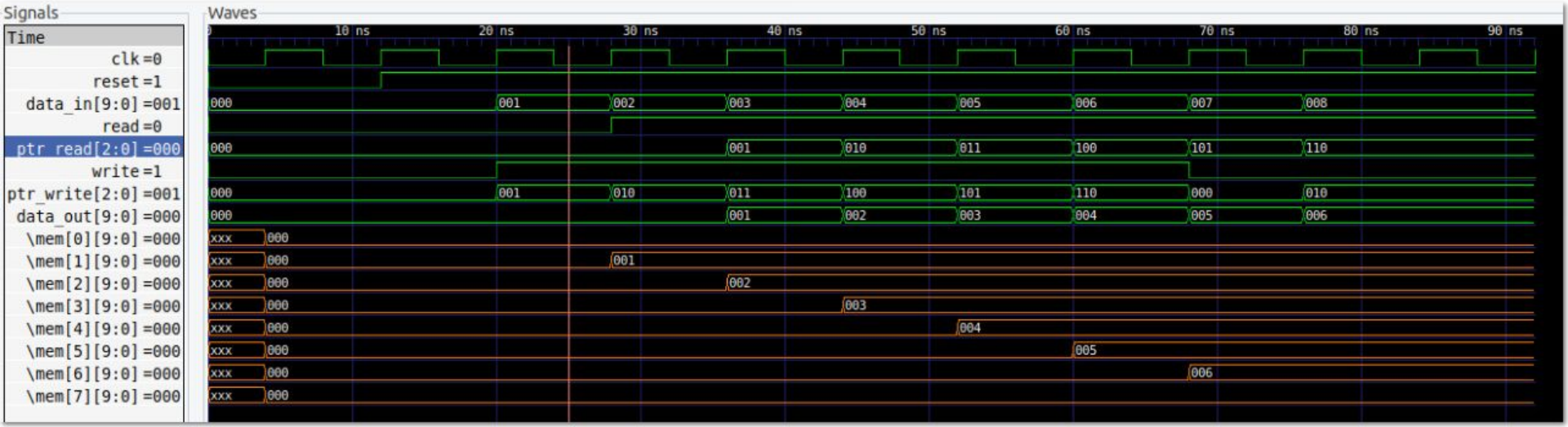
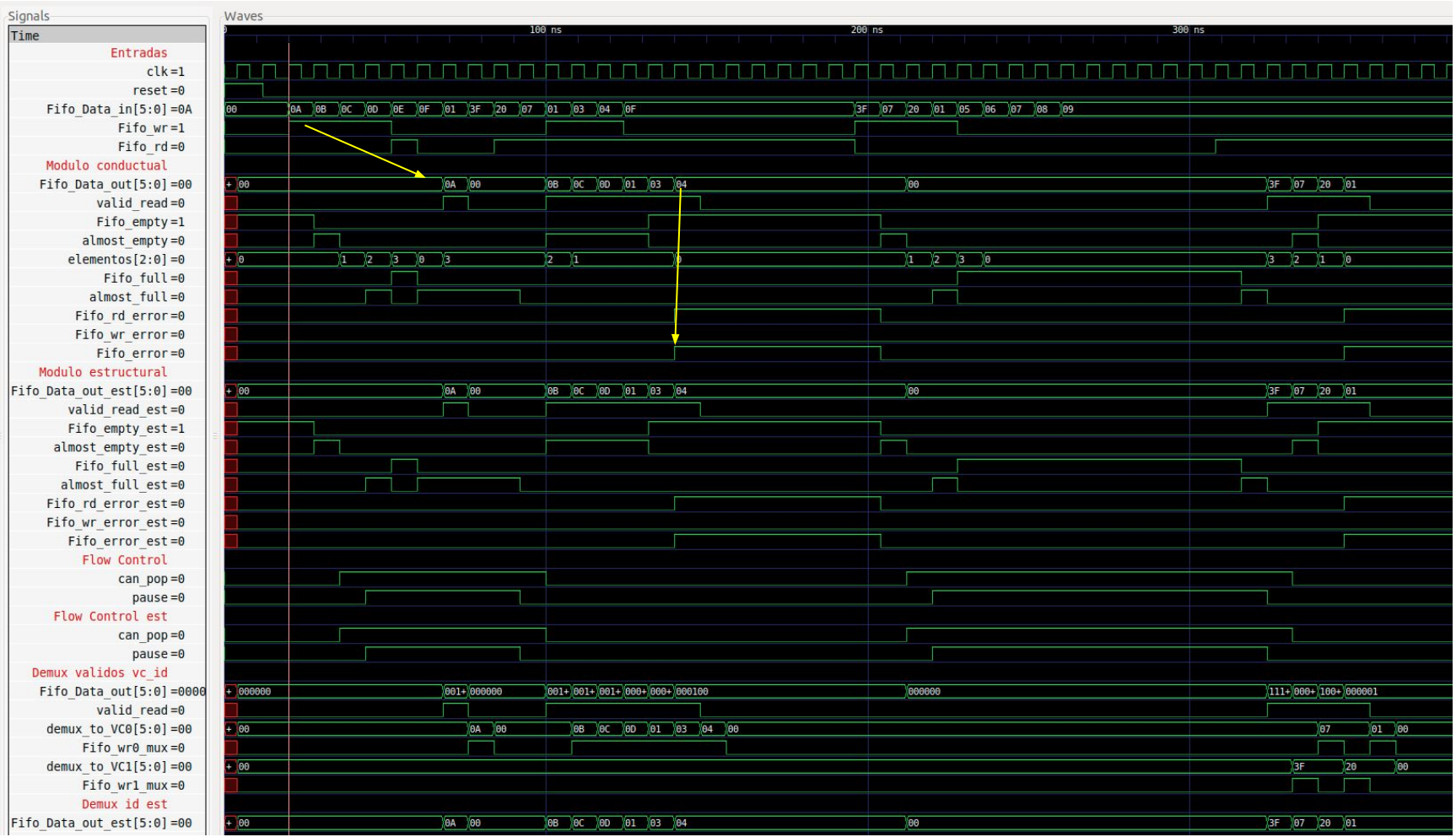
Proyecto 2: Capa Transacción PCIE

Por: Luis Diego Pacheco Chamberlain
Sebastián Palacino Chacón

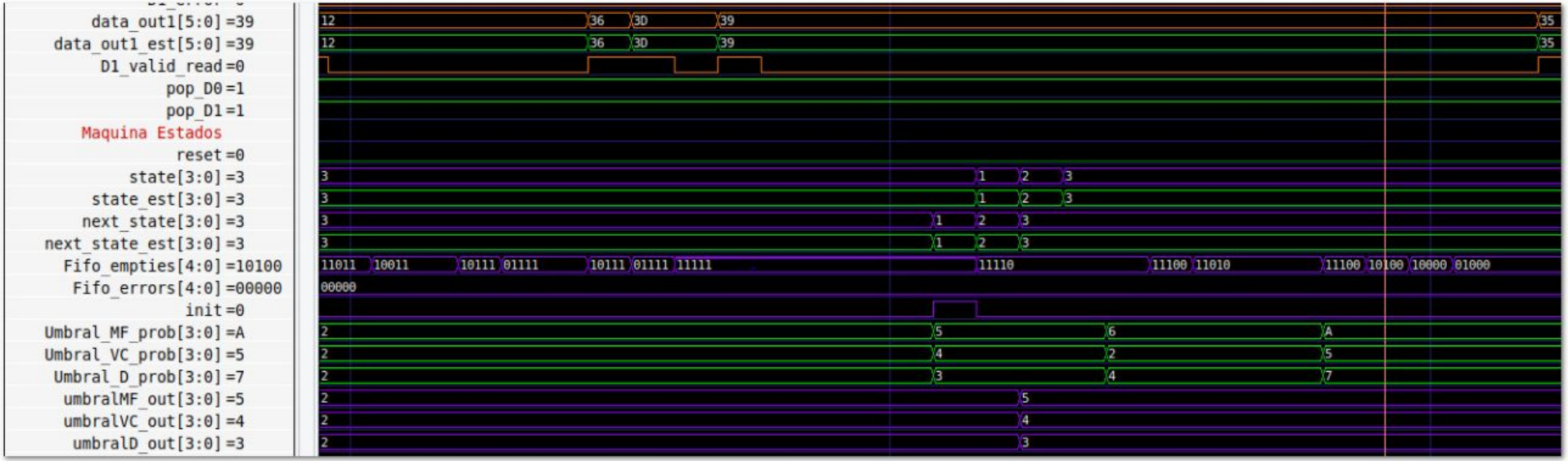
Memoria


Signals
Waves
Time
clk =0
reset =1
data_in[9:0] =001
read =0
ptr_read[2:0] =000
write =1
ptr_write[2:0] =001
data_out[9:0] =000
\mem[0][9:0] =000
\mem[1][9:0] =000
\mem[2][9:0] =000
\mem[3][9:0] =000
\mem[4][9:0] =000
\mem[5][9:0] =000
\mem[6][9:0] =000
\mem[7][9:0] =000
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns
90 ns

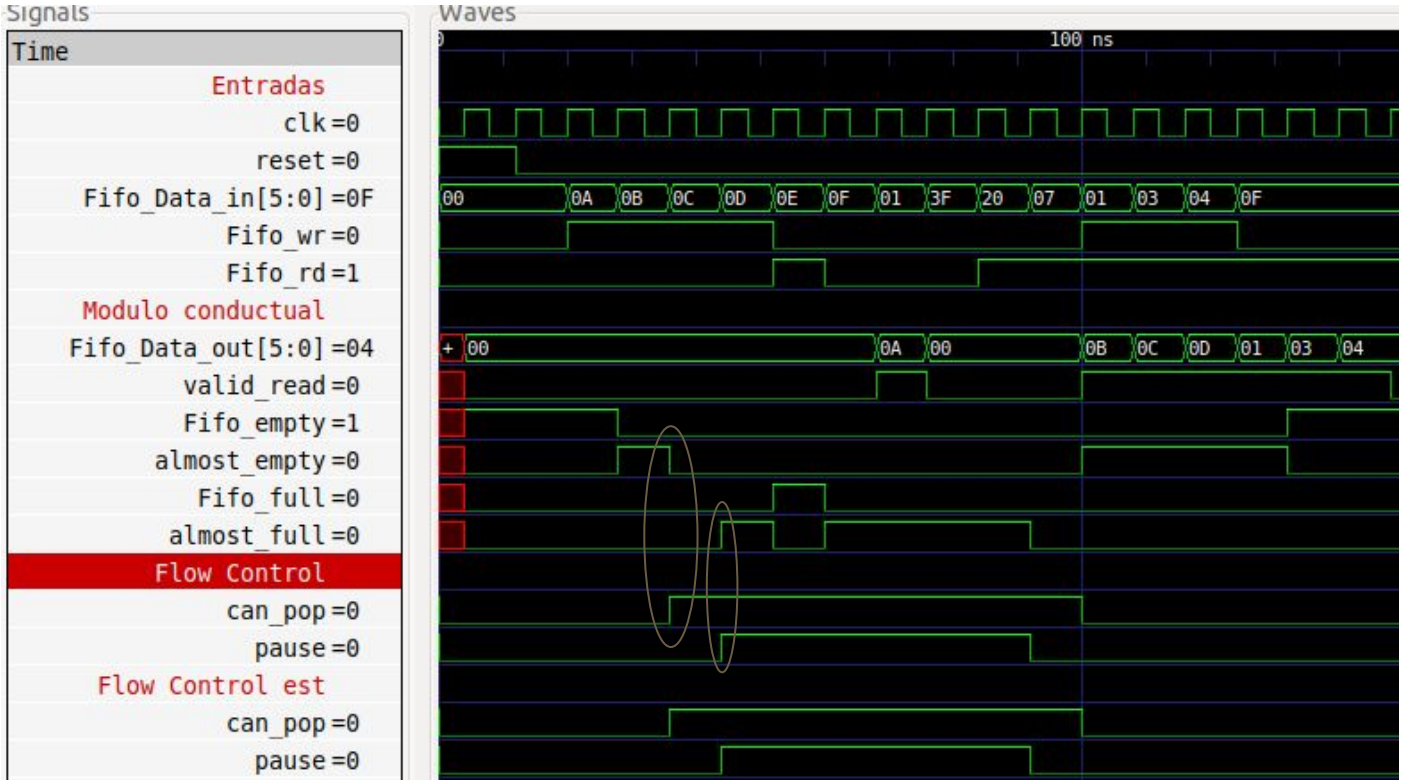
Fifo


Signals
Waves
Time
Entradas
clk =1
reset =0
Fifo_Data_in[5:0] =0A
Fifo_wr =1
Fifo_rd =0
Modulo conductual
Fifo_Data_out[5:0] =00
valid_read =0
Fifo_empty =1
almost_empty =0
elementos[2:0] =0
Fifo_full =0
almost_full =0
Fifo_rd_error =0
Fifo_wr_error =0
Fifo_error =0
Modulo estructural
Fifo_Data_out_est[5:0] =00
valid_read_est =0
Fifo_empty_est =1
almost_empty_est =0
Fifo_full_est =0
almost_full_est =0
Fifo_rd_error_est =0
Fifo_wr_error_est =0
Fifo_error_est =0
Flow Control
can_pop =0
pause =0
Flow Control est
can_pop =0
pause =0
Demux validos vc_id
Fifo_Data_out[5:0] =0000
valid_read =0
demux_to_VC0[5:0] =00
Fifo_wr0_mux =0
demux_to_VC1[5:0] =00
Fifo_wr1_mux =0
Demux id est
Fifo_Data_out_est[5:0] =00
100 ns
200 ns
300 ns

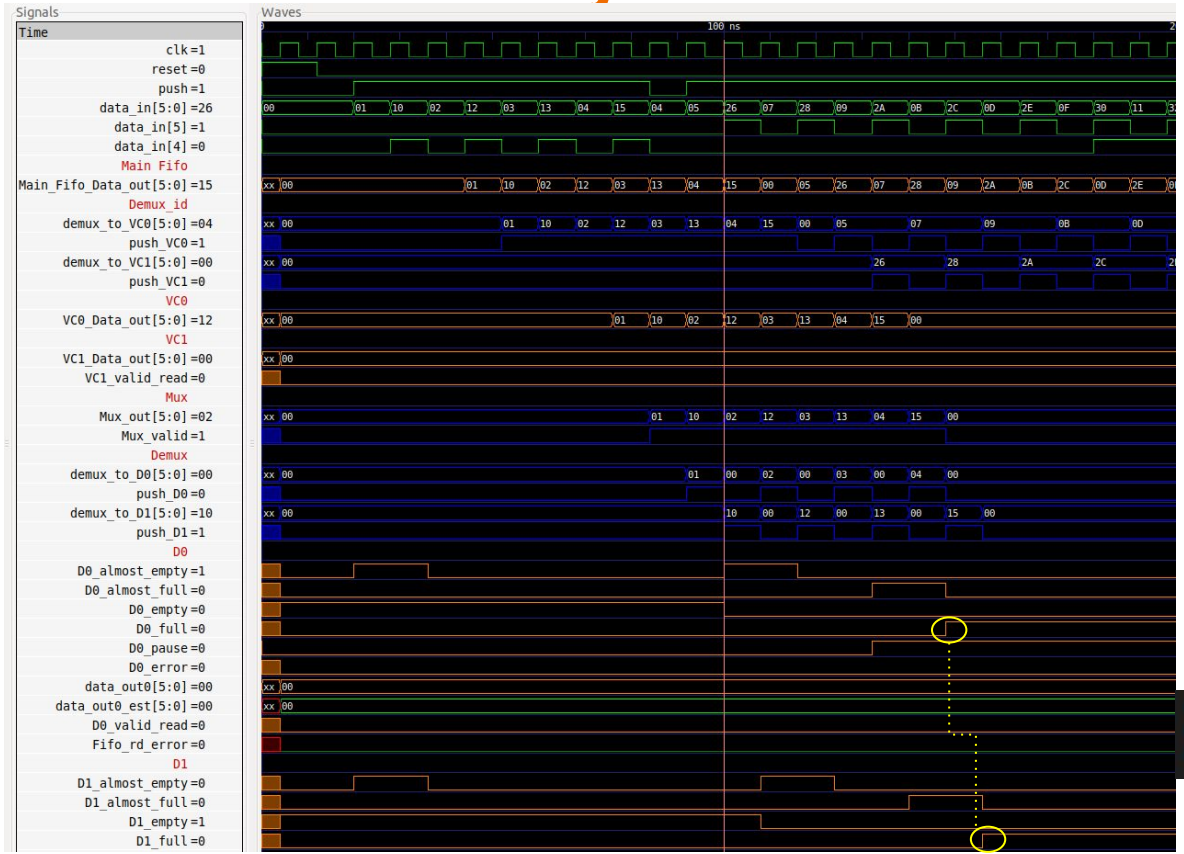
Máquina de Estados



Flow Control

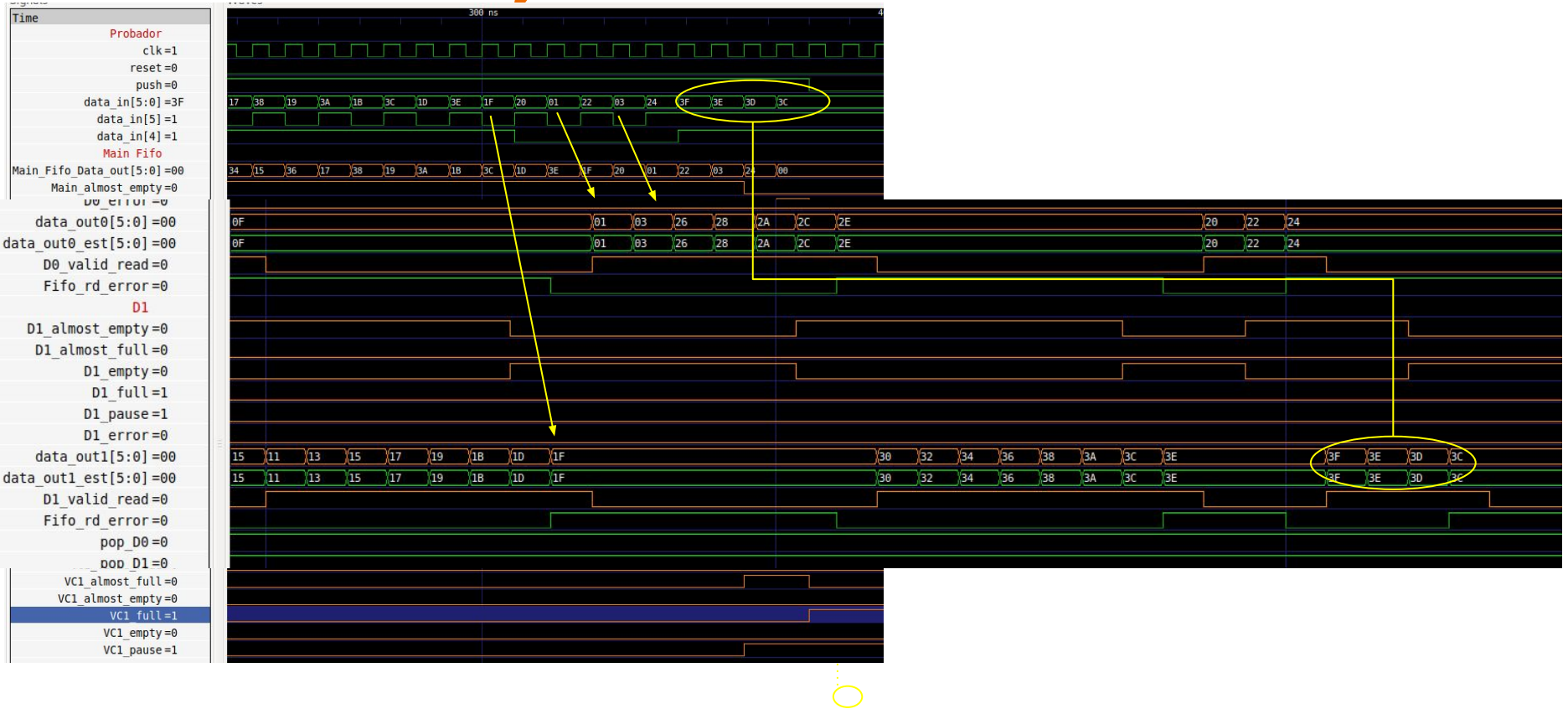


Prueba: Llenado y vacío

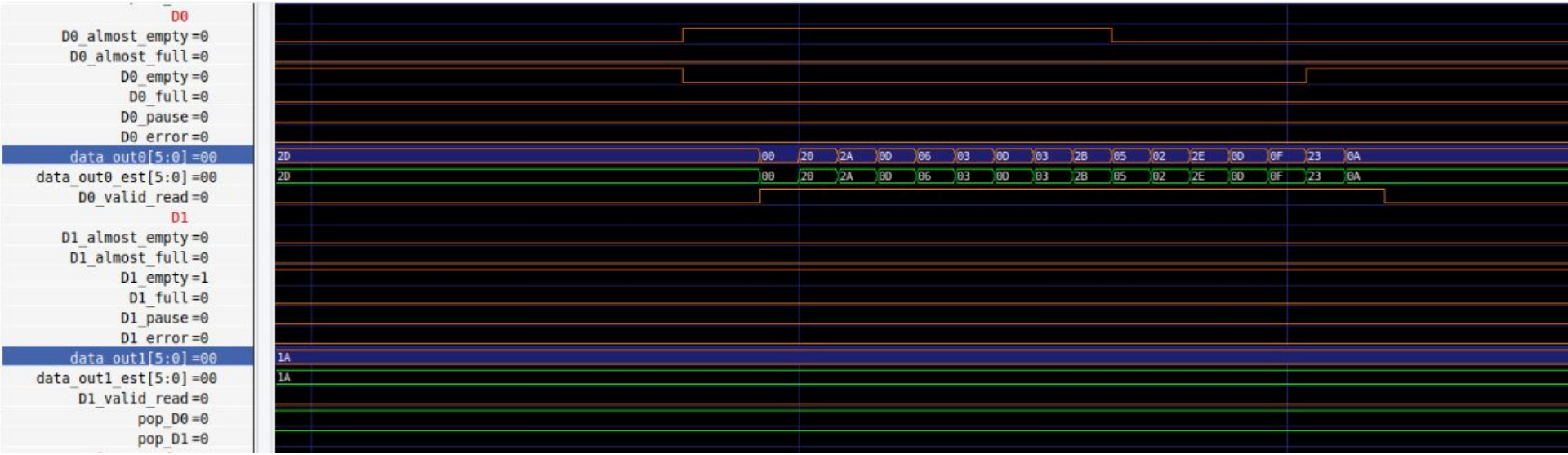



```
assign pop_main = !(VC0_pause || VC1_pause) && !(Main_fifo_empty);
assign pop_VC0 = !(D0_pause || D1_pause) && !(VC0_empty);
assign pop_VC1 = !(D0_pause || D1_pause) && !(VC1_empty) && (VC0_emp
```

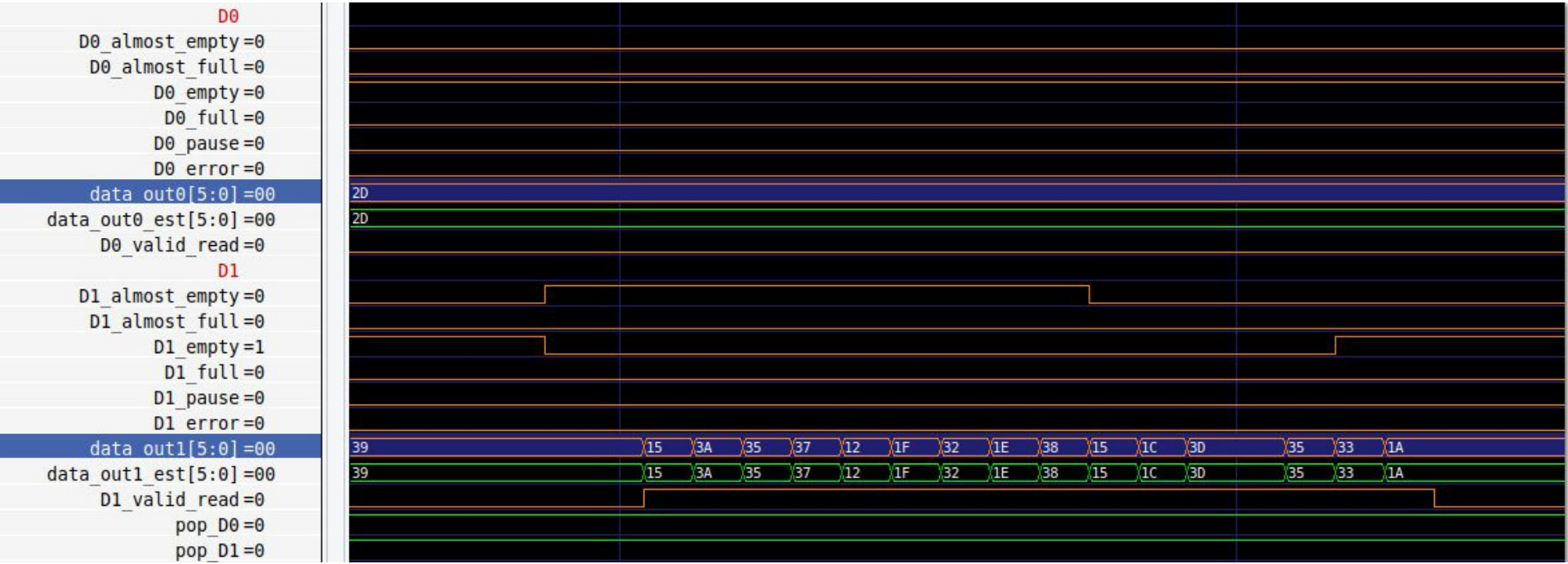
Prueba: Llenado y vacío



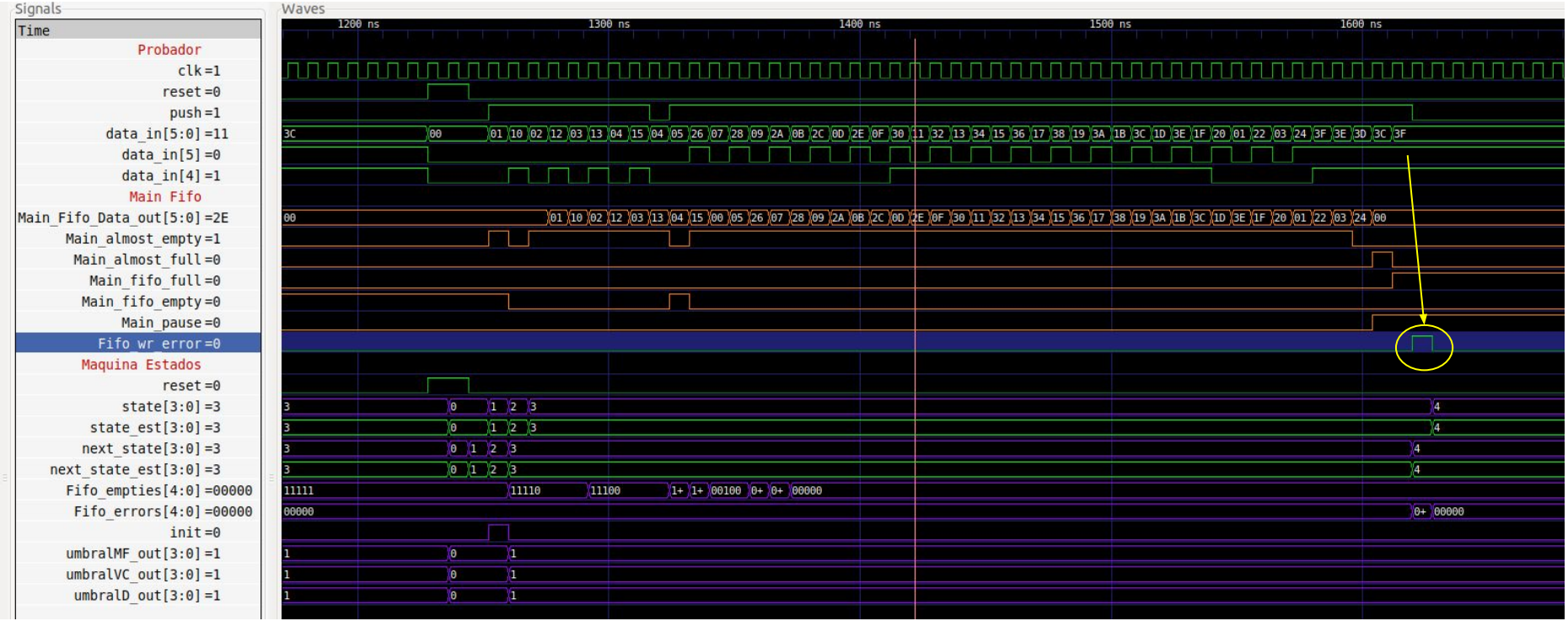
Prueba: Único tipo de tráfico



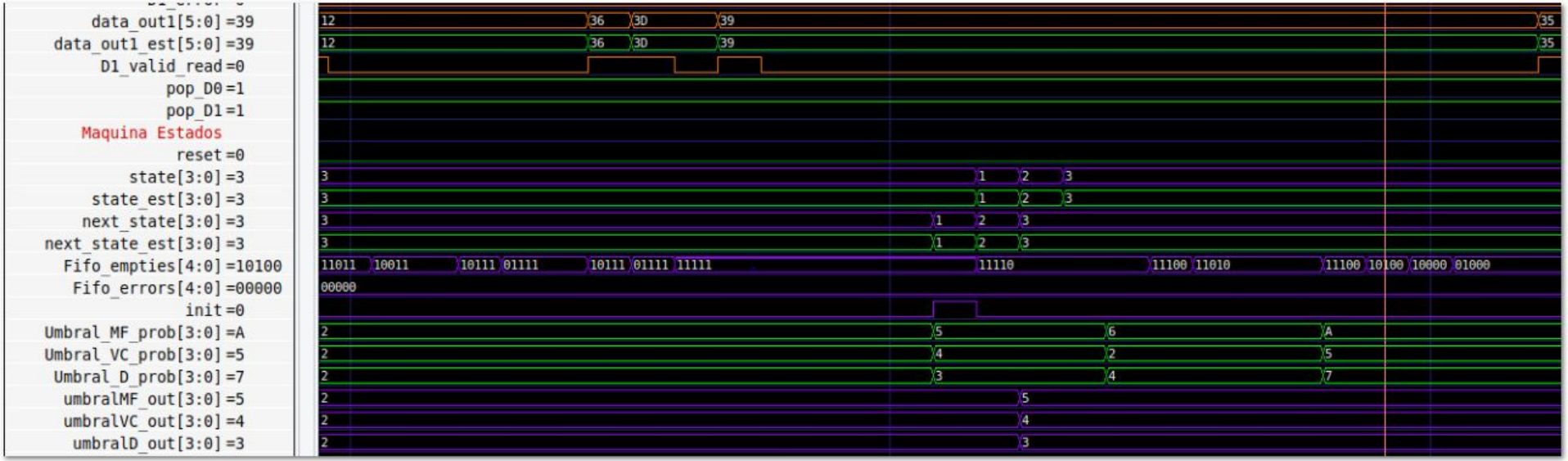
Prueba: Único tipo de tráfico



Prueba: Errores



Prueba: Umbrales distintos



Latencia y Tasa de Datos

- Se duran 18 ciclos de reloj de un dato de entrada a salida. Cada ciclo dura 8ns, por ende, la Latencia es de **144 ns**
- 24 bits logran salir en 32 ns, lo cual da una tasa de datos de **0.75 bits/ns**

Particularidades y Desafíos

- Durante la interconexión de los módulos se presentaron errores en la máquina de estados. Esto causó un problema para los estudiantes que ameritó que se reunieran para resolverlo.
- En el FIFO se utilizó un “=” dentro de un always @(posedge clk) que no se encontró hasta que todo estaba conectado por lo que se tuvo que hacer una corrección y acomodar esa lógica en un always @(*).

```
if (Fifo_wr && !Fifo_full) begin
    wr_ptr <= wr_ptr + 1;
    //contador = contador +1;
    Fifo_wr_error <= 0;
    q0 <= Fifo_Data_in;
end
```

```
always @(*) begin
    cont = conor;
    if(Fifo_rd && !Fifo_empty)
        cont = cont - 1;
    if (Fifo_wr && !Fifo_full)
        cont = cont + 1;
end
```

Conclusiones y Recomendaciones

- Se utilizó métodos de comunicación asertiva para repartición de tareas y solucionar problemas que se presentaron durante el proyecto.
- Se recomienda a las personas que deseen implementar un proyecto igual que se utilice el método de valid.
- Además de esto se recomienda un buen trabajo de equipo para poder aprovechar al máximo las fortalezas y debilidades de cada integrante.

Bitácora

- 24/10/2020: Luis Diego y Sebastián se reunieron para discutir el plan de pruebas y asignar tareas. Sebastián se encargará de la memoria y Luis Diego del FIFO.
- 26/10/2020: Sebastián subió la memoria al Git
- 27/10/2020: Luis Diego subió el FIFO al Git
- 01/11/2020: Luis Diego y Sebastián se reunieron para discutir los siguientes pasos del proyecto. Sebastián se encargará de la máquina de estados y Luis Diego del Flow Control.
- 03/11/2020: Sebastián subió la máquina de estados al Git.
- 03/11/2020: Luis Diego subió el Flow Control al Git.
- 05/11/2020: Sebastián y Luis Diego se reunieron para discutir las siguientes etapas del proyecto.
- 06/11/2020: Sebastián arregló errores en la máquina de estados

Bitácora

- 06/11/2020: Luis Diego realizó la interconexión y la subió al Git.
- 09/11/2020: Luis Diego arregló unos errores en la máquina de estados.
- 11/11/2020: Luis Diego y Sebastián se repartieron las pruebas. Luis Diego se encargará de las pruebas A y C, y Sebastián de las B, D y E.
- 12/11/2020: Luis Diego y Sebastián se reunieron para analizar errores dentro del código
- 13/11/2020: Sebastián logró realizar la prueba B.
- 14/11/2020: Sebastián logró realizar la prueba D y E. Luis Diego logró realizar la A y C.
- 24/11/2020: Luis Diego y Sebastián crearon la presentación del proyecto.

- **QoS (Calidad de Servicio):** Medición del desempeño de un producto o servicio desde el punto de vista de los usuarios. Medida cualitativa. Toma en cuenta factores como uso de recursos, tráfico, pérdida de paquetes.
- **Arbitraje en sistemas digitales:** dispositivos electrónicos que se encargan de alocar el acceso a recursos compartidos.
- **Priority Flow Control:** Un mecanismo que permite selectivamente pausar el tráfico de paquetes que proviene de un bus para darle prioridad a otro con una clase distinta.
- **¿Cómo se relacionan los créditos con Flow Control?:** Los créditos establecen cuánto espacio hay disponible en un buffer para que se le puedan enviar la cantidad exacta de datos necesarios sin que haya una señal cuando el buffer se llena.