PROYECTO
DE ENCRIPTACIÓN Y DESENCRIPTACIÓN EN HARDWARE

Testbenches de Encriptación/Desencriptación

TE2002B.401

ENCRIPTADORES

Líder* - *Angelo Segura Ibarra

AddRoundKey

Emiliano Camacho Ponce
Obiel Emiliano Rangel Moreno
Evan André Santana Pacheco
Alfredo Alejandro Soto Herrera

MixColumns

Yumee Chung
Ana Carolina Coronel Herrera
Adrián Márquez Núñez
Andrés Ignacio Zegales Taborga

ShiftRows

Dalila Fonseca Maya
Santiago Isai González Arista
Victoria Lilian Robles Vargas
Angelo Segura Ibarra

Líder - *Andrés Ignacio Zegales Taborga*

SubBytes

Ricardo Antonio Enguilo Sánchez
Mariana Hernández Díaz
Lizeth Niniz Corona

Key Schedule

Anell Moreno Quilatán
Mario Armando Moscoso
Fernando Hernández
Hernández

DESENCRIPTADORES

Líder- ***Omar Reyes Barrueta***

Inv SubBytes

Matías Piedra Pichardo

Josemaría De Alba Arguelles

Ricardo Ruiz Cano

Samantha Erin Medina Muñoz

Fernando García Padilla

Inv ShiftRows

Socorro Alejandra Montelongo Loredo

José Fernando Vivanco Benavides

Juan César Nuño Rivera

Gabriel Enrique Lascurain Flores

Arturo Urías Jiménez

Inv MixColumns

Arturo Balboa Alvarado

Omar Ryes Barrueta

Líder- ***Arturo Balboa Alvarado***

Inv AddRoundKey

Arturo Balboa Alvarado

Anell Moreno Quilantán

Omar Reyes Barrueta

Mario Armando Moscoso Becerra

Fernando Hernández Hernández

Key Schedule

Anell Moreno Quilatán

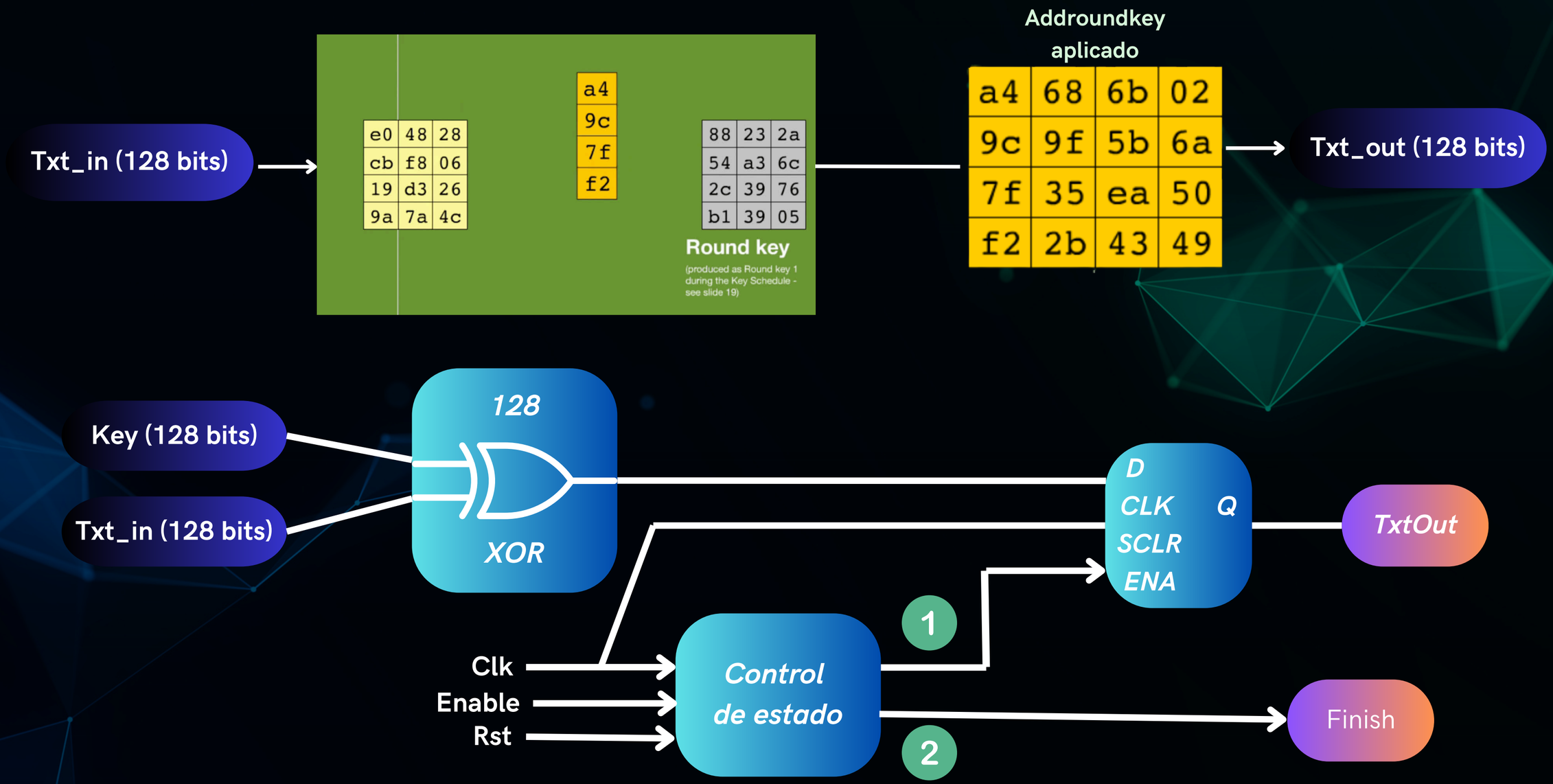
Mario Armando Moscoso Becerra

Fernando Hernández Hernández

Para garantizar el correcto funcionamiento de cada módulo del sistema de encriptación y desencriptación, desarrollamos un conjunto de **testbenches**.
Cada testbench es un entorno de prueba que simula el comportamiento real del módulo, permitiéndo verificar que la lógica implementada sea correcta antes de llevarla al hardware.

ENCRIPTACIÓN

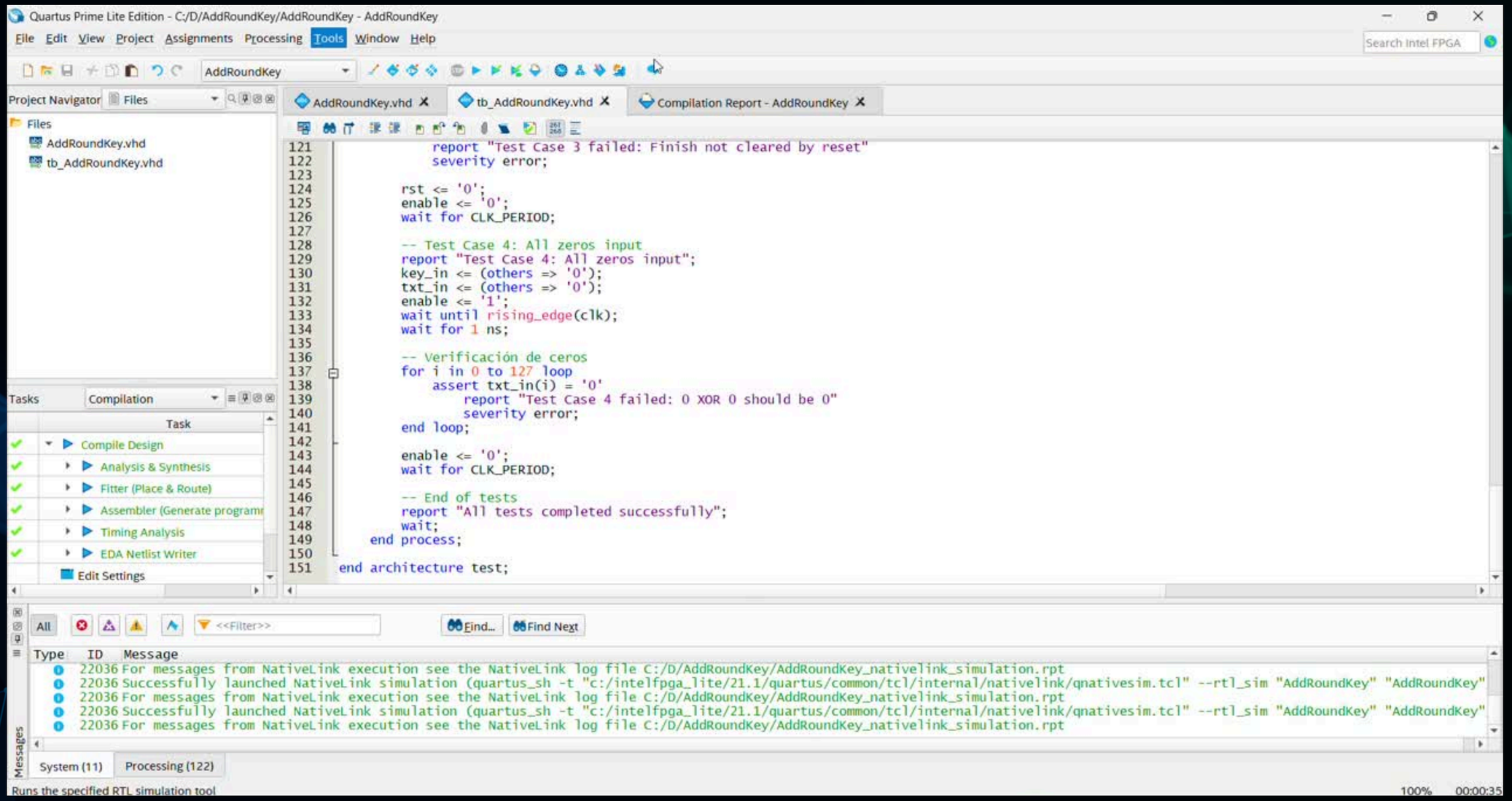
ADDROUNDKEY



SIMULACIÓN

Addroundkey

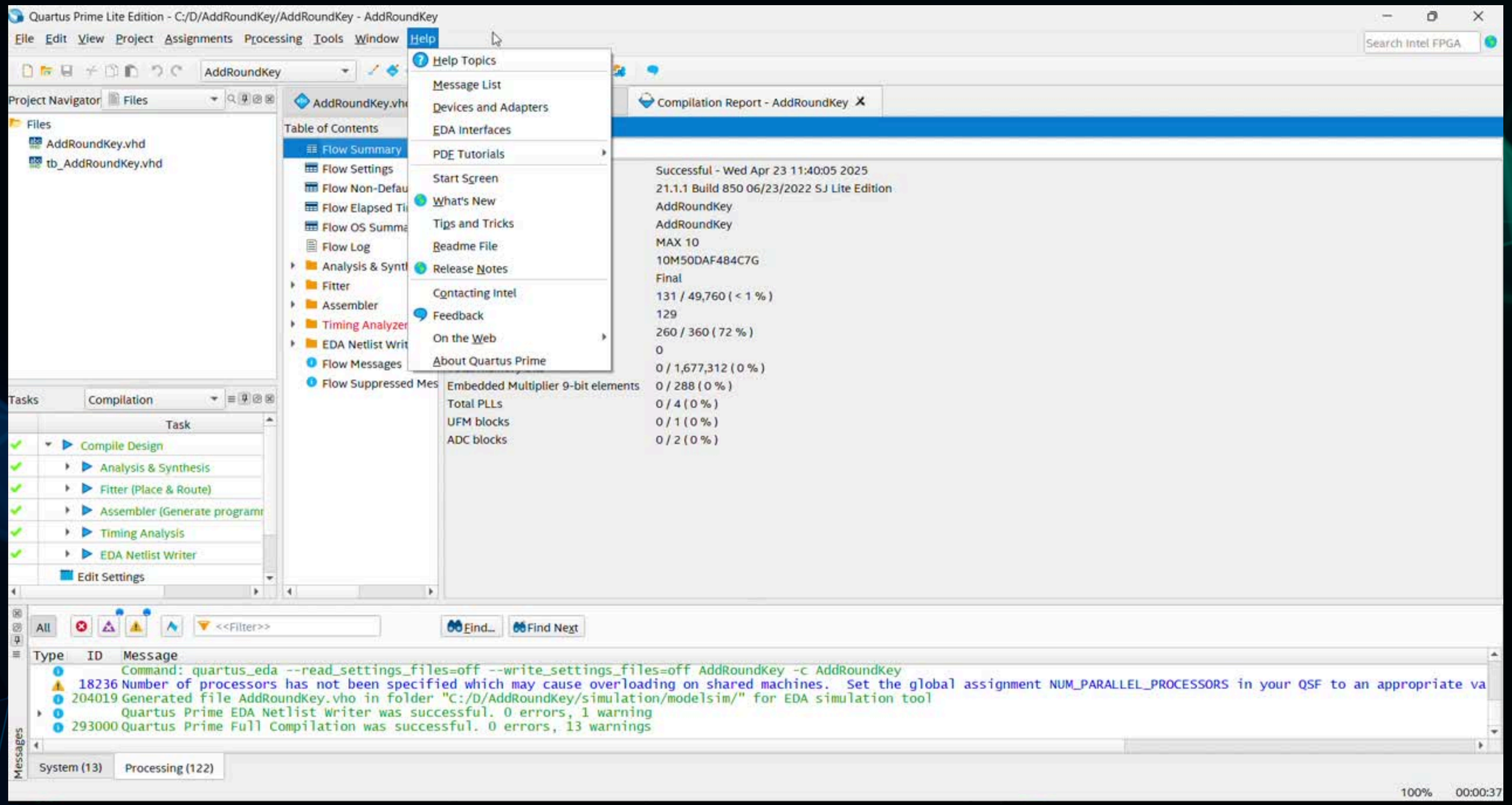
Prueba 1



SIMULACIÓN

Addroundkey

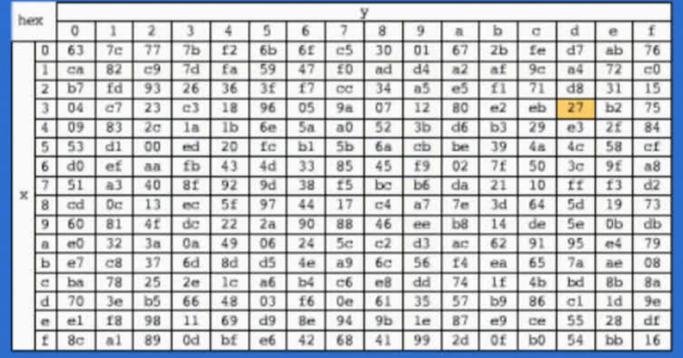
Prueba 2



SUBBYTES

Txt_in (128 bits)

d4	a0	9a	e9
27	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08



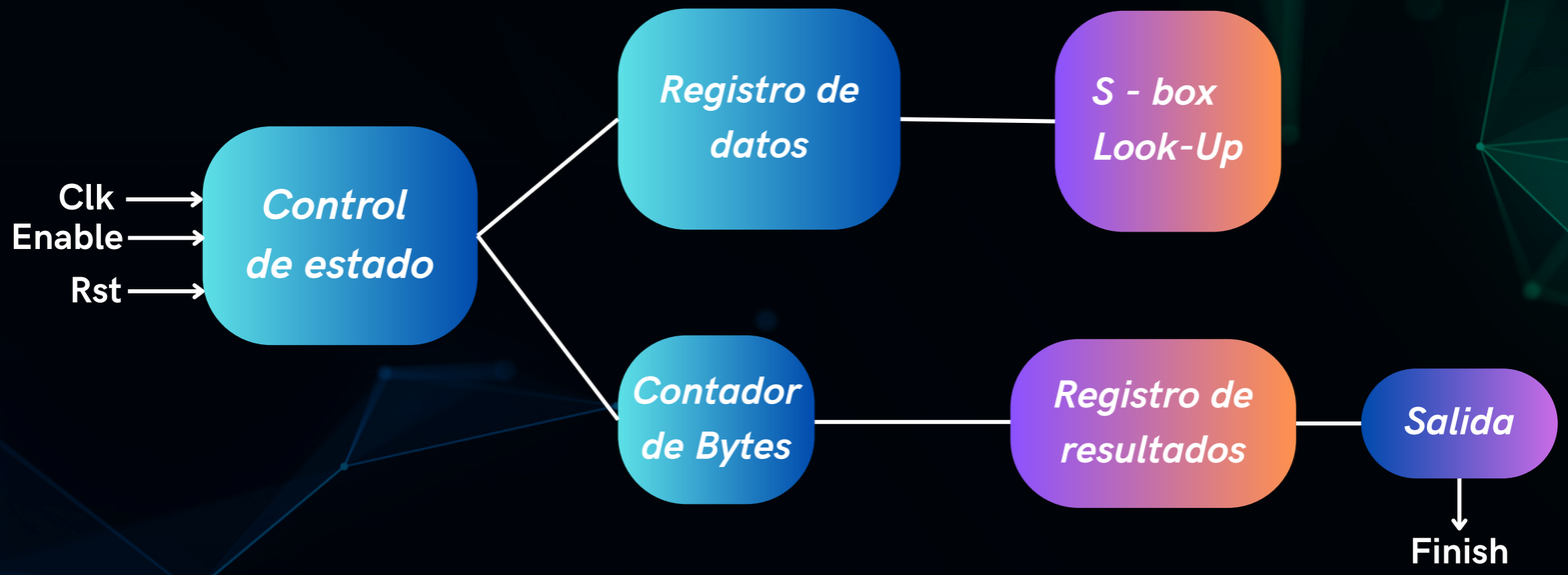
Txt_out (128 bits)

S-BOX

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

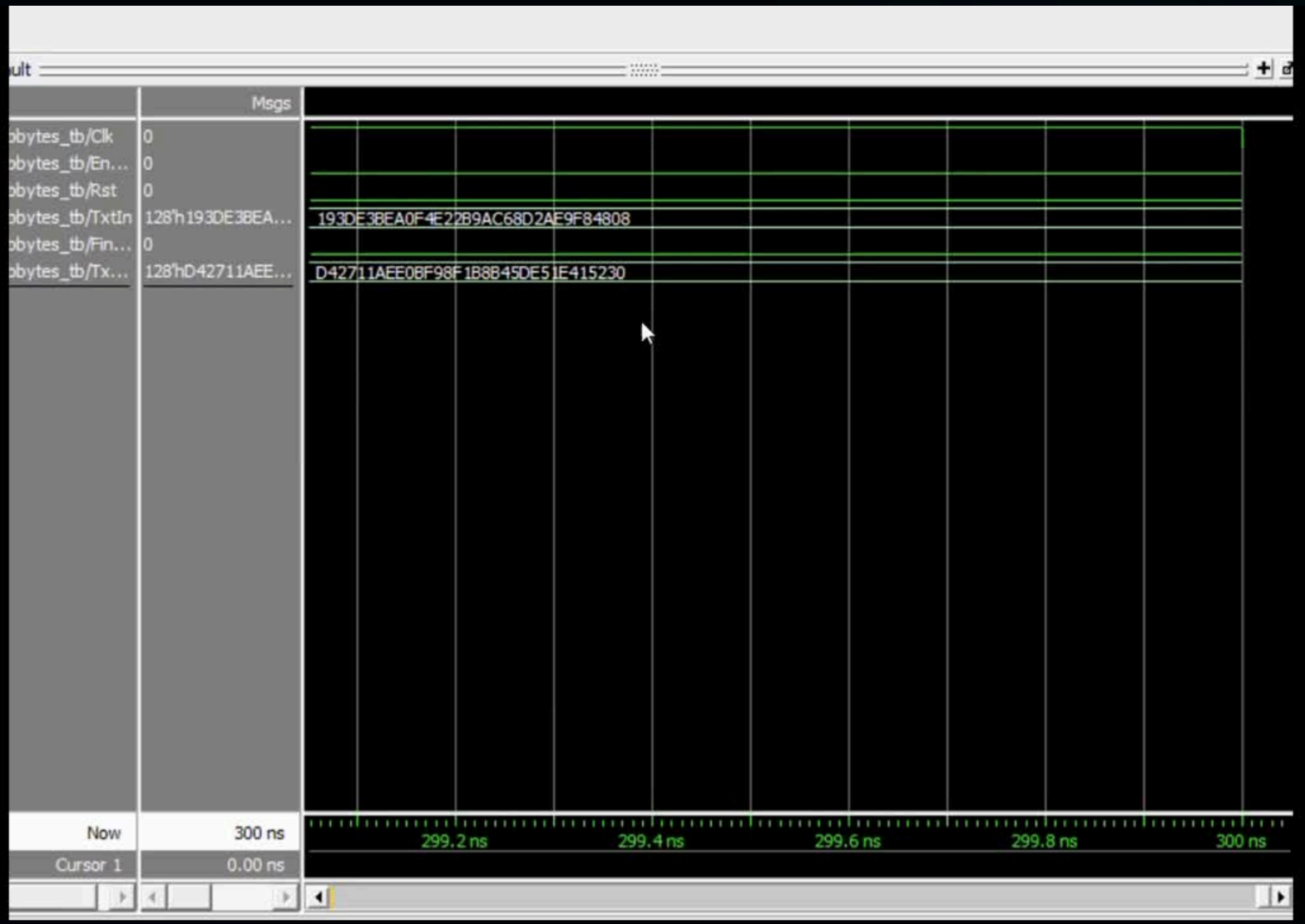
SUBBYTES

Diagrama de funcionamiento

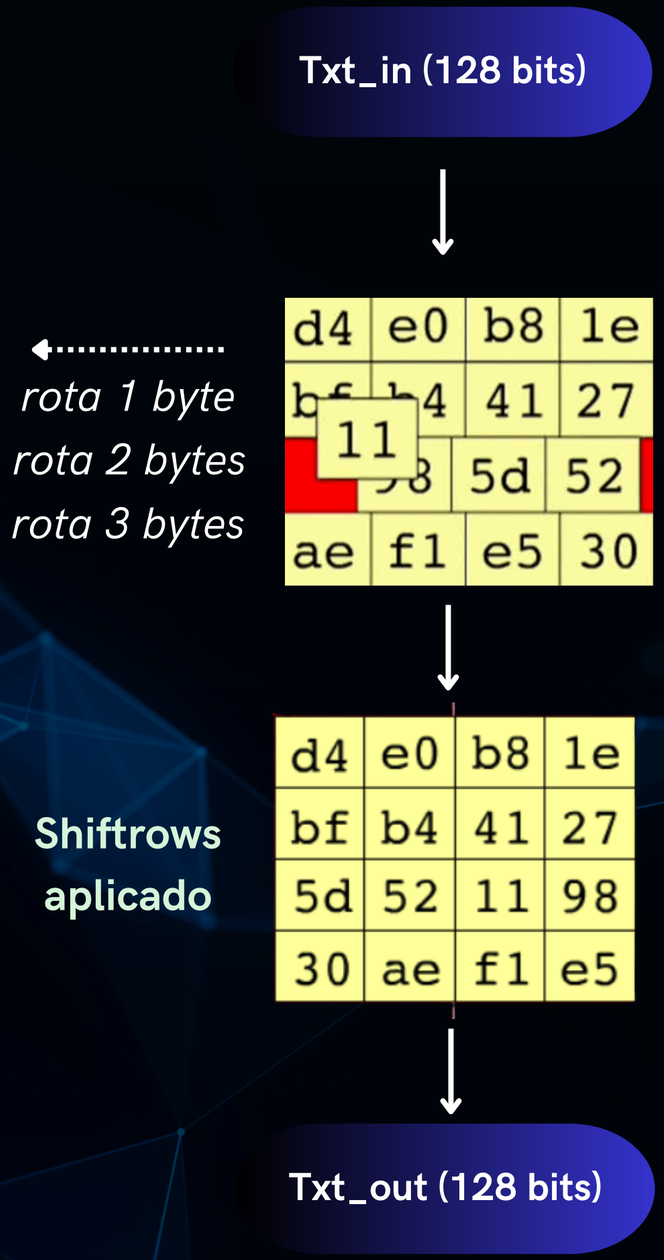


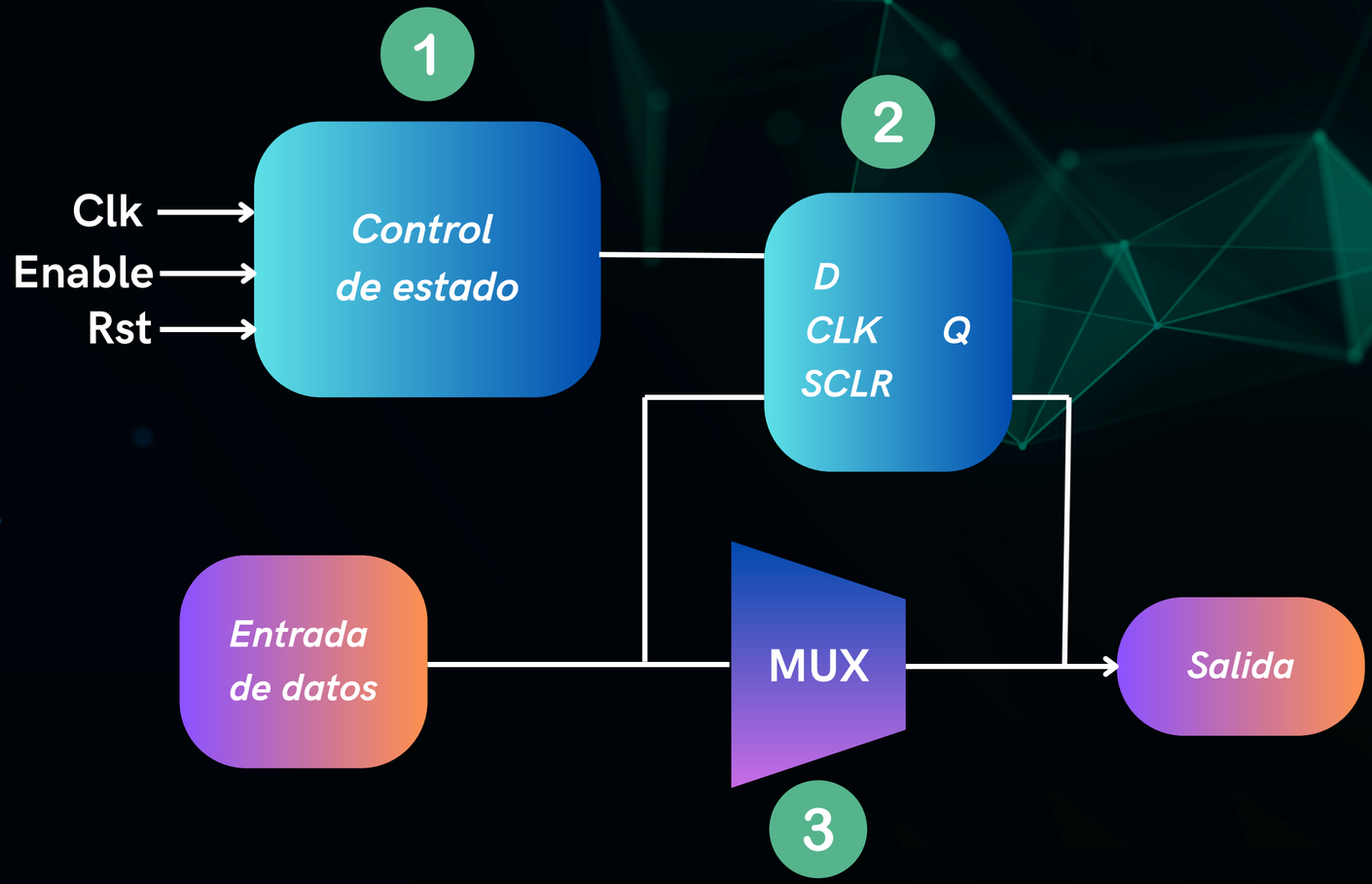
SIMULACIÓN

Subbytes



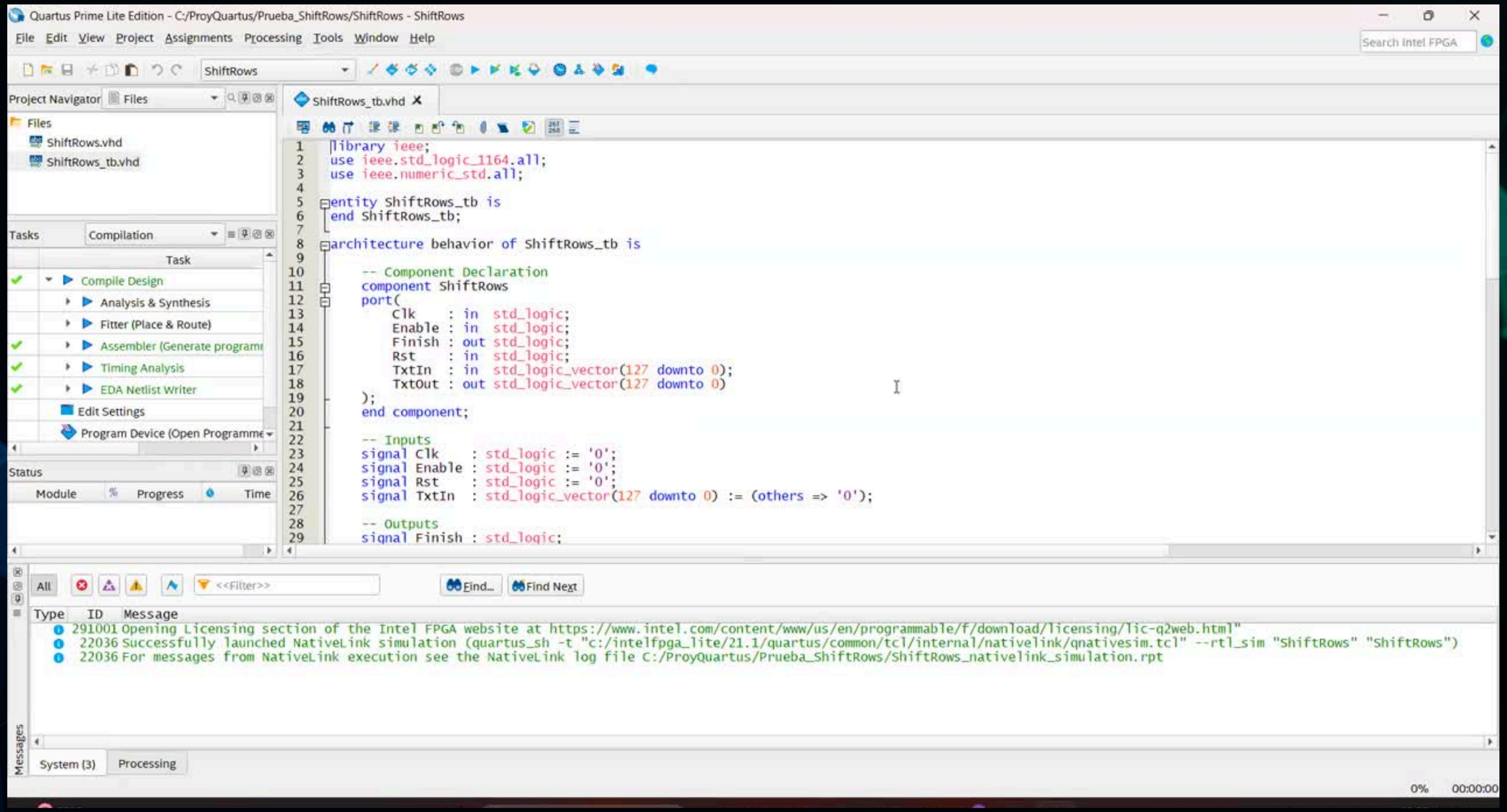
SHIFTROWS


Txt_in (128 bits)
rota 1 byte
rota 2 bytes
rota 3 bytes
d4 e0 b8 1e
bf b4 41 27
11
5d 52
ae f1 e5 30
Shiftrows aplicado
d4 e0 b8 1e
bf b4 41 27
5d 52 11 98
30 ae f1 e5
Txt_out (128 bits)


1
2
3
Clk
Enable
Rst
Control de estado
D
CLK
SCLR
Q
Entrada de datos
MUX
Salida

SIMULACIÓN

Shiftrows



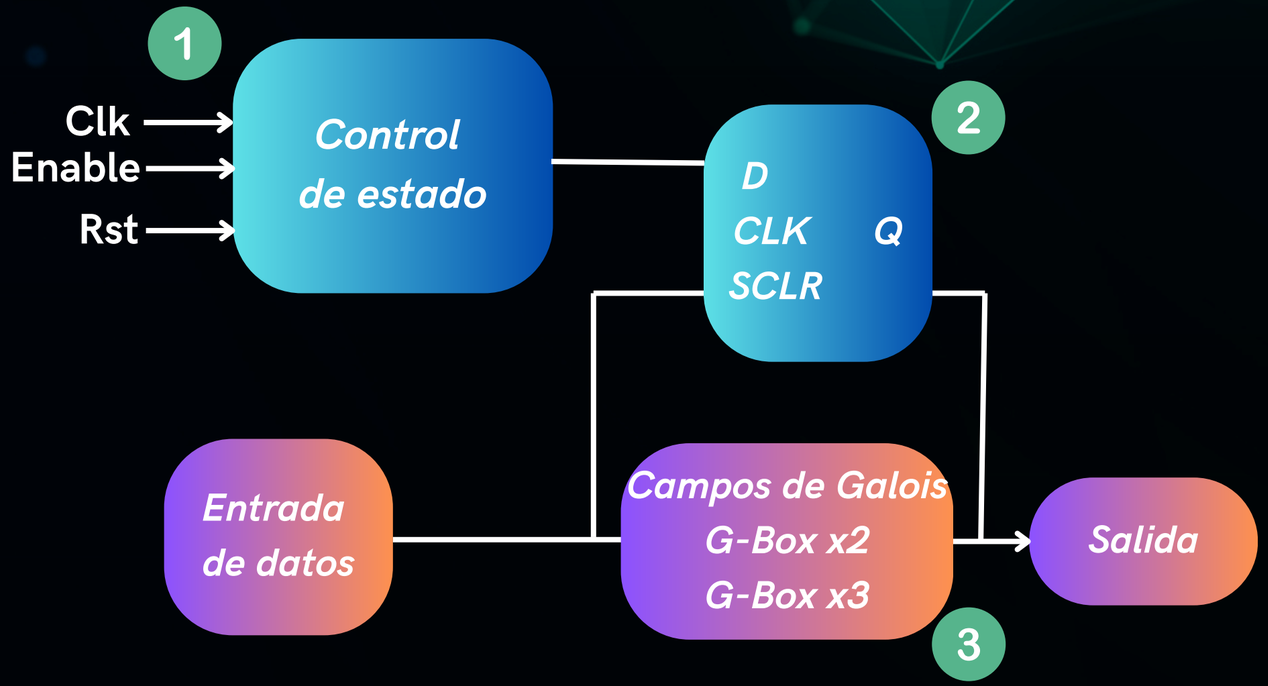
MIXCOLUMN

Txt_in (128 bits)

04	e0	b8	1e
66	b4	41	27
81	52	11	98
e5	ae	f1	e5

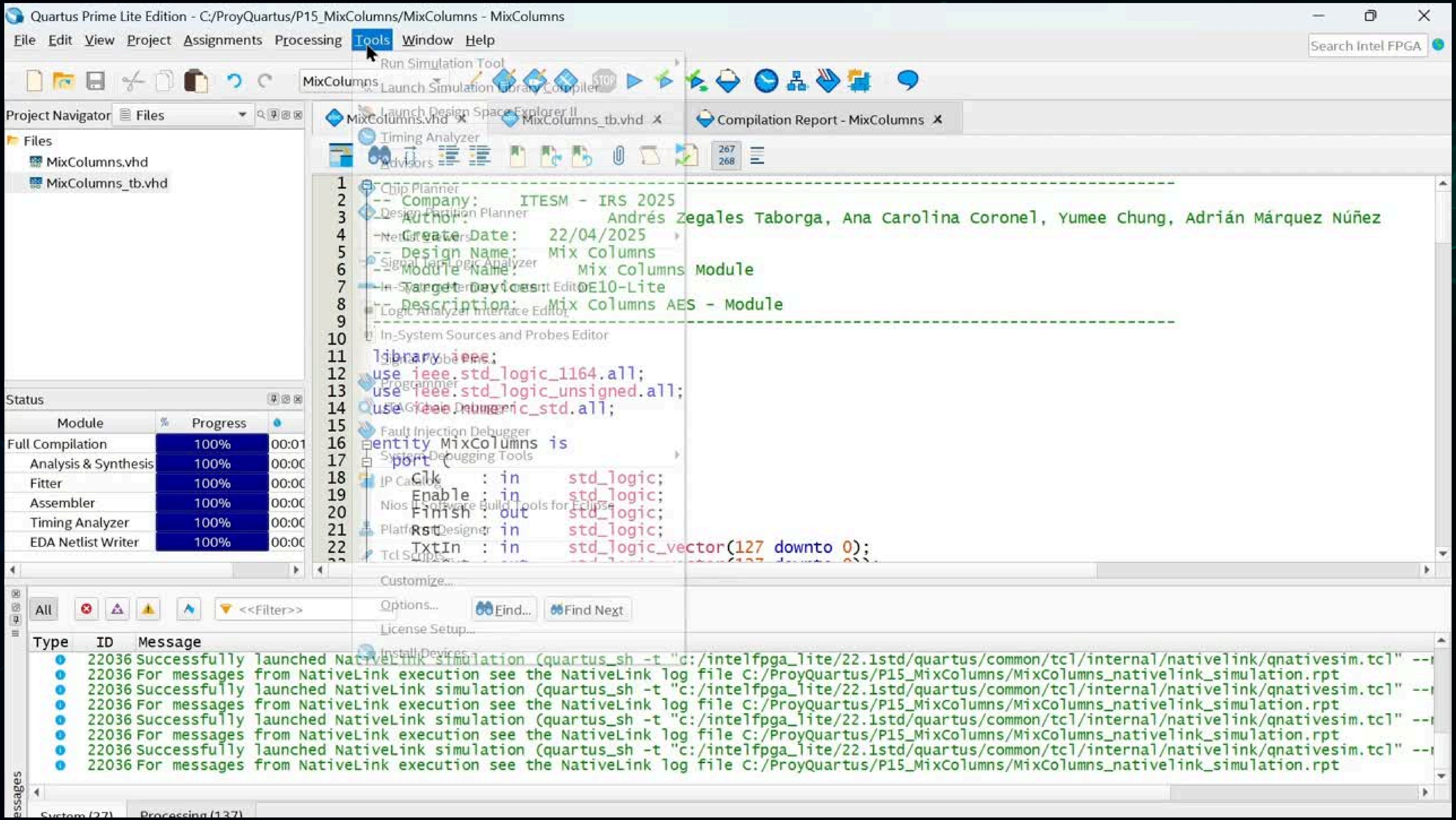
The MixColumns step along with the ShiftRows step is the primary source of diffusion in Rijndael.

Txt_out (128 bits)



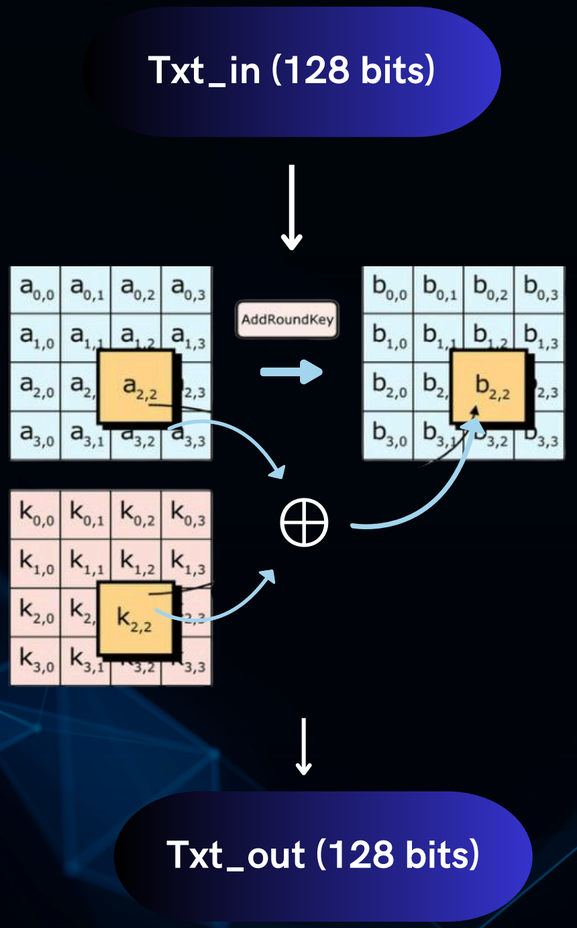
SIMULACIÓN

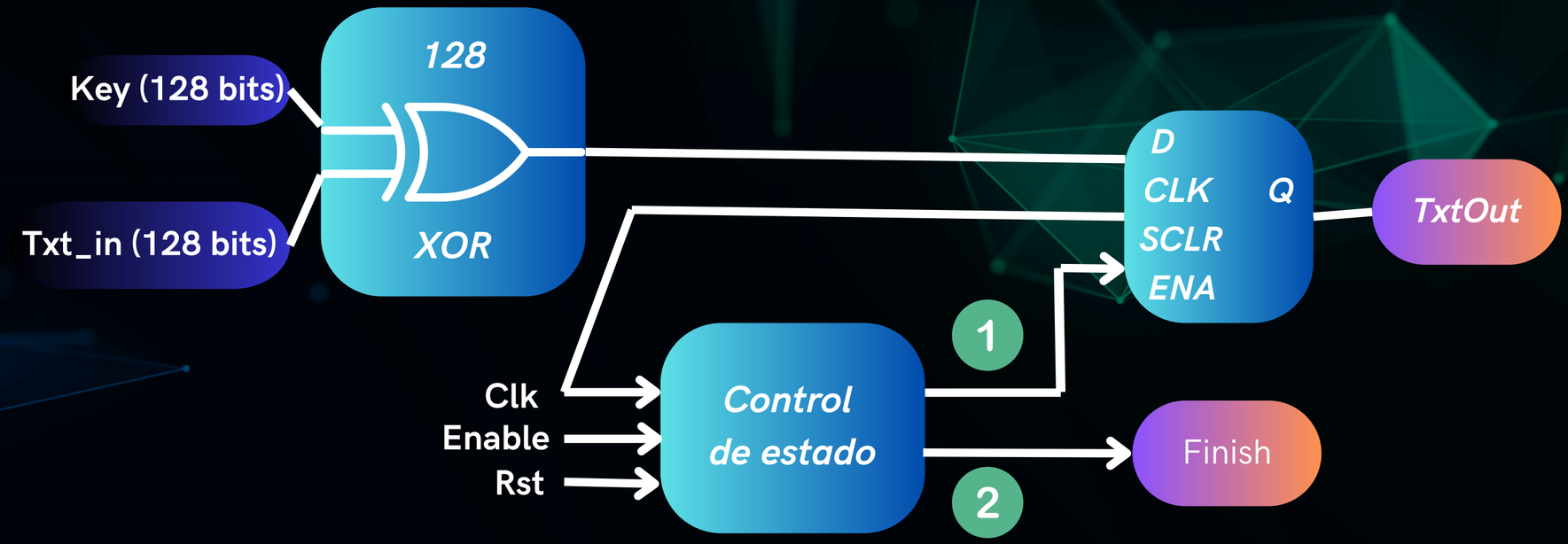
Mix Column

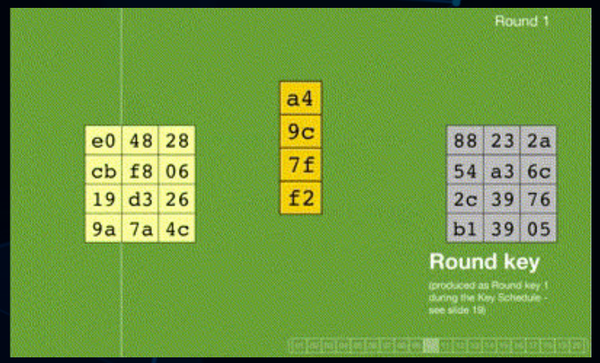


DESENCRIPTACIÓN

INV ADDROUNDKEY

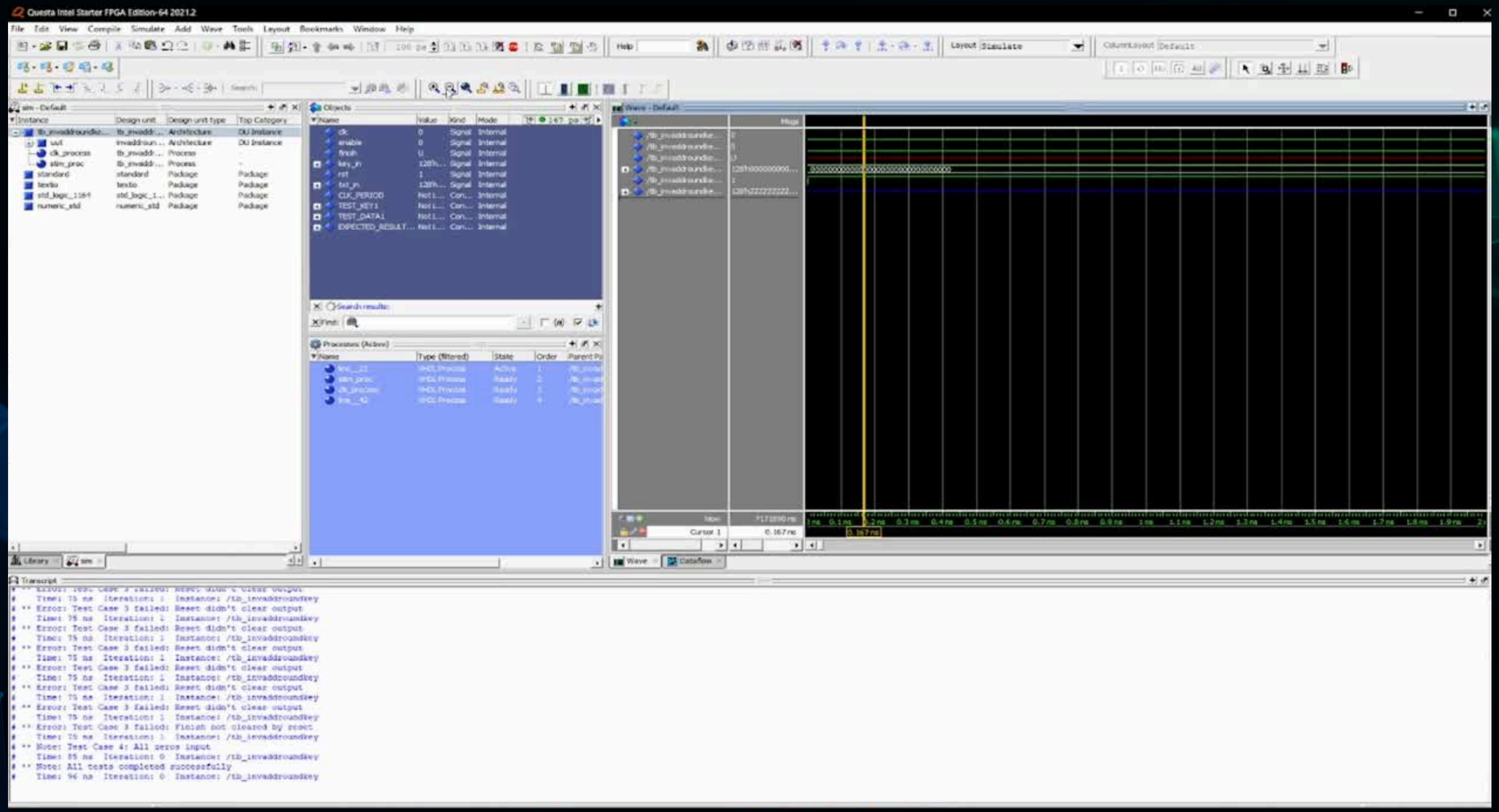




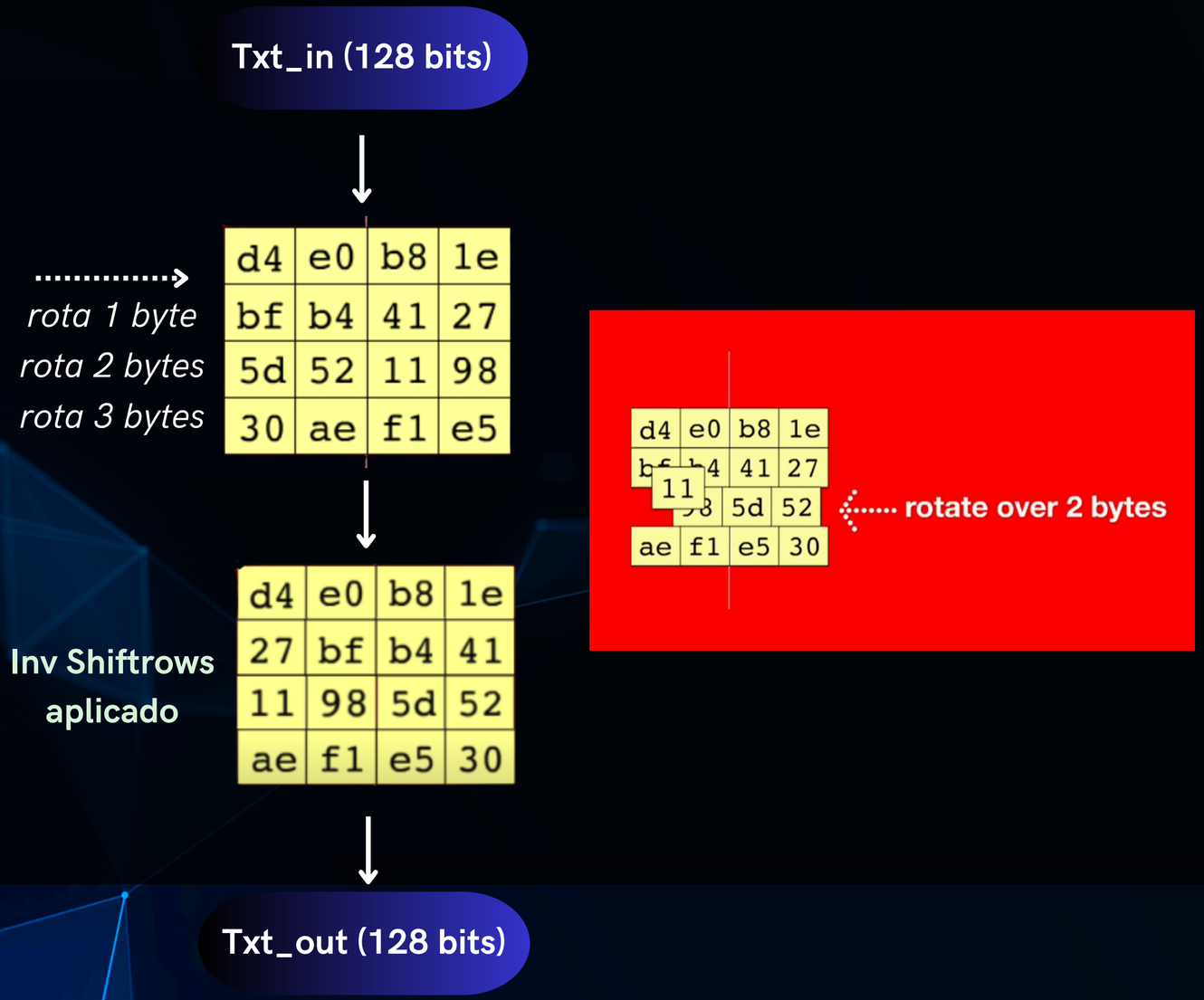


SIMULACIÓN

Inv Addroundkey

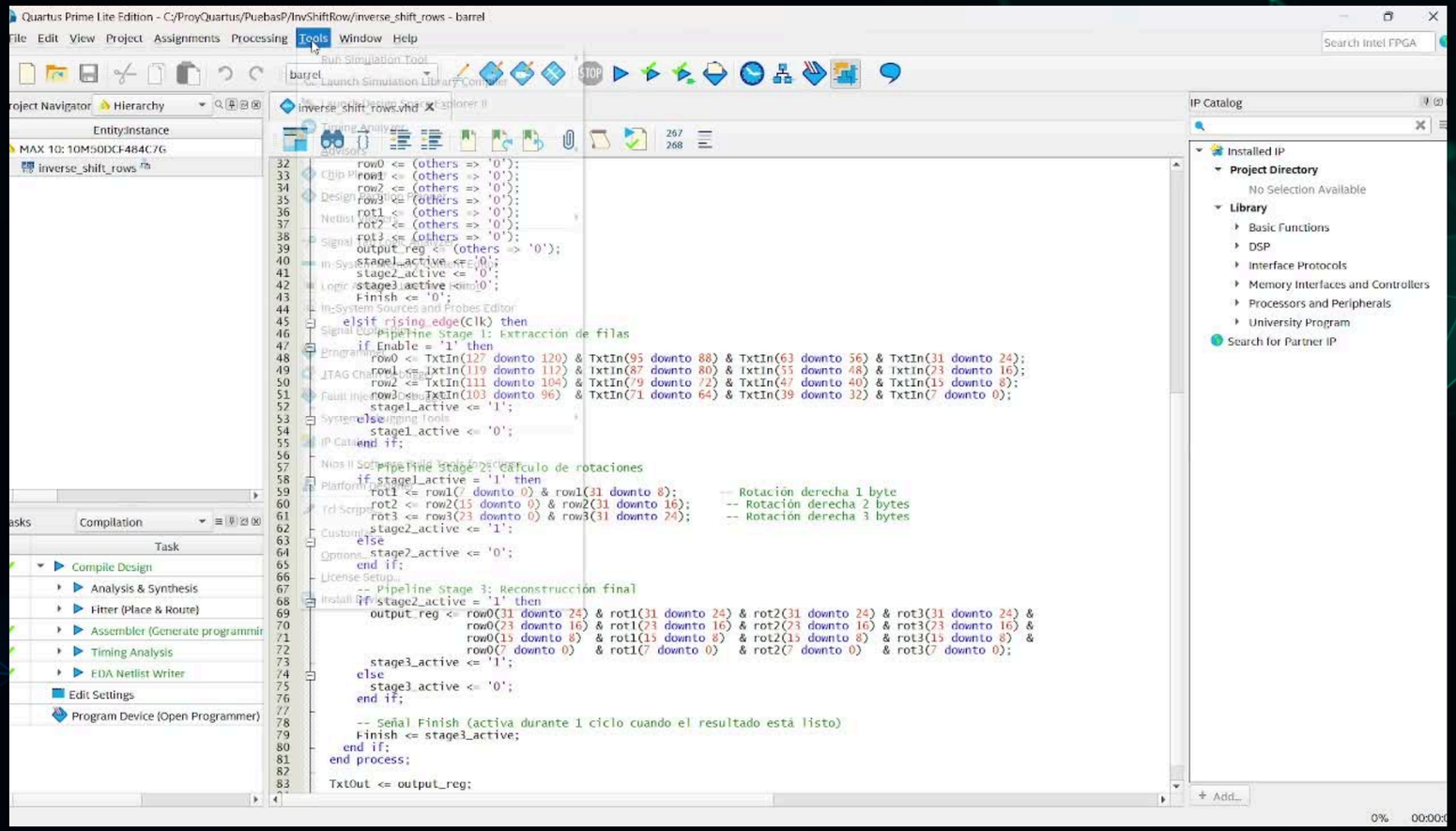


INV SHIFTROWS



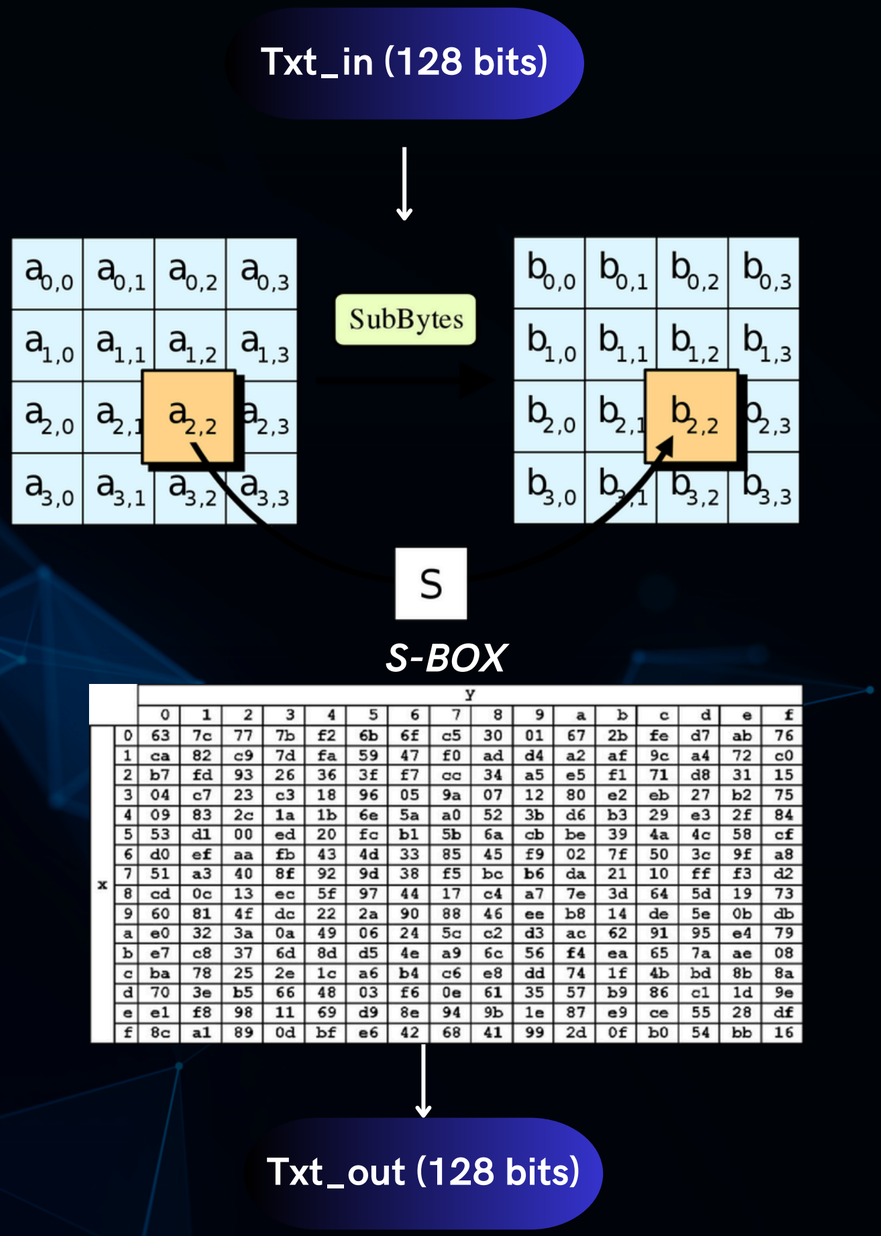
SIMULACIÓN

Inv Shiftrows

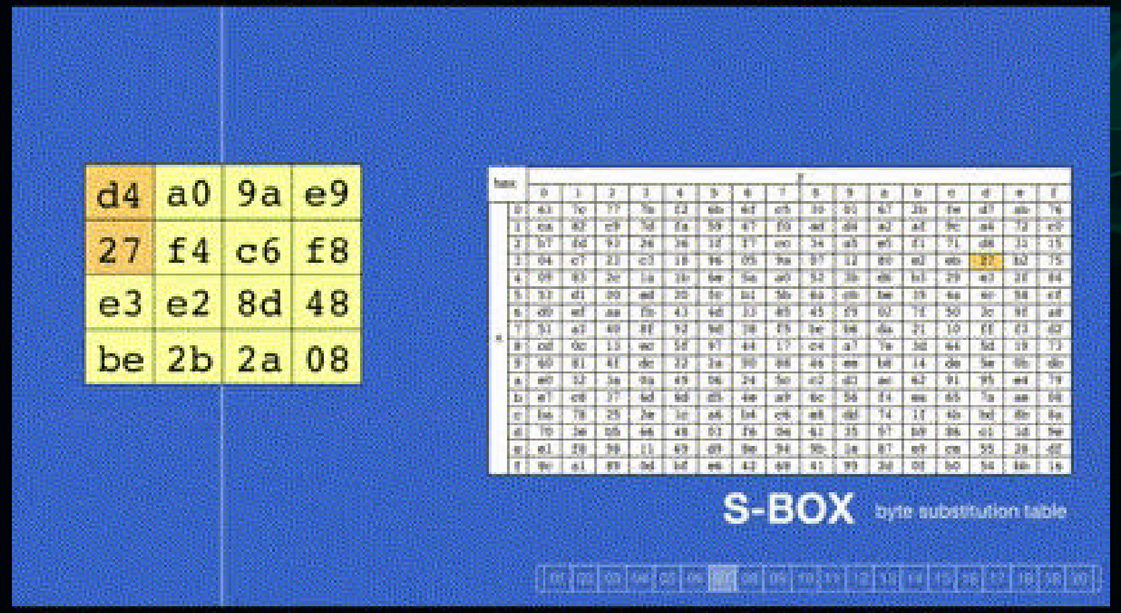



```
        row0 <= (others => '0');
        row1 <= (others => '0');
        row2 <= (others => '0');
        row3 <= (others => '0');
        rot1 <= (others => '0');
        rot2 <= (others => '0');
        rot3 <= (others => '0');
        output_reg <= (others => '0');
        stage1_active <= '0';
        stage2_active <= '0';
        stage3_active <= '0';
        Finish <= '0';
    elsif rising_edge(Clk) then
        -- Pipeline Stage 1: Extracción de filas
        if Enable = '1' then
            row0 <= TxtIn(127 downto 120) & TxtIn(95 downto 88) & TxtIn(63 downto 56) & TxtIn(31 downto 24);
            row1 <= TxtIn(119 downto 112) & TxtIn(87 downto 80) & TxtIn(55 downto 48) & TxtIn(23 downto 16);
            row2 <= TxtIn(111 downto 104) & TxtIn(79 downto 72) & TxtIn(47 downto 40) & TxtIn(15 downto 8);
            row3 <= TxtIn(103 downto 96)  & TxtIn(71 downto 64) & TxtIn(39 downto 32) & TxtIn(7 downto 0);
            stage1_active <= '1';
        else
            stage1_active <= '0';
        end if;

        -- Pipeline Stage 2: Cálculo de rotaciones
        if stage1_active = '1' then
            rot1 <= row1(7 downto 0) & row1(31 downto 8);       -- Rotación derecha 1 byte
            rot2 <= row2(15 downto 0) & row2(31 downto 16);     -- Rotación derecha 2 bytes
            rot3 <= row3(23 downto 0) & row3(31 downto 24);     -- Rotación derecha 3 bytes
            stage2_active <= '1';
        else
            stage2_active <= '0';
        end if;
        -- Pipeline Stage 3: Reconstrucción final
        if stage2_active = '1' then
            output_reg <= row0(31 downto 24) & rot1(31 downto 24) & rot2(31 downto 24) & rot3(31 downto 24) &
                          row0(23 downto 16) & rot1(23 downto 16) & rot2(23 downto 16) & rot3(23 downto 16) &
                          row0(15 downto 8)  & rot1(15 downto 8)  & rot2(15 downto 8)  & rot3(15 downto 8)  &
                          row0(7 downto 0)   & rot1(7 downto 0)   & rot2(7 downto 0)   & rot3(7 downto 0);
            stage3_active <= '1';
        else
            stage3_active <= '0';
        end if;

        -- Señal Finish (activa durante 1 ciclo cuando el resultado está listo)
        Finish <= stage3_active;
    end if;
end process;

TxtOut <= output_reg;
```

INV SUBBYTES

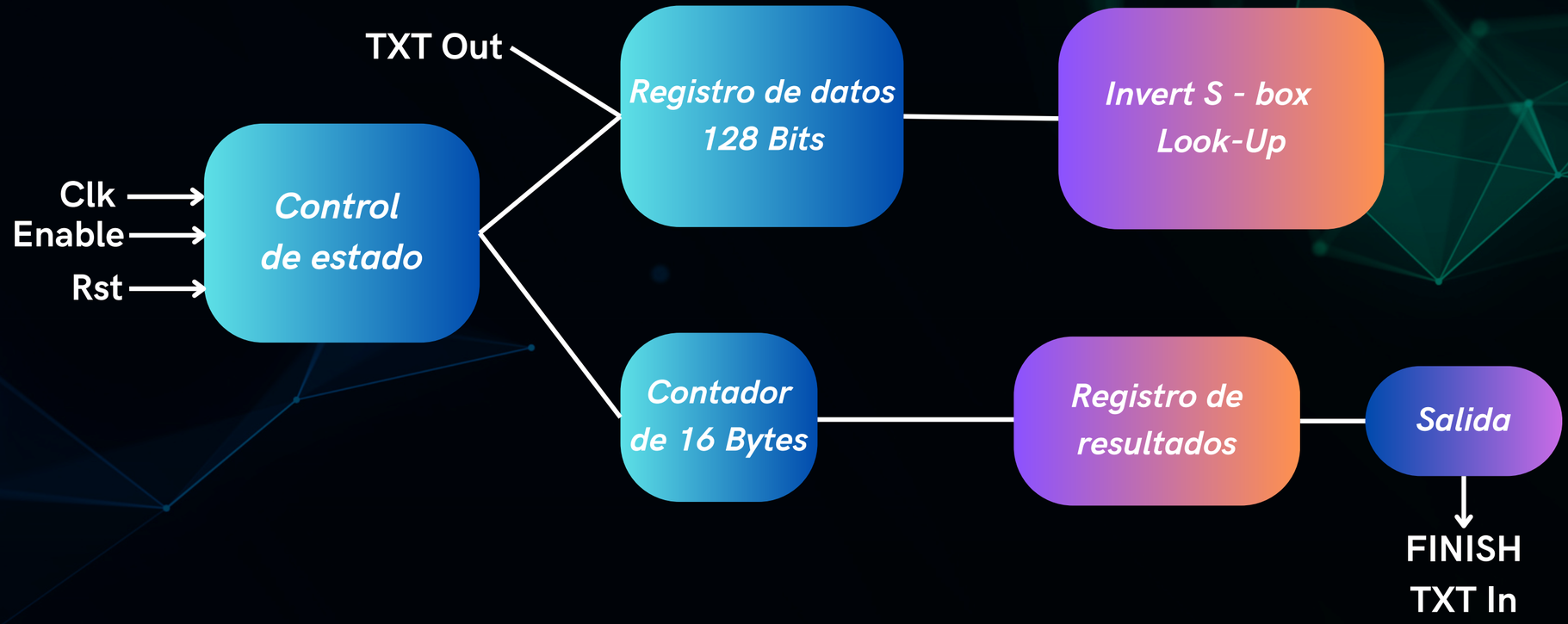


	y															
x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16



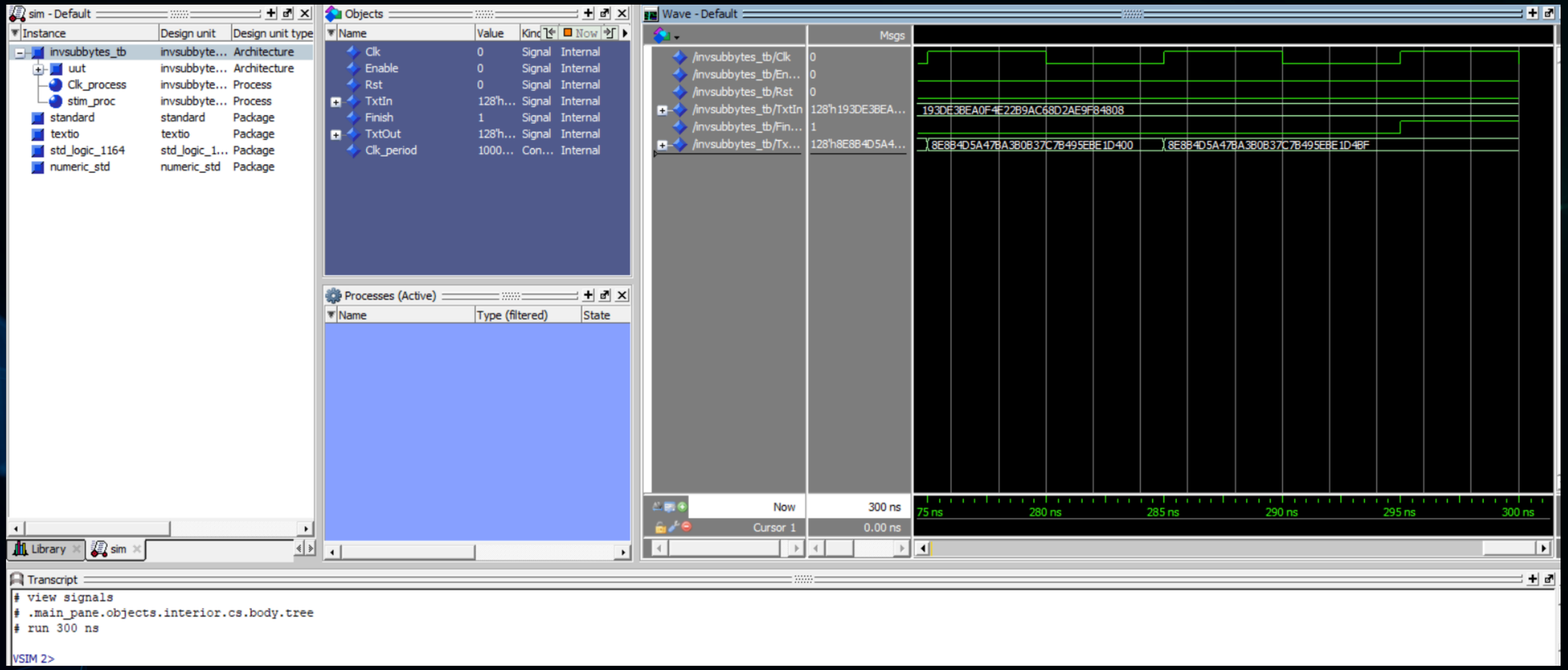
DIAGRAMA

Inv Subbytes



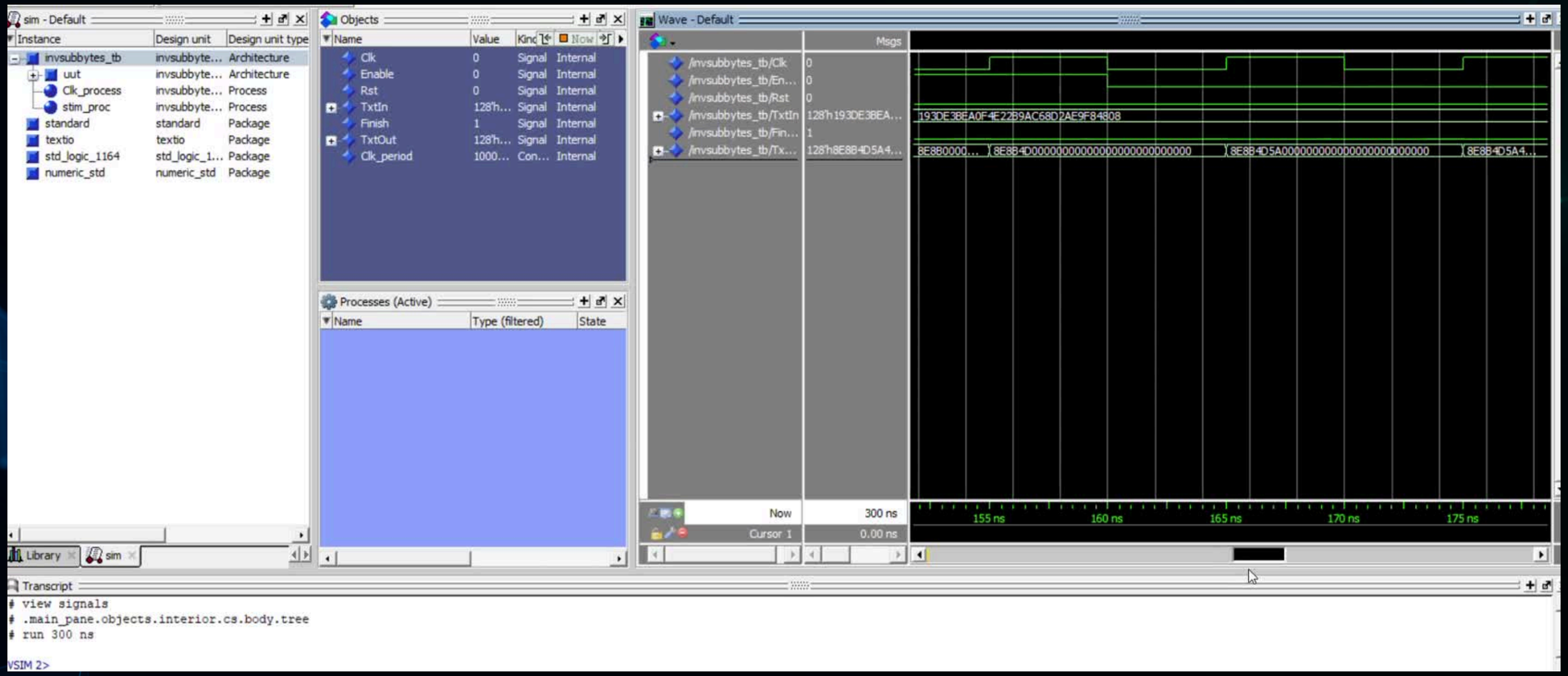
SIMULACIÓN

Inv Subbytes

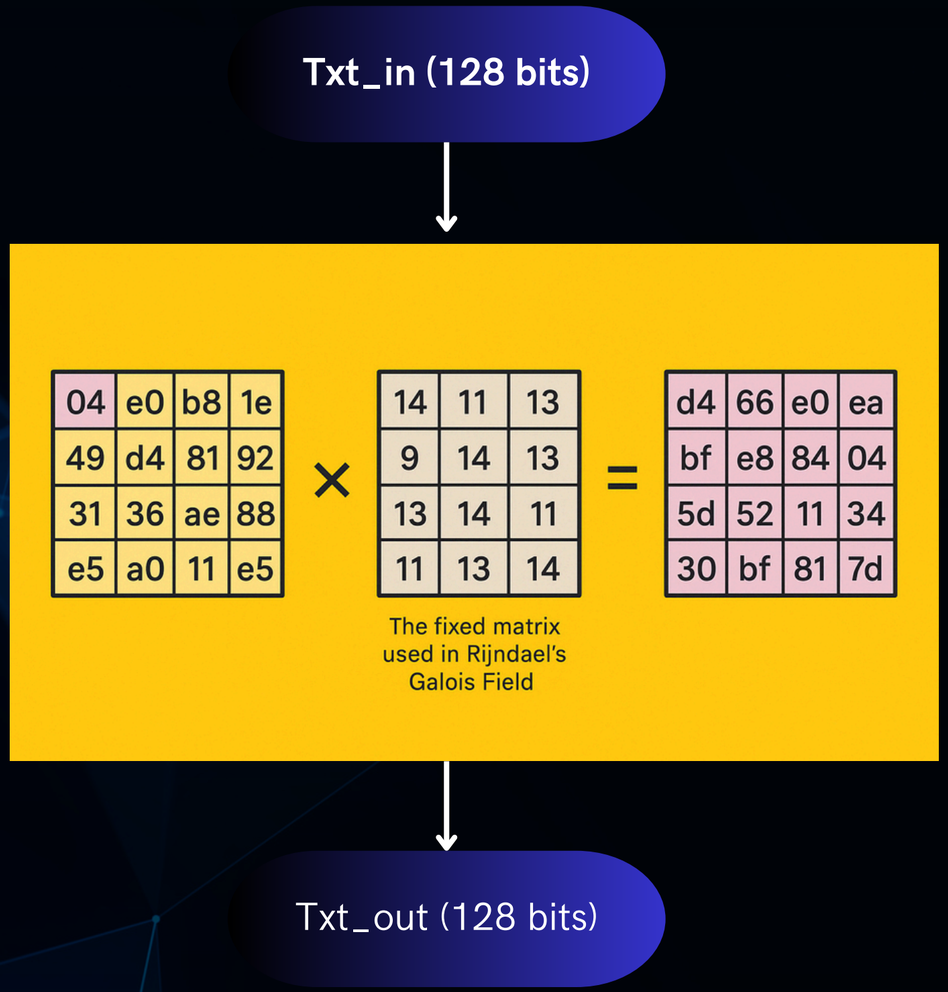


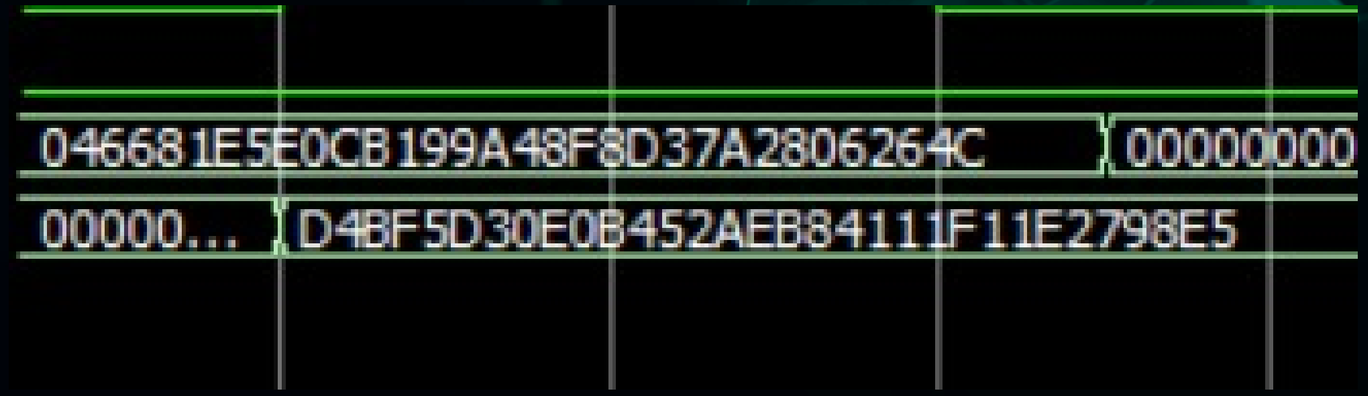
SIMULACIÓN

Inv Subbytes



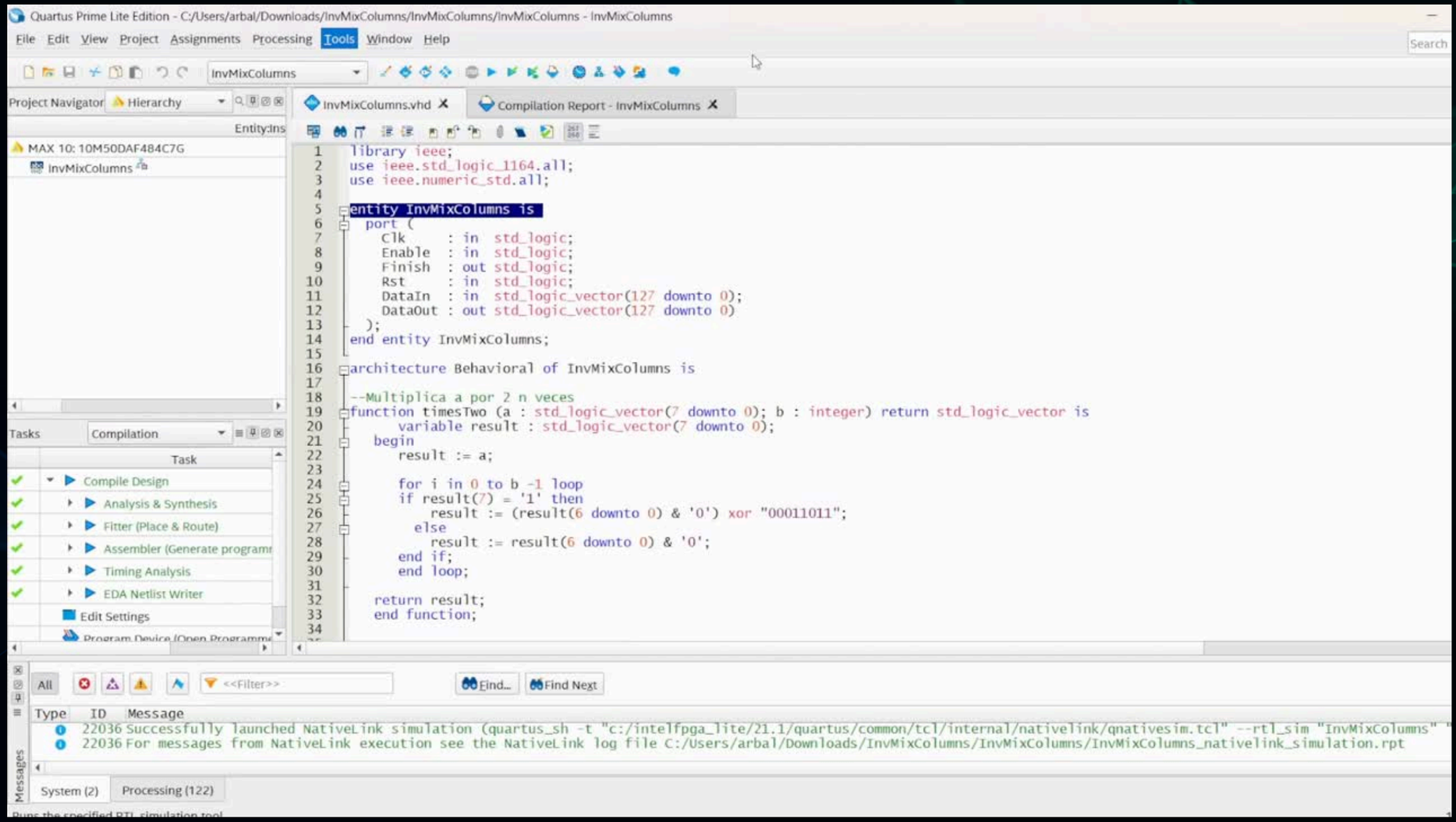
INV MIX COLUMN



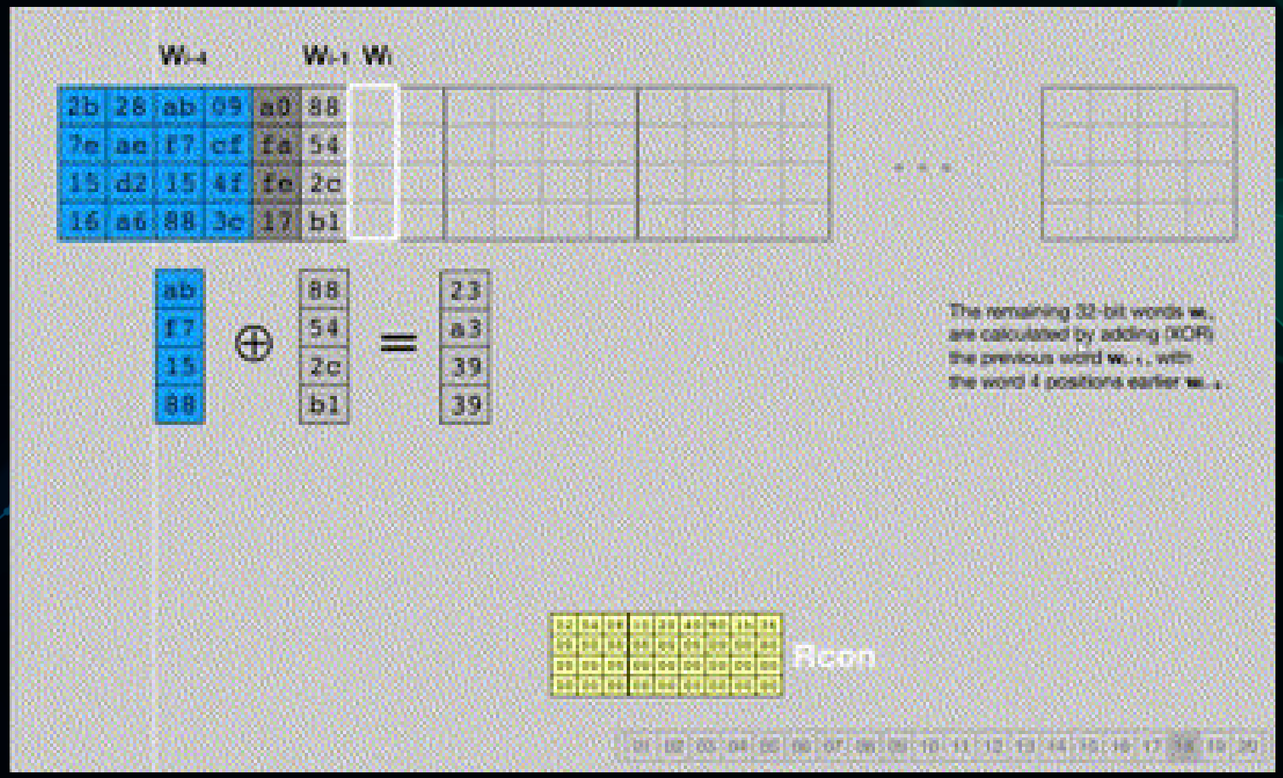


SIMULACIÓN

Inv Mix Column

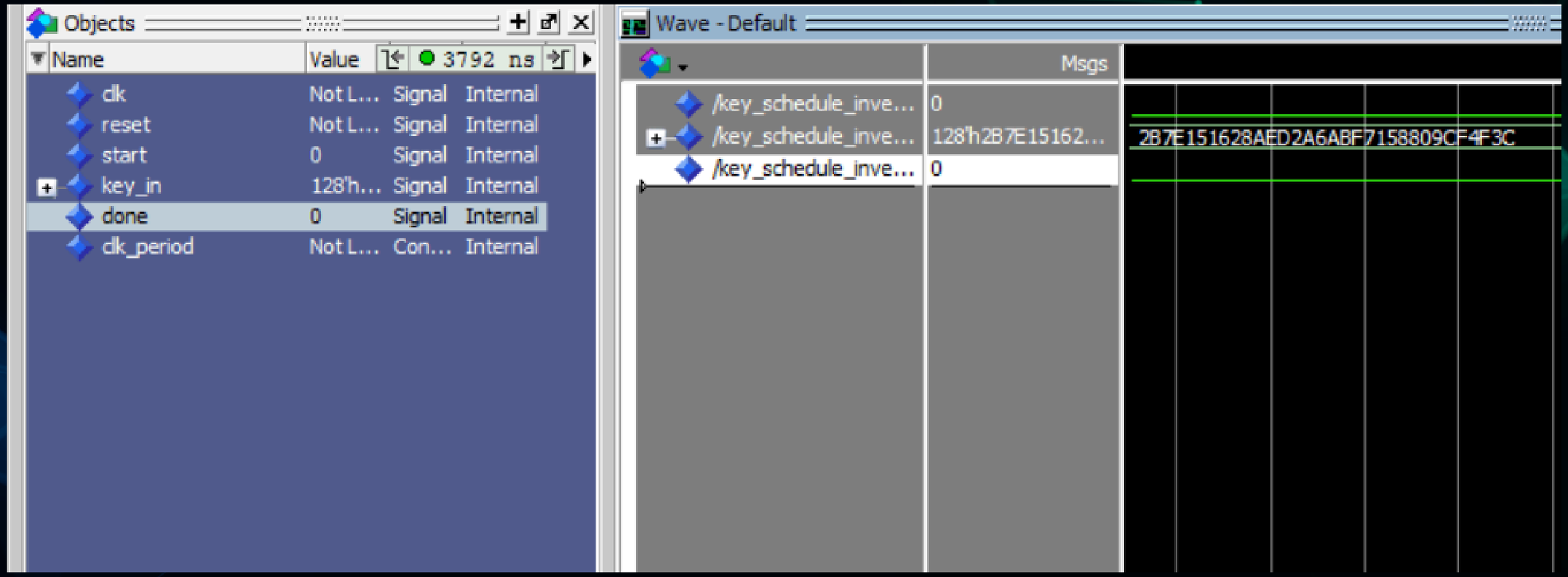


INV KEY SCHEDULE



SIMULACIÓN

Inv Key Schedule



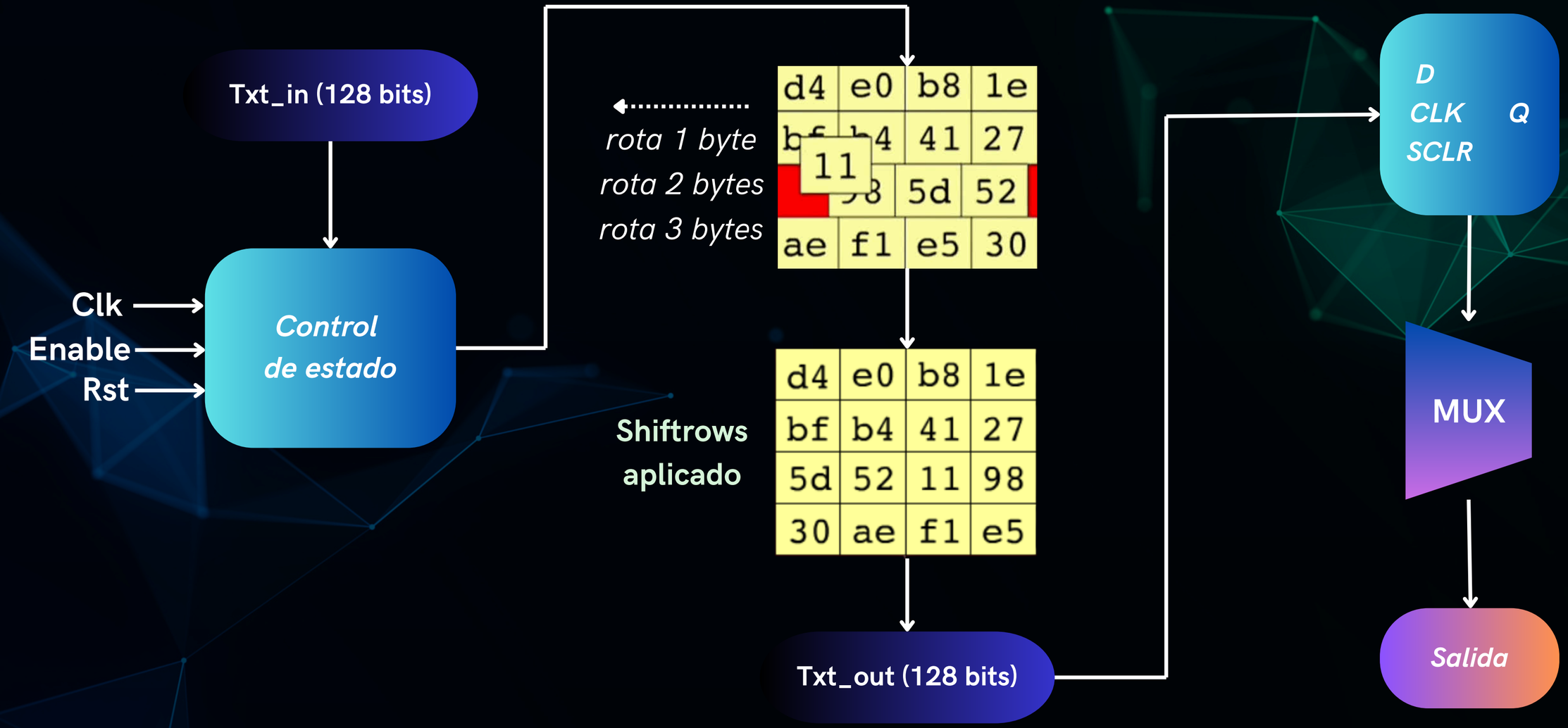
MUCHAS GRACIAS!

BIBLIOGRAFIA

EITCA. (2023) ¿Cuál es el papel del polinomio irreducible en la operación de multiplicación en los campos de Galois? https://eitca.org/cybersecurity/eitc-is-ccf-classical-cryptography-fundamentals/aes-block-cipher-cryptosystem/introduction-to-galois-fields-for-the-aes/examination-review-introduction-to-galois-fields-for-the-aes/what-is-the-role-of-the-irreducible-polynomial-in-the-multiplication-operation-in-galois-fields/

EITCA. (2023). ¿Cómo se realiza la multiplicación en Galois Fields en el contexto del algoritmo AES? https://es.eitca.org/cybersecurity/eitc-is-ccf-classical-cryptography-fundamentals/aes-block-cipher-cryptosystem/introduction-to-galois-fields-for-the-aes/examination-review-introduction-to-galois-fields-for-the-aes/how-is-multiplication-performed-in-galois-fields-in-the-context-of-the-aes-algorithm/

SHIFTROWS


Txt_in (128 bits)
Clk
Enable
Rst
Control de estado
rota 1 byte
rota 2 bytes
rota 3 bytes
d4 e0 b8 1e
bf b4 41 27
11
5d 52
ae f1 e5 30
Shiftrows aplicado
d4 e0 b8 1e
bf b4 41 27
5d 52 11 98
30 ae f1 e5
Txt_out (128 bits)
D
CLK
SCLR
Q
MUX
Salida

SIMULACIÓN

Shiftrows