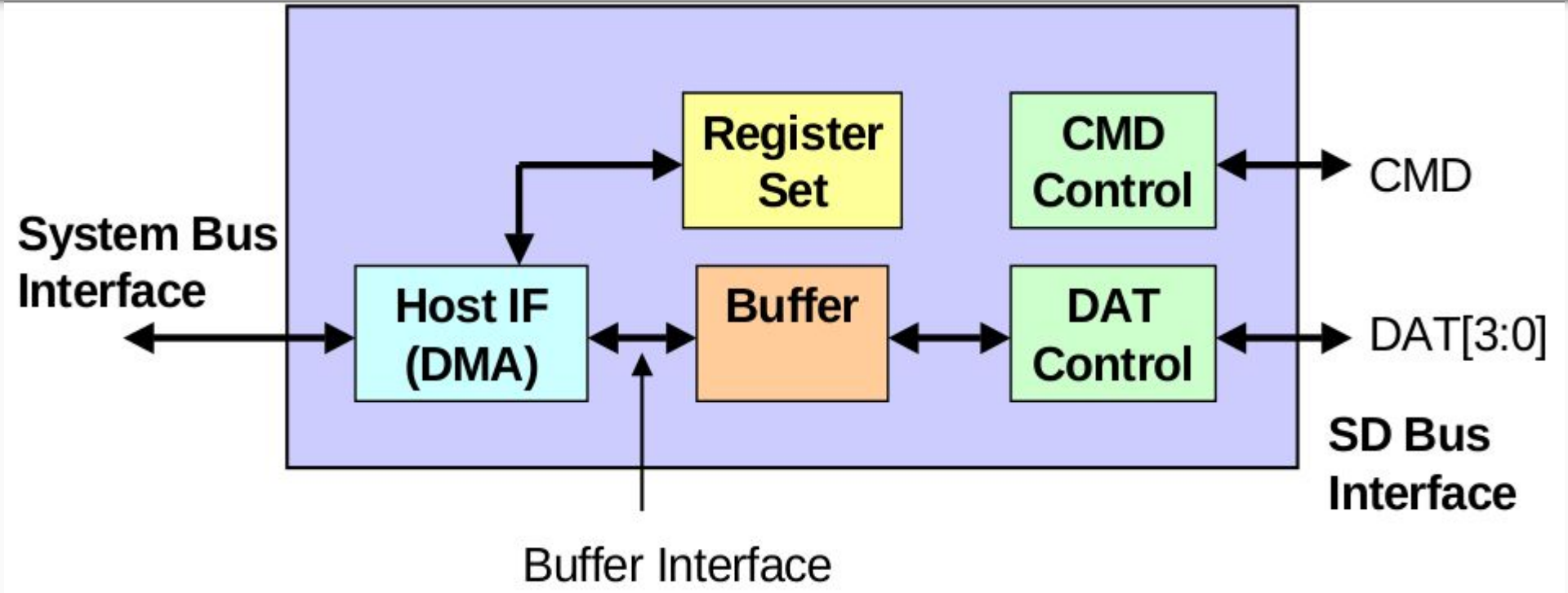
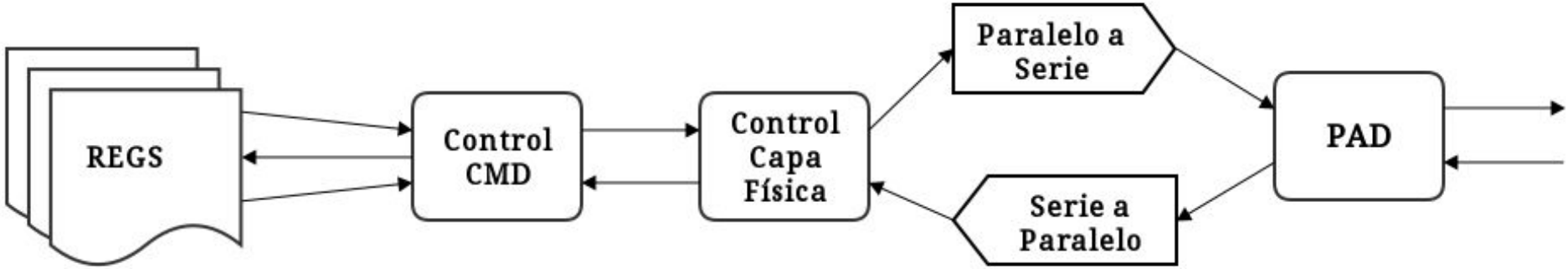
SD Host Controller

Alberto Alfaro Degracia B40167
Juan José Delgado Quesada B42250
Jose Pablo Delgado Chinchilla B32241
Leonardo Hernández Chacón B43262

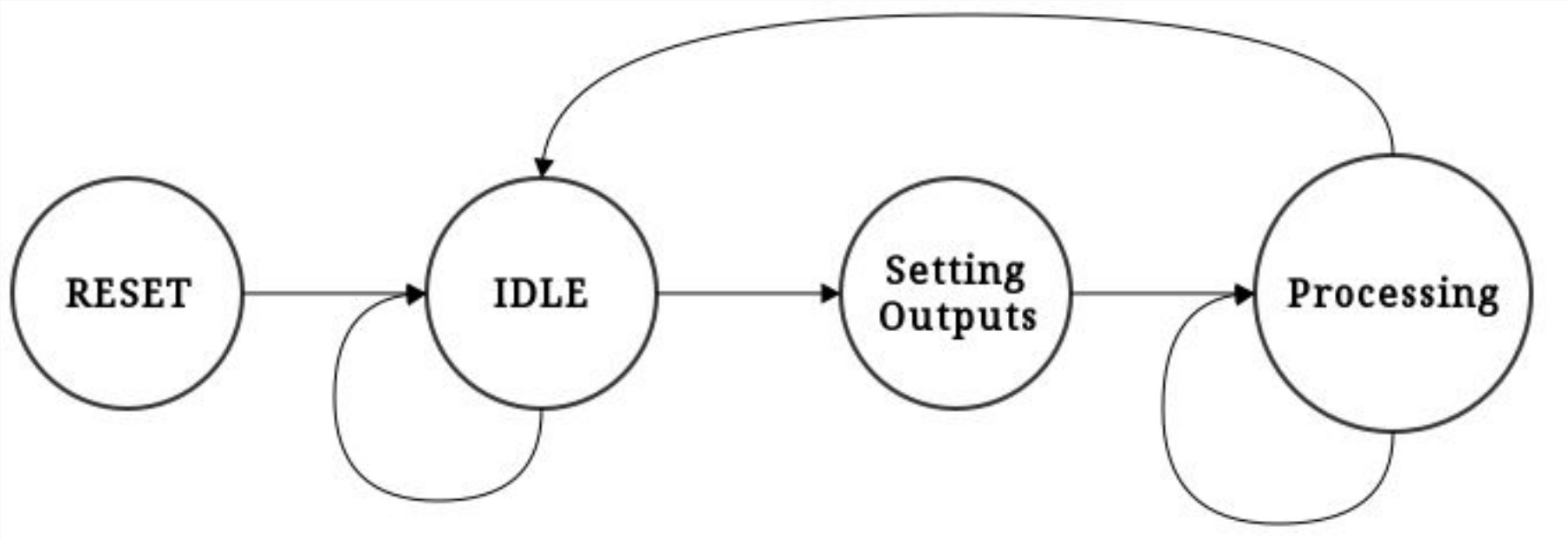
Partes de un SD Host Controller



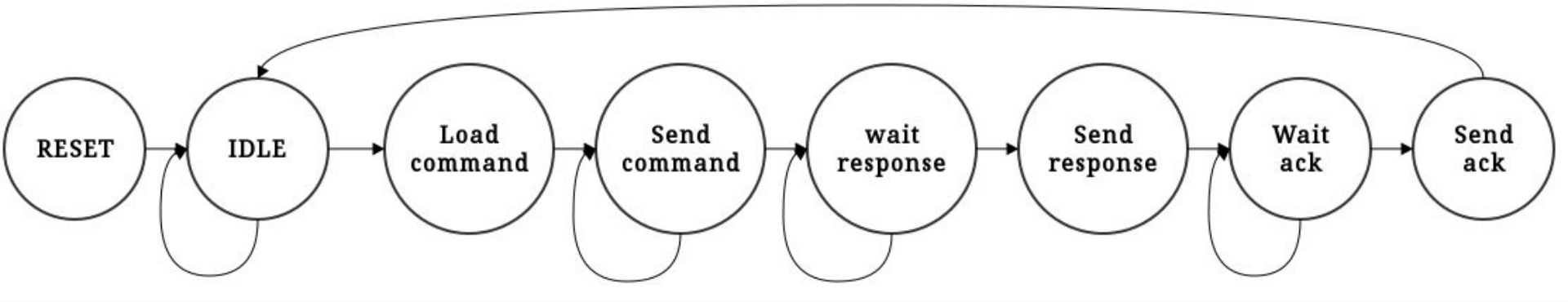
CMD Control


REGS
Control
CMD
Control
Capa
Física
Paralelo a
Serie
Serie a
Paralelo
PAD

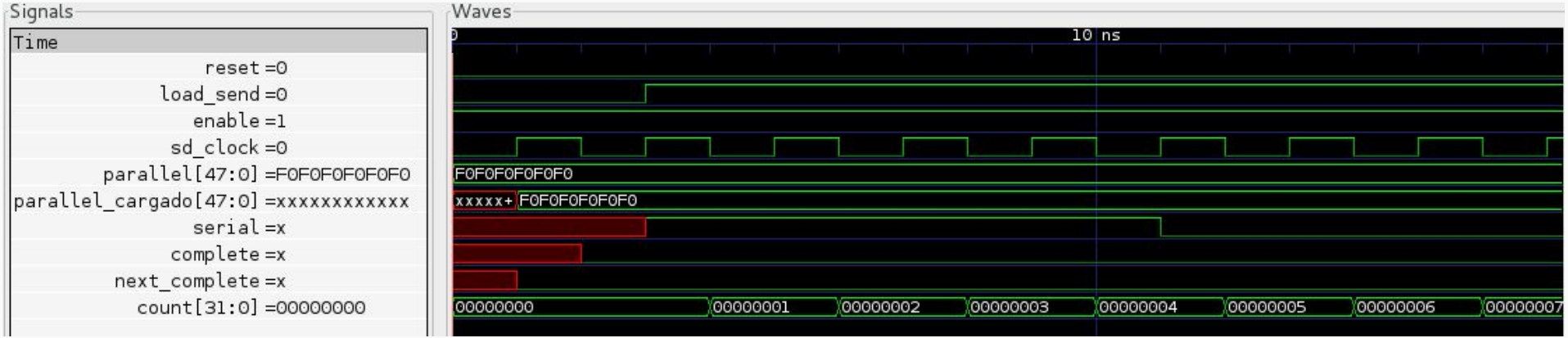
Control del CMD



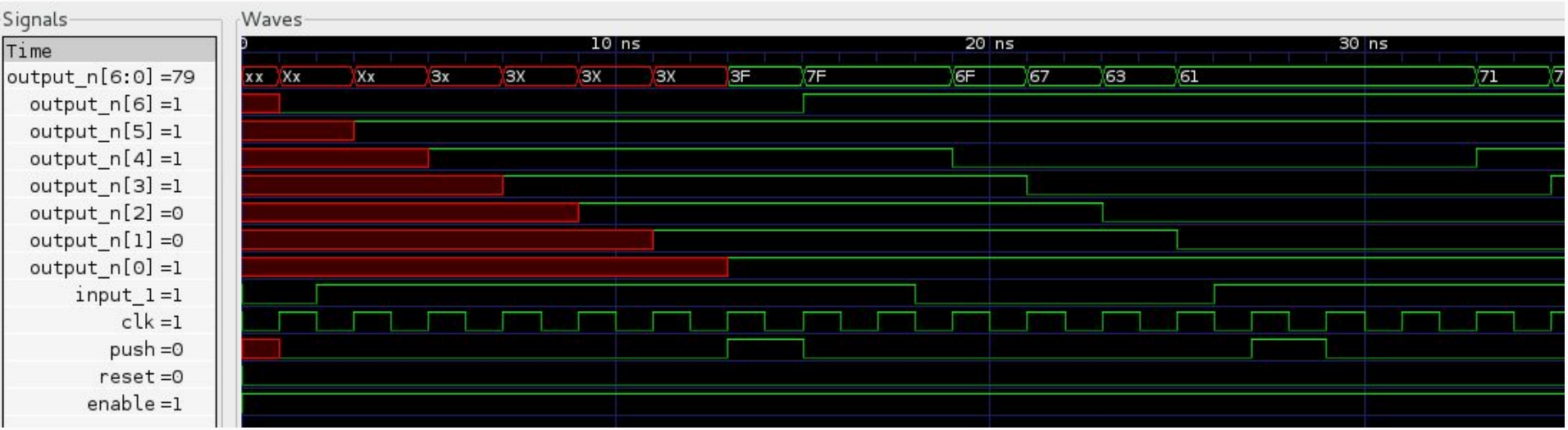
Control Capa Física



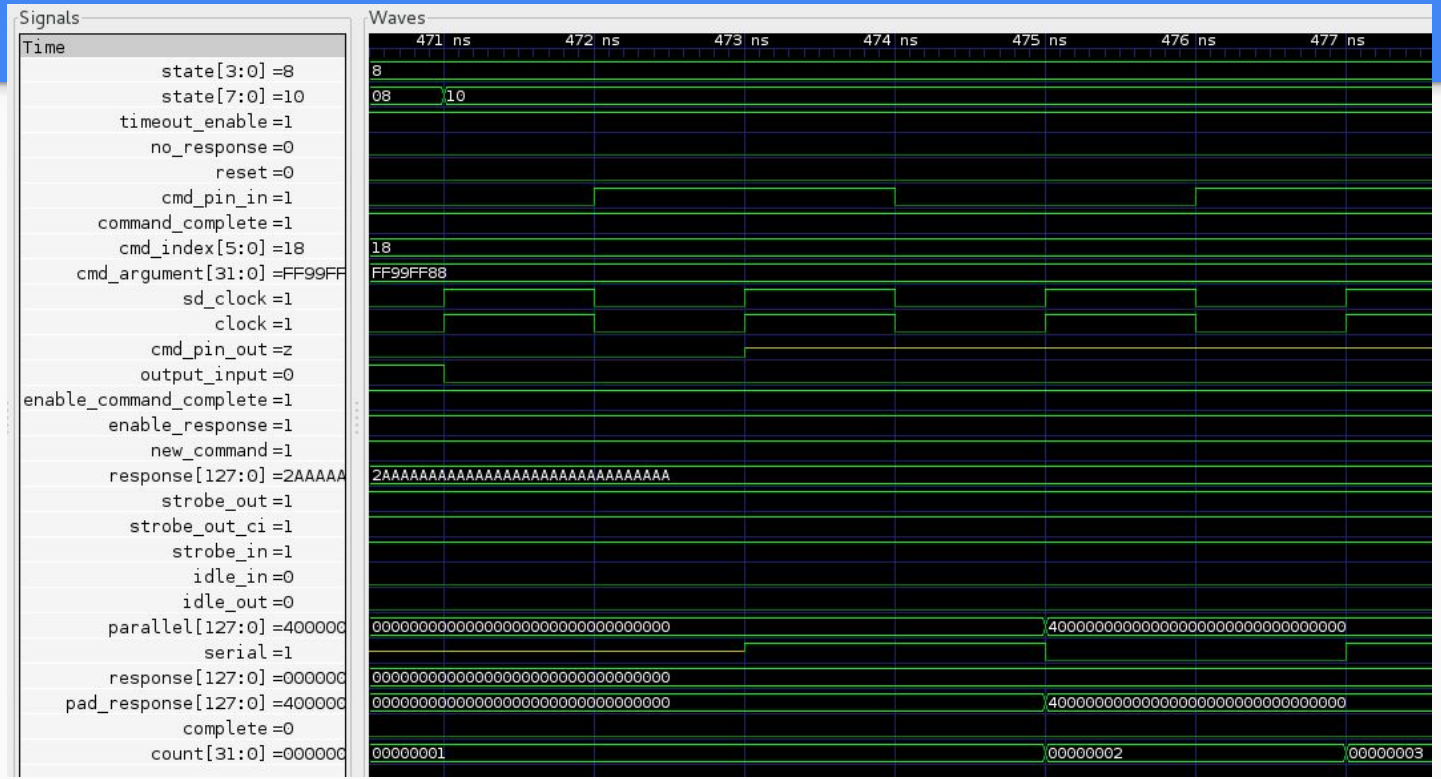
Wrapper Paralelo - Serial



Wrapper serial - Paralelo



Simulación proceso del CMD



SD Host Standard Register

- Protocolo de Comunicación
- Driver
- Bloques
- CPU
- Reservados

Offset	15-08 bit	07-00 bit	Offset	15-08 bit	07-00 bit
002h	SDMA System Address (High)		000h	SDMA System Address (Low)	
006h	Block Count		004h	Block Size	
00Ah	Argument1		008h	Argument0	
00Eh	Command		00Ch	Transfer Mode	
012h	Response1		010h	Response0	
016h	Response3		014h	Response2	
01Ah	Response5		018h	Response4	
01Eh	Response7		01Ch	Response6	
022h	Buffer Data Port1		020h	Buffer Data Port0	
026h	Present State		024h	Present State	
02Ah	Wakeup Control	Block Gap Control	028h	Power Control	Host Control
02Eh	Software Reset	Timeout Control	02Ch	Clock Control	
032h	Error Interrupt Status		030h	Normal Interrupt Status	
036h	Error Interrupt Status Enable		034h	Normal Interrupt Status Enable	
03Ah	Error Interrupt Signal Enable		038h	Normal Interrupt Signal Enable	
03Eh	---		03Ch	Auto CMD12 Error Status	
042h	Capabilities		040h	Capabilities	
046h	Capabilities (Reserved)		044h	Capabilities (Reserved)	
04Ah	Maximum Current Capabilities		048h	Maximum Current Capabilities	
04Eh	Maximum Current Capabilities (Reserved)		04Ch	Maximum Current Capabilities (Reserved)	
052h	Force Event for Error Interrupt Status		050h	Force Event for Auto CMD12 Error Status	
	---		054h	---	ADMA Error Status
05Ah	ADMA System Address [31:16]		058h	ADMA System Address [15:00]	
05Eh	ADMA System Address [63:48]		05Ch	ADMA System Address [47:32]	
---	---		---	---	
0F2h	---		0F0h	---	
---	---		---	---	
0FEh	Host Controller Version		0FCh	Slot Interrupt Status	

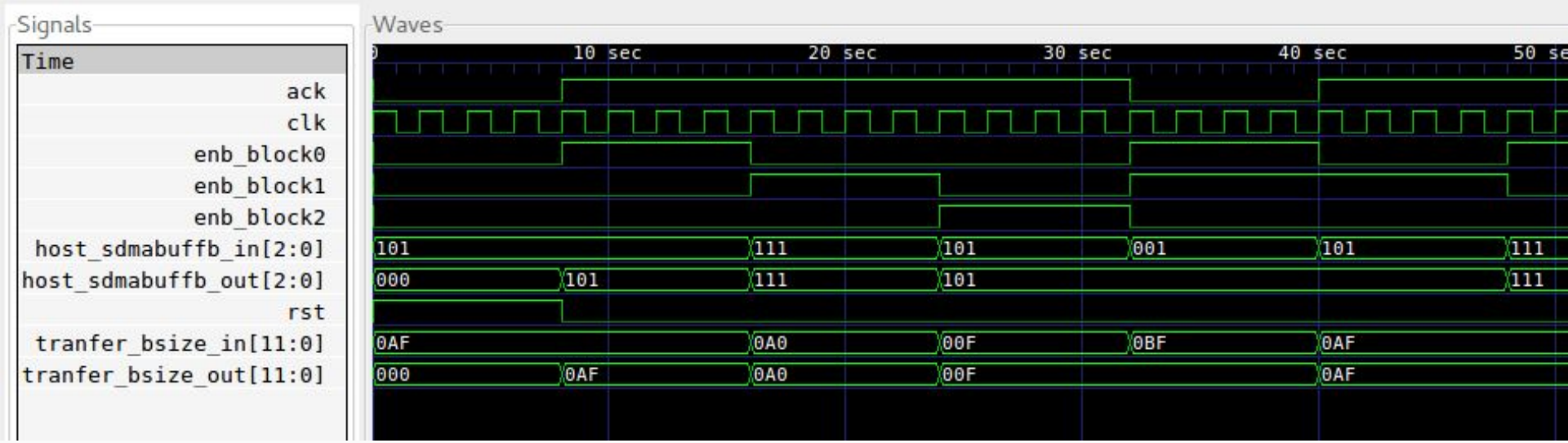
Single register structure

D31 – D25	D24	D23 – D20	D19	D18	D17	D16
Rsvd	CMD Line Signal Level	DAT[3:0] Line Signal Level	Write Protect Switch Pin Level	Card Detect Pin Level	Card State Stable	Card Inserted

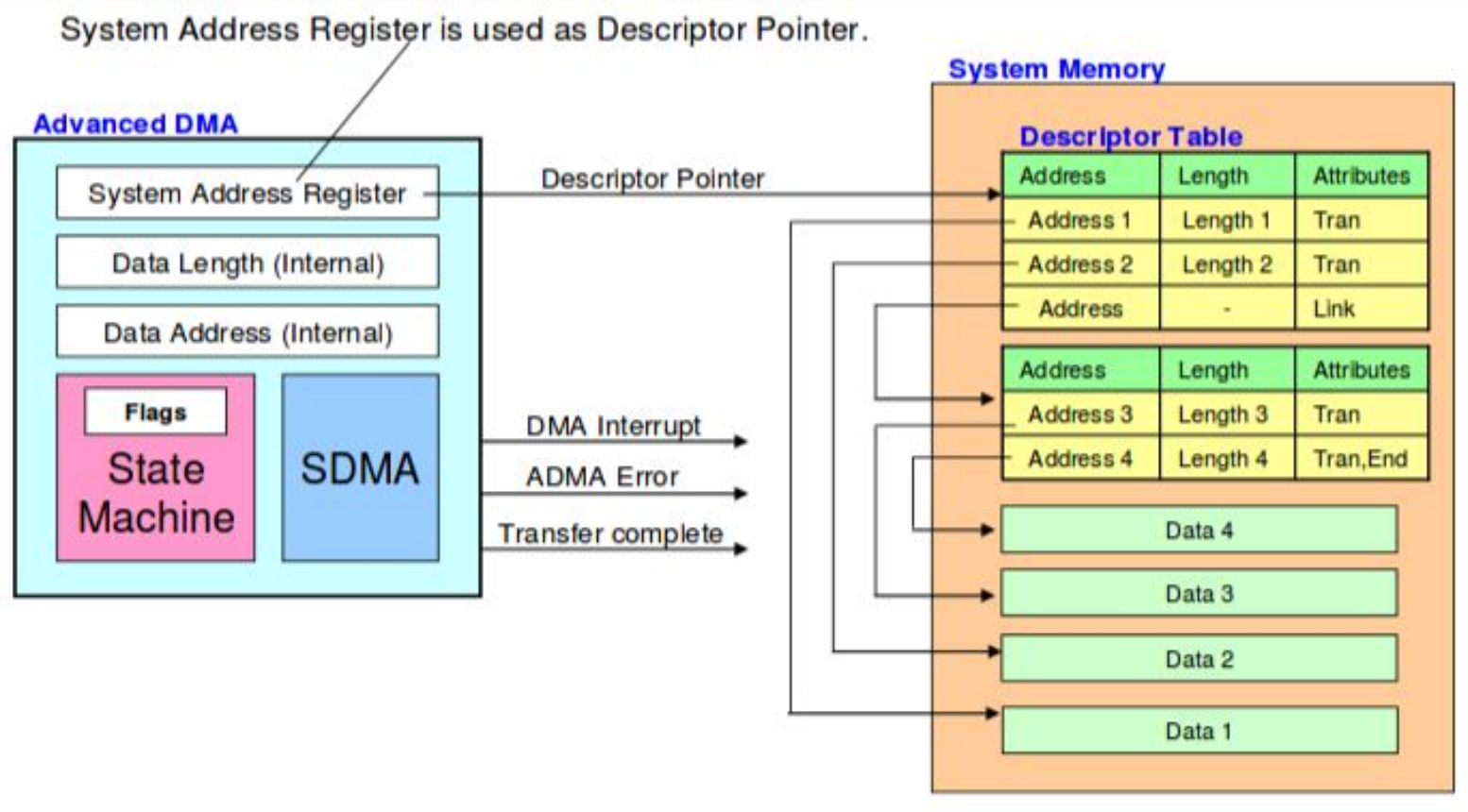
D15 – D12	D11	D10	D09	D08	D07 – D03	D02	D01	D00
Rsvd	Buffer Read Enable	Buffer Write Enable	Read Transfer Active	Write Transfer Active	Rsvd	DAT Line Active	Command Inhibit (DAT)	Command Inhibit (CMD)

```
//REG SIZE
wire [(width-1):0] data_in;

//WIRES
    //Regular Blocks
    wire rst;
    wire clk;
    wire enb_block0;
    wire enb_block1;
    wire enb_block2;
    reg ack;
    //INPUTS
    wire [6:0] SDCLKfrqselect_in;
    wire SDCLKenable_in;
    wire InternalClockStable_in;
    wire InternalClockEnable_in;

    //OUTPUTS
    wire [6:0] SDCLKfrqselect_out;
    wire SDCLKenable_out;
    wire InternalClockStable_out;
    wire InternalClockEnable_out;

//REG
    //OUTPUTS
    reg [(width-1):0] data_out;
```

Simulación de registros



ADMA2


System Address Register is used as Descriptor Pointer.
Advanced DMA
System Address Register
Data Length (Internal)
Data Address (Internal)
Flags
State Machine
SDMA
Descriptor Pointer
DMA Interrupt
ADMA Error
Transfer complete
System Memory
Descriptor Table
Address
Length
Attributes
Address 1
Length 1
Tran
Address 2
Length 2
Tran
Address
-
Link
Address
Length
Attributes
Address 3
Length 3
Tran
Address 4
Length 4
Tran,End
Data 4
Data 3
Data 2
Data 1

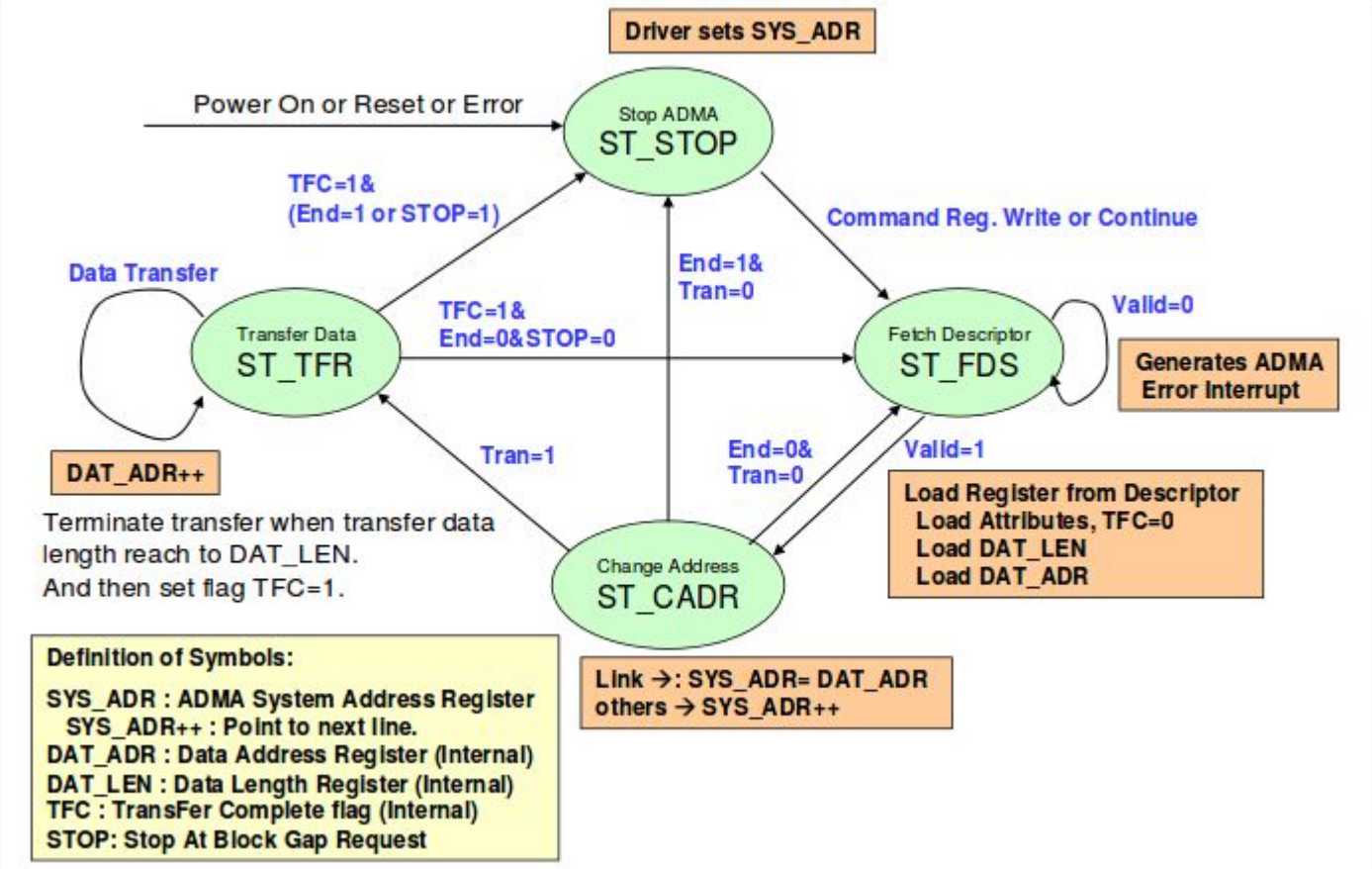
Descriptor Line

Address Field		Length		Reserved		Attribute					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit Length		000000		Act2	Act1	0	Int	End	Valid

Valid	Valid=1 indicates this line of descriptor is effective. If Valid=0 generate ADMA Error Interrupt and stop ADMA to prevent runaway.
End	End=1 indicates to end of descriptor. The Transfer Complete Interrupt is generated when the operation of the descriptor line is completed.
Int	INT=1 generates DMA Interrupt when the operation of the descriptor line is completed.

Act2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Do not execute current line and go to next line.
0	1	rsv	reserved	(Same as Nop. Do not execute current line and go to next line.)
1	0	Tran	Transfer Data	Transfer data of one descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

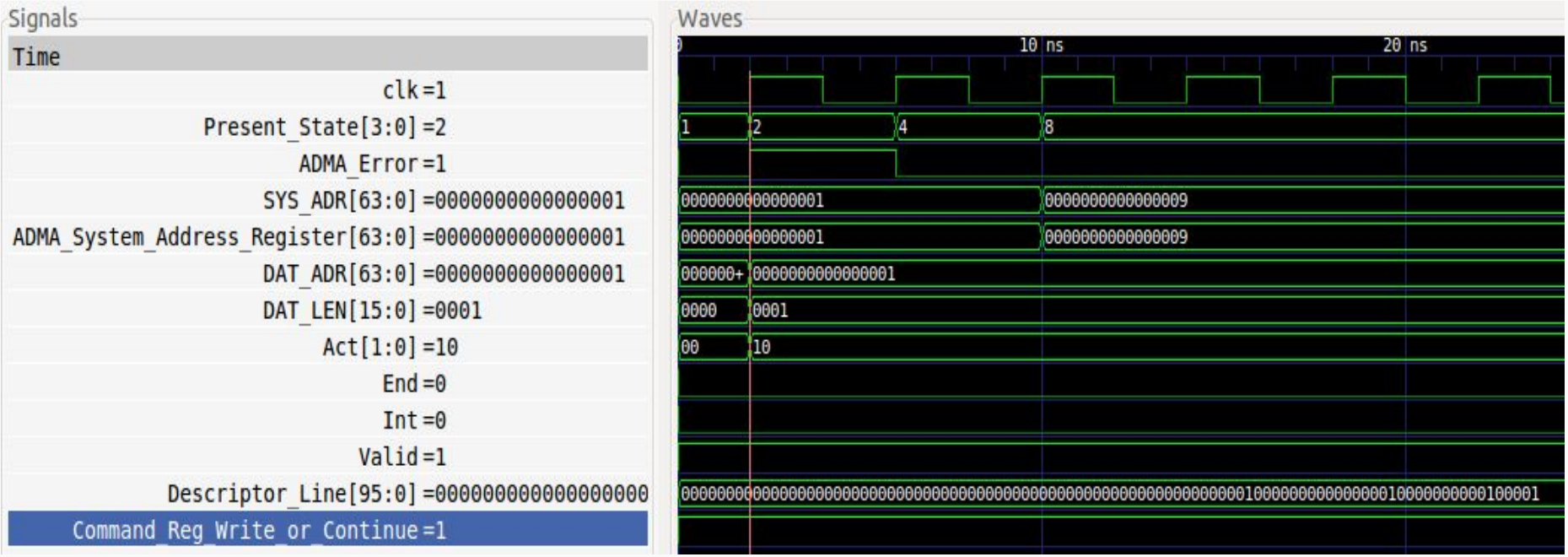
Máquina de Estados.



Estados

State Name	Operation
ST_FDS (Fetch Descriptor)	ADMA2 fetches a descriptor line and set parameters in internal registers. Next go to ST_CADR state.
ST_CADR (Change Address)	Link operation loads another Descriptor address to *ADMA System Address* register. In other operations, *ADMA System Address* register is incremented to point next descriptor line. If End=0, go to ST_TFR state. ADMA2 shall not be stopped at this state even if some errors occur.
ST_TFR (Transfer Data)	Data transfer of one descriptor line is executed between system memory and SD card. If data transfer continues (End=0) go to ST_FDS state. If data transfer completes, go to ST_STOP state.
ST_STOP (Stop DMA)	ADMA2 stays in this state in following cases: (1) After Power on reset or software reset. (2) All descriptor data transfers are completed If a new ADMA2 operation is started by writing Command register, go to ST_FDS state.

ST_FDS


Signals
Time
clk =1
Present_State[3:0] =2
ADMA_Error =1
SYS_ADR[63:0] =0000000000000001
ADMA_System_Address_Register[63:0] =0000000000000001
DAT_ADR[63:0] =0000000000000001
DAT_LEN[15:0] =0001
Act[1:0] =10
End =0
Int =0
Valid =1
Descriptor_Line[95:0] =00000000000000000
Command_Reg_Write_or_Continue =1
Waves
10 ns
20 ns
1
2
4
8
0000000000000001
0000000000000009
0000000000000001
0000000000000009
000000+
0000000000000001
0000
0001
00
10

Lectura/Escritura a registros

```
input [15:0] Block_Size_Register;     //Offset 004h, interesa bit [11:0]
input [15:0] Block_Count_Register;    //Offset 006h, interesan todos los bits
input [15:0] Transfer_Mode_Register;  //Offset 00Ch, interesan los bits 05, 04, 01,00.
input [31:0] Present_State_Register;  //Offset 024h, interesan los bits 09,08.
input [7:0]  Block_Gap_Control_Register;    //Offset 02Ah, interesan los bits 0,1.
input [15:0]  Command_Register;       //Offset 00Eh, interesan los bits 6 y 7.

output enb_DMA_Interrupt;
output enb_ADMA_Error;
output enb_Transfer_complete;
output enb_ADMA_System_Address_Register;
output DMA_Interrupt;     //Normal_Interrupt_Status_Register   Offset  030h  bit 3.
output ADMA_Error;        //Error Interrupt Status Register    Offset 032h bit 9.
output Transfer_complete; //Normal_Interrupt_Status_Register   Offset  030h  bit 1.
output [63:0] ADMA_System_Address_Register; // (Offset 058h)
```

```
SYS_ADR<=Initial_ADMA_System_Address;
ADMA_System_Address_Register<=SYS_ADR;
enb_ADMA_System_Address_Register<=1;

if(ack_ADMA_System_Address_Register==1)begin

  Present_State <= Next_State;
  enb_ADMA_System_Address_Register<=0;
 end

else begin
  Present_State <= Present_State;
     end
```

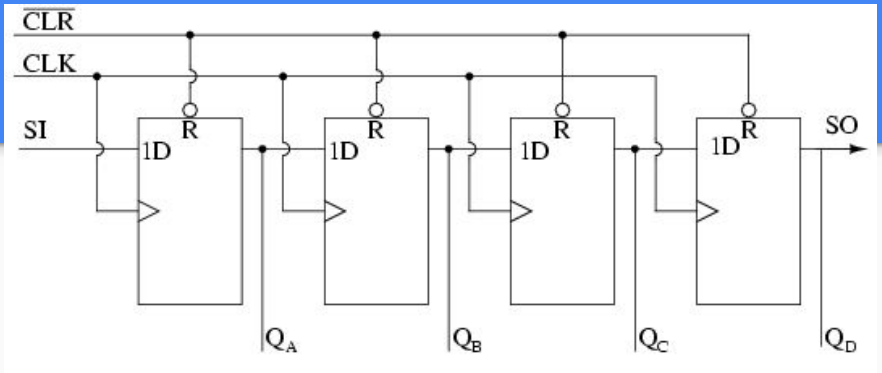
Transferencia

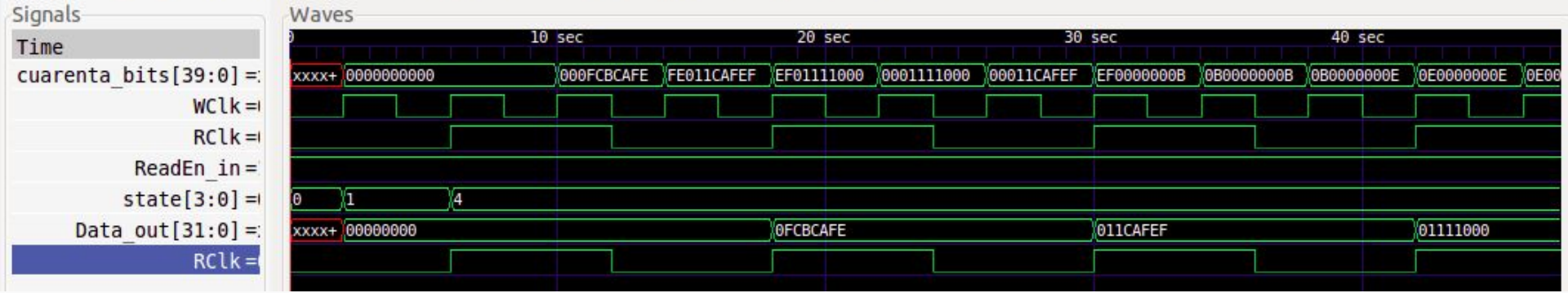
```
case(Data_Transfer_Direction_Select)
      0: begin
         Transfer_Mode_Direction<=HOST_TO_CARD;
         end

      1: begin
         Transfer_Mode_Direction<=CARD_TO_HOST;
         end

      default: begin
         Transfer_Mode_Direction<=WAIT;
               end

endcase
```

Buffer <-------> Memoria

Comunicación con DAT.

FIFO


CLR
CLK
SI
1D
R
SO
Q_A
Q_B
Q_C
Q_D

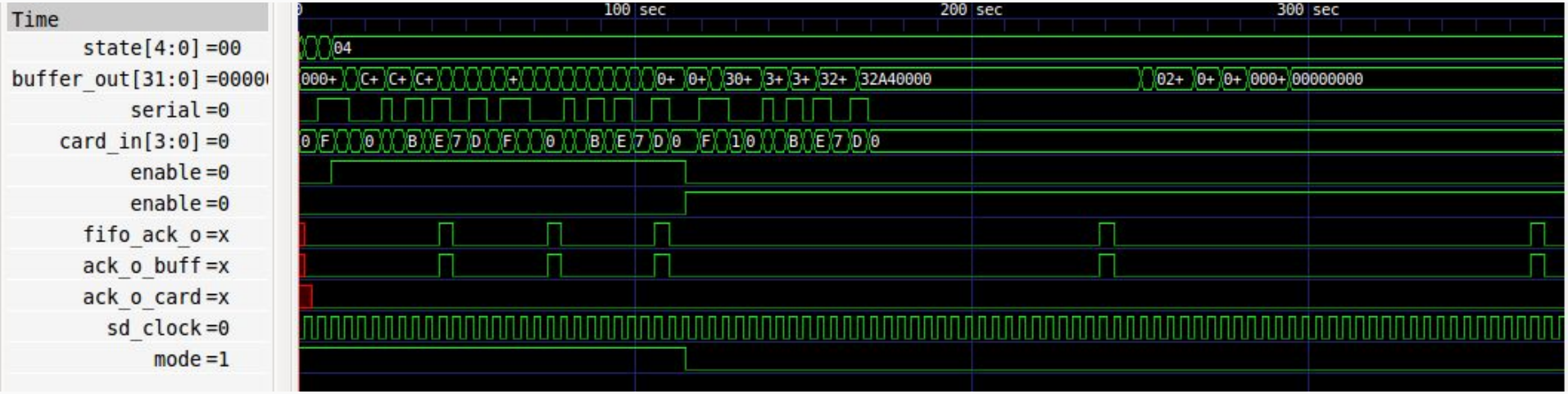

Signals
Waves
Time
cuarenta_bits[39:0]
WClk
RClk
ReadEn_in
state[3:0]
Data_out[31:0]
RClk
10 sec
20 sec
30 sec
40 sec
xxxx+ 0000000000 000FCBCAFE FE011CAFEF EF01111000 0001111000 00011CAFEF EF0000000B 0B0000000B 0B0000000E 0E0000000E 0E00
0 1 4
xxxx+ 00000000 0FCBCAFE 011CAFEF 01111000

Módulo DAT

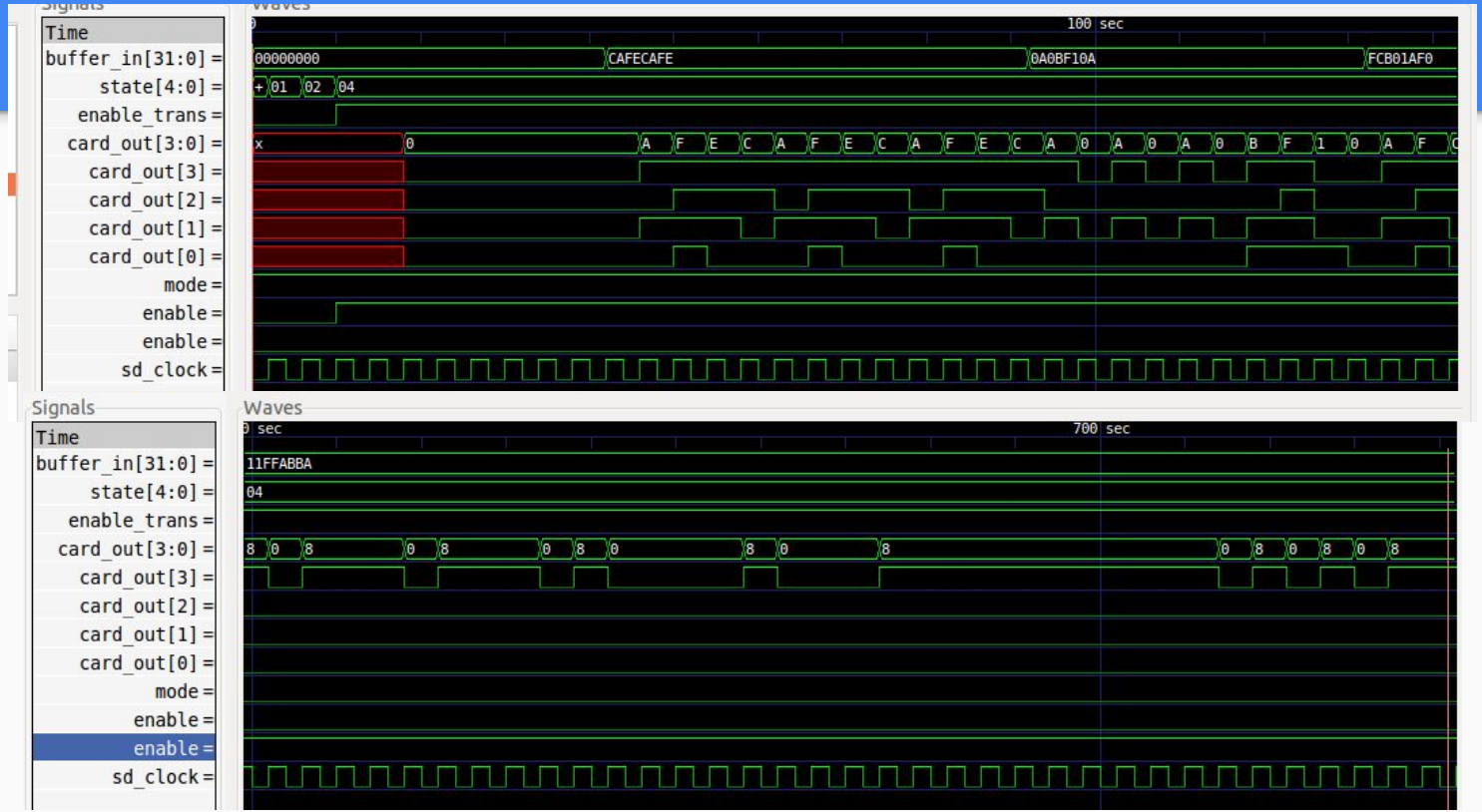
4 Módulos para conversión de datos

Transición de estados, para habilitar y seleccionar módulos y señales

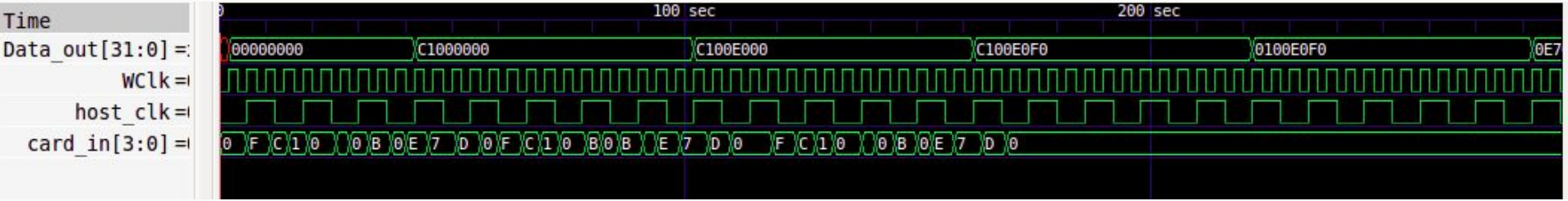
Prueba escritura en FIFO



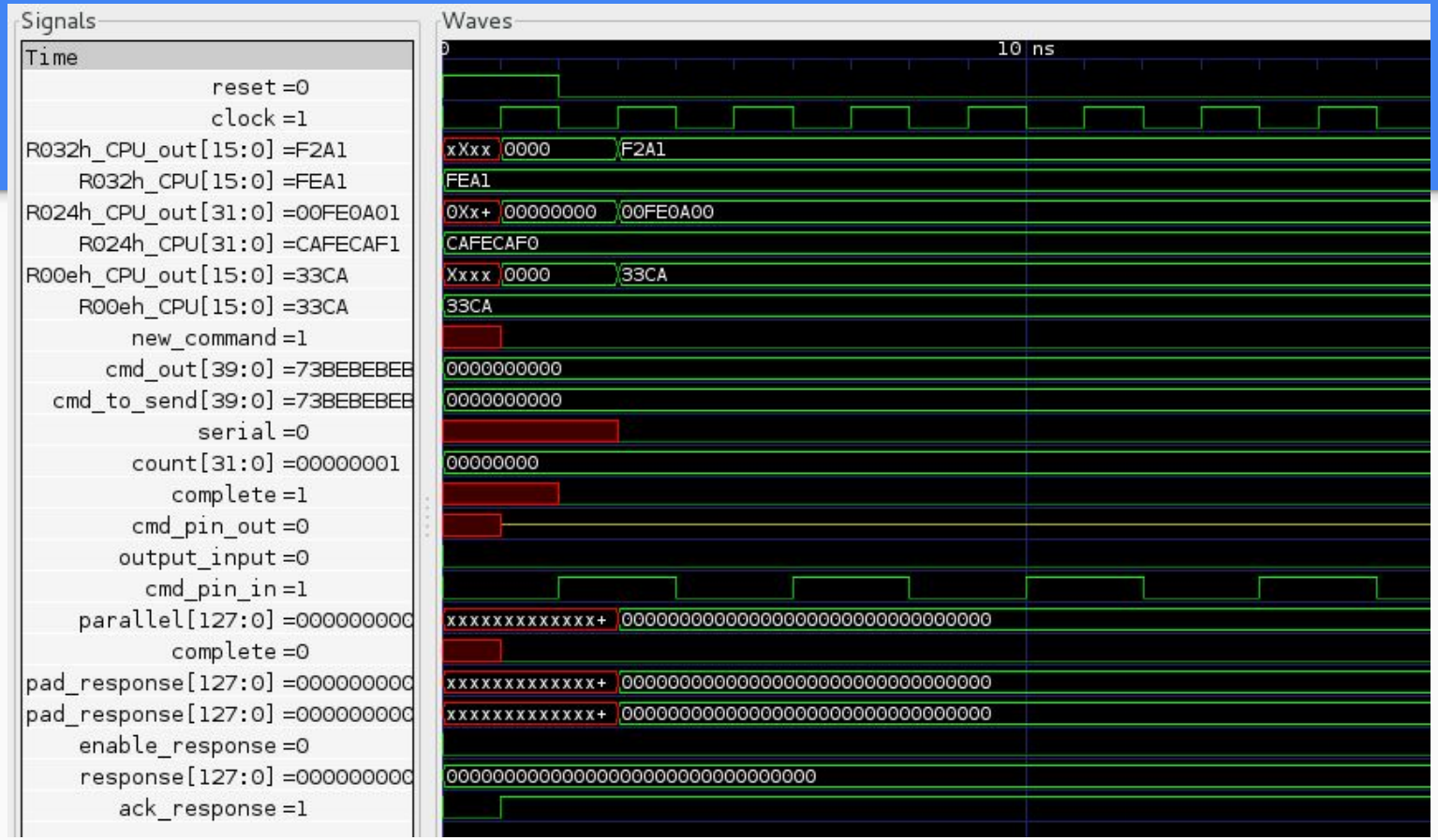
Prueba escritura en SD card



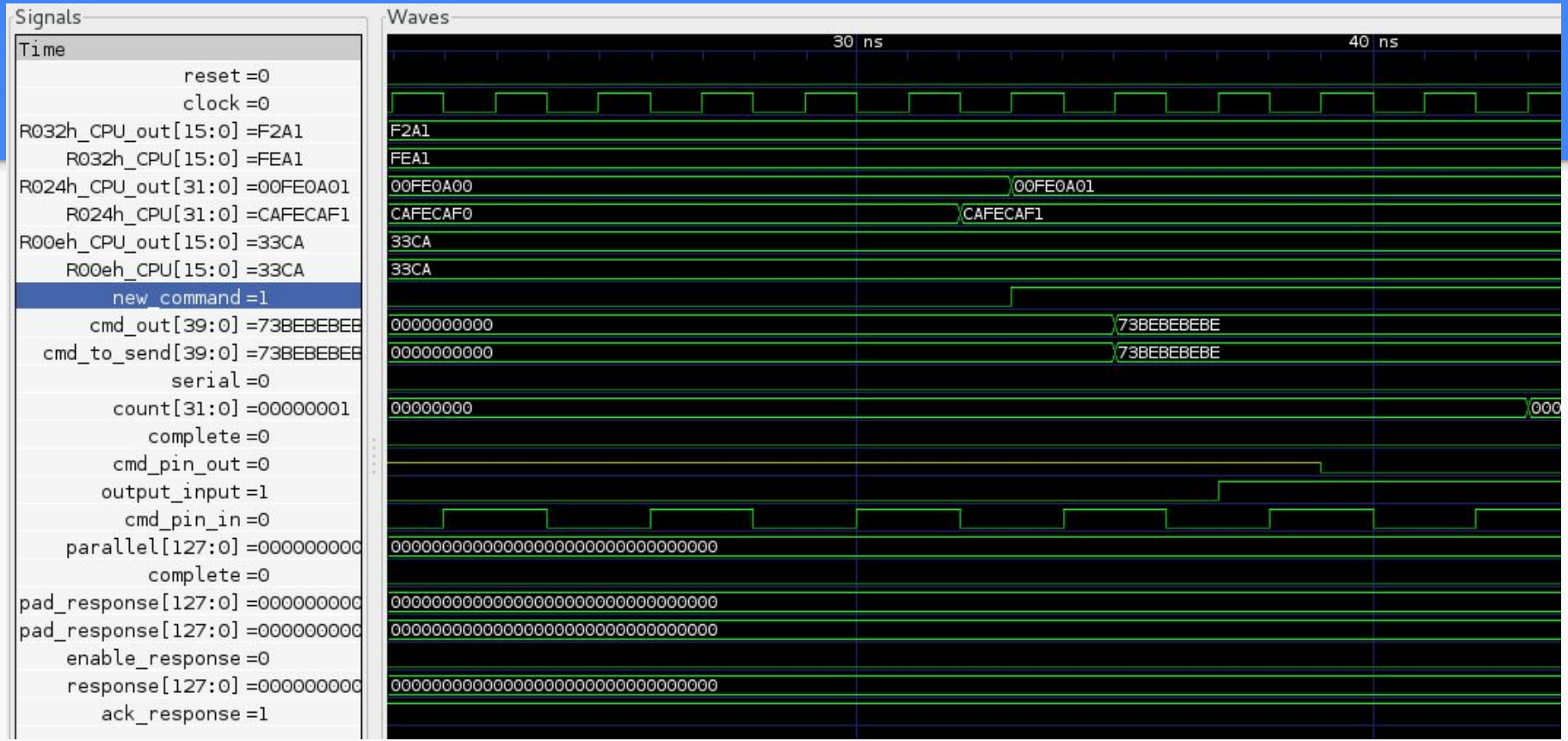
Comunicación SD card - DAT - FIFO



SD HOST REGS-CMD



SD HOST REGS-CMD



Organización y Retos

- CMD (Juan José)
- DMA (Leonardo)
- DAT (Alberto)
- Reg (Pablo)

- Reuniones Semanales
- Utilización de Github

- Utilización de diferentes Buffers
- Simulación del CPU
- Latch inferidos (yosys)