Diseño de Circuitos con Dispositivos Lógicos Programables

UART MODO RÁFAGA

Diseño, Integración y Verificación de periférico en laRVa

Valladolid – Noviembre 2025

Iván Herrero Alonso, Lucía López García, Víctor Rueda Domínguez

CONTENIDOS

01 

Objetivo

02 

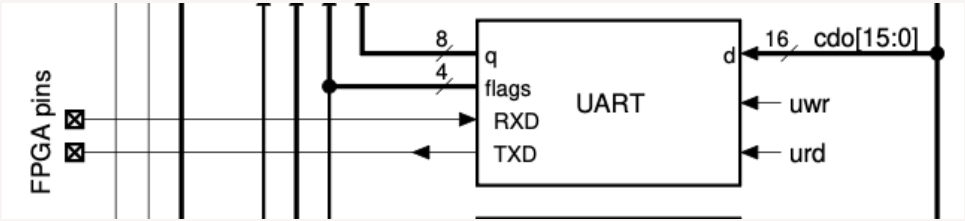
UART aislada

Fundamento
Componentes
Modificaciones
Cronogramas

03 

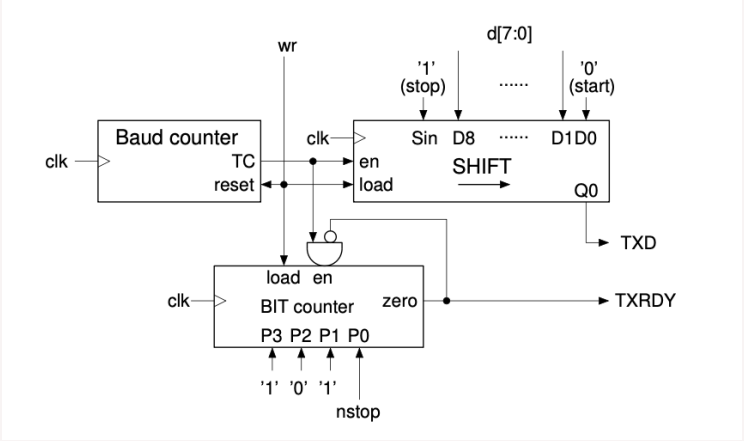
UART integrada

Fundamento
Componentes
Modificaciones
Cronogramas



PROBLEMA

La UART base es ineficiente. Requiere 4 operaciones de store de la CPU para enviar una palabra de 32 bits



SOLUCIÓN

Diseñar una UART que soporte “modo ráfaga” para optimizar transmisión de datos. Realizará una sola operación de store para 32 bits para reducir tráfico de bus E/S.

NUEVO PERIFÉRICO UARTB

La nueva UART podrá soportar dos modos para enviar, que se configurará con un registro llamado MODO

Modo Normal

CPU store(8 bits) → UART
CPU store(8 bits) → UART
CPU store(8 bits) → UART
CPU store(8 bits) → UART

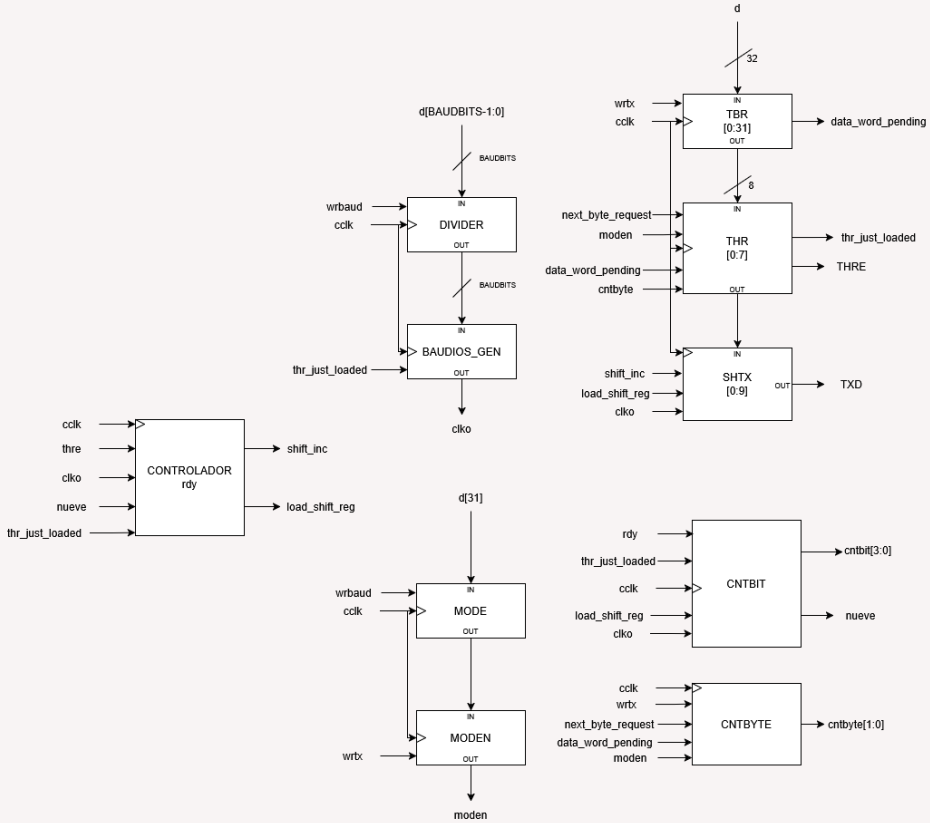
4 operaciones por palabra:
Lento y Alta carga de CPU

Modo Ráfaga

CPU store(32 bits) → UARTB → Envío de 4 bytes
Envío separado por bit STOP y bit START

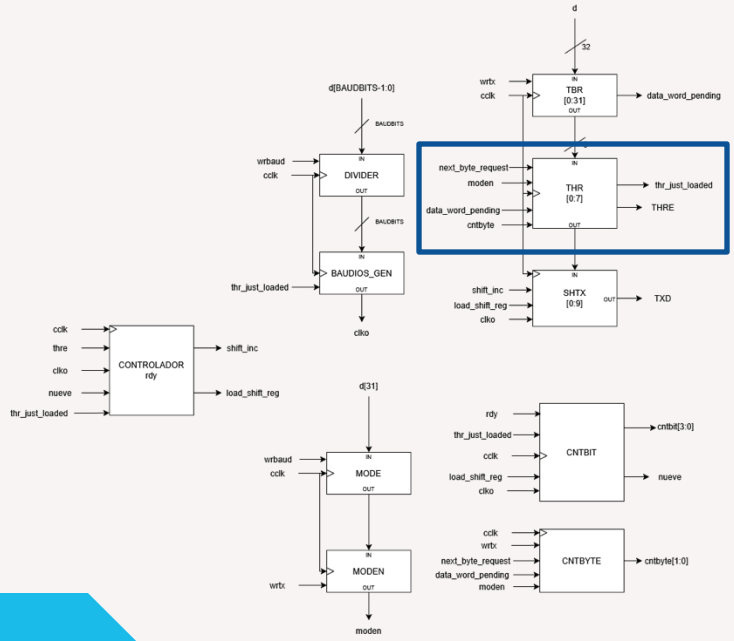
1 operación por palabra:
Rápido y Mínima carga de CPU

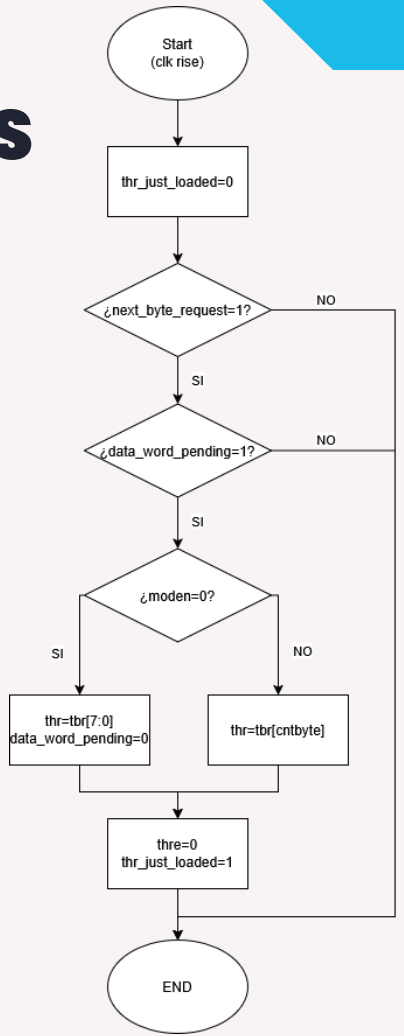
COMPONENTES TRANSMISOR



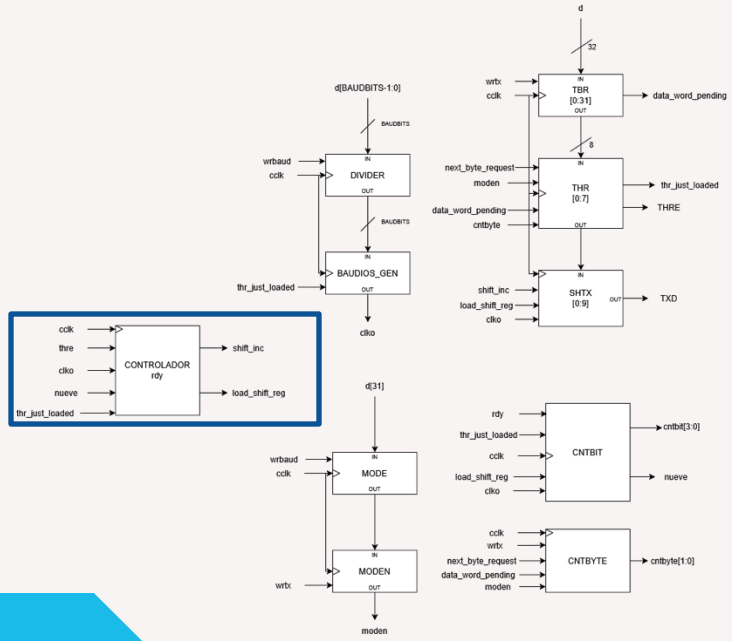
shift_inc = clko & (~rdy) & (nueve)

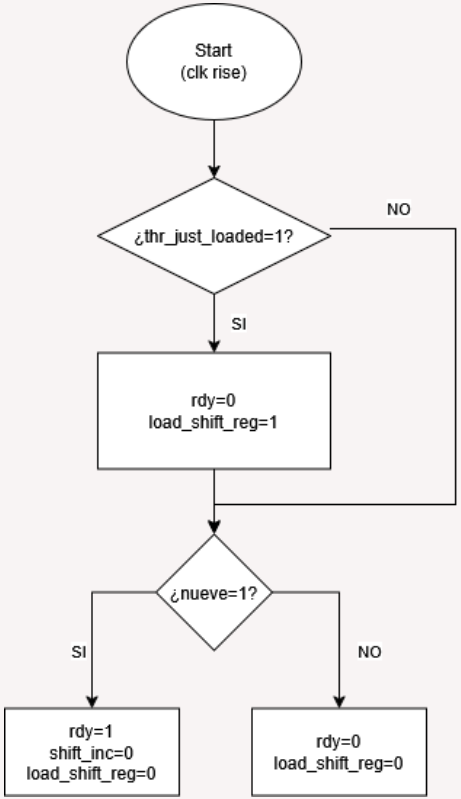
COMPONENTES TRANSMISOR





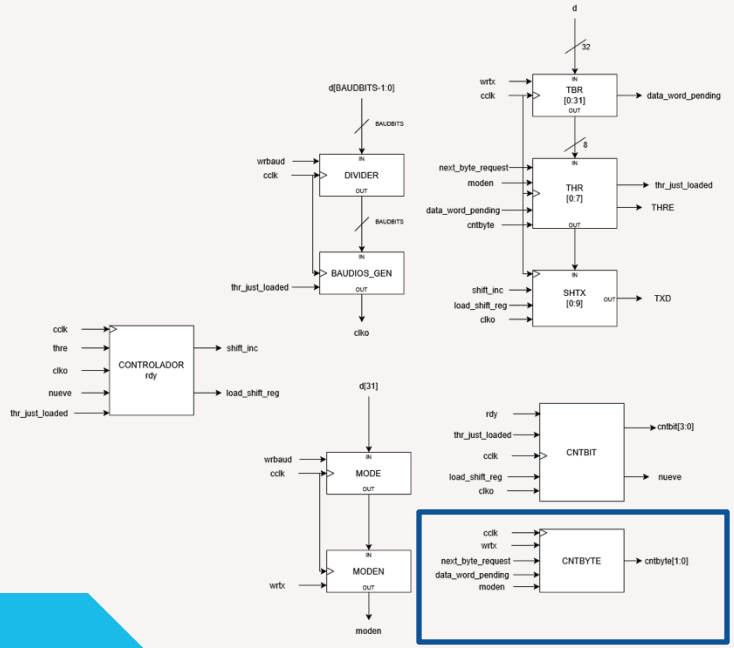
COMPONENTES TRANSMISOR

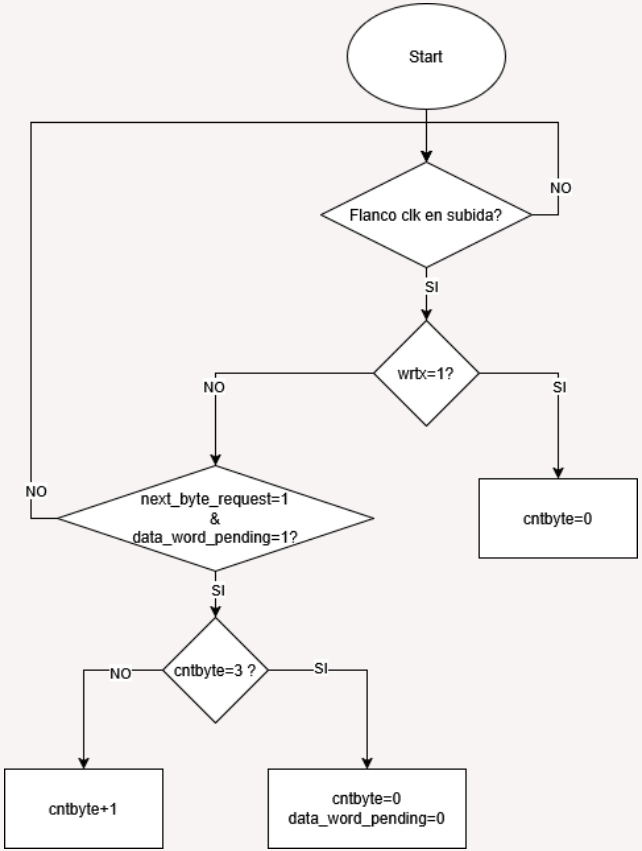




shift_inc = clko & (~rdy) & (nueve)

COMPONENTES TRANSMISOR





MODIFICACIONES

```
// Registros para implementar modos
reg [31:0]tbr;
reg mode;
reg moden;
reg [1:0] cntbyte;
reg data_word_pending = 1'b0;
reg thr_just_loaded = 1'b0;
```

```
// Lógica de extracción de byte de TBR a THR
wire next_byte_request;
assign next_byte_request = thre & rdy;

  always @(posedge clk) begin
    thr_just_loaded <= 1'b0;
    if (next_byte_request) begin
      if (data_word_pending) begin
```

MODO NORMAL

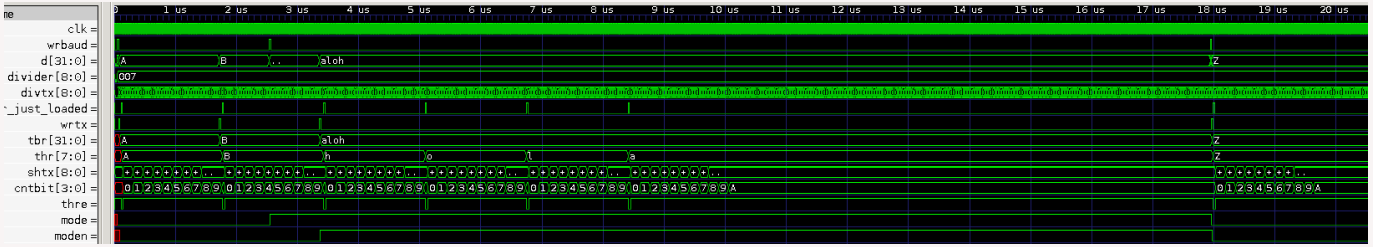
```
        if (moden == 1'b0) begin

`ifdef SIMULATION
                $write ("%c",tbr&255);
                $fflush ( );
`endif
                thr <= tbr[7:0];
                thre <= 1'b0; // THR ocupado
                thr_just_loaded <= 1'b1;
                data_word_pending <= 1'b0;
            end
```

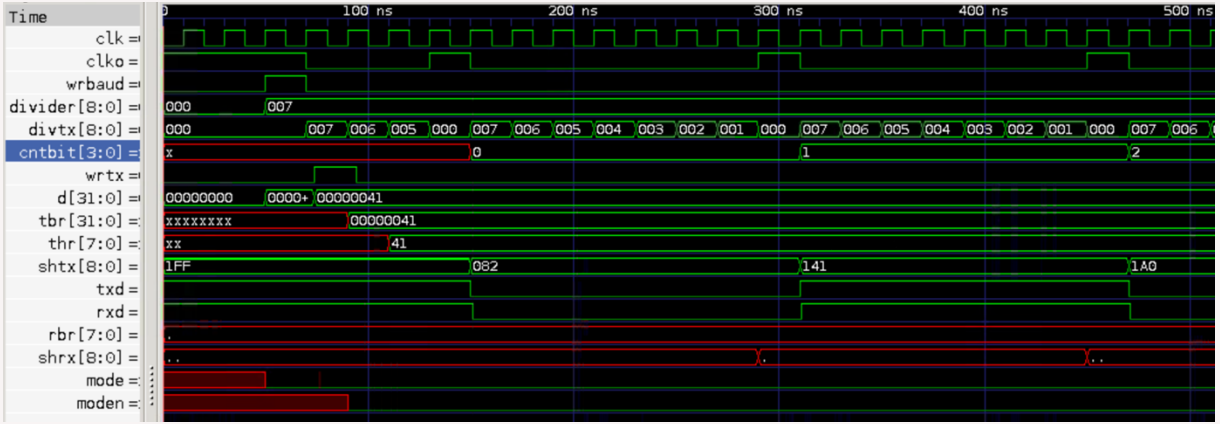
MODO RÁFAGA

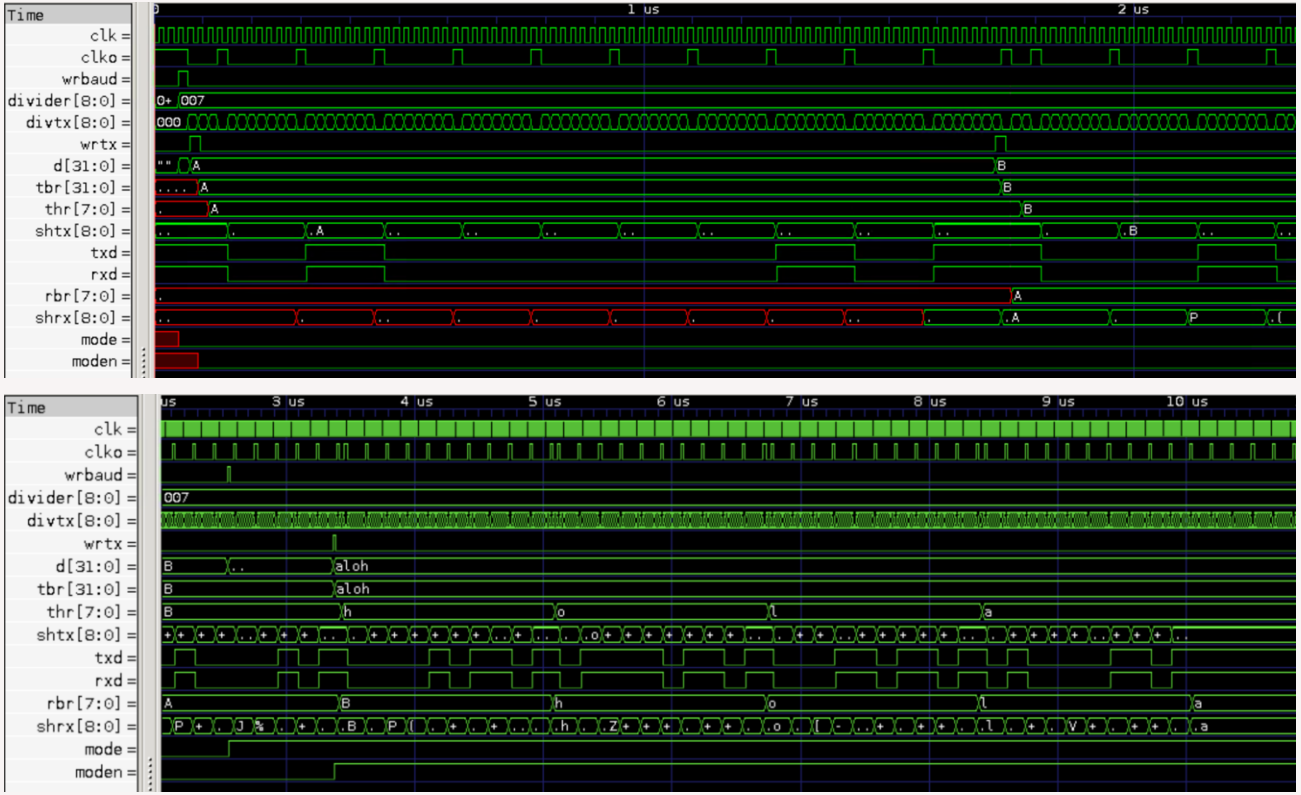
```
else begin

  case (cntbyte)
        2'b00: thr <= tbr[7:0];
        2'b01: thr <= tbr[15:8];
        2'b10: thr <= tbr[23:16];
        2'b11: thr <= tbr[31:24];
        default: thr <= 8'hXX;
    endcase

    thre <= 1'b0;
    thr_just_loaded <= 1'b1;

    if (cntbyte == 2'b11) begin
        data_word_pending <= 1'b0;
        cntbyte <= 2'b00;
    end else begin
        cntbyte <= cntbyte + 1;
    end
end
```

CRONOGRAMA

MODO normal: envío A, B - MODO ráfaga: envío hola - MODO normal: envío Z

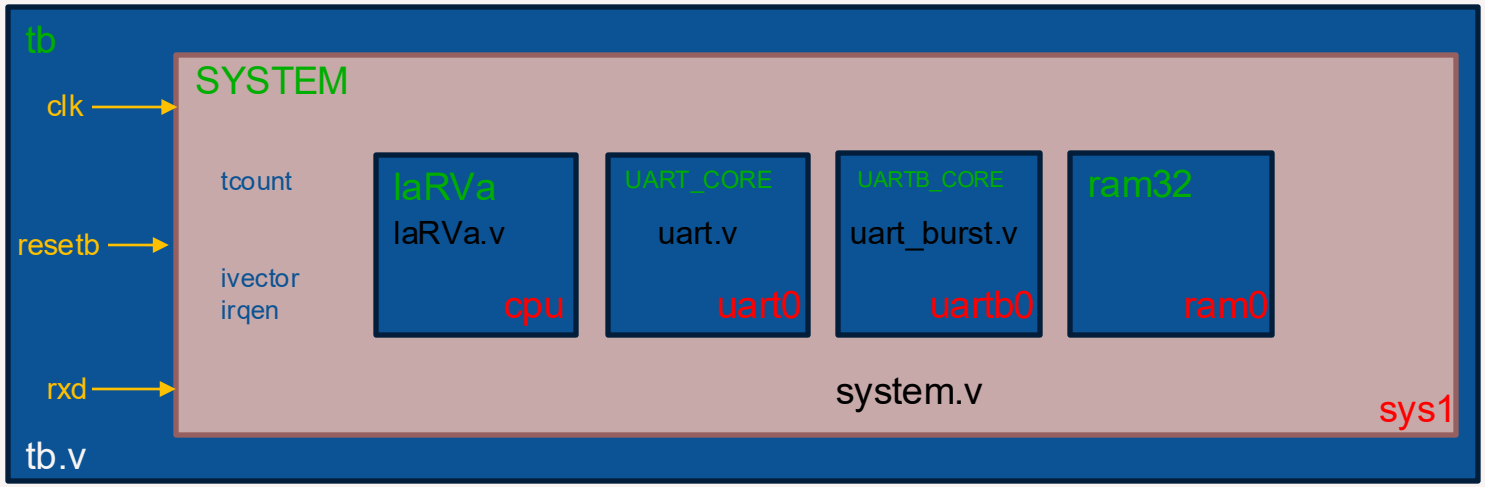





Time
clk
clko
wrbaud
divider[8:0]
divtx[8:0]
wrtx
d[31:0]
tbr[31:0]
thr[7:0]
shtx[8:0]
txd
rxd
rbr[7:0]
shrx[8:0]
mode
moden
1 us
2 us
3 us
4 us
5 us
6 us
7 us
8 us
9 us
10 us
007
aloh
h
o
l
a

INTEGRACIÓN DE LA UART

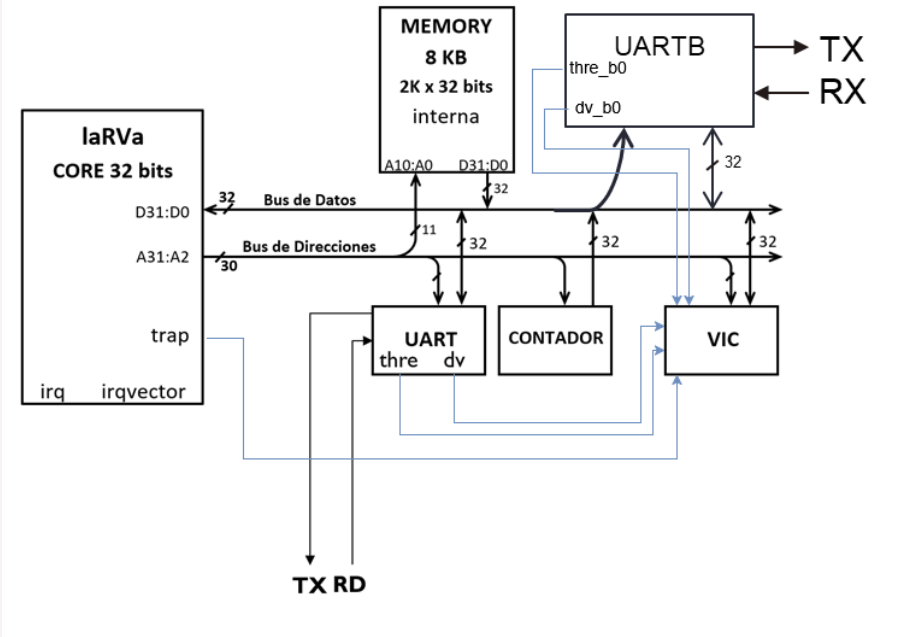
La nueva UART se incorporará en el sistema LaRVa de la siguiente manera:



- Módulo (clase)
- Objeto (materialización)
- Fichero
- Otras variables equivalentes a bloques
- Puertos (estímulos)

HARDWARE DE LA INTEGRACIÓN

La nueva UART se incorporará en la dirección 0xE0000080 del sistema LaRVa y constará de dos puertos, por lo que ocupará el rango de direcciones hasta el 0XE0000087



MODIFICACIONES

Periféricos

```
wire uartb0cs;
assign uartb0cs = iocs & (ca[7:5]==3'b100);
always @* begin
  casex (ca[7:2])
    // --- UART0 original ----------------------
    6'b000xx0: iodo <= {24'hx, uart_do};
    6'b000xx1: iodo <= {27'hx, ove, fe, tend,
                               thre, dv};

    // --- Nueva UARTB0 ------------------------
    6'b100xx0: iodo <= {24'hx, uartb0_do};
    6'b100xx1: iodo <= {27'hx, ove_b0, fe_b0,
                               tend_b0, thre_b0,
                               dv_b0};

    // --- Timer e IRQEN -----------------------
    6'b011xxx: iodo <= tcount;
    6'b111xxx: iodo <= {28'hx, irqen};

    default:  iodo <= 32'hxxxxxxxx;
  endcase
end
```

Mapeo de registros

```
assign uwrtx_b0    = uartb0cs & (~ca[2]) & mwe[0];
assign urd_b0      = uartb0cs & (~ca[2]) & (mwe==4'b0000);
assign uwrbaud_b0 = uartb0cs & ( ca[2]) & mwe[0] & mwe[1];
```

Nueva UARTB

```
assign rxd_b0 = txd_b0;
```

```
UARTB_CORE #(.BAUDBITS(BAUDBITS)) uartb0 (
    .clk     (cclk       ),
    .txd     (txd_b0     ),
    .rxd     (rxd_b0     ),
    .d       (cdo        ),
    .wrtx    (uwrtx_b0   ),
    .wrbaud  (uwrbaud_b0),
    .rd      (urd_b0     ),
    .q       (uartb0_do  ),
    .dv      (dv_b0      ),
    .fe      (fe_b0      ),
    .ove     (ove_b0     ),
    .tend    (tend_b0    ),
    .thre    (thre_b0    )
```

MODIFICACIONES

Control de interrupciones

Registro de habilitación de interrupciones

```
reg [4:0] irqen = 5'b0;

always @(posedge cclk or posedge reset) begin
    if (reset) begin
        irqen <= 5'b0;
    end else if (irqcs & (~ca[4]) & mwe[0]) begin
        irqen <= cdo[4:0];
    end
end
```

Encoder de prioridad

```
reg [2:0] vecn;
always @* begin
    if (trap) begin
        vecn = 3'b000;
    end else if (irqpen[0]) begin
        vecn = 3'b001;
    end else if (irqpen[1]) begin
        vecn = 3'b010;
    end else if (irqpen[2]) begin
        vecn = 3'b011;
    end else if (irqpen[3]) begin
        vecn = 3'b100;
    end else begin
        vecn = 3'bxxx;
    end
end
```

Señales de petición de interrupción

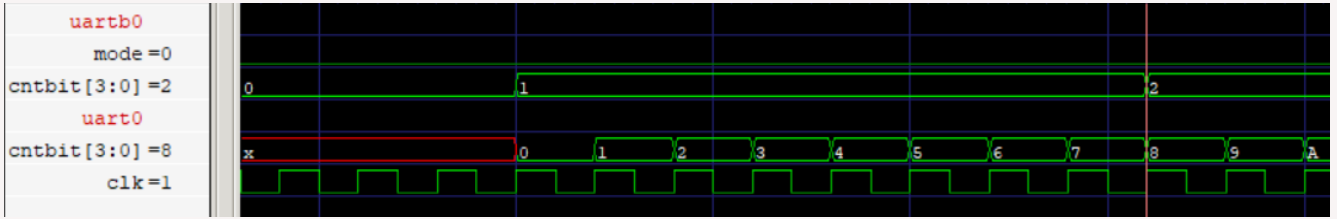
```
wire [4:0] irqpen;
assign irqpen[0] = irqen[0] & dv;
assign irqpen[1] = irqen[1] & thre;
assign irqpen[2] = irqen[2] & dv_b0;
assign irqpen[3] = irqen[3] & thre_b0;
assign irqpen[4] = 1'b0;
```

Vectores de interrupción

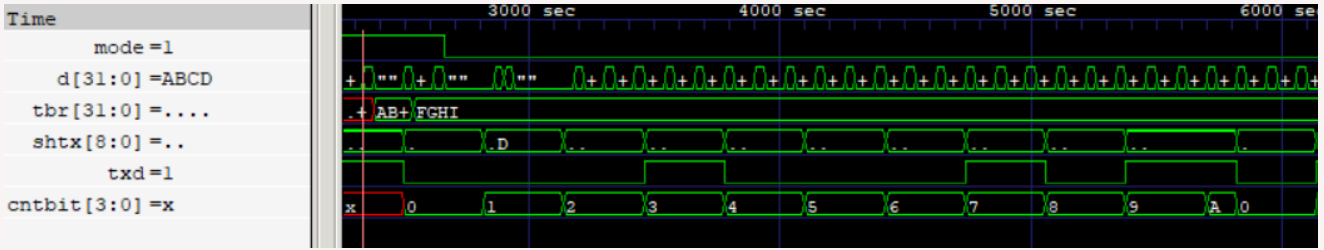
```
reg [31:2] irqvect [0:4];
always @(posedge cclk) begin
    // Vectores 0..3 mapeados en 0xE00000F0-0xE00000FC
    if (irqcs & ca[4] & (mwe==4'b1111))
        irqvect[ca[3:2]] <= cdo[31:2];

    // Vector 4 mapeado en 0xE00000E4
    if (irqcs & (~ca[4]) & (ca[3:2]==2'b01) & (mwe==4'b1111))
        irqvect[4] <= cdo[31:2];
end
```

CRONOGRAMAS

Decremento en 8 de la velocidad de la UARTB frente a la UART y activación modo 0



Establecer modo ráfaga enviando ABCD y después FGHI



Diseño de Circuitos con Dispositivos Lógicos Programables

GRACIAS POR SU ATENCIÓN

Iván Herrero Alonso, Lucía López García, Víctor Rueda Domínguez