Project VLSI 2: Knowles Adder 32-bit

Ομάδα 9
Γεώργιος Ταρνάρας - 1092751
Γεώργιος Σουκαράς - 1092660

ECE UPatras - 2025

1 Εισαγωγή

Σκοπός της εργασίας είναι η σχεδίαση και υλοποίηση ενός αθροιστή Knowles 32 bit. Ο αθροιστής θα υλοποιηθεί στη γλώσσα περιγραφής υλικού VHDL και στη συνέχεια θα γίνει η σύνθεση του κυκλώματος σε τεχνολογίες 45 και 7 nm με το εργαλείο σύνθεσης Genus. Επιπλέον, θα γίνει το placement και η διασύνδεση των standard cell του κυκλώματος με το εργαλείο Innovus. Τέλος, για την επαλήθευση της σωστής λειτουργίας του αθροιστή το κύκλωμα θα περάσει από τα flows του Xcelium και Logic Equivalence Checking (LEC).

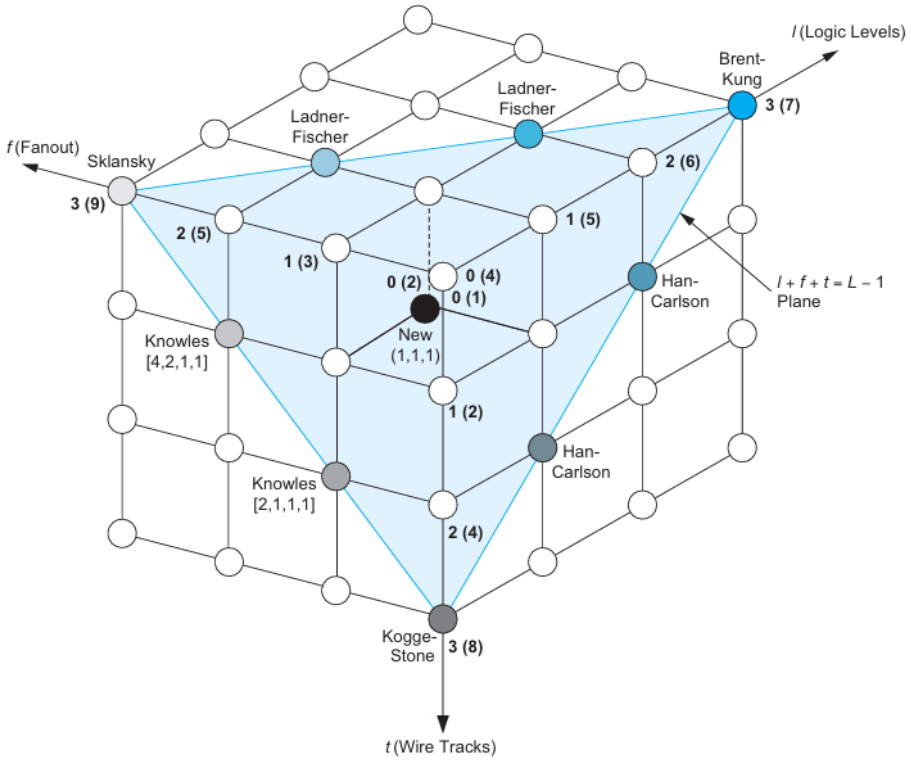
2 Αθροιστές δένδρου

Για τους αθροιστές μεγάλου εύρους (N > 16), η καθυστέρηση των αθροιστών πρόβλεψης (ή παράκαμψης, ή επιλογής) κρατουμένου κυριαρχείται από την καθυστέρηση που συνεπάγεται το πέρασμα του κρατουμένου από τα στάδια πρόβλεψης. Αυτή η καθυστέρηση μειώνεται με ένα δεύτερο στάδιο πρόβλεψης επί των μπλόκ που κάνουν την αρχική πρόβλεψη. Έτσι, μπορεί να κατασκευαστεί ένα πολυεπίπεδο δένδρο δομών πρόβλεψης και να επιτύχει λογαριθμική καθυστέρηση. Οι αθροιστές αυτού του είδους αναφέρονται στη βιβλιογραφία ως αθροιστές δένδρου ή αθροιστές παράλληλου προθέματος και ο Knowles είναι ένας από αυτούς. Γενικά, υπάρχουν πολλές διαφορετικές υλοποιήσεις του δένδρου και κάθε μία βασίζεται σε διαφορετικούς συμβιβασμούς μεταξύ του πλήθους επιπέδων λογικής, του πλήθους πυλών, του βαθμού οδήγησης και της ποσότητας αγωγών διασύνδεσης.

Κυριότερες Αρχιτεκτονικές
1960: J. Sklansky– conditional adder
1973: Kogge-Stone adder
1980: Ladner-Fisher adder
1982: Brent-Kung adder
1987: Han Carlson adder
1999: S. Knowles

Ο Knowles στο paper "A Family of Adders" πρότεινε μία οικογένεια δικτύων ανάμεσα στα δέντρα Kogge-Stone (πολλοί αγωγοί) και Sklanky (high fanout). Σε αυτή την εργασία θα υλοποιήσουμε δένδρο με βαθμούς οδήγησης εξόδων [2,1,1,1,1], το οποίο υποδιπλασιάζει

το πλήθος αγωγών στην τελική διαδρομή σε σχέση με ένα δένδρο Kogge Stone με κόστος το διπλασιασμό του φορτίου σε αυτούς τους αγωγούς.



Σχήμα 1: Ταξινόμηση των αθροιστών δικτύου προθέματος

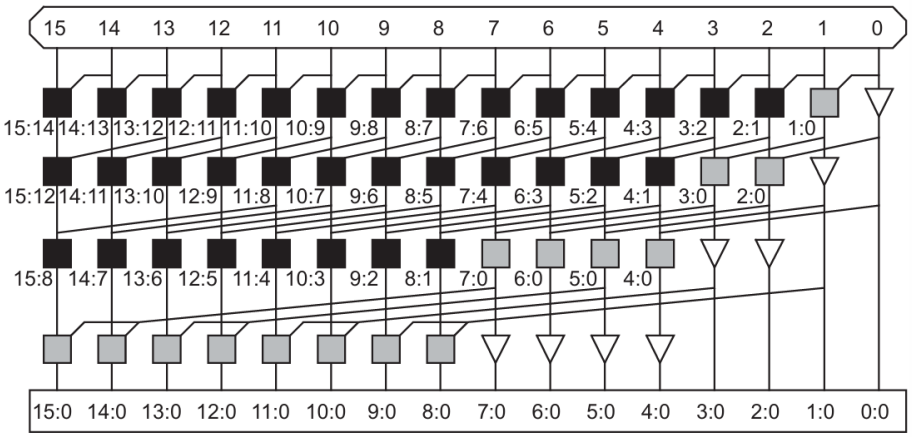
3 Δίκτυο PG

Για να υλοποίησουμε τον αθροιστή θα πρέπει να υπολογίσουμε τα σήματα διάδοσης και γέννησης κρατουμένου για τα οποία ισχύουν οι σχέσεις:

$$\begin{aligned} g_i &= A_i \cdot B_i \\ p_i &= A_i \oplus B_i \end{aligned} \tag{1}$$

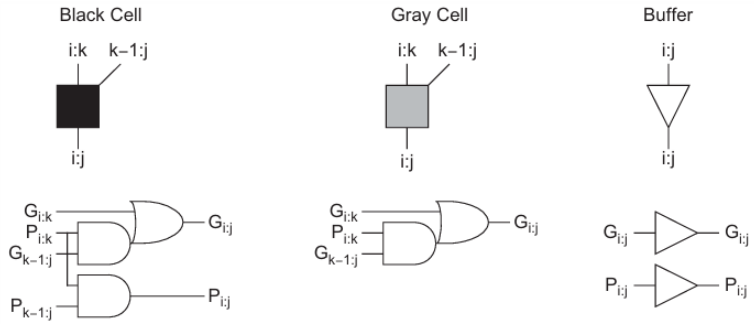
$$\begin{aligned} G_{i:j} &= G_{i:k} + P_{i:k} \cdot G_{k-1:j} \\ P_{i:j} &= P_{i:k} \cdot P_{k-1:j} \end{aligned} \tag{2}$$

Σε έναν Knowles adder αυτό επιτυγχάνεται με το παρακάτω δίκτυο:



Σχήμα 2: Δίκτυο [2,1,1,1] για 16 bit

Μπορούμε να επεκτήνουμε το παραπάνω δίκτυο στα 32 bit προσθέτοντας ένα επιπλέον στάδιο με βαθμό οδήγησης 1 και έτσι να κατασκευάσουμε μία δομή [2,1,1,1,1]. Τα κύτταρα που θα χρησιμοποιήσουμε εκτελούν τους παρακάτω υπολογισμούς.



Σχήμα 3: Μαύρο, γκρι κύτταρο, buffer

Τέλος το άθροισμα σε κάθε θέση υπολογίζεται με τη σχέση $S_i = P_i \oplus G_{i-1:0}$ και το κρατούμενο εξόδου $C_{out} = G_{N-1} + P_{N-1:0}C_{in}$.

4 Κύκλωμα υπολογισμού p, g - pg_calculation.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity pg_calculation is
5    generic ( m : integer := 32 );
6    port(
7      A, B : in  std_logic_vector( m-1 downto 0 );
8      P, G : out std_logic_vector( m-1 downto 0 )
9    );
10 end pg_calculation;
11
```

```
architecture my_arch of pg_calculation is
  begin
    generate_label:
    for i in 0 to m-1 generate
      P(i) <= A(i) XOR B(i);
      G(i) <= A(i) AND B(i);
    end generate;

end my_arch;
```

5 black_cell.vhd, grey_cell.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity black_cell is
  port(
    g1   : in std_logic; -- p_(k-1)_j
    p1   : in std_logic; -- g_(k-1)_j
    g0   : in std_logic; -- p_i_k
    p0   : in std_logic; -- g_i_k
    g2   : out std_logic; --g_i_j
    p2   : out std_logic  --p_i_j
  );
end black_cell;

architecture my_arch of black_cell is
  begin
    g2 <= g1 OR ( p1 AND g0 );
    p2 <= p1 AND p0;

end my_arch;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity gray_cell is
  port(
    g1   : in  std_logic; -- p_i_k
    p1   : in  std_logic; -- g_i_k
    g0   : in  std_logic; -- g_(k-1)_j
    g2   : out std_logic -- g_i_j
  );
end gray_cell;

architecture my_arch of gray_cell is
  begin
    g2 <= g1 OR ( p1 AND g0 );

```

```
end my_arch;
```

6 Παραμετρική υλοποίηση stage που οδηγεί 1 άλλη στήλη - knowles_stage.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity knowles_stage is
  generic(
    k  : integer := 32;
    gn : integer := 1 --gray cell number
  );
  port(
    P_in, G_in   : in  std_logic_vector( k-1 downto 0 );
    P_out, G_out : out std_logic_vector( k-1 downto 0 )
  );
end knowles_stage;

architecture my_arch of knowles_stage is

  component black_cell is
    port(
      g1   : in std_logic; -- p_(k-1)_j
      p1   : in std_logic; -- g_(k-1)_j
      g0   : in std_logic; -- p_i_k
      p0   : in std_logic; -- g_i_k
      g2   : out std_logic; --g_i_j
      p2   : out std_logic  --p_i_j
    );
  end component;

  component gray_cell is
    port(
      g1   : in  std_logic; -- p_i_k
      p1   : in  std_logic; -- g_i_k
      g0   : in  std_logic; -- g_(k-1)_j
      g2   : out std_logic -- g_i_j
    );
  end component;

  begin

    generate_buffer_label:
    for j in 0 to gn-1 generate
      P_out(j) <= P_in(j);
      G_out(j) <= G_in(j);
    end generate;
```

```
45
46      generate_gray_label:
47      for j in gn to 2*gn-1 generate
48        gray_j : gray_cell port map( G_in(j), P_in(j), G_in(j-gn), G_out(j) );
49        P_out(j) <= P_in(j);
50      end generate;
51
52      generate_black_label:
53      for j in 2*gn to k-1 generate
54        black_j : black_cell port map( G_in(j), P_in(j), G_in(j-gn), P_in(j-gn),
        G_out(j), P_out(j) );
55      end generate;
56
57  end my_arch;
```

Από το Σχήμα 2 φαίνεται ότι ο αριθμός των γκρί κυττάρων και απομονωτών αυξάνεται εκθετικά σε κάθε στάδιο. Έτσι, σχεδιάσαμε μια παραμετρική υλοποίηση για τα πρώτα 4 στάδια. Στο πρώτο στάδιο θα έχουμε 1 απομονωτή, 1 γκρί κύτταρο και 30 μαύρα. Στο δεύτερο θα έχουμε 2 απομονωτές, 2 γκρι κύτταρα και 28 μαύρα κτλ.

7 Δίκτυο PG - knowles_pg_32.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity knowles_pg_32 is
5    port(
6      P, G  : in  std_logic_vector( 31 downto 0 );
7      G_0   : out std_logic_vector( 31 downto 0 )
8    );
9  end knowles_pg_32;
10
11 architecture my_arch of knowles_pg_32 is
12
13   component knowles_stage is
14     generic(
15       k  : integer := 32;
16       gn : integer := 1 --gray cell number
17     );
18     port(
19       P_in, G_in   : in  std_logic_vector( k-1 downto 0 );
20       P_out, G_out : out std_logic_vector( k-1 downto 0 )
21     );
22   end component;
23
24   component gray_cell is
25     port(
26       g1   : in  std_logic; -- p_i_k
27       p1   : in  std_logic; -- g_i_k
28       g0   : in  std_logic; -- g_(k-1)_j
```

```
29      g2    : out std_logic -- g_i_j
30    );
31  end component;
32
33  signal P1, P2, P3, P4, G1, G2, G3, G4 : std_logic_vector( 31 downto 0 );
34
35  begin
36
37    -- stages
38    stage_0 : knowles_stage generic map(32, 1) port map( P, G, P1, G1 );
39    stage_1 : knowles_stage generic map(32, 2) port map( P1, G1, P2, G2 );
40    stage_2 : knowles_stage generic map(32, 4) port map( P2, G2, P3, G3 );
41    stage_3 : knowles_stage generic map(32, 8) port map( P3, G3, P4, G4 );
42
43    -- last stage buffers
44    generate_buffer_label:
45    for j in 0 to 15 generate
46      G_0(j) <= G4(j);
47    end generate;
48
49    -- last stage gray cells
50    generate_gray_label:
51    for j in 0 to 7 generate
52      gray_even : gray_cell port map( G4(16+2*j), P4(16+2*j), G4(2*j+1), G_0
    (16+ 2*j) );
53      gray_odd  : gray_cell port map( G4(16+2*j+1), P4(16+2*j+1), G4(2*j+1),
    G_0(16+ 2*j+1) );
54    end generate;
55
56  end my_arch;
```

Στα πρώτα 4 στάδια κάθε στήλη οδηγεί άλλη μία στήλη το πολύ. Στο τελευταίο στάδιο κάποιες στήλες οδηγούν άλλες δύο στήλες γιατί έχουμε δομή [2,1,1,1,1]. Ξεκινώντας από τις στήλες 16 και 17 τα γκρι κύτταρα τους οδηγούνται ανα δύο από την ίδια στήλη κατα αντιστοιχία με το Σχήμα 2.

8 Λογική υπολογισμού αθροίσματος και κρατούμενου εξόδου - sum_logic.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity sum_logic is
5    generic ( m : integer := 32 );
6    port(
7      P, C  : in  std_logic_vector( m-1 downto 0 );
8      G_last: in  std_logic;
9      S   : out std_logic_vector( m-1 downto 0 );
10     Cout  : out std_logic
11   );
```

```
end sum_logic;

architecture my_arch of sum_logic is
  begin
    generate_label:
    for i in 0 to m-1 generate
      S(i) <= C(i) XOR P(i);
    end generate;

    Cout <= ( P(m-1) AND C(m-1) ) OR G_last;

end my_arch;
```

9 Τελικό κύκλωμα αθροιστή - knowles_adder32.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity knowles_adder32 is
  port(
    A, B  : in  std_logic_vector( 31 downto 0 );
    Cin   : in std_logic;
    S   : out std_logic_vector( 31 downto 0 );
    Cout  : out std_logic
  );
end knowles_adder32;

architecture my_arch of knowles_adder32 is

  component pg_calculation is
    generic ( m : integer := 32 );
    port(
      A, B : in  std_logic_vector( m-1 downto 0 );
      P, G : out std_logic_vector( m-1 downto 0 )
    );
  end component;

  component knowles_pg_32 is
    port(
      P, G  : in  std_logic_vector( 31 downto 0 );
      G_0   : out std_logic_vector( 31 downto 0 )
    );
  end component;

  component sum_logic is
    generic ( m : integer := 32 );
    port(
      P, C  : in  std_logic_vector( m-1 downto 0 );
      G_last: in  std_logic;
```

```
      S   : out std_logic_vector( m-1 downto 0 );
      Cout  : out std_logic
    );
  end component;

  signal P_int, G_int : std_logic_vector( 32 downto 0 );
  signal G_0_int : std_logic_vector( 31 downto 0 );

  begin
    P_int(0) <= '0';
    G_int(0) <= Cin;
    pg_calc :  pg_calculation generic map(32) port map(A, B, P_int(32 downto 1), G_int(32 downto 1) );
    pg_logic : knowles_pg_32 port map( P_int(31 downto 0), G_int(31 downto 0), G_0_int );
    sum : sum_logic generic map(32) port map( G_0_int, P_int(32 downto 1), G_int(32), S, Cout );

end my_arch;
```

Αρχικά γίνεται ο υπολογισμός των σημάτων διάδοσης και γέννησης κρατουμένου με βάση τα δεδομένα εισόδου. Τα p_i, g_i περνούν μέσα από το PG δίκτυο και έτσι παράγονται τα $G_{i:0}$ σήματα. Τέλος, υπολογίζεται το άθροισμα. Για να περάσουμε το κύκλωμα από τα διάφορα flows προσθέτουμε καταχωρητές εισόδου/εξόδου.

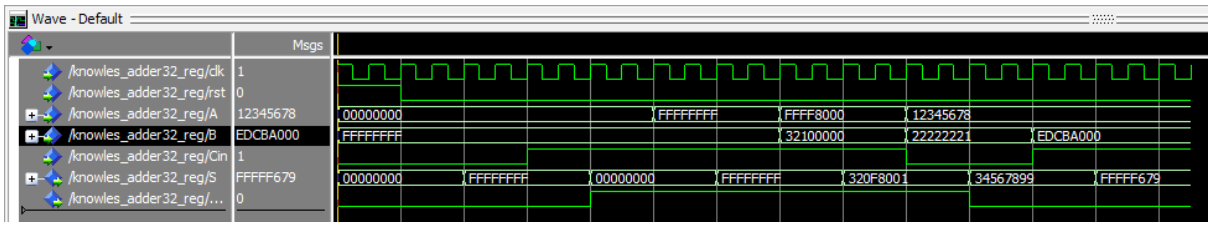
```
library ieee;
use ieee.std_logic_1164.all;

entity knowles_adder32_reg is
  port(
    A, B : in  std_logic_vector( 31 downto 0 );
    Cin  : in  std_logic;
    clk, rst    : in std_logic;
    S : out std_logic_vector( 31 downto 0 );
    Cout : out std_logic
  );
end knowles_adder32_reg;


architecture my_arch of knowles_adder32_reg is

  component QD
      generic (n:integer:=4);
      PORT(q:in std_logic_vector(n-1 downto 0);
           clk,rst:std_logic;
           d:out std_logic_vector(n-1 downto 0));
      end component;

  component knowles_adder32
    port(
      A, B  : in  std_logic_vector( 31 downto 0 );
      Cin   : in std_logic;
```

```
28        S    : out std_logic_vector( 31 downto 0 );
29        Cout  : out std_logic
30      );
31    end component;
32
33    signal A_Q, B_Q, S_Q: std_logic_vector(31 downto 0);
34    signal Cin_Q, Cout_Q, CoutVec, CinVec: std_logic_vector(1 downto 0);
35
36    begin
37      CinVec(1) <= Cin;
38      CinVec(0) <= '0';
39      Reg1: QD generic map(32) port map(A, clk, rst, A_Q);
40        Reg2: QD generic map(32) port map(B, clk, rst, B_Q);
41        Reg3: QD generic map(2) port map(CinVec, clk, rst, Cin_Q);
42        Knowles: knowles_adder32 port map(A_Q, B_Q, Cin_Q(1), S_Q, Cout_Q(1));
43        Cout_Q(0) <= '0';
44        Reg4: QD generic map(32) port map(S_Q, clk, rst, S);
45      Reg5: QD generic map(2) port map(Cout_Q, clk, rst, CoutVec);
46      Cout <= CoutVec(1);
47  end my_arch;
```

10 Προσομοίωση στο Modelsim

Η ορθή λειτουργία του αθροιστή επιβεβαιώθηκε με προσομοίωση στο modelsim. Οι τιμές των εισόδων και οι αντίστοιχες τιμές των εξόδων συνοψίζονται στον ακόλουθο πίνακα.

A	B	Cin	S	Cout
00000000	FFFFFFFF	0	FFFFFFFF	0
00000000	FFFFFFFF	1	00000000	1
FFFFFFFF	FFFFFFFF	1	FFFFFFFF	1
FFFF8000	32100000	1	320F8001	1
12345678	22222221	0	34567899	0
12345678	EDCBA000	0	FFFFF679	0



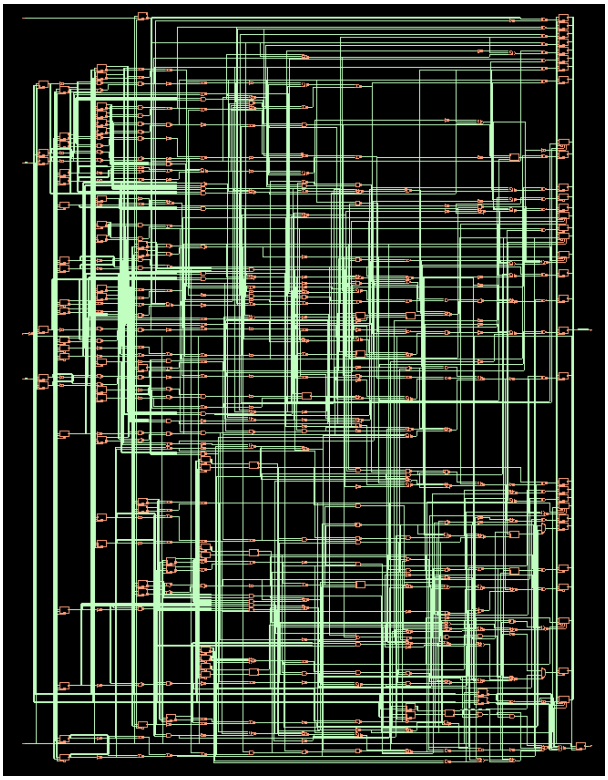
Σχήμα 4: modelsim simulation

11 Genus

11.1 Τεχνολογία 45nm

Synthesis

Το τελικό κύκλωμα που προκύπτει στο γραφικό περιβάλλον μετά την σύνθεση είναι το ακόλουθο:



Σχήμα 5: syn_opt 45nm

Area

Εκτελώντας την εντολή "report area", προκύπτει ότι η συγκεκριμένη υλοποίηση καταλαμβάνει συνολική επιφάνεια 2559 τ.μ .

```
@genus:root: 132> report area
=============================================================
  Generated by:           Genus(TM) Synthesis Solution 22.13-s093_1
  Generated on:           Jul 06 2025  01:43:00 pm
  Module:                 knowles_adder32_reg
  Technology libraries:   slow_1v0
                          pll 0.0
                          CDK_S128x16 0.0
                          CDK_S256x16 0.0
                          CDK_R512x16 0.0
                          physical_cells
  Operating conditions:   slow
  Interconnect mode:      global
  Area mode:              physical library
=============================================================

      Instance      Module  Cell Count  Cell Area  Net Area   Total Area
-------------------------------------------------------------------------
knowles_adder32_reg                507   1978.624   580.573     2559.196
```

Σχήμα 6: report area 45nm

Power

Εκτελώντας την εντολή "report power", προκύπτει ότι η συγκεκριμένη υλοποίηση καταναλώνει συνολική ισχύ $1.42mW$.

```
Instance: /knowles_adder32_reg
Power Unit: W
PDB Frames: /stim#0/frame#0
-------------------------------------------------------------------------------
   Category         Leakage     Internal    Switching        Total    Row%
-------------------------------------------------------------------------------
     memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
   register     4.01815e-08  7.94368e-04  8.77484e-05  8.82157e-04  62.13%
      latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
      logic     7.83545e-08  2.68733e-04  2.38349e-04  5.07160e-04  35.72%
       bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
      clock     0.00000e+00  0.00000e+00  3.05640e-05  3.05640e-05   2.15%
        pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
         pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
-------------------------------------------------------------------------------
   Subtotal     1.18536e-07  1.06310e-03  3.56661e-04  1.41988e-03 100.00%
 Percentage           0.01%       74.87%       25.12%      100.00% 100.00%
-------------------------------------------------------------------------------
```

Σχήμα 7: report power 45nm

Timing

Εκτελώντας την εντολή "report timing", έχοντας θέσει την περίοδο $T = 1.5ns$, προκύπτει μηδενικό slack, επομένως αξιοποιείται ολόκληρη η περίοδος για την άθροιση χωρίς να σπαταλάται χρόνος.

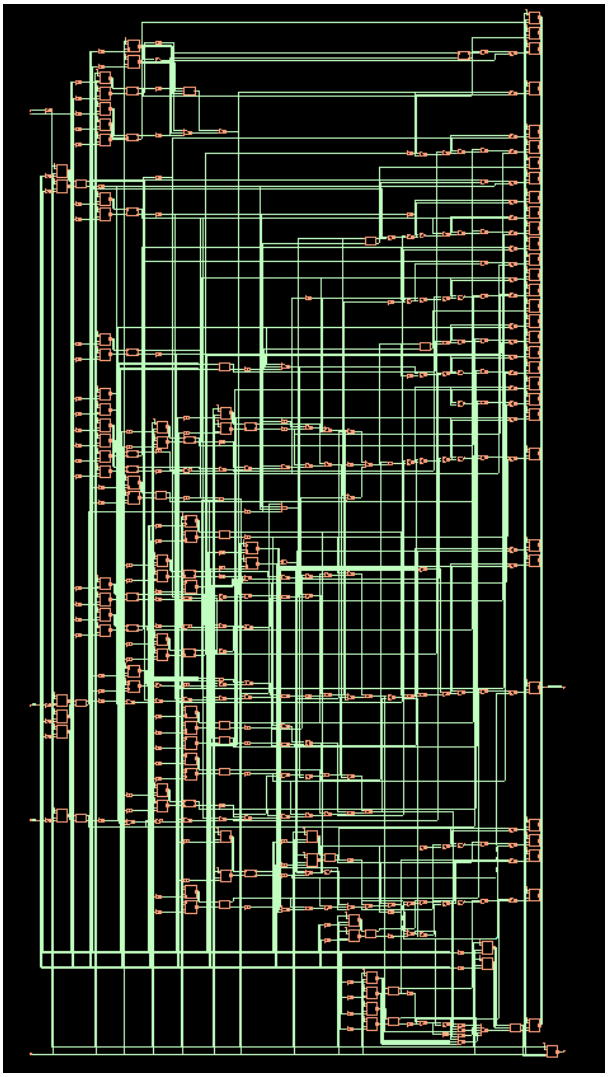
```
                  Capture       Launch
      Clock Edge:+    1500            0
      Src Latency:+      0            0
      Net Latency:+      0 (I)        0 (I)
          Arrival:=   1500            0

            Setup:-    263
     Required Time:=  1237
      Launch Clock:-     0
         Data Path:-  1237
             Slack:=     0

#-------------------------------------------------------------------------------------------------------
#   Timing Point    Flags    Arc   Edge   Cell      Fanout Load Trans Delay Arrival Instance
#                                                     (fF)  (ps)  (ps)   (ps)  Location
#-------------------------------------------------------------------------------------------------------
  Reg1_d_reg[0]/CK  -        -       R     (arrival)     98    -     0     0       0    (-,-)
  Reg1_d_reg[0]/QN  -        CK->QN  F     DFFRX4         1  8.5    56   355     355    (-,-)
  g7457__3680/Y     -        B->Y    R     NOR2X6         1  8.5    70    54     409    (-,-)
  g7405__1666/Y     -        A->Y    F     NOR2X6         2  9.5    49    53     462    (-,-)
  g7233__5122/Y     -        A1->Y   R     OAI21X4        2 12.9   141    90     552    (-,-)
  g7221/Y           -        A->Y    F     CLKINVX6       3 16.8    76    82     634    (-,-)
  g7209__6161/Y     -        B->Y    R     NOR2X6         1  8.4    71    60     694    (-,-)
  g7175__7482/Y     -        B->Y    F     NOR2X6         3 12.3    61    54     748    (-,-)
  g7169__5115/Y     -        B->Y    R     NOR2X4         1  8.4    90    66     814    (-,-)
  g7156__3680/Y     -        B->Y    F     NOR2X6         4 14.0    80    63     877    (-,-)
  g7135__1881/Y     -        A1->Y   F     OA21X4         3 10.9    82   170    1047    (-,-)
  g7109__2346/Y     -        A2->Y   R     OAI31X1        1  5.4   309   190    1237    (-,-)
  Reg4_d_reg[16]/SE <<<       -       R     SDFFRHQX1      1    -     -     0    1237    (-,-)
#-------------------------------------------------------------------------------------------------------
```

Σχήμα 8: report timing 45nm

11.2 Τεχνολογία 7nm

Synthesis

Το τελικό κύκλωμα που προκύπτει στο γραφικό περιβάλλον μετά την σύνθεση είναι το ακόλουθο:



Σχήμα 9: syn_opt 7nm

Area

Εκτελώντας την εντολή "report area", προκύπτει ότι η συγκεκριμένη υλοποίηση καταλαμβάνει συνολική επιφάνεια 58 τ.μ .

```
@genus:root: 132> report area
====================================================
  Generated by:           Genus(TM) Synthesis Solution 22.13-s093_1
  Generated on:           Jul 06 2025  01:59:34 pm
  Module:                 knowles_adder32_reg
  Technology libraries:   asap7sc7p5t_SIMPLE_RVT_TT_ccs_211120 1.0
                          asap7sc7p5t_INVBUF_RVT_SS_ccs_211120 1.0
                          asap7sc7p5t_SEQ_RVT_TT_ccs_220123 1.0
                          asap7sc7p5t_SIMPLE_RVT_TT_ccs_211120 1.0
                          asap7sc7p5t_INVBUF_RVT_SS_ccs_211120 1.0
                          asap7sc7p5t_SEQ_RVT_TT_ccs_220123 1.0
  Operating conditions:   PVT_0P7V_25C (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================

      Instance       Module  Cell Count  Cell Area  Net Area   Total Area   Wireload
----------------------------------------------------------------------------------------
knowles_adder32_reg                393     58.189     0.000       58.189 <none> (D)
```

Σχήμα 10: report area 7nm

Power

Εκτελώντας την εντολή "report power", προκύπτει ότι η συγκεκριμένη υλοποίηση καταναλώνει συνολική ισχύ $0.53mW$.

```
Instance: /knowles_adder32_reg
Power Unit: W
PDB Frames: /stim#0/frame#0
-------------------------------------------------------------------------------------
   Category         Leakage     Internal    Switching        Total     Row%
-------------------------------------------------------------------------------------
     memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
   register     2.98828e-08  3.90907e-04  8.75875e-06  3.99695e-04   75.40%
      latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
      logic     7.94963e-08  3.03250e-05  4.00043e-05  7.04088e-05   13.28%
       bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
      clock     0.00000e+00  0.00000e+00  6.00250e-05  6.00250e-05   11.32%
        pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
         pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00    0.00%
-------------------------------------------------------------------------------------
   Subtotal     1.09379e-07  4.21232e-04  1.08788e-04  5.30129e-04  100.00%
 Percentage          0.02%       79.46%       20.52%      100.00%  100.00%
-------------------------------------------------------------------------------------
```

Σχήμα 11: report power 7nm

Timing

Εκτελώντας την εντολή "report timing", έχοντας θέσει την περίοδο $T = 0.4ns$, προκύπτει $slack = 2ps$, επομένως αμελητέο μέρος της περιόδου δεν αξιοποιείται από το κύκλωμα.

```
      Clock Edge:+     400            0
       Src Latency:+     0            0
       Net Latency:+     0 (I)        0 (I)
           Arrival:=   400            0

             Setup:-    25
     Required Time:=   375
      Launch Clock:-     0
         Data Path:-   374
             Slack:=     2

#-------------------------------------------------------------------------------------------------------
#   Timing Point    Flags    Arc    Edge          Cell            Fanout Load Trans Delay Arrival Instance
#                                                                  (fF)  (ps)  (ps)   (ps)  Location
#-------------------------------------------------------------------------------------------------------
  Reg2_d_reg[0]/CLK -        -      R     (arrival)                 98     -     0     0       0    (-,-)
  Reg2_d_reg[0]/QN  -        CLK->QN F    DFFASRHQNx1_ASAP7_75t_R    1   2.0    32    60      60    (-,-)
  g5957/CON         -        B->CON  R    FAx1_ASAP7_75t_R           1   0.4    33    26      86    (-,-)
  g3/Y              -        A->Y    F    INVxp67_ASAP7_75t_R        2   1.4    28    26     112    (-,-)
  g5621__6417/Y     -        C->Y    R    NAND3xp33_ASAP7_75t_R      1   0.5    19    18     130    (-,-)
  g5603__8246/Y     -        B->Y    F    NAND2xp5_ASAP7_75t_R       3   2.0    29    21     150    (-,-)
  g5593__5477/Y     -        A->Y    R    NAND2xp5_ASAP7_75t_R       1   0.5    17    16     167    (-,-)
  g5587__9315/Y     -        A->Y    F    NAND2xp5_ASAP7_75t_R       3   1.8    27    18     185    (-,-)
  g5574__5122/Y     -        B->Y    R    NAND2xp33_ASAP7_75t_R      1   0.5    22    20     204    (-,-)
  g5566__7410/Y     -        A->Y    F    NAND2xp5_ASAP7_75t_R       2   0.9    17    14     218    (-,-)
  g5552__9315/Y     -        A->Y    R    NAND3xp33_ASAP7_75t_R      1   0.5    19    13     231    (-,-)
  g5542__2802/Y     -        B->Y    F    NAND2xp5_ASAP7_75t_R       2   0.8    16    14     245    (-,-)
  g5519__7410/Y     -        A->Y    R    NOR2xp33_ASAP7_75t_R       3   1.1    34    22     267    (-,-)
  g5508__7098/Y     -        B->Y    F    NAND2xp5_ASAP7_75t_R       2   1.5    27    22     289    (-,-)
  g5492__5107/Y     -        B->Y    R    NAND2xp5_ASAP7_75t_R       1   0.5    18    14     303    (-,-)
  g5483__1881/Y     -        B->Y    F    NAND2xp5_ASAP7_75t_R       2   1.3    21    17     320    (-,-)
  g5464__6783/Y     -        B->Y    R    NAND2xp33_ASAP7_75t_R      1   0.5    35    18     338    (-,-)
  g5463__4319/Y     -        B->Y    F    NAND2xp5_ASAP7_75t_R       1   1.0    22    18     356    (-,-)
  g5449__2398/Y     -        A->Y    R    XNOR2xp5_ASAP7_75t_R       1   0.6    32    17     374    (-,-)
  Reg4_d_reg[26]/D  <<<      -       R    DFFASRHQNx1_ASAP7_75t_R    1    -     -     0      374    (-,-)
#-------------------------------------------------------------------------------------------------------
```

Σχήμα 12: report timing 7nm

12 LEC

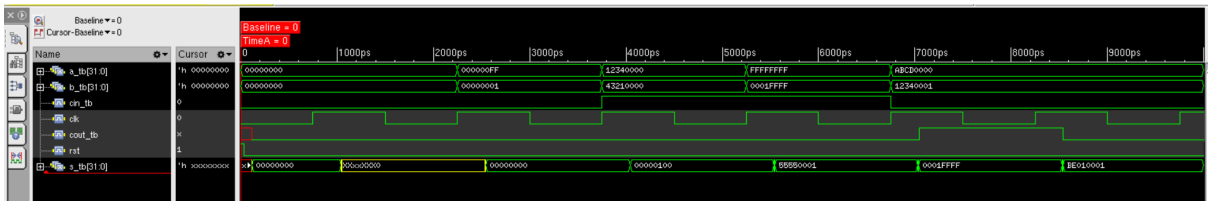
Αυτό σημαίνει ότι το RTL design στη VHDL και το gate level design που παράγει το Genus έχουν την ίδια συμπεριφορά.

```
LEC> report verification
================================================================================
                          Verification Report
--------------------------------------------------------------------------------
Category                                                                  Count
--------------------------------------------------------------------------------
1. Non-standard modeling options used:                                        0
--------------------------------------------------------------------------------
2. Incomplete verification:                                                   0
--------------------------------------------------------------------------------
3. User modification to design:                                               0
--------------------------------------------------------------------------------
4. Conformal Constraint Designer clock domain crossing checks recommended:   0
--------------------------------------------------------------------------------
5. Design ambiguity:                                                          0
--------------------------------------------------------------------------------
6. Compare Results:                                                        PASS
================================================================================
```

Σχήμα 13: Logic Equivalence Checking report

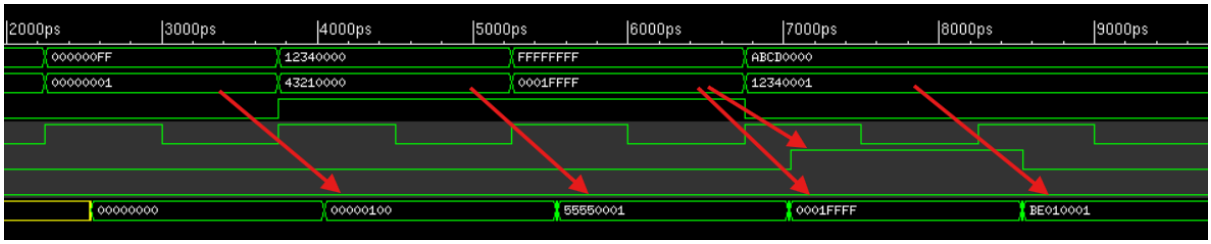
13 Xcelium

Φτιάξαμε το testbench σε verilog και χρησιμοποιήσαμε το xcelium για να προσομοιώσουμε το κύκλωμα που παράγει το genus (σε gate level). Για το clock χρησιμοποιήσαμε 50 % duty cycle με περίοδο 1.5ns όπως έγινε στη σύνθεση στα 45nm.



Σχήμα 14: Waveform window

A	B	Cin	S	Cout
000000FF	00000001	0	00000100	0
12340000	43210000	1	55550001	0
FFFFFFFF	0001FFFF	1	0001FFFF	1
ABCD0000	12340001	0	BE010001	0

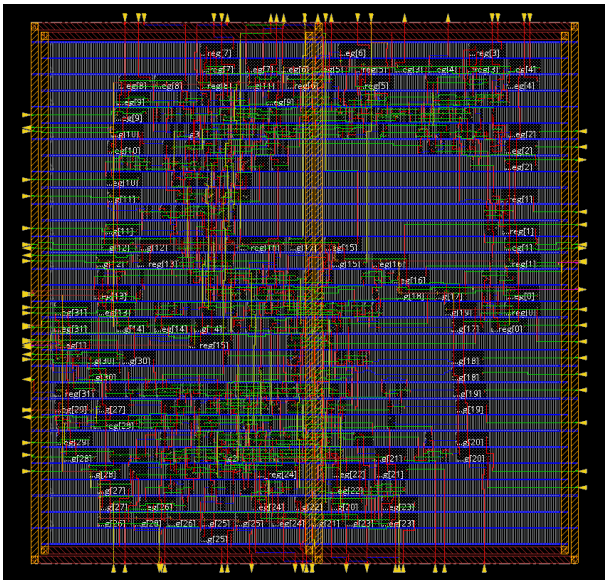


Σχήμα 15: Είσοδοι-Έξοδοι

Ο αθροιστής δίνει σωστά αποτελέσματα και έτσι επιβεβαιώνεται η σωστή λειτουργία του. Ωστόσο, η έξοδος δεν εμφανίζεται στην ακμή του ρολογιού αλλά καθυστερεί λίγο κάτι το οποίο δεν θα έπρεπε να γίνεται. Αυτό πιθανότατα οφείλεται σε κάποιο λάθος μας κατά τη ρύθμιση του xcelium.

14 Innovus

Το τελικό κύκλωμα, με χρήση τεχνολογίας 45 nm, στο innovus μετά το post route optimization και της προσθήκης των filler cells είναι το ακόλουθο:



Σχήμα 16: Υλοποίηση στο innovus με τεχνολογία 45nm

Timing

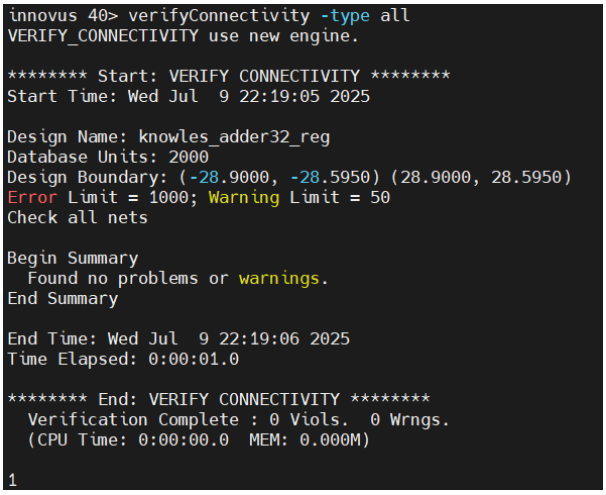
Εκτελώντας την εντολή "report_timing":

```
Path 1: VIOLATED Setup Check with Pin Reg4_d_reg[26]/CK
Endpoint:   Reg4_d_reg[26]/D (^) checked with  leading edge of 'clk'
Beginpoint: Reg2_d_reg[0]/Q  (^) triggered by  leading edge of 'clk'
Path Groups: {clk}
Analysis View: worst_case
Other End Arrival Time          -0.001
- Setup                          0.142
+ Phase Shift                    1.500
= Required Time                  1.357
- Arrival Time                   1.398
= Slack Time                    -0.041
     Clock Rise Edge                     0.000
     + Clock Network Latency (Prop)     -0.000
     = Beginpoint Arrival Time          -0.000
     +-------------------------------------------------------------------------+
     |    Instance    |     Arc      |   Cell    | Delay | Arrival | Required |
     |                |              |           |       |  Time   |   Time   |
     |----------------+--------------+-----------+-------+---------+----------|
     | Reg2_d_reg[0]  | CK ^         |           |       |  -0.000 |   -0.041 |
     | Reg2_d_reg[0]  | CK ^ -> Q ^  | DFFRX1    | 0.319 |   0.318 |    0.278 |
     | g7457__3680    | A ^ -> Y v   | NOR2X1    | 0.053 |   0.371 |    0.330 |
     | g7405__1666    | A v -> Y ^   | NOR2X1    | 0.070 |   0.441 |    0.400 |
     | g7233__5122    | A1 ^ -> Y v  | OAI21X1   | 0.084 |   0.525 |    0.484 |
     | FE_OFC8_n_201  | A v -> Y ^   | INVX1     | 0.078 |   0.603 |    0.562 |
     | g7209__6161    | B ^ -> Y v   | NOR2X1    | 0.054 |   0.657 |    0.616 |
     | g7175__7482    | B v -> Y ^   | NOR2X1    | 0.084 |   0.742 |    0.701 |
     | g7169__5115    | B ^ -> Y v   | NOR2X1    | 0.080 |   0.822 |    0.781 |
     | g7156__3680    | B v -> Y ^   | NOR2X2    | 0.084 |   0.906 |    0.865 |
     | g7149__6161    | B ^ -> Y ^   | OR2X1     | 0.109 |   1.015 |    0.974 |
     | g7136__6260    | A ^ -> Y v   | NAND2X2   | 0.107 |   1.122 |    1.081 |
     | g7111__6131    | A1N v -> Y v | OAI2BB1X1 | 0.160 |   1.282 |    1.241 |
     | g7092__6260    | S0 v -> Y ^  | MXI2XL    | 0.116 |   1.398 |    1.357 |
     | Reg4_d_reg[26] | D ^          | DFFRHQX1  | 0.000 |   1.398 |    1.357 |
     +-------------------------------------------------------------------------+
```

Σχήμα 17: report_timing

Geometry check

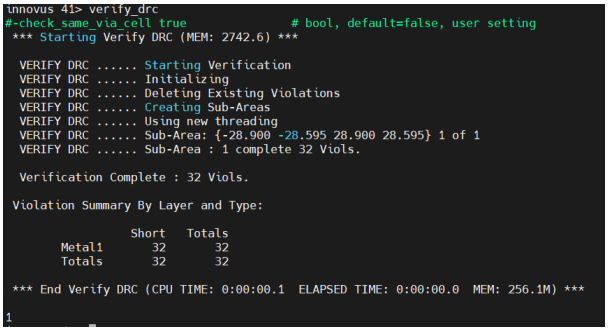
Εκτελώντας την εντολή "verifyConnectivity -type all " βλέπουμε ότι δεν γίνεται καμία παραβίαση:



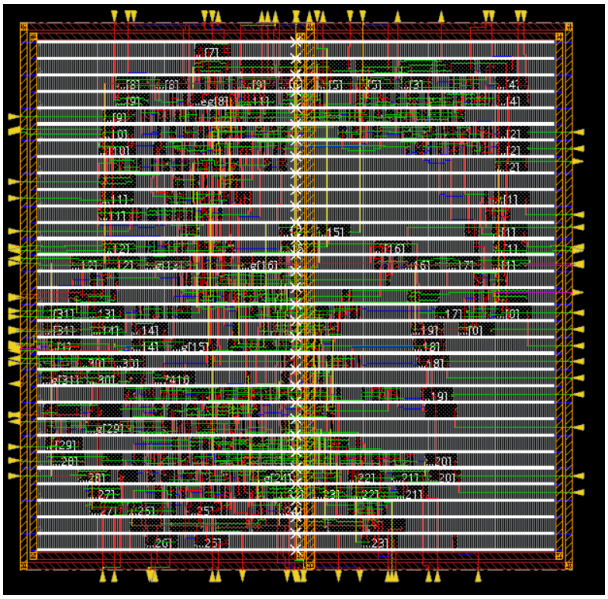
Σχήμα 18: verifyConnectivity -type all

DRC

Εκτελώντας τις εντολές "verify_drc", "highlight_drc" βλέπουμε 32 παραβίασεις πάνω στα stripes στη μέση του κυκλώματος.Αυτό μπορεί να γίνεται επειδή τα μέταλλα είναι πολύ κοντά μεταξύ τους ή επειδή θα έπρεπε να είναι πίο πλατιά ή και για κάποιο άλλο λόγο:



Σχήμα 19: verify_drc



Σχήμα 20: highlight_drc

Αναφορές

[1] Neil Weste, David Harris *Σχεδίαση ολοκληρωμένων κυκλωμάτων CMOS VLSI, 4th edition.*

[2] Simon Knowles *A Family of adders, IEEE, 2001.*

[3] Kostas Vitoroulis, 2006. Presented to Dr. A. J. Al-Khalili.Concordia University. *Parallel prefix adders.*