTEGER:=8);
        PORT (          R:              IN      STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                        L, E, W:IN      STD_LOGIC;
                        Clock:  IN      STD_LOGIC;
                        Q:              BUFFER STD_LOGIC_VECTOR(N-1 DOWNTO 0));
end Component;
Component regN
                GENERIC (N: INTEGER:=16);
                PORT (          D:      IN      STD_LOGIC_VECTOR(N-1 DOWNTO 0);
                                Resetn, Clock, E:       IN      STD_LOGIC;
                                Q:              OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
end Component;
Component upcount
                GENERIC (Modulo : INTEGER := 8 ) ;
                PORT (          R :             IN INTEGER RANGE 0 to Modulo-1 ;
                                Clock, Resetn, E, L :   IN STD_LOGIC ;
                                Q :             BUFFER INTEGER RANGE 0 to Modulo-1 ) ;
end Component;
Component RAM_LPM
                PORT ( address: IN STD_LOGIC_VECTOR (4 DOWNTO 0);
                        clock           : IN STD_LOGIC ;
                                data            : IN STD_LOGIC_VECTOR (6 DOWNTO 0);
                                wren            : IN STD_LOGIC ;
                                q               : OUT STD_LOGIC_VECTOR (6 DOWNTO 0));
end Component;
BEGIN
FSM_transitions: PROCESS ( Resetn, Clock )
        BEGIN
                IF Resetn = '0' THEN y <= S1 ;
                ELSIF (Clock'EVENT AND Clock = '1') THEN
                        CASE y IS
```

```
                        WHEN S1 =>  y <= S2 ;
                        WHEN S2 =>  IF start = '0' THEN y <= S2 ; ELSE y <= S3 ; END IF ;
                        WHEN S3 =>  y <= S4 ;
                        WHEN S4 =>  IF Menora100 = '0' THEN y <= S3 ; ELSE y <= S5 ; END IF;
                        WHEN S5 =>  IF Fin = '0' THEN y <= S6 ; ELSE y <= S13 ; END IF ;
                        WHEN S6 =>  IF Ingresar = '0' THEN y <= S5 ; ELSE y <= S7 ; END IF ;
                        WHEN S7 =>  IF Ingresar = '0' THEN y <= S8 ; ELSE y <= S7 ; END IF ;
                        WHEN S8 =>  IF DataA_OK = '0' THEN y <= S10 ; ELSE y <= S9 ; END IF ;
                        WHEN S9 =>  IF DatoMayor = '0' THEN y <= S5 ; ELSE y <= S11 ; END IF ;
                        WHEN S10 =>  IF Ingresar = '0' THEN y <= S10 ; ELSE y <= S7 ; END IF ;
                        WHEN S11 =>  y <= S12 ;
                        WHEN S12 =>  IF Datos32 = '0' THEN y <= S3 ; ELSE y <= S13 ; END IF ;
                        WHEN S13 =>   y <= S14 ;
                        WHEN S14 =>  IF AesB = '0' THEN y <= S14 ; ELSE y <= S1 ; END IF ;
                END CASE ;
        END IF ;
    END PROCESS ;
FSM_outputs: PROCESS ( y, Load,AesB )
    BEGIN
        LR<='0'; ER<='0'; EA<='0'; EC<='0'; WE<='0'; EDA<='0'; LC<='0'; EOuts<='0'; Error<='0';
        CASE y IS
            WHEN S1 =>    IF Load = '1' THEN LR<='1'; ER<='1'; END IF ;
            WHEN S2 =>
            WHEN S3 =>    ER <= '1' ;
            WHEN S4 =>
            WHEN S5 =>
            WHEN S6 =>
            WHEN S7 =>    EA <= '1' ;
            WHEN S8 =>
            WHEN S9 =>
            WHEN S10 =>  Error <= '1' ;
            WHEN S11 =>  WE <= '1' ;
            WHEN S12 =>  EC <= '1' ;
            WHEN S13 =>  EDA<='1'; LC<='1'; EC<='1';
            WHEN S14 =>  EC<='1';EOuts<='1';  IF AesB = '1' THEN LC<='1';EC<='1'; END IF;
        END CASE ;
    END PROCESS ;
-- Procesador de Datos
RC <= 0 ;   WR <= (QR(0) XOR QR(4));   Menora100 <= '1' WHEN (QR < 100) ELSE '0';
DatoMayor <= '1' WHEN (QR < QA) ELSE '0';   DataA_OK <= '1' WHEN (QA < 100) ELSE '0';
Datos32 <= '1' WHEN (Aciertos = 31) ELSE '0';   AesB <= '1' WHEN (AdRAM = Acertados) ELSE '0';
AdRAM <= conv_std_logic_vector(Aciertos, 5);
DespDerecha:shiftrne GENERIC MAP ( N => 7 ) PORT MAP ( DataR, LR, ER, WR, Clock, QR );
RegistroA:      regN     GENERIC MAP ( N => 7 ) PORT MAP ( DataA,Resetn,Clock,EA,QA);
ContadorUP: upcount   GENERIC MAP ( Modulo => 32 ) PORT MAP (RC,Clock, Resetn,EC,LC,Aciertos);
MemoriaRAM:RAM_LPM  PORT MAP (AdRAM,Clock,QA,WE,Nseudo);
RegistroOuts:   regN     GENERIC MAP ( N => 7 ) PORT MAP (Nseudo,Resetn,Clock,EOuts,Outs);
RegAcertados:   regN     GENERIC MAP ( N => 5 ) PORT MAP ( AdRAM,Resetn,Clock,EDA,Acertados);
Presentacion: PROCESS (EOuts)
    BEGIN
        IF EOuts='1' THEN
            MuxOut1 <= Outs(3 Downto 0);
            MuxOut2 <= '0' & Outs(6 Downto 4);
            MuxAcert1 <= AdRAM(3 Downto 0);
            MuxAcert2 <= "000" & AdRAM(4);
        ELSE    MuxOut1 <= "1111";
            MuxOut2 <= "1111";
            MuxAcert2 <= "1111";
            MuxAcert2 <= "1111";
        END IF;
    END PROCESS;
EOuts0<=EOuts;          END mixta ;
```



PROBLEMA # 3 (17 p)

Diseñe una **MSA**, en modo fundamental, que sirve para detectar dos secuencias diferentes en la entrada **Dat**. Si la entrada **S**=0, en **Dat** se debe detectar en forma consecutiva la secuencia "1 0". Si la entrada **S**=1, en **Dat** se debe detectar en forma consecutiva la secuencia "0 1". En ambos casos, si la secuencia detectada es la correcta, se debe activar la salida **Ok** (**Ok**=1) y luego quedar listo para detectar nuevamente cualquiera de las dos secuencias. Si mientras se ingresa una secuencia, cambia el valor de **S**, el sistema debe detectar la respectiva nueva secuencia desde el inicio.



Presentar:

1. Diagrama de Estados Primitivo (Formato: **S Dat / Ok).** Tabla de Estados Primitivo.Tabla de Implicantes. Diagrama de Equivalencia máxima.
2. Diagrama de Estados Reducido. Mapa de asignación de Código de Estados.
3. Mapa de Excitación. Mapas para las variables **Y1** y **Y0** y para la salida **Ok**.
4. Diagrama de tiempo para la salida **Ok** asumiendo los valores de las entradas **S** y **Dat** dados. Indique claramente los periodos de tiempo correspondiente a cada estado de su Diagrama de Estados Reducido.
5. Indique si su circuito corre riesgo de tener **Hazards Estáticos** o no. ¿Como se puede evitar?