МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ІКНІ
Кафедра **ПЗ**

ЗВІТ

до лабораторної роботи № 1
на тему: *"*Синтез та моделювання основних типів комбінаційних схем в системі Proteus*"*
з дисципліни: *"Архітектура комп'ютера та комп'ютерних мереж"*

Лектор:
доц. каф. ПЗ
Крук О.Г.

Виконала:
ст. гр. ПЗ-26
Пелих О. Р.

Прийняв:
доц. каф. ПЗ
Крук О.Г.

« ____ » _____ 2024 р.

Львів – 2024

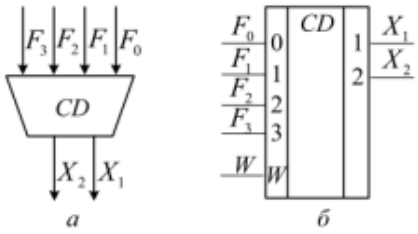
Тема роботи: синтез та моделювання основних типів комбінаційних схем в системі Proteus

Мета роботи: набути практичних навиків моделювання логічних схем в середовищі системи програм Proteus; поглибити знання про основні типи комбінаційних схем: шифратори, дешифратори, мультиплексори і демультиплексори; опанувати їх синтез; дослідити роботу синтезованих схем в системі програм Proteus.

Теоретичні відомості

Шифратор - це цифровий пристрій, призначений для перетворення вхідного m-розрядного унітарного коду у вихідний n-розрядний двійковий позиційний код. Унітарний код – це двійковий код, що має лише одну одиницю, а решта значень – нулі.

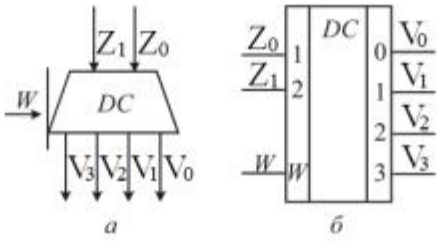
Шифратор позначається так:



Умовні графічні позначення шифратора:
а - на функціональних схемах; б - на електричних принципових схемах

Дешифратор - це цифровий пристрій, призначений для перетворення вхідного n-розрядного двійкового позиційного коду у вихідний m-розрядний унітарний код.

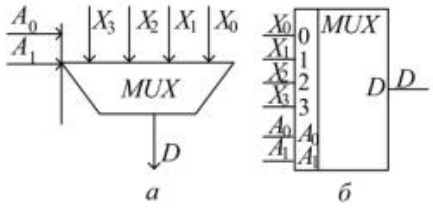
Дешифратор позначається так:



Умовні графічні позначення дешифратора:
а – на функціональних схемах; б – на електричних принципових схемах

Мультиплексор - це комбінаційний цифровий пристрій, призначений для комутування (перемикання) логічних сигналів від одного з n інформаційних X-входів на єдиний D-вихід. Номер конкретного інформаційного входу, який

повинен під’єднуватися до виходу в певний момент часу, вказується за допомогою адресних А-входів.



Умовні графічні позначення мультиплексора "4 в 1":
а – на функціональних схемах; б – на електричних принципових схемах

Індивідуальне завдання

Індивідуальний варіант (ПЗ-26, 17 варіант)

17		0	d_1	0	d_2	d_3	d_4	0	d_0	131	$F_3, F_7, F_2, F_4, F_1, F_6, F_5$

Хід роботи

1. Створіть в програмі Proteus Professional версії не нижче 8.9 новий проєкт. Для цього запустіть програму Proteus Professional і клацніть на кнопці New Project. У вікні New Project Wizard: Start в поле Name введіть ім’я проєкту, до прикладу LR_1_a, далі клацніть на кнопці Browse і виберіть або створіть папку 1_AK, в якій буде збережено проєкт (шлях до папки відобразиться в полі Path) і клацніть на кнопці Next. Імена проєкту і всіх папок в шляху набирайте латинськими літерами. Якщо появиться вікно Project ALready Exists на запитання Replace existing proect? виберіть Так. В наступному вікні New Project Wizard: Schematic Design виберіть Create a schematic from the selected template, виокремте рядок DEFAULT і клацніть на кнопці Next. В новому вікні New Project Wizard: PCB Layout виберіть Do not create a PCB Layout і клацніть на кнопці Next. Далі у вікні New Project Wizard: Firmware виберіть No Firmware Projecti клацніть на кнопці Next. Нарешті у вікні New Project Wizard: Summary клацніть на кнопці Finish.
2. Синтезуйте і введіть в систему програм Proteus схему пріоритетного шифратора 8 × 3 відповідно до варіанту індивідуального завдання.

2.1. Складіть за аналогією до (1.2) вирази для проміжних змінних відповідно до заданого в індивідуальному завданні пріоритету вхідних сигналів Fi, i = 1,…7.

Пріоритет: F1, F6, F5, F3, F7, F2, F4

H3 = F3

H7 = ¬F3∧F7

H2 = ¬F3∧¬F7∧F2

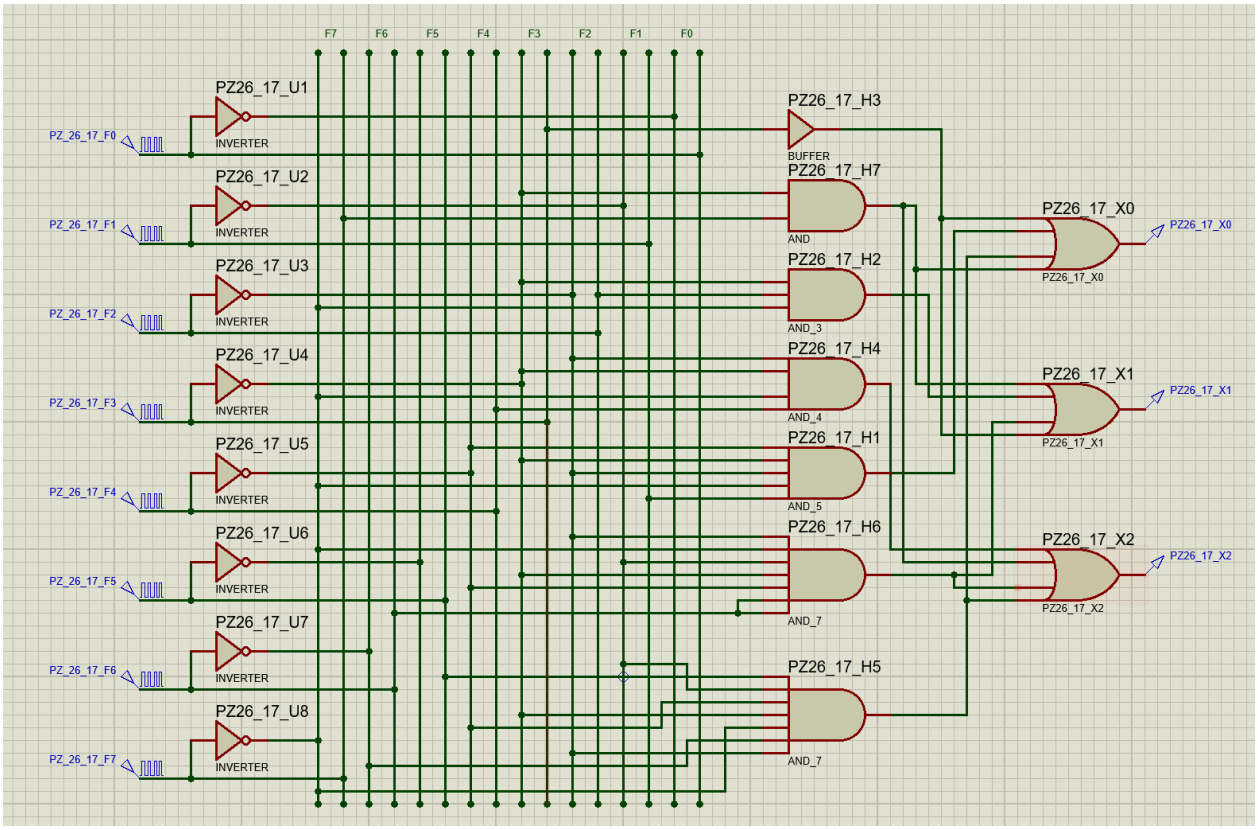
H4 = ¬F3∧¬F7∧¬F2∧F4

H1 = ¬F3∧¬F7∧¬F2∧¬F4∧F1
H6 = ¬F3∧¬F7∧¬F2∧¬F4∧¬F1∧F6
H5 = ¬F3∧¬F7∧¬F2∧¬F4∧¬F1∧¬F6∧F5

2.2. Виразіть вихідні сигнали пріоритетного шифратора x0, x1, x2 через проміжні змінні (рівняння (1.3)).

X0 = H1 ∨ H3 ∨ H5 ∨ H7,
X1 = H2 ∨ H3 ∨ H6 ∨ H7,
X2 = H4 ∨ H5 ∨ H6 ∨ H7.

2.3. Використовуючи отримані рівняння, створіть схему пріоритетного шифратора в робочій області системи Proteus, за основу візьміть схему, наведену на рис. 1.18. В іменах всіх елементів схеми та генераторів повинен бути ваш ідентифікатор PZ27_13_, якщо ваша група ПЗ-27, а номер в списку групи – 13.



3. Задайте прямі та інверсні значення вхідних сигналів F0…F7.

3.1. Генератори вхідних сигналів F0…F7 розташуйте зліва вертикально в порядку зростання (як на рис. 1.18).

3.2. Розрахуйте період цифрового сигналу T =1/f (значення частоти f переведіть в Гц, результат заокругліть до трьох значущих цифр). Визначіть

ширину (тривалість) елементарного імпульса τ = T/8, результат заокругліть до трьох значущих цифр.

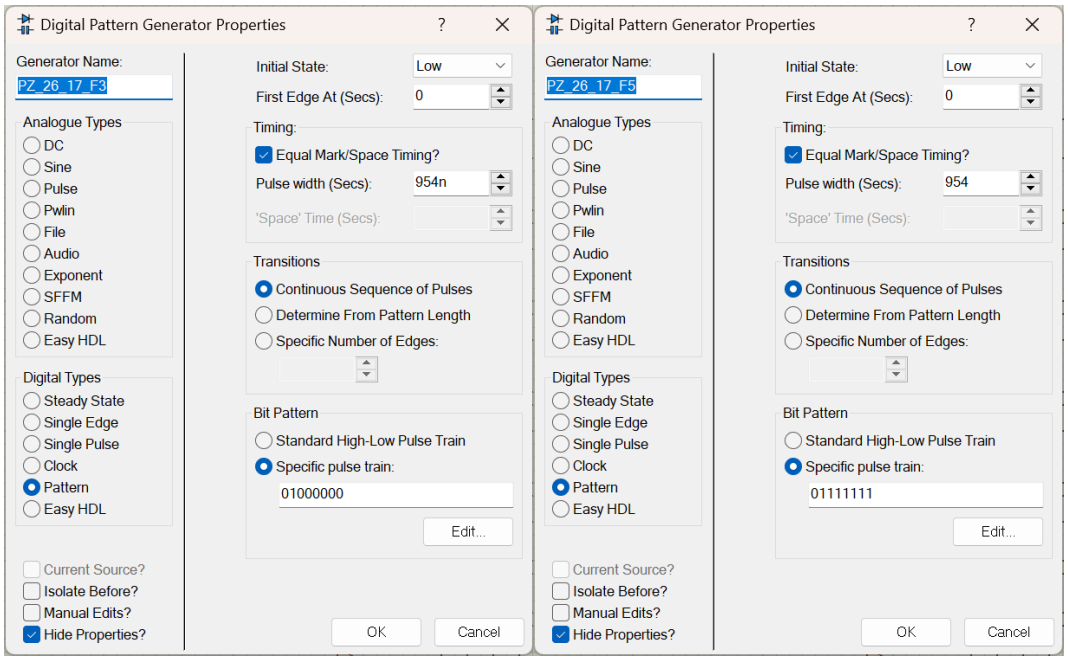
- період цифрового сигналу T = 1/f = 1/131000 = 0.00000763с;
- ширина елементарного імпульса t = T/8 = 0.00000763/8 = 0.000000953с

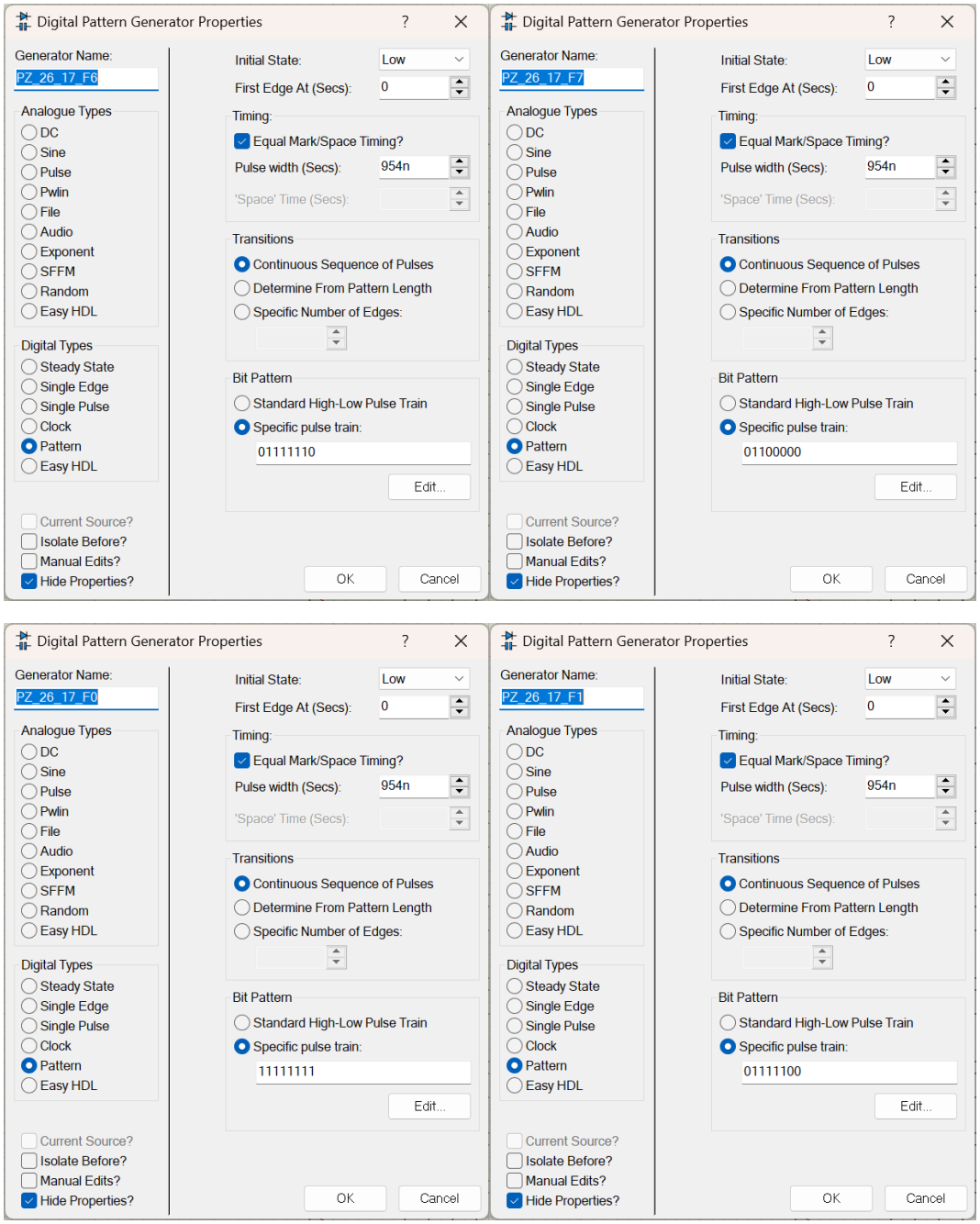
3.3. Сигнал F0 задайте за допомогою цифрового генератора (рис. 1.26 а). В полі Pulse width необхідно ввести значення τ.

3.4. Тепер задайте сигнал на вході, який має найвищий пріоритет (стоїть на першому місці в заданій послідовності) (рис. 1.26 б). Зверніть увагу на значення в полі Specific pulse train.

3.5. Наступним задайте сигнал на вході, який є другим в заданій послідовності, в поле Specific pulse train введіть значення 01100000.

3.6. Послідовно задавайте сигнали на всіх решта входах відповідно до їх пріоритетів. Для входу з найнижчим пріоритетом в полі Specific pulse train має бути значення 01111111.





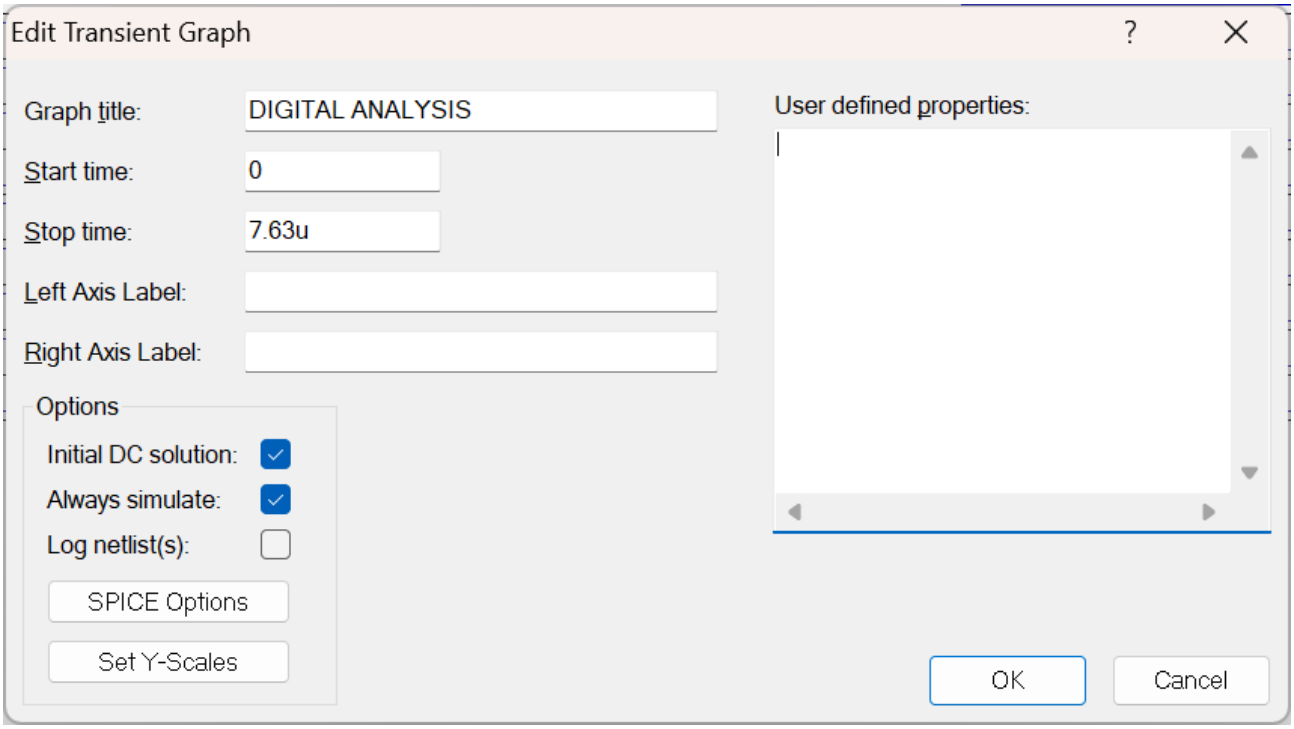
4. Побудуйте цифровий графік на часовому проміжку 0…T, на якому виведіть вхідні та вихідні сигнали в такій послідовності: F0…F7, X0…X2.

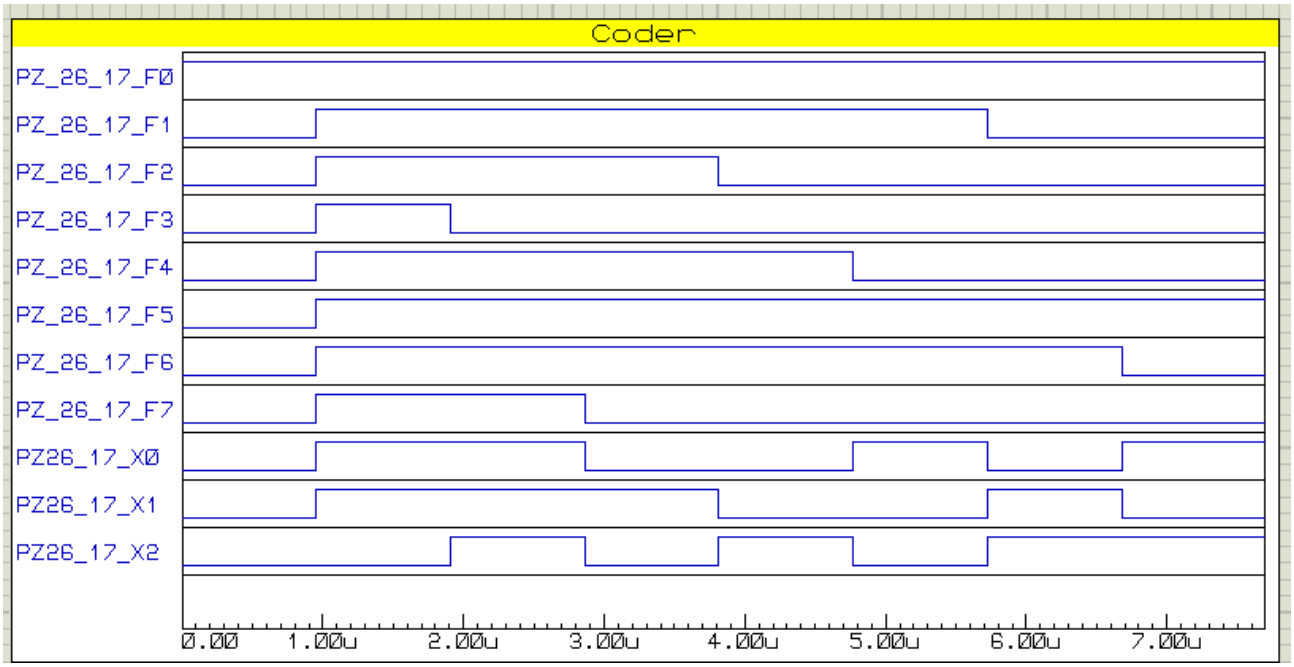
4.1. Для побудови графіка вкажіть прямокутник довжиною не менше 1/3 ширини робочої області.

4.2. В заголовку графіка вкажіть задану послідовність за спаданням пріоритету. Задайте на графіку часовий проміжок 0…T.

4.3. Налаштуйте графік: в рядку головного меню системи Proteus клацніть на елементі Template, виберіть в спадному меню рядок Set Graph & Trace Colours, у вікні Graph Colour Configuration вкажіть наступні параметри. Рис. 1.27. Налаштування параметрів графіка

4.4. Побудуйте цифровий графік – виконайте команду Simulate Graph з контекстного меню.





5. З отриманих часових діаграм запишіть послідовно вихідний двійковий код x2x1x0 на кожному з восьми проміжків: kτ…(k+1)τ, k = 0, …7.

Двійковий код	Сигнали, які набувають значення одиниці на даному проміжку
000	F0
110	F0,F1,F2,F3,F4,F5,F6,F7
111	F0,F1,F2,F4,F5,F6,F7
010	F0,F1,F2,F4,F5,F6
001	F0,F1,F4,F5,F6
100	F0,F1,F5,F6
011	F0,F5,F6
101	F0,F5

7. Створіть новий проєкт LR_1_b в цій же папці 1_AK.
8. Синтезуйте і введіть в систему програм Proteus схему лінійного дешифратора 3 × 8 відповідно до варіанту індивідуального завдання в табл. 1.6 (прийміть: z0 ≡ a0, z1 ≡ a1, z2 ≡ a2; v0 ≡ d0, v1 ≡ d1, v2 ≡ d2, v3 ≡ d3, v4 ≡ d4).

8.1. За аналогією до (1.4) складіть рівняння для кожного з виходів дешифратора V0…V7 на підставі таблиці істинності з входами z2, z1, z0, заданої в індивідуальному завданні.

V0=Z2 ∧ Z1 ∧ Z0
V1= ¬ Z2 ∧ ¬ Z1 ∧ Z0
V2= ¬Z2 ∧ Z1 ∧ Z0
V3= Z2 ∧ ¬Z1 ∧ ¬ Z0
V4= Z2 ∧ ¬ Z1 ∧ Z0

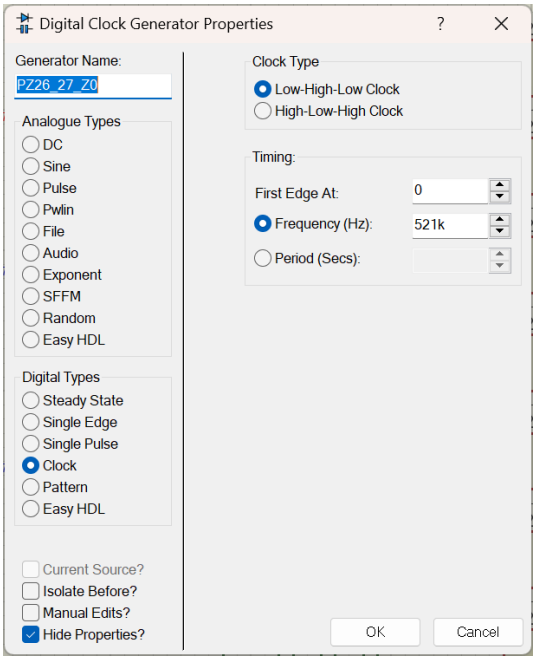
8.2. Використовуючи отримані рівняння, створіть схему лінійного дешифратора в робочій області системи Proteus (в іменах всіх елементів схеми та генераторів повинен бути ваш ідентифікатор).
Аналіз таблиці демонструє, що кожному вхідному коду відповідає активація лише одного вихідного сигналу. Це однозначно вказує на коректність побудови дешифратора. Така унікальна відповідність між вхідними кодами та вихідними сигналами підтверджує, що дешифратор функціонує згідно з очікуваною логікою, правильно перетворюючи закодовану інформацію у відповідні вихідні сигнали.
9. Задайте значення вхідних сигналів z0…z2.
Задала частоту сигналів Z0, Z1, Z2:

f0 = 4*f = 4*118000 = 472 000 Гц;
f1 = 2*f = 2*118000 = 236 000 Гц;
f2 = f = 118 000 Гц.

9.1. Генератори вхідних сигналів z0…z2 розташуйте зліва вертикально в порядку зростання (як на рис. 1.18).

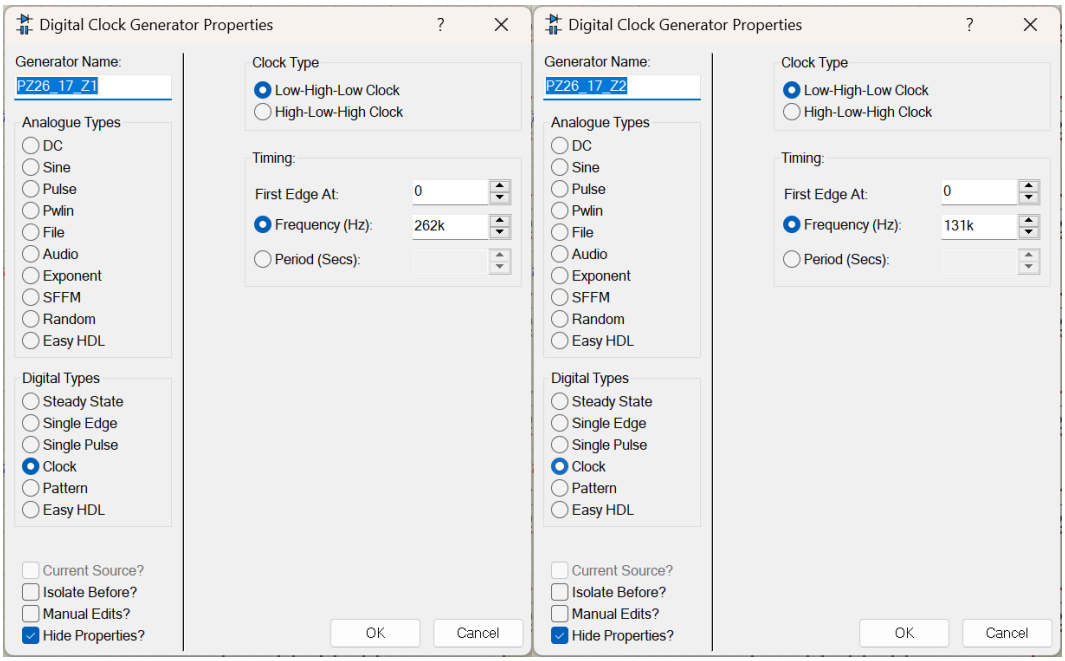
9.2. Сигнал z0 з частотою f0 = 4*f задайте за допомогою цифрового генератора

131 000*4 = 524 000



9.3. Аналогічно задайте сигнал z1 з частотою f1 = 2*f (рис. 1.28 б), а також сигнал z2 з частотою f2 = f.

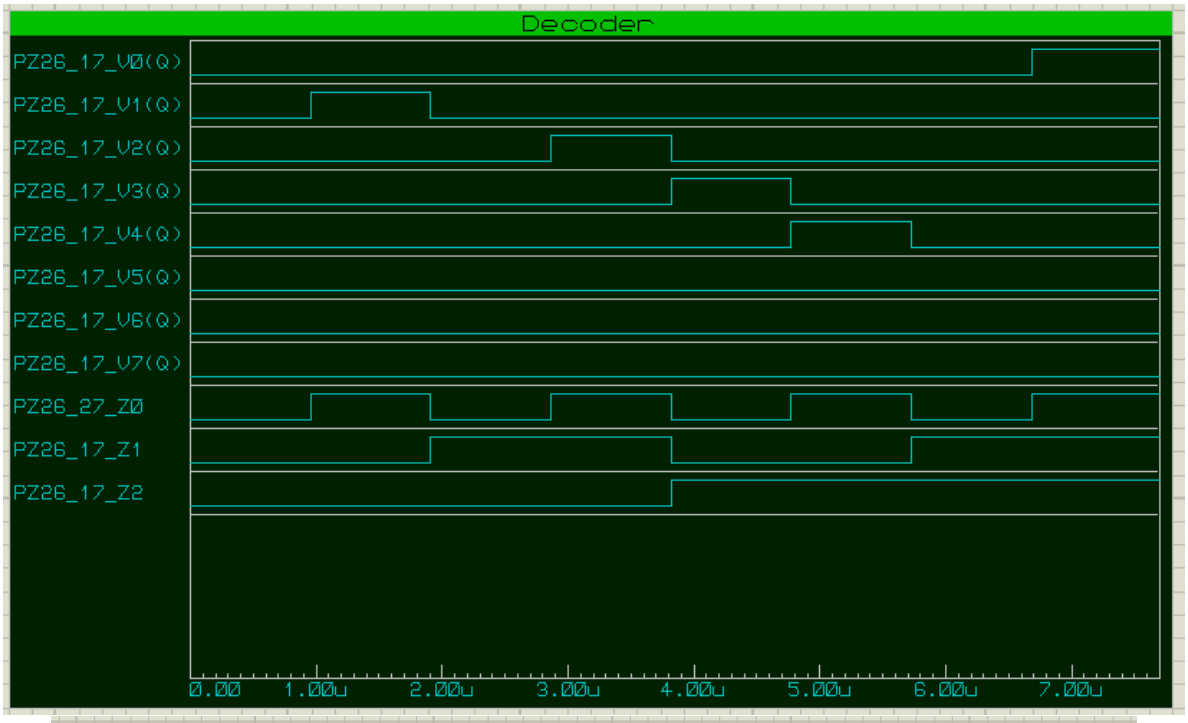
131 000*2 = 261 000

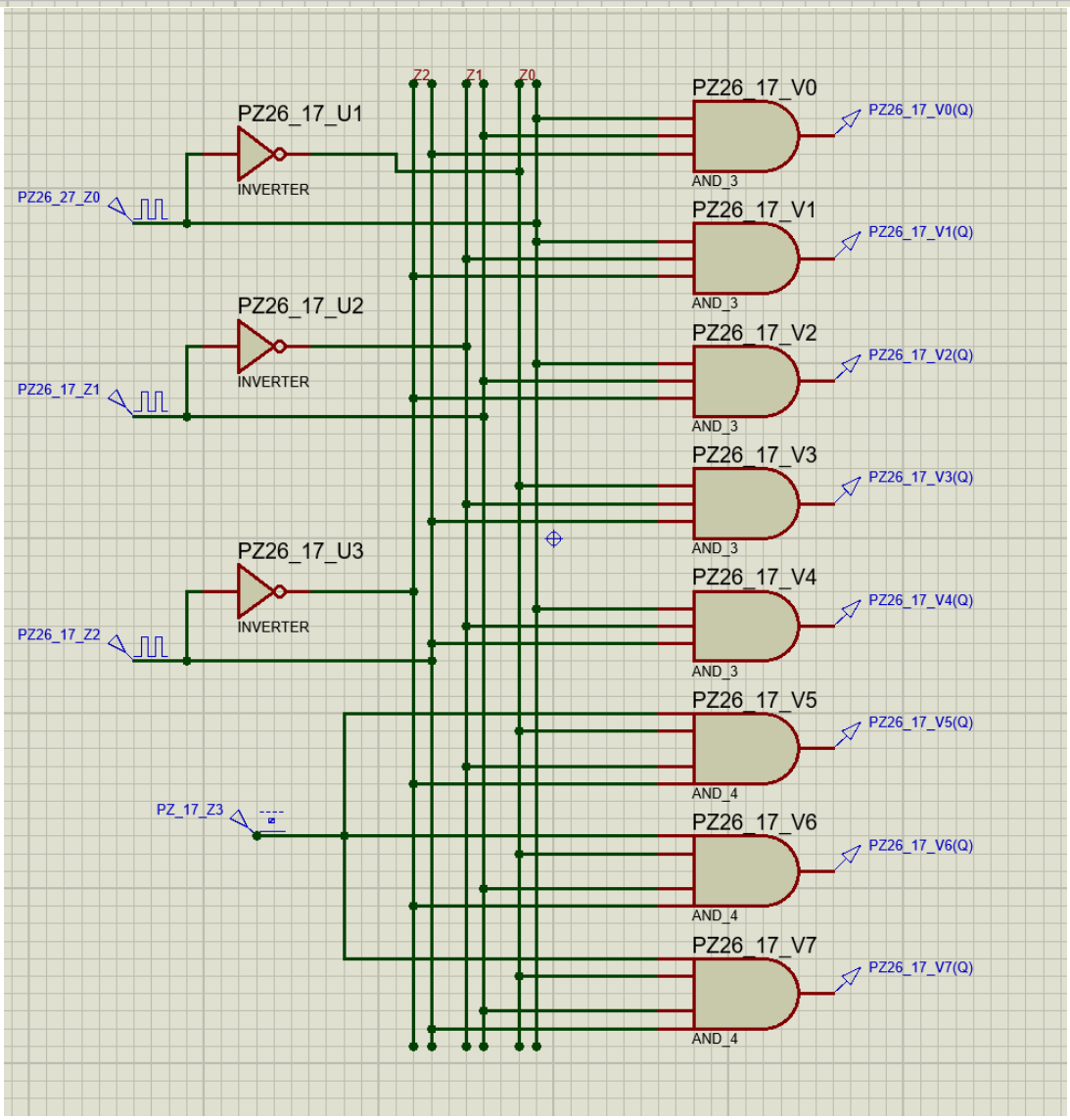


10. Побудуйте аналогічно до п. 4 цифровий графік з заголовком Decoder на часовому проміжку 0…T, на якому виведіть вхідні та вихідні сигнали в такій послідовності: z0… z2, v0…v7.

Побудувала цифровий графік лінійного дешифратора на часовому проміжку 0…T, де T = 0.00000847:

11. Проаналізуйте отримані часові діаграми вхідних і вихідних сигналів на кожному з восьми проміжків: kτ…(k+1)τ, k = 0, …7 і складіть таблицю істинності.


Decoder
PZ26_17_V0(Q)
PZ26_17_V1(Q)
PZ26_17_V2(Q)
PZ26_17_V3(Q)
PZ26_17_V4(Q)
PZ26_17_V5(Q)
PZ26_17_V6(Q)
PZ26_17_V7(Q)
PZ26_27_Z0
PZ26_17_Z1
PZ26_17_Z2
0.00
1.00u
2.00u
3.00u
4.00u
5.00u
6.00u
7.00u


Z2
Z1
Z0
PZ26_17_U1
INVERTER
PZ26_27_Z0
PZ26_17_U2
INVERTER
PZ26_17_Z1
PZ26_17_U3
INVERTER
PZ26_17_Z2
PZ_17_Z3
PZ26_17_V0
AND_3
PZ26_17_V0(Q)
PZ26_17_V1
AND_3
PZ26_17_V1(Q)
PZ26_17_V2
AND_3
PZ26_17_V2(Q)
PZ26_17_V3
AND_3
PZ26_17_V3(Q)
PZ26_17_V4
AND_3
PZ26_17_V4(Q)
PZ26_17_V5
AND_4
PZ26_17_V5(Q)
PZ26_17_V6
AND_4
PZ26_17_V6(Q)
PZ26_17_V7
AND_4
PZ26_17_V7(Q)

Таблиця істинності

Z2	Z1	Z0	V7	V6	V5	V4	V3	V2	V1	V0
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	1

Аналіз таблиці демонструє, що кожному вхідному коду відповідає активація лише одного вихідного сигналу. Це однозначно вказує на коректність побудови дешифратора. Така унікальна відповідність між вхідними кодами та вихідними сигналами підтверджує, що дешифратор функціонує згідно з очікуваною логікою, правильно перетворюючи закодовану інформацію у відповідні вихідні сигнали.

12. Порівняйте отриману таблицю істинності з заданою. Зробіть висновки про синтезований лінійний дешифратор. Збережіть проєкт в системі Proteus.

13. Створіть новий проєкт з іменем LR_2c в цій же папці 1_AK.

14. Синтезуйте і введіть в систему програм Proteus схему мультиплексора "5 в 1" за заданою таблицею істинності відповідно до індивідуального завдання в табл. 1.6 (a2, a1, a0 - адресні входи; d4, d3, d2, d1, d0 - інформаційні входи).

14.1. Запишіть вихідний сигнал мультиплексора D в досконалій диз’юнктивній нормальній формі.

$D = (a0 \wedge a1 \wedge a2 \wedge d0) \vee (a0 \wedge \neg a1 \wedge \neg a2 \wedge d1) \vee (a0 \wedge a1 \wedge \neg a2 \wedge d2) \vee (\neg a0 \wedge \neg a1 \wedge a2 \wedge d3) \vee \vee (a0 \wedge \neg a1 \wedge a2 \wedge d4)$

14.2. Використовуючи отриману логічну функцію, створіть схему мультиплексора в робочій області системи Proteus (в іменах всіх елементів схеми та генераторів повинен бути ваш ідентифікатор).

15. Задайте значення сигналів на адресних входах a0…a2 аналогічно як для сигналів z0…z2.

16. Задайте значення сигналів на інформаційних входах d0…d4.

16.1. Сигнал d0 задайте за допомогою цифрового генератора DPATTERN як на рис. 1.26 б, однак ширина імпульсу має бути T/64 сек.

16.2. Сигнали d1…d4 задайте аналогічно, в поле Specific pulse train введіть відповідно значення 01100000 для d1, 01110000 для d2, 01111000 для d3, і 01111100 для d4.

Задала значення сигналів на інформаційних входах d0…d4:

d0 = 01000000

d1 = 01100000

d2 = 01110000

d3 = 01111000

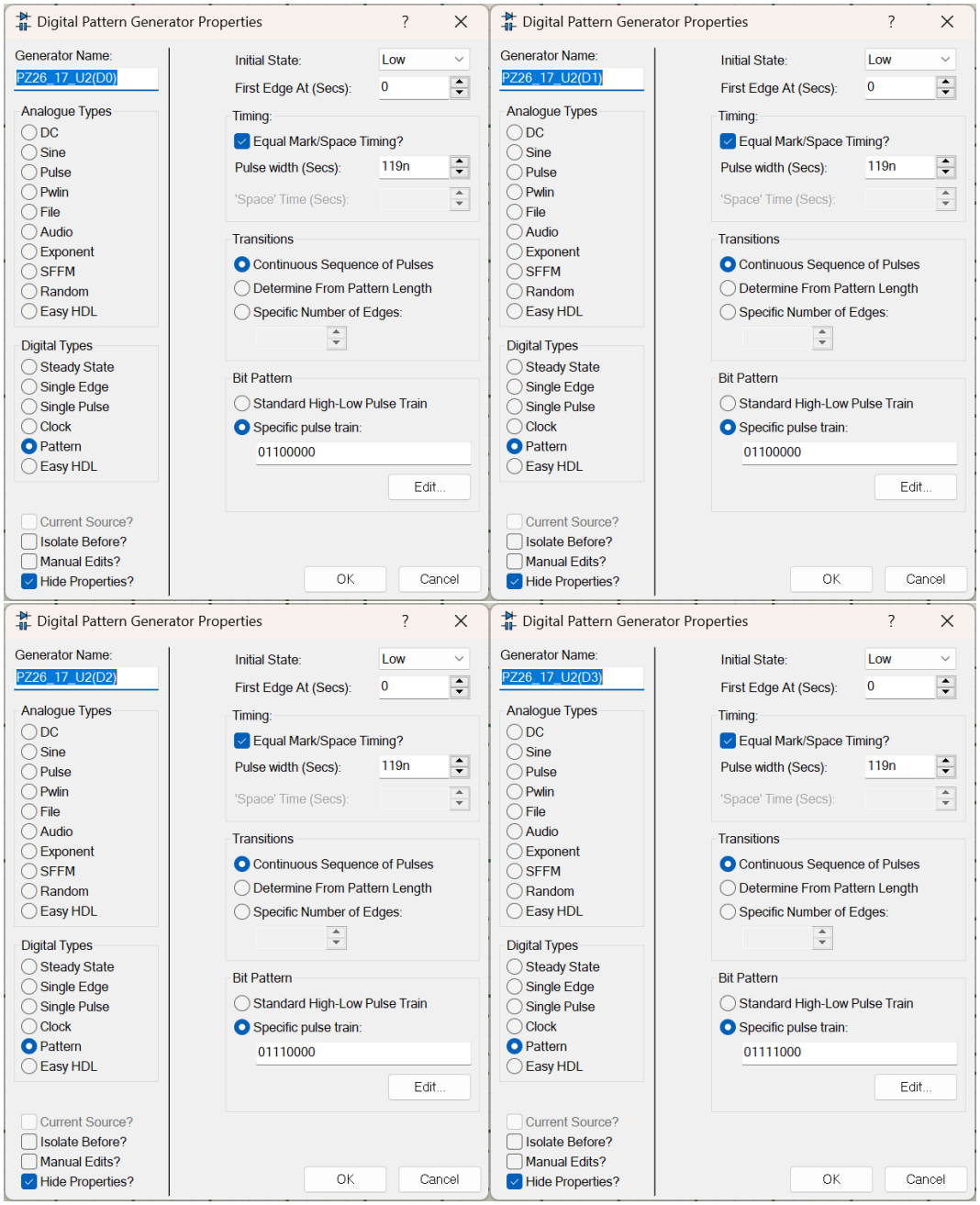
d4 = 01111100

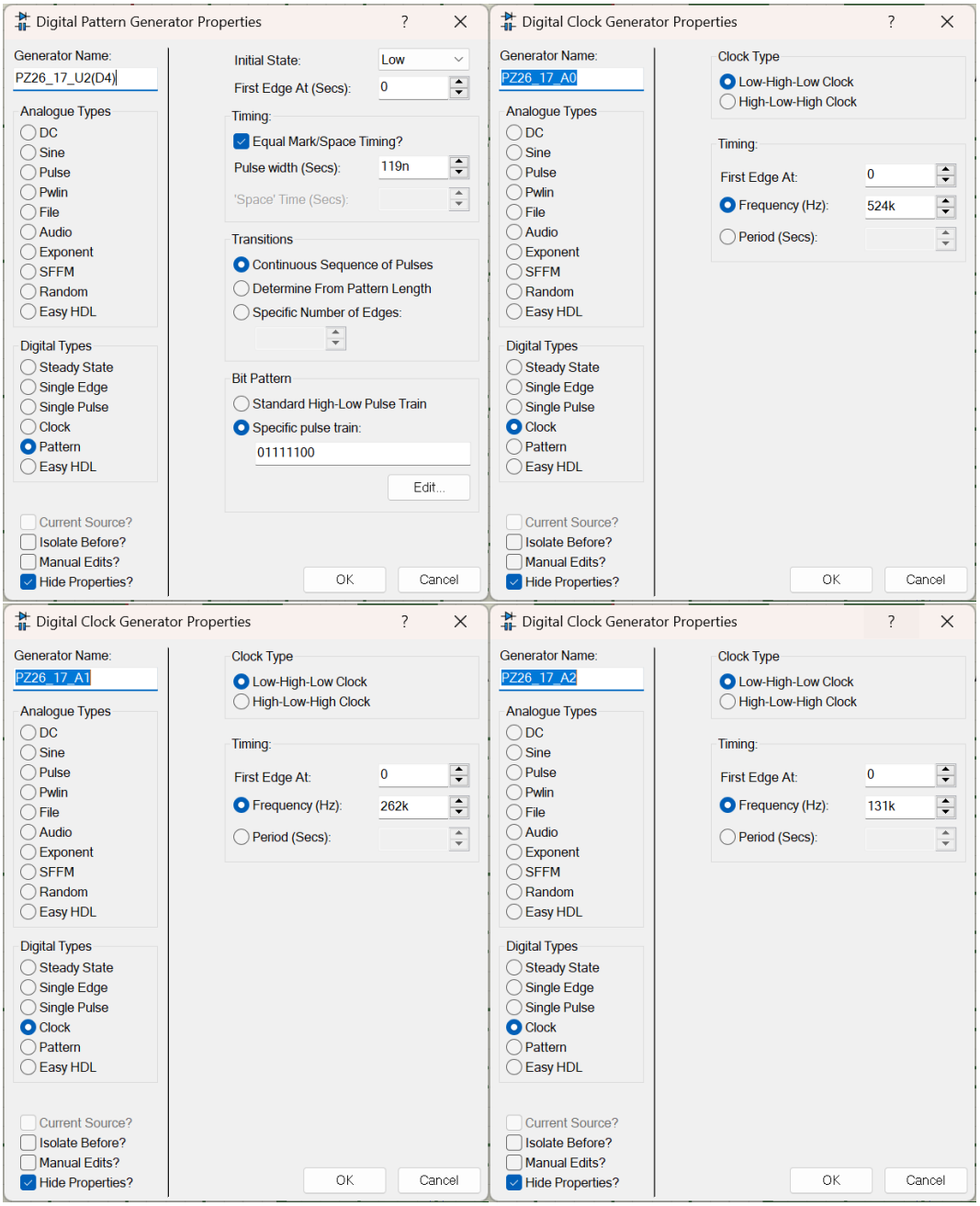
A0 = 131 000*4 = 524 000

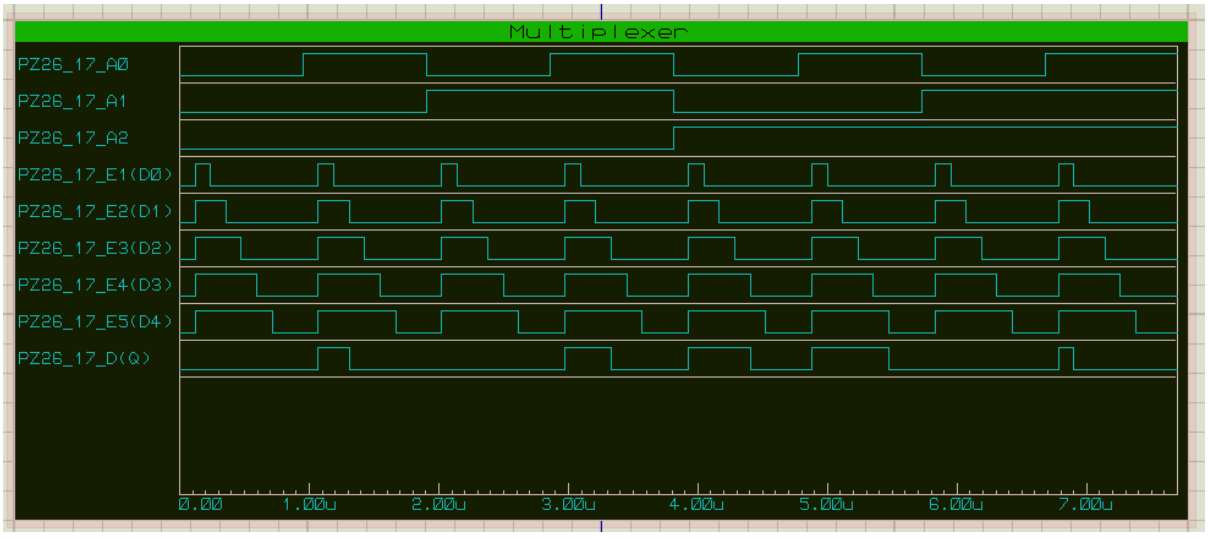
A1 = 131 000*2 = 262 000

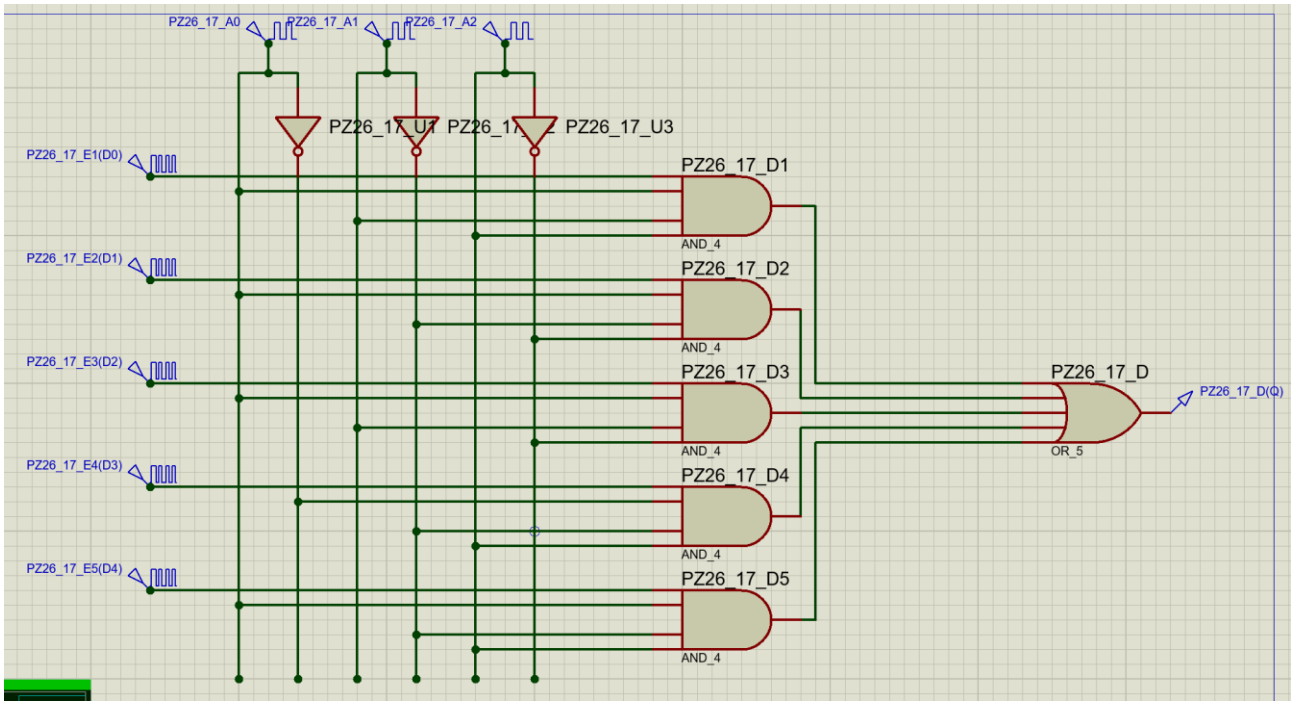
A3 = 131 000

17. Побудуйте аналогічно до п. 4 цифровий графік з заголовком Multiplexer на часовому проміжку 0…T, на якому виведіть вхідні та вихідні сигнали в такій послідовності: a0… a2, d0…d4, D.


Digital Pattern Generator Properties
Generator Name:
PZ26_17_U2(D0)
Analogue Types
DC
Sine
Pulse
Pwlin
File
Audio
Exponent
SFFM
Random
Easy HDL
Digital Types
Steady State
Single Edge
Single Pulse
Clock
Pattern
Easy HDL
Current Source?
Isolate Before?
Manual Edits?
Hide Properties?
Initial State:
Low
First Edge At (Secs):
0
Timing:
Equal Mark/Space Timing?
Pulse width (Secs):
119n
'Space' Time (Secs):
Transitions
Continuous Sequence of Pulses
Determine From Pattern Length
Specific Number of Edges:
Bit Pattern
Standard High-Low Pulse Train
Specific pulse train:
01100000
Edit...
OK
Cancel
Digital Pattern Generator Properties
Generator Name:
PZ26_17_U2(D1)
Analogue Types
DC
Sine
Pulse
Pwlin
File
Audio
Exponent
SFFM
Random
Easy HDL
Digital Types
Steady State
Single Edge
Single Pulse
Clock
Pattern
Easy HDL
Current Source?
Isolate Before?
Manual Edits?
Hide Properties?
Initial State:
Low
First Edge At (Secs):
0
Timing:
Equal Mark/Space Timing?
Pulse width (Secs):
119n
'Space' Time (Secs):
Transitions
Continuous Sequence of Pulses
Determine From Pattern Length
Specific Number of Edges:
Bit Pattern
Standard High-Low Pulse Train
Specific pulse train:
01100000
Edit...
OK
Cancel
Digital Pattern Generator Properties
Generator Name:
PZ26_17_U2(D2)
Analogue Types
DC
Sine
Pulse
Pwlin
File
Audio
Exponent
SFFM
Random
Easy HDL
Digital Types
Steady State
Single Edge
Single Pulse
Clock
Pattern
Easy HDL
Current Source?
Isolate Before?
Manual Edits?
Hide Properties?
Initial State:
Low
First Edge At (Secs):
0
Timing:
Equal Mark/Space Timing?
Pulse width (Secs):
119n
'Space' Time (Secs):
Transitions
Continuous Sequence of Pulses
Determine From Pattern Length
Specific Number of Edges:
Bit Pattern
Standard High-Low Pulse Train
Specific pulse train:
01110000
Edit...
OK
Cancel
Digital Pattern Generator Properties
Generator Name:
PZ26_17_U2(D3)
Analogue Types
DC
Sine
Pulse
Pwlin
File
Audio
Exponent
SFFM
Random
Easy HDL
Digital Types
Steady State
Single Edge
Single Pulse
Clock
Pattern
Easy HDL
Current Source?
Isolate Before?
Manual Edits?
Hide Properties?
Initial State:
Low
First Edge At (Secs):
0
Timing:
Equal Mark/Space Timing?
Pulse width (Secs):
119n
'Space' Time (Secs):
Transitions
Continuous Sequence of Pulses
Determine From Pattern Length
Specific Number of Edges:
Bit Pattern
Standard High-Low Pulse Train
Specific pulse train:
01111000
Edit...
OK
Cancel


Digital Pattern Generator Properties
Generator Name:
PZ26_17_U2(D4)
Initial State:
Low
First Edge At (Secs):
0
Timing:
Equal Mark/Space Timing?
Pulse width (Secs):
119n
'Space' Time (Secs):
Transitions
Continuous Sequence of Pulses
Determine From Pattern Length
Specific Number of Edges:
Bit Pattern
Standard High-Low Pulse Train
Specific pulse train:
01111100
Edit...
Analogue Types
DC
Sine
Pulse
Pwlin
File
Audio
Exponent
SFFM
Random
Easy HDL
Digital Types
Steady State
Single Edge
Single Pulse
Clock
Pattern
Easy HDL
Current Source?
Isolate Before?
Manual Edits?
Hide Properties?
OK
Cancel
Digital Clock Generator Properties
Generator Name:
PZ26_17_A0
Clock Type
Low-High-Low Clock
High-Low-High Clock
Timing:
First Edge At:
0
Frequency (Hz):
524k
Period (Secs):
Digital Clock Generator Properties
Generator Name:
PZ26_17_A1
Clock Type
Low-High-Low Clock
High-Low-High Clock
Timing:
First Edge At:
0
Frequency (Hz):
262k
Period (Secs):
Digital Clock Generator Properties
Generator Name:
PZ26_17_A2
Clock Type
Low-High-Low Clock
High-Low-High Clock
Timing:
First Edge At:
0
Frequency (Hz):
131k
Period (Secs):


Multiplexer
PZ26_17_A0
PZ26_17_A1
PZ26_17_A2
PZ26_17_E1(D0)
PZ26_17_E2(D1)
PZ26_17_E3(D2)
PZ26_17_E4(D3)
PZ26_17_E5(D4)
PZ26_17_D(Q)
0.00
1.00u
2.00u
3.00u
4.00u
5.00u
6.00u
7.00u



18. Проаналізуйте отримані часові діаграми всіх сигналів на кожному з восьми проміжків: kτ…(k+1)τ, k = 0, …7 і складіть таблицю істинності для мультиплексора.
19. Порівняйте отриману таблицю істинності з заданою. Зробіть висновки про синтезований мультиплексор. Збережіть проєкт в системі Proteus.

A0	0	1	0	1	0	1	0	1
A1	0	0	1	1	0	0	1	1
A2	0	0	0	0	1	1	1	1
D	0	1	0	1	1	1	0	1
di	0	d1	0	d2	d3	d4	0	d0

Аналіз таблиці та графіка демонструє правильність синтезу мультиплексора. Це підтверджується тим, що в областях поширення сигналу D входи d0, d1, d2, d3, d4 набувають значення одиниці за тих самих комбінацій аргументів, які були визначені у вихідній таблиці істинності. Така відповідність свідчить про коректну реалізацію функціональності мультиплексора згідно з початковими вимогами.

20. Створіть новий проєкт з іменем LR_2d в цій же папці 1_AK.
21. Синтезуйте і введіть в систему програм Proteus схему демультиплексора "1 в 5".

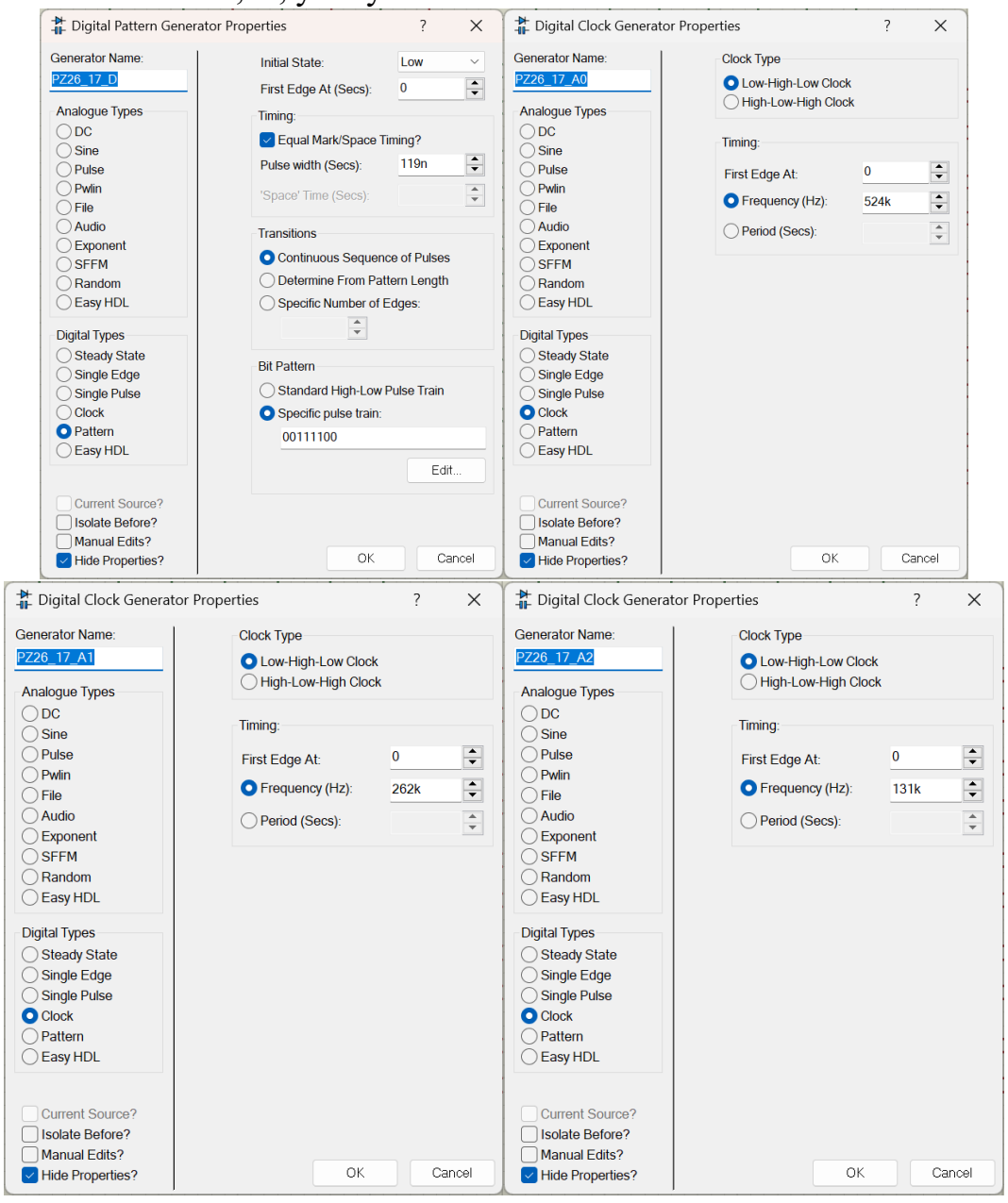
21.1. Запишіть за заданою таблицею істинності відповідно до індивідуального завдання в табл. 1.6 сигнал на кожному виході де

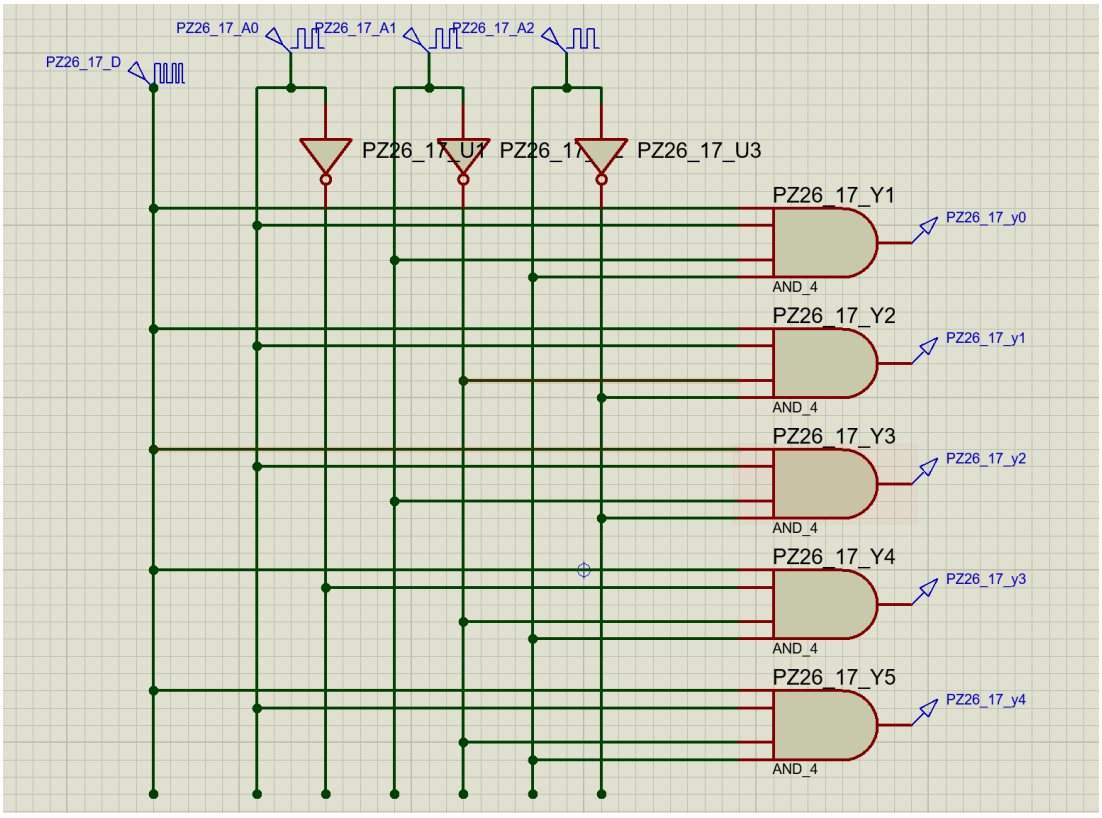
мультиплексора в досконалій диз'юнктивній нормальній формі (a2, a1, a0 - адресні входи; y4, y3, y2, y1, y0 - інформаційні виходи; прийміть: y0 ≡ d0, y1 ≡ d1, y2 ≡ d2, y3 ≡ d3, y4 ≡ d4).

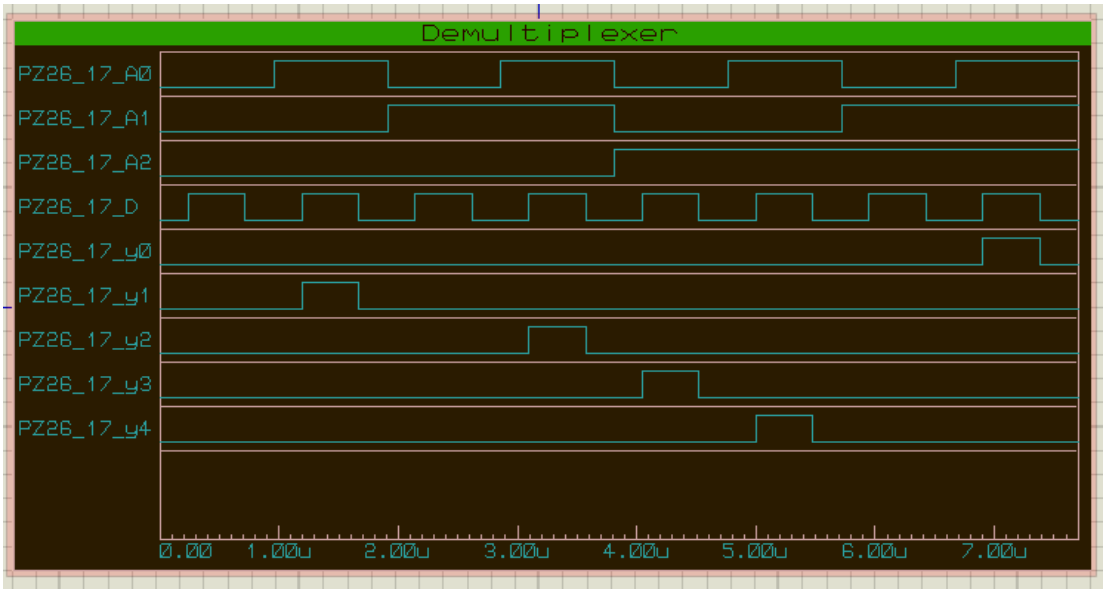
21.2. Використовуючи отримані логічні функції, створіть схему демультиплексора в робочій області системи Proteus (в іменах всіх елементів схеми та генераторів повинен бути ваш ідентифікатор).

22. Задайте значення сигналу D на інформаційному вході демультиплексора за допомогою цифрового генератора DPATTERN: в поле Specific pulse train введіть відповідно значення 00111100.

23. Побудуйте аналогічно до п. 4 цифровий графік з заголовком Demultiplexer на часовому проміжку 0…T, на якому виведіть вхідні та вихідні сигнали в такій послідовності: a0… a2, D, y0…y4







24. Проаналізуйте отримані часові діаграми всіх сигналів на кожному з восьми проміжків: kτ…(k+1)τ, k = 0, …7 і складіть таблицю істинності для демультиплексора.

A0	0	1	0	1	0	1	0	1
A1	0	0	1	1	0	0	1	1
A2	0	0	0	0	1	1	1	1
D	0	1	0	1	1	1	0	1
yi	0	y1	0	y2	y3	y4	0	y0

25. Порівняйте отриману таблицю істинності з заданою. Зробіть висновки про синтезований демультиплексор. Збережіть проєкт в системі Proteus.

Аналіз таблиці та графіка підтверджує коректність синтезу демультиплексора. Сигнал D розподіляється на виходи y0, y1, y2, y3, y4, які набувають значення одиниці за тих самих комбінацій вхідних аргументів, що вказані в початковій таблиці істинності. Це свідчить про правильну роботу схеми відповідно до заданих умов.

Висновок

Під час виконання цієї лабораторної роботи я навчилася моделювати схеми шифратора, дешифратора, мультиплексора та демультиплексора. На основі результатів роботи отриманих схем було створено графік, який наочно демонструє їх функціонування. Для перевірки коректності синтезу кожної схеми я використала таблиці істинності. Це дозволило мені переконатися, що всі схеми були складені правильно і працюють відповідно до заданих параметрів. Така практична робота надала мені можливість не лише теоретично вивчити, але й застосувати на практиці знання про ці важливі компоненти цифрової електроніки.