Icono

Descripción generada automáticamente con confianza bajaUn dibujo de una persona

Descripción generada automáticamente con confianza bajaINSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

MATERIA

Arquitectura de Computadoras

PROFESOR

Vega García Nayeli

GRUPO

3CV11

Práctica 9

ALUMNOS

Cedillo Hernández Juan Daniel

Martínez Coronel Brayan Yosafat

Rodríguez Acosta Alan

FECHA DE ENTREGA

05/06/2021

# Unidad de control

# Código de implementación

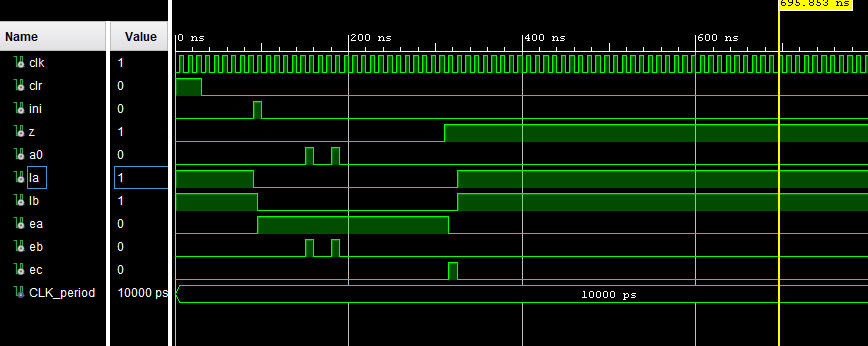
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** UC **is**
5. **Port (**
6. clk,clr,ini,z,a0:**in** std\_logic;
7. lb,la,ea,eb,ec:**out** std\_logic
8. );
9. **end** UC;
11. **architecture** Behavioral **of** UC **is**
12. **type** estados **is** (e0,e1,e2);
13. **signal** sig,act:estados;
15. **begin**
17. **process**(clr,clk)
18. **begin**
19. **if**(clr='1')**then**
20. **act <= e0;**
21. **elsif**(**rising\_edge**(clk))**then**
22. act<=sig;
23. **end** **if**;
24. **end** **process**;
26. **process**(act,ini,z,a0)
27. **begin**
28. la<='0';
29. lb<='0';
30. **ea<='0';**
31. eb<='0';
32. ec<='0';
33. **case** act **is**
34. **when** e0 =>
35. **lb<='1';**
36. **if** ini='0' **then**
37. la<='1';
38. sig<=e0;
39. **elsif** ini='1' **then**
40. **sig<=e1;**
41. **end** **if**;
42. **when** e1 =>
43. ea<='1';
44. **if** z='1' **then**
45. **sig<=e2;**
46. **elsif** z='0' **then**
47. **if** a0='1' **then**
48. eb<='1';
49. sig<=e1;
50. **elsif a0='0' then**
51. sig<=e1;
52. **end** **if**;
53. **end** **if**;
54. **when** e2 =>
55. **ec<='1';**
56. **if** ini='0' **then**
57. sig<=e0;
58. **elsif** ini='1' **then**
59. sig<=e2;
60. **end if;**
61. **when** **others** => sig<=e0;
62. **end** **case**;
63. **end** **process**;
65. **end Behavioral;**

# Código de simulación

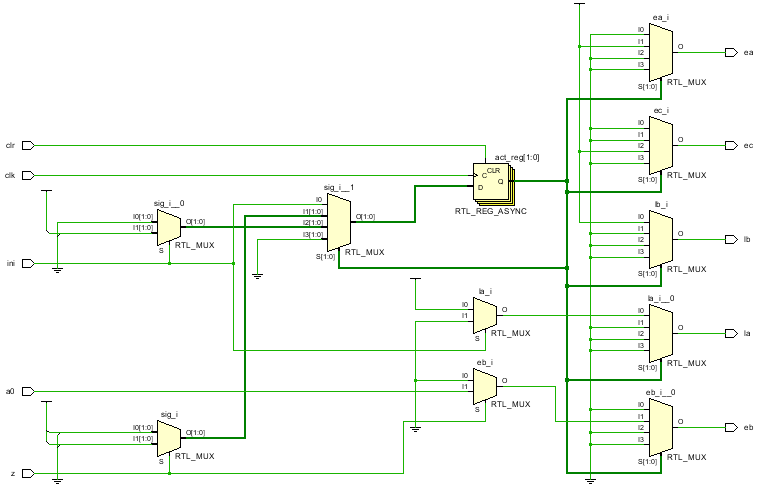
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;

5. **entity UCTB is**
6. *--  Port ( );*
7. **end** UCTB;
9. **architecture** Behavioral **of** UCTB **is**
10. **component UC is**
11. **Port** (
12. clk,clr,ini,z,a0:**in** std\_logic;
13. lb,la,ea,eb,ec:**out** std\_logic
14. );
15. **end component;**
16. **constant** CLK\_period : time := 10 ns;
17. **signal** clk,clr,ini,z,a0,lb,la,ea,eb,ec:std\_logic:='0';
18. **begin**
20. **uct:uc port map(**
21. clk=>clk,
22. clr=>clr,
23. ini=>ini,
24. z=>z,
25. **a0=>a0,**
26. lb=>lb,
27. la=>la,
28. ea=>ea,
29. eb=>eb,
30. **ec=>ec**
31. );
33. CLK\_process :**process**
34. **begin**
35. **CLK <= '0';**
36. **wait** **for** CLK\_period/2;
37. CLK <= '1';
38. **wait** **for** CLK\_period/2;
39. **end** **process**;
41. **process**
42. **begin**
43. clr<='1';
44. **wait** **for** 30 ns;
45. **clr<='0';**
46. **wait** **for** 60 ns;
47. ini<='1';
48. **wait** **for** 10 ns;
49. ini<='0';
50. **wait for 50 ns;**
51. a0<='1';
52. **wait** **for** 10 ns;
53. a0<='0';
54. **wait** **for** 20 ns;
55. **a0<='1';**
56. **wait** **for** 10 ns;
57. a0<='0';
58. **wait** **for** 120 ns;
59. z<='1';
60. **wait;**
61. **end** **process**;
63. **end** Behavioral;

# Forma de onda



# Diagrama RTL



# Registros

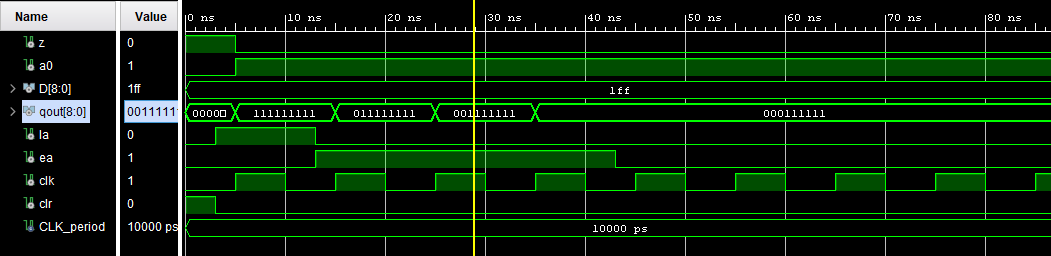
# Código de implementación

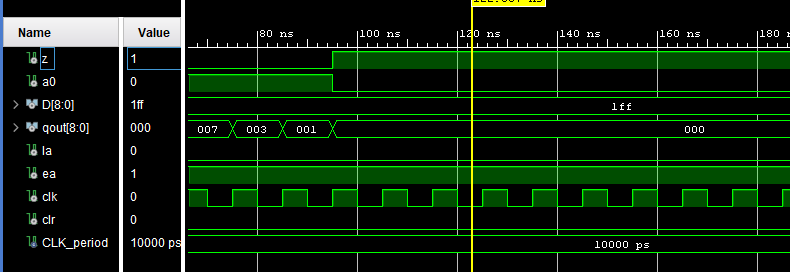
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** Registros **is**
5. **Port (**
6. z,a0:**out** std\_logic;
7. D:**in** std\_logic\_vector(8 **downto** 0);
8. qout:**out** std\_logic\_vector(8 **downto** 0);
9. la,ea,clk,clr:**in** std\_logic
10. **);**
11. **end** Registros;
13. **architecture** Behavioral **of** Registros **is**
14. **signal** q:std\_logic\_vector(8 **downto** 0);
15. **begin**
16. qout<=q;
18. **process**(clr,clk)
19. **begin**
20. **if(clr='1')then**
21. q<=(**others**=>'0');
22. **elsif**(**rising\_edge**(clk))**then**
23. multiplexor: **for** i **in** 0 **to** 8 **loop**
24. **case** la **is**
25. **when '0' =>**
26. **if** ea='0' **then**
27. q(i)<=q(i);
28. **elsif** ea='1' **then**
29. **if**(i<8) **then**
30. **q(i)<=q(i+1);**
31. **else**
32. q(i)<='0';
33. **end** **if**;
34. **end** **if**;
35. **when '1' =>**
36. **if** ea='0' **then**
37. q(i)<=D(i);
38. **end** **if**;
39. **when** **others** =>q(i)<='-';
40. **end case;**
41. **end** **loop**;
42. **end** **if**;
43. **end** **process**;
45. **process(q)**
46. **variable** aux: std\_logic;
47. **begin**
48. aux:=q(8);
50. **for i in 0 to 7 loop**
51. aux:=aux **or** q(i);
52. **end** **loop**;
53. z<= **not** aux;
54. **end** **process**;
56. a0<=q(0);
58. **end** Behavioral;

# Código de simulación

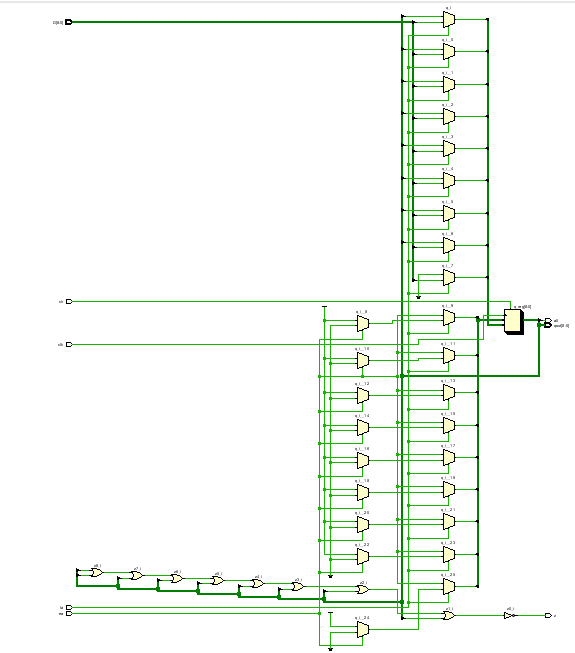
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** regtb **is**
5. ***--  Port ( );***
6. **end** regtb;
8. **architecture** Behavioral **of** regtb **is**
9. **component** Registros **is**
10. **Port (**
11. z,a0:**out** std\_logic;
12. D:**in** std\_logic\_vector(8 **downto** 0);
13. qout:**out** std\_logic\_vector(8 **downto** 0);
14. la,ea,clk,clr:**in** std\_logic
15. **);**
16. **end** **component**;
17. **signal**    z,a0: std\_logic;
18. **signal**    D: std\_logic\_vector(8 **downto** 0);
19. **signal**    qout: std\_logic\_vector(8 **downto** 0);
20. **signal la,ea,clk,clr: std\_logic;**
21. **constant** CLK\_period : time := 10 ns;
22. **begin**
24. regs:Registros **port** **map**(
25. **z=>z,**
26. D=>D,
27. qout=>qout,
28. la=>la,
29. ea=>ea,
30. **clk=>clk,**
31. clr=>clr,
32. a0=>a0
33. );
35. **CLK\_process :process**
36. **begin**
37. CLK <= '0';
38. **wait** **for** CLK\_period/2;
39. CLK <= '1';
40. **wait for CLK\_period/2;**
41. **end** **process**;
43. **process**
44. **begin**
45. **la<='0';**
46. ea<='0';
47. clr<='1';
48. d<="111111111";
49. **wait** **for** 3 ns;
50. **clr<='0';**
51. la<='1';
52. **wait** **for** 10 ns;
53. la<='0';
54. ea<='1';
55. **wait for 30 ns;**
56. ea<='0';
57. **wait**;
59. **end** **process**;
61. **end** Behavioral;

# Forma de onda





# Diagrama RTL



# Contador

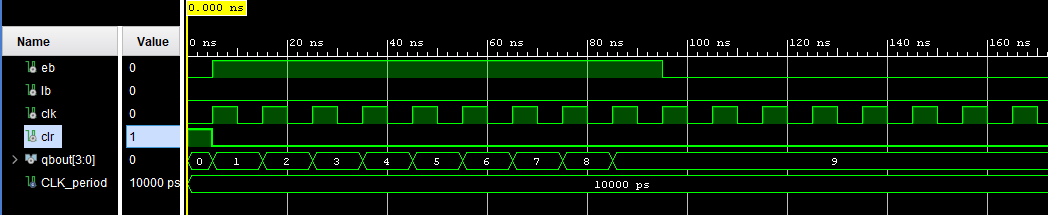
# Código de implementación

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
3. **use** ieee.std\_logic\_unsigned.**all**;
4. **use** ieee.std\_logic\_arith.**all**;
6. **entity** contador **is**
7. **Port** (
8. eb,lb,clk,clr: **in** std\_logic;
9. qbout: **out** std\_logic\_vector(3 **downto** 0));
10. **end contador;**
12. **architecture** Behavioral **of** contador **is**
13. **constant** db: std\_logic\_vector(3 **downto** 0):="0000";
14. **signal** qb:std\_logic\_vector(3 **downto** 0);
15. **begin**
17. qbout<=qb;
19. **process**(clk,clr)
20. **begin**
21. **if**(clr='1') **then**
22. qb<="0000";
23. **elsif**(**rising\_edge**(clk)) **then**
24. **if**(lb='0' **and** eb='0') **then**
25. **qb<=qb;**
26. **elsif**(lb='1' **and** eb='0') **then**
27. qb<=db;
28. **elsif**(lb='0' **and** eb='1') **then**
29. **if** qb<9 **then**
30. **qb<=qb+1;**
31. **else**
32. qb<="0000";
33. **end** **if**;
34. **end** **if**;
35. **end if;**
36. **end** **process**;
38. **end** Behavioral;

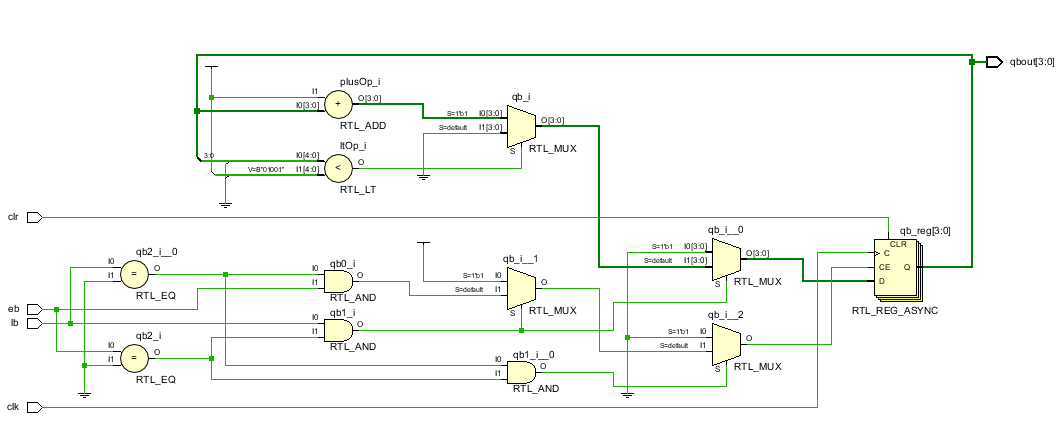
# Código de simulación

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** contadortb **is**
5. ***--  Port ( );***
6. **end** contadortb;
8. **architecture** Behavioral **of** contadortb **is**
9. **component** contador **is**
10. **Port (**
11. eb,lb,clk,clr: **in** std\_logic;
12. qbout: **out** std\_logic\_vector(3 **downto** 0));
13. **end** **component**;
15. **signal eb,lb,clk,clr: std\_logic;**
16. **signal** qbout: std\_logic\_vector(3 **downto** 0);
17. **constant** CLK\_period : time := 10 ns;
18. **begin**
20. **cont:contador port map(**
21. eb=>eb,
22. lb=>lb,
23. clk=>clk,
24. clr=>clr,
25. **qbout=>qbout**
26. );
28. CLK\_process :**process**
29. **begin**
30. **CLK <= '0';**
31. **wait** **for** CLK\_period/2;
32. CLK <= '1';
33. **wait** **for** CLK\_period/2;
34. **end** **process**;
36. **process**
37. **begin**
38. lb<='0';
39. eb<='0';
40. **clr<='1';**
41. **wait** **for** 5 ns;
42. clr<='0';
43. eb<='1';
44. **wait** **for** 90 ns;
45. **eb<='0';**
46. **wait**;
47. **end** **process**;
49. **end** Behavioral;

# Forma de onda



# Diagrama RTL



# Decodificador y multiplexor

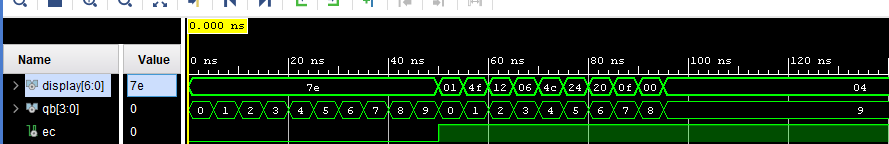
# Código de implementación

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** decoymux **is**
5. **Port (**
6. display: **out** std\_logic\_vector(6 **downto** 0);
7. qb:**in** std\_logic\_vector(3 **downto** 0);
8. ec:**in** std\_logic
9. );
10. **end decoymux;**
12. **architecture** Behavioral **of** decoymux **is**
14. **begin**
16. deco: **process**(qb,ec)
17. **begin**
18. **case** qb&ec **is**
19. **when** "00001" =>display<="0000001";
20. **when "00011" =>display<="1001111";**
21. **when** "00101" =>display<="0010010";
22. **when** "00111" =>display<="0000110";
23. **when** "01001" =>display<="1001100";
24. **when** "01011" =>display<="0100100";
25. **when "01101" =>display<="0100000";**
26. **when** "01111" =>display<="0001111";
27. **when** "10001" =>display<="0000000";
28. **when** "10011" =>display<="0000100";
29. **when** "00000" =>display<="1111110";
30. **when "00010" =>display<="1111110";**
31. **when** "00100" =>display<="1111110";
32. **when** "00110" =>display<="1111110";
33. **when** "01000" =>display<="1111110";
34. **when** "01010" =>display<="1111110";
35. **when "01100" =>display<="1111110";**
36. **when** "01110" =>display<="1111110";
37. **when** "10000" =>display<="1111110";
38. **when** "10010" =>display<="1111110";
39. **when** **others** =>display<="-------";
40. **end case;**
41. **end** **process** deco;
43. **end** Behavioral;

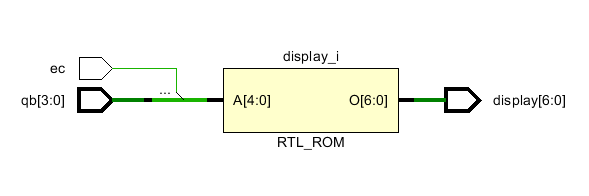
# Código de simulación

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** decoymuxtb **is**
5. ***--Port ( );***
6. **end** decoymuxtb;
8. **architecture** Behavioral **of** decoymuxtb **is**
9. **component** decoymux **is**
10. **Port (**
11. display: **out** std\_logic\_vector(6 **downto** 0);
12. qb:**in** std\_logic\_vector(3 **downto** 0);
13. ec:**in** std\_logic
14. );
15. **end component;**
16. **signal** display: std\_logic\_vector(6 **downto** 0);
17. **signal** qb: std\_logic\_vector(3 **downto** 0);
18. **signal** ec: std\_logic;
19. **begin**
21. dym:decoymux **port** **map**(
22. display=>display,
23. qb=>qb,
24. ec=>ec
25. **);**
27. **process**
28. **begin**
29. ec<='0';
30. **qb<="0000";**
31. **wait** **for** 5 ns;
32. qb<="0001";
33. **wait** **for** 5 ns;
34. qb<="0010";
35. **wait for 5 ns;**
36. qb<="0011";
37. **wait** **for** 5 ns;
38. qb<="0100";
39. **wait** **for** 5 ns;
40. **qb<="0101";**
41. **wait** **for** 5 ns;
42. qb<="0110";
43. **wait** **for** 5 ns;
44. qb<="0111";
45. **wait for 5 ns;**
46. qb<="1000";
47. **wait** **for** 5 ns;
48. qb<="1001";
49. **wait** **for** 5 ns;
50. **ec<='1';**
51. qb<="0000";
52. **wait** **for** 5 ns;
53. qb<="0001";
54. **wait** **for** 5 ns;
55. **qb<="0010";**
56. **wait** **for** 5 ns;
57. qb<="0011";
58. **wait** **for** 5 ns;
59. qb<="0100";
60. **wait for 5 ns;**
61. qb<="0101";
62. **wait** **for** 5 ns;
63. qb<="0110";
64. **wait** **for** 5 ns;
65. **qb<="0111";**
66. **wait** **for** 5 ns;
67. qb<="1000";
68. **wait** **for** 5 ns;
69. qb<="1001";
70. **wait;**
71. **end** **process**;
73. **end** Behavioral;

# Forma de onda



# Diagrama RTL



# Arquitectura completa

# Paquete

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **package** paquetecasm **is**
6. **component** Registros **is**
7. **Port** (
8. z,a0:**out** std\_logic;
9. D:**in** std\_logic\_vector(8 **downto** 0);
10. **qout:out std\_logic\_vector(8 downto 0);**
11. la,ea,clk,clr:**in** std\_logic
12. );
13. **end** **component**;
15. **component UC is**
16. **Port** (
17. clk,clr,ini,z,a0:**in** std\_logic;
18. lb,la,ea,eb,ec:**out** std\_logic
19. );
20. **end component;**
22. **component** contador **is**
23. **Port** (
24. eb,lb,clk,clr: **in** std\_logic;
25. **qbout: out std\_logic\_vector(3 downto 0));**
26. **end** **component**;
28. **component** decoymux **is**
29. **Port** (
30. **display: out std\_logic\_vector(6 downto 0);**
31. qb:**in** std\_logic\_vector(3 **downto** 0);
32. ec:**in** std\_logic
33. );
34. **end** **component**;
36. **end** **package**;

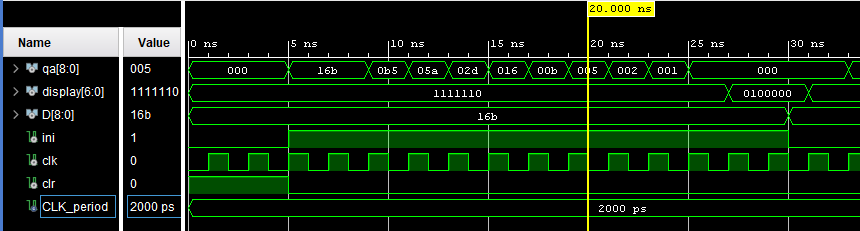
# Código de implementación

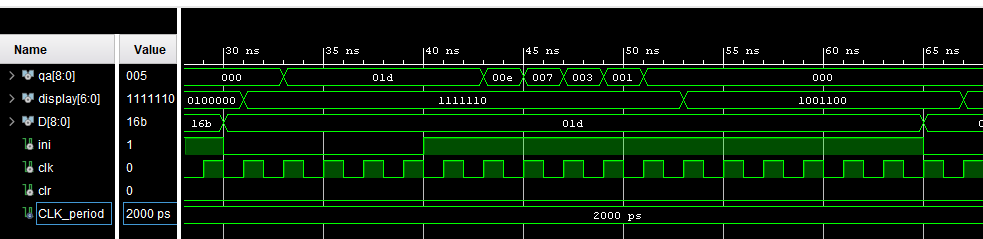
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
3. **library** work;
4. **use** work.paquetecasm.**all**;
6. **entity** CASM2 **is**
7. **Port** (
8. qa: **out** std\_logic\_vector(8 **downto** 0);
9. display:**out** std\_logic\_vector(6 **downto** 0);
10. **D: in  std\_logic\_vector(8 downto 0);**
11. ini,clk,clr: **in** std\_logic
12. );
13. **end** CASM2;
15. **architecture Behavioral of CASM2 is**
16. **signal** z,a0: std\_logic;
17. *--signal D: std\_logic\_vector(8 downto 0);*
18. **signal** qout: std\_logic\_vector(8 **downto** 0);
19. **signal** lb,la,ea,eb,ec: std\_logic;
20. **signal qb: std\_logic\_vector(3 downto 0);**
22. **begin**
24. uncon:uc **port** **map**(
25. **clk=>clk,**
26. clr=>clr,
27. ini=>ini,
28. z=>z,
29. a0=>a0,
30. **la=>la,**
31. lb=>lb,
32. ea=>ea,
33. eb=>eb,
34. ec=>ec
35. **);**
37. cont:contador **port** **map**(
38. eb=>eb,
39. lb=>lb,
40. **clk=>clk,**
41. clr=>clr,
42. qbout=>qb
43. );
45. **dec: decoymux port map(**
46. display=>display,
47. qb=>qb,
48. ec=>ec
49. );
51. reg: registros **port** **map**(
52. D=>D,
53. z=>z,
54. a0=>a0,
55. **qout=>qa,**
56. clk=>clk,
57. clr=>clr,
58. la=>la,
59. ea=>ea
60. **);**
62. **end** Behavioral;

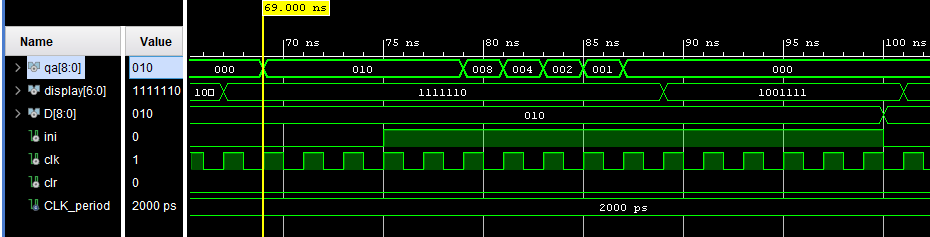
# Código de simulación

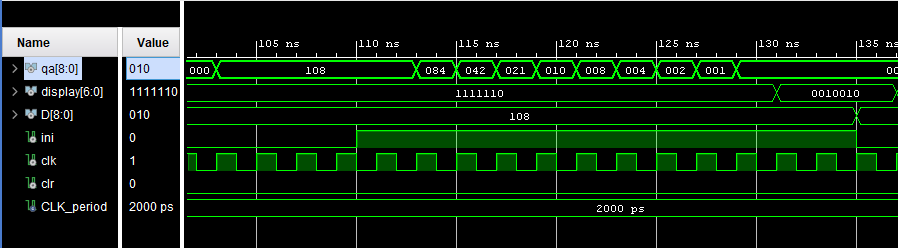
1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** CASMTB **is**
5. ***--  Port ( );***
6. **end** CASMTB;
8. **architecture** Behavioral **of** CASMTB **is**
9. **component** CASM2 **is**
10. **Port (**
11. qa: **out** std\_logic\_vector(8 **downto** 0);
12. display:**out** std\_logic\_vector(6 **downto** 0);
13. D: **in**  std\_logic\_vector(8 **downto** 0);
14. ini,clk,clr: **in** std\_logic
15. **);**
16. **end** **component**;
17. **signal** qa: std\_logic\_vector(8 **downto** 0);
18. **signal** display: std\_logic\_vector(6 **downto** 0);
19. **signal** D: std\_logic\_vector(8 **downto** 0);
20. **signal ini,clk,clr: std\_logic;**
21. **constant** CLK\_period : time := 2 ns;
22. **begin**
24. CLK\_process :**process**
25. **begin**
26. CLK <= '0';
27. **wait** **for** CLK\_period/2;
28. CLK <= '1';
29. **wait** **for** CLK\_period/2;
30. **end process;**
32. c12:casm2 **port** **map**(
33. qa=>qa,
34. display=>display,
35. **D=>D,**
36. ini=>ini,
37. clk=>clk,
38. clr=>clr
39. );
41. **process**
42. **begin**
43. D<="101101011";
44. ini<='0';
45. **clr<='1';**
46. **wait** **for** 5 ns;
47. clr<='0';
48. ini<='1';
49. **wait** **for** 25 ns;
50. **D<="000011101";**
51. ini<='0';
52. **wait** **for** 10 ns;
53. ini<='1';
54. **wait** **for** 25 ns;
55. **D<="000010000";**
56. ini<='0';
57. **wait** **for** 10 ns;
58. ini<='1';
59. **wait** **for** 25 ns;
60. **D<="100001000";**
61. ini<='0';
62. **wait** **for** 10 ns;
63. ini<='1';
64. **wait** **for** 25 ns;
65. **D<="000000000";**
66. ini<='0';
67. **wait** **for** 10 ns;
68. ini<='1';
69. **wait**;
71. **end** **process**;
73. **end** Behavioral;

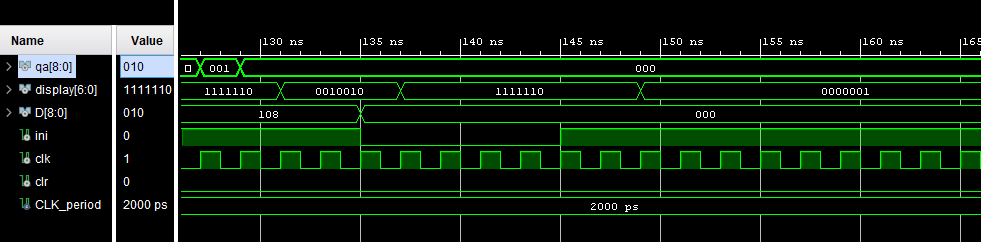
# Forma de onda



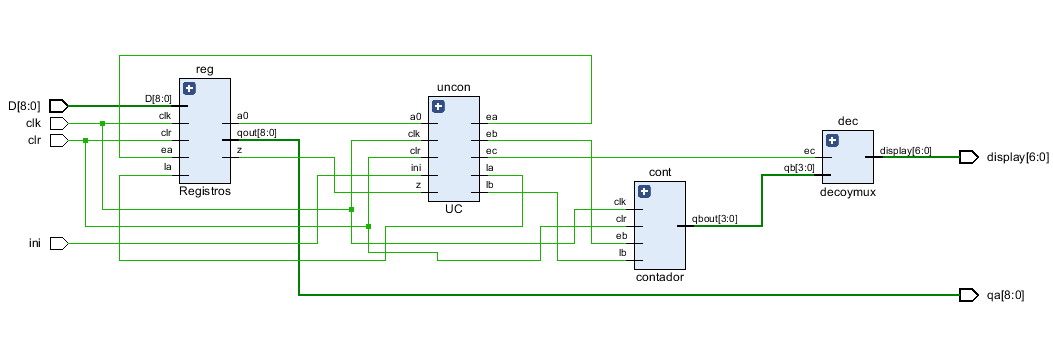








# Diagrama RTL



# Divisor de frecuencia

# Código de implementación

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** divisorDeFrecuencia **is**
5. **Port (**
6. *--en,*
7. clk,clr:**in** std\_logic;
8. q:**inout** std\_logic\_vector(25 **downto** 0)
9. );
10. **end divisorDeFrecuencia;**
12. **architecture** Behavioral **of** divisorDeFrecuencia **is**
13. **begin**
14. **process**(clk,clr)
15. **variable aux:std\_logic;**
16. **begin**
17. **if**(clr='1') **then**
18. q<=(**others**=>'0');
19. **elsif**(**rising\_edge**(clk)) **then**
20. **aux:='1';**
21. l: **for** i **in** 0 **to** 25 **loop**
22. **if**(i>0) **then**
23. **for** j **in** 0 **to** i-1 **loop**
24. aux:=aux **and** q(j);
25. **end loop;**
26. **end** **if**;
27. *--aux:=aux and en;*
28. q(i)<=q(i) **xor** aux;
29. **end** **loop**;
30. **end if;**
31. **end** **process**;
32. **end** Behavioral;

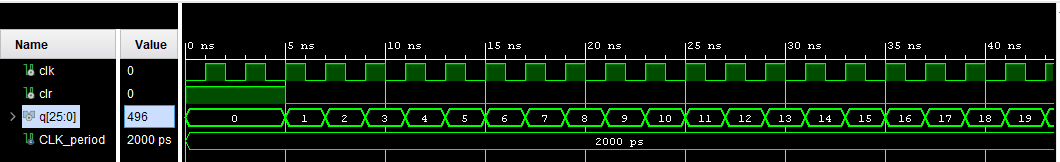
# Modificando para la Practica

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** divisorDeFrecuencia **is**
5. **Port (**
6. *--en,*
7. clk,clr:**in** std\_logic;
8. clkout:**out** std\_logic
9. );
10. **end divisorDeFrecuencia;**
12. **architecture** Behavioral **of** divisorDeFrecuencia **is**
13. **signal** q: std\_logic\_vector(25 **downto** 0);
14. **begin**
15. **clkout<=q(26);**
16. **process**(clk,clr)
17. **variable** aux:std\_logic;
18. **begin**
19. **if**(clr='1') **then**
20. **q<=(others=>'0');**
21. **elsif**(**rising\_edge**(clk)) **then**
22. aux:='1';
23. l: **for** i **in** 0 **to** 26 **loop**
24. **if**(i>0) **then**
25. **for j in 0 to i-1 loop**
26. aux:=aux **and** q(j);
27. **end** **loop**;
28. **end** **if**;
29. *--aux:=aux and en;*
30. **q(i)<=q(i) xor aux;**
31. **end** **loop**;
32. **end** **if**;
33. **end** **process**;
34. **end** Behavioral;

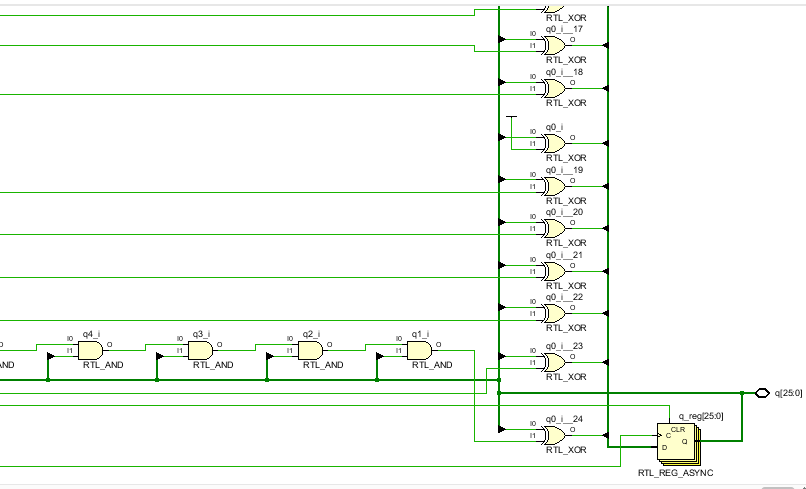
# Código de simulación

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **entity** divtb **is**
5. ***--  Port ( );***
6. **end** divtb;
8. **architecture** Behavioral **of** divtb **is**
9. **component** divisorDeFrecuencia **is**
10. **Port (**
11. *--en,*
12. clk,clr:**in** std\_logic;
13. q:**inout** std\_logic\_vector(25 **downto** 0)
14. );
15. **end component;**
16. **constant** CLK\_period : time := 2 ns;
17. **signal** clk,clr: std\_logic;
18. **signal** q: std\_logic\_vector(25 **downto** 0);
19. **begin**
21. CLK\_process :**process**
22. **begin**
23. CLK <= '0';
24. **wait** **for** CLK\_period/2;
25. **CLK <= '1';**
26. **wait** **for** CLK\_period/2;
27. **end** **process**;
29. df:divisorDeFrecuencia **port** **map**(
30. **clk=>clk,**
31. clr=>clr,
32. q=>q
33. );
35. **process**
36. **begin**
38. clr<='1';
39. **wait** **for** 5 ns;
40. **clr<='0';**
41. **wait**;
43. **end** **process**;
45. **end Behavioral;**

# Forma de onda



# Diagrama RTL



# Arquitectura completa con divisor

# Paquete

1. **library** IEEE;
2. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **package** packpr9 **is**
5. **component CASM2 is**
6. **Port** (
7. qa: **out** std\_logic\_vector(8 **downto** 0);
8. display:**out** std\_logic\_vector(6 **downto** 0);
9. D: **in**  std\_logic\_vector(8 **downto** 0);
10. **ini,clk,clr: in std\_logic**
11. );
12. **end** **component**;
14. **component** divisorDeFrecuencia **is**
15. **Port (**
16. *--en,*
17. clk,clr:**in** std\_logic;
18. clkout:**out** std\_logic
19. );
20. **end component;**
21. **end** **package**;

# Código de implementación

1. **library** IEEE;
2. **library** work;
3. **use** IEEE.STD\_LOGIC\_1164.**ALL**;
4. **use** work.packpr9.**all**;
6. **entity** practica9 **is**
7. **Port** ( ini : **in** STD\_LOGIC;
8. clk : **in** STD\_LOGIC;
9. clr : **in** STD\_LOGIC;
10. **D : in STD\_LOGIC\_VECTOR (8 downto 0);**
11. display: **out** STD\_LOGIC\_VECTOR (6 **downto** 0);
12. qa: **out** STD\_LOGIC\_VECTOR (8 **downto** 0)
13. );
14. **end** practica9;
16. **architecture** Behavioral **of** practica9 **is**
17. **signal** clkout: std\_logic;
18. **begin**
20. **div:divisorDeFrecuencia port map(**
21. clk=>clk,
22. clr=>clr,
23. clkout=>clkout
24. );
26. cas:CASM2 **port** **map**(
27. ini=>ini,
28. clk=>clkout,
29. clr=>clr,
30. **D=>D,**
31. qa=>qa,
32. display=>display
33. );
35. **end Behavioral;**

# Diagrama RTL

