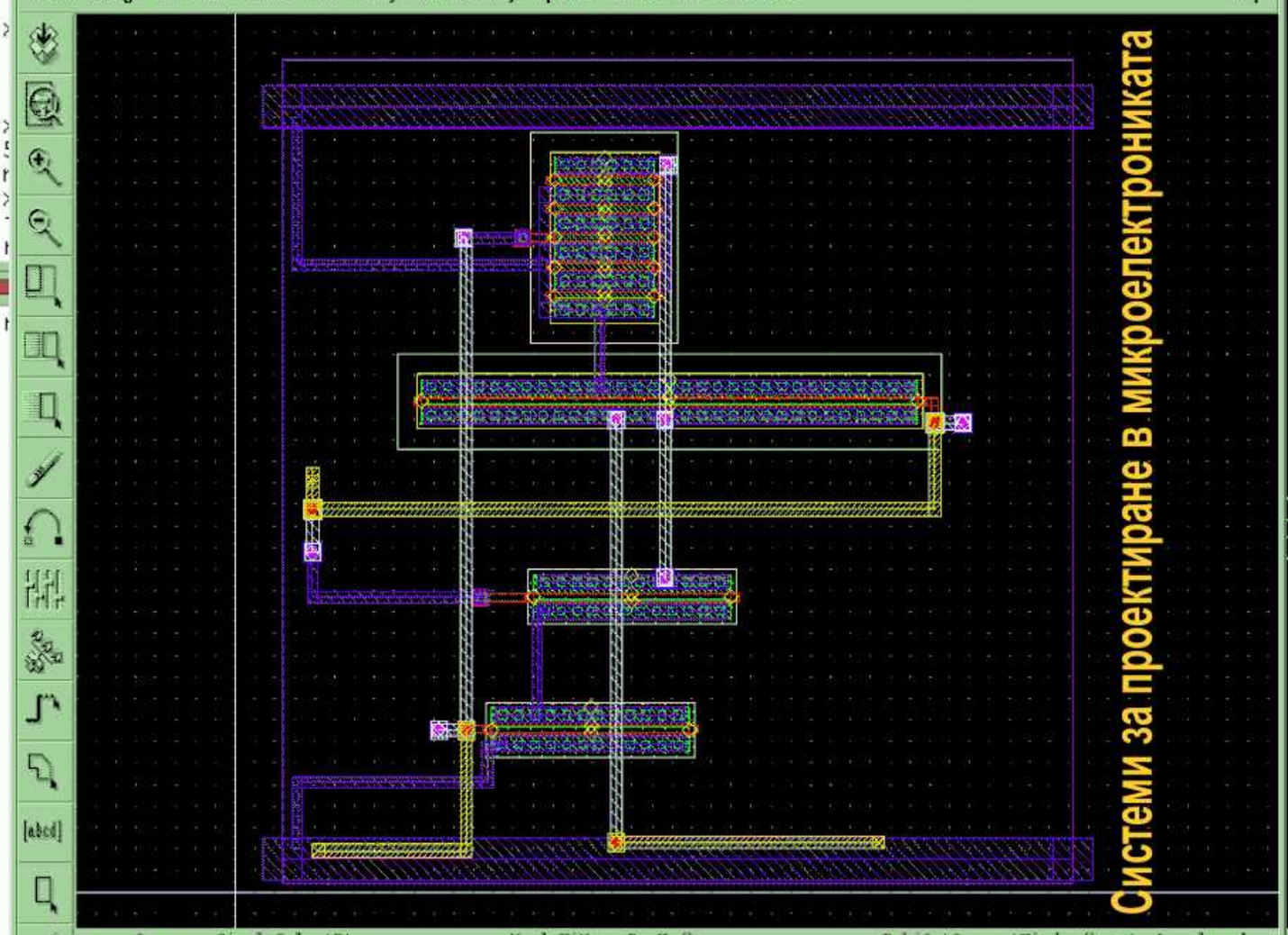
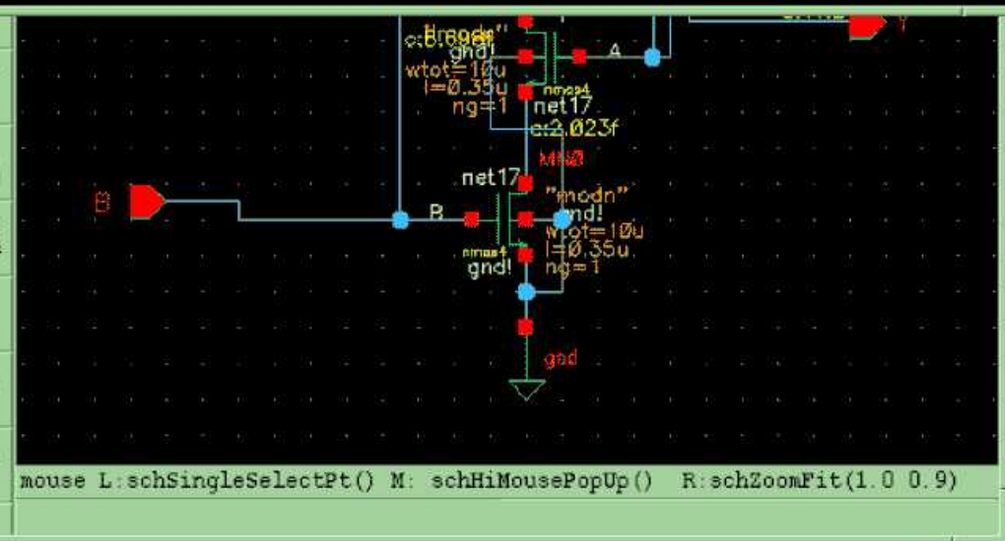
М. Христов, Р. Радонов, Б. Дончев, К. Михайлова,
Д. Пукнева, О. Антонова, Д. Арабаджиев, О. Манчев,
И. Партенов

Системи за проектиране в микроелектрониката

Ръководство за лабораторни упражнения





Марин Христов, Росен Радонов, Благомир Дончев,
Кристи Михайлова, Диана Пукнева, Олга Антонова,
Добромир Арабаджиев, Огнян Манчев, Илия Партенов

Ръководство

за лабораторни упражнения
по

Системи за проектиране в микроелектрониката

София, 2004

Ръководството за лабораторни упражнения по "Системи за проектиране в микроелектрониката" е предназначено за студентите от специалността "Електроника" на Технически университет – София при подготовката им по дисциплините "Системи за проектиране в микроелектрониката" – за образователно-квалификационна степен "магистър" и "Автоматизация на инженерния труд в микроелектрониката" и "Проектиране на интегрални цифрови схеми и системи" за образователно-квалификационна степен "бакалавър". То може да бъде използвано и от студентите от други сходни специалности, докторанти, инженери, проектанти и др., интересуващи се от работата и приложението на индустриални CAD системи.

У В О Д

Проектирането на съвременни микроелектронни схеми и устройства е немислимо без широкото използване на автоматизирани компютърни системи. Съществува огромно разнообразие от системи за проектиране, както по отношение на създаваните продукти – електронни системи и печатни платки, интегрални схеми, микромеханични устройства, системи върху чип, така и подпомаганите етапи за синтез, проектиране, симулация, моделиране и др. Разбира се голяма роля играят и използваните операционни и компютърни системи.

В настоящето ръководство за лабораторни упражнения се описват практически проекти с наложилите се като световен индустриален стандарт продукти: CADENCE, SYNOPSYS, MENTOR GRAPHICS, XILINX. За разбиране същността на описаните практически методики и процедури е необходимо предварително запознаване със съответния материал от учебника по “Системи за проектиране в микроелектрониката”.

Обръщаме специално внимание на факта, че представените софтуерни системи постоянно се променят и усъвършенстват поради бързото и непрекъснато развитие на използваните методи, алгоритми, технология за производство, хардуерни платформи и др.

Разработените лабораторни упражнения условно могат да бъдат разделени на две групи: изследване и проектиране на аналогови елементи и схеми и цифрови схеми и системи.

Настоящата книга обобщава десетгодишния опит на колектива на ECAD лабораторията при Факултета по Електронна техника и технологии в Техническия университет – София. Изразяваме нашата голяма благодарност и признателност за приноса и помощта на многобройните сътрудници на лабораторията, голяма част от които в момента работят в други фирми и организации в България и чужбина.

Ще приемем с внимание и интерес всички мнения, препоръки, предложения, целящи подобряването и осъвременяването на настоящето ръководство.

София, 2004г. *АВТОРИТЕ*

Упражнение № 1

Въведение в системата за проектиране CADENCE. Основни анализи.

I. Въведение в системата за проектиране CADENCE. Въвеждане на схема.

CADENCE DESIGN FRAMEWORK II изисква X-windows (CDE/Openwindows върху Sun SPARC станции). Тя е общ интерфейс към пълния набор от програмни продукти за проектиране на интегрални схеми в системата **CADENCE**. Използването на общ потребителски интерфейс и обща база данни позволява лесно преминаване между различни етапи при проектиране.

1. Стартиране на CADENCE.

Първата стъпка след влизане в **UNIX** и отваряне на терминал е създаването на директория, в която ще се работи с **CADENCE** и където ще се записват данните от направените схеми и симулации:

1) **mkdir** <*име на директорията*>

Например: mkdir cadence

Следващата стъпка е влизане в създадената директория:

2) **cd** <*име на директорията*>

Например: cd cadence

3) Стартиране на **CADENCE**:

source ~radonov/scripts/.cshrc-cad446
ams_cds –tech csx –mode fb

Опцията "**–tech csx**" определя технологията, параметрите на която ще се използват в процеса на проектиране с **CADENCE**. В случая се използва "**csx**" за AMS 0.35μm Si CMOS технология. Тази опция е задължителна, само когато се стартира **CADENCE** за първи път, а опцията "**–mode fb**" зарежда всички библиотеки необходими за топологично проектиране.

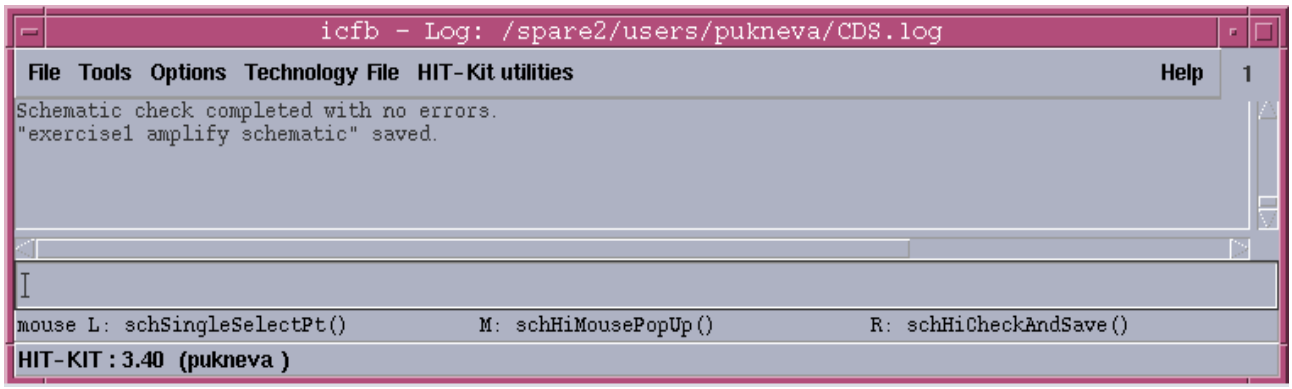
Няколко секунди след стартирането на **CADENCE** се появяват два прозореца:

*__CIW__ – **Command Interpreter Window** или команден прозорец за въвеждане и изпълнение на командите (фиг. 1).

*__LMW__ – **Library Manager Window** или прозорец на библиотечния браузър (фиг. 2).

2. Използване на CIW.

CIW е основния команден прозорец на системата за проектиране **CADENCE**. На фиг. 1 е показан неговият общ вид.



Фиг. 1. Команден прозорец на **CADENCE** - **CIW**

- **Име на прозореца** - показва името на съответния редактор на **CADENCE** и пълното име на log файла, където се записват текущите сесии.
- **Лента с менюта** – съдържа менютата с команди за достъп до продуктите на **CADENCE DESIGN FRAMEWORK II**.
- **Поле за извеждане на информацията** - показва команди от текущата сесия и резултатите от тяхното изпълнение.
- **Команден ред** - в него могат директно да се въвеждат команди и изрази. Използва се **SKILL** – програмен език на **CADENCE**.
- **Ред, описващ текущите настройки на бутоните на мишката**.
- **Информационен ред** - подсказва каква трябва да бъде следващата стъпка при изпълнение на текущата команда.

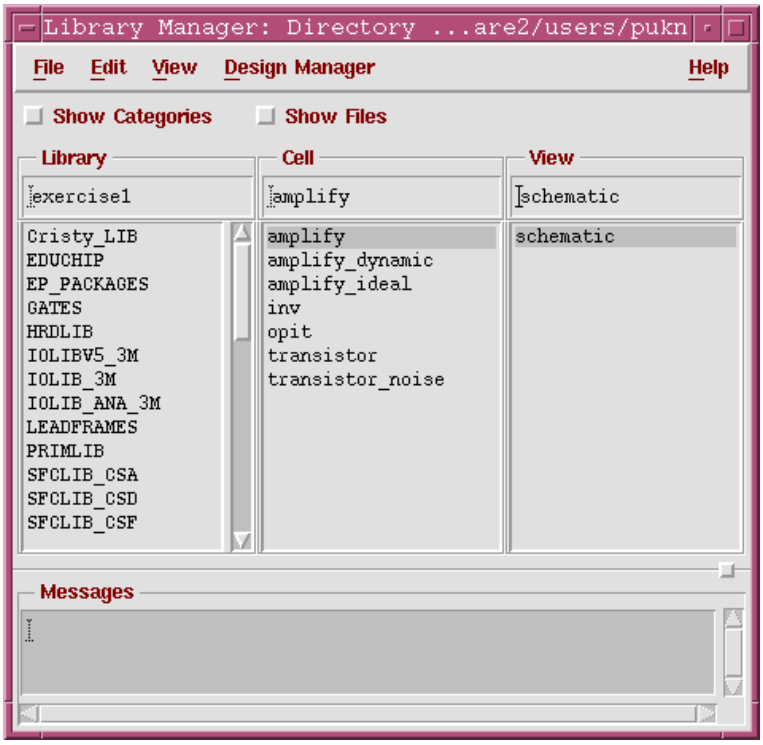
Спирането на **CADENCE** може да стане по два начина:

- Изписване на **exit** в командния ред:
- Избиране на командата **exit** от менюто **File**.

3. Използване на LMW.

LMW е библиотечния редактор на **CADENCE**. В него се визуализират всички налични библиотеки, могат да се създават нови такива, да се редактират или изтриват, както и да се извършват операции по създаване, съхранение, преместване и т.н. на различните клетки (**cells**) и техните представяния (**cellviews**).

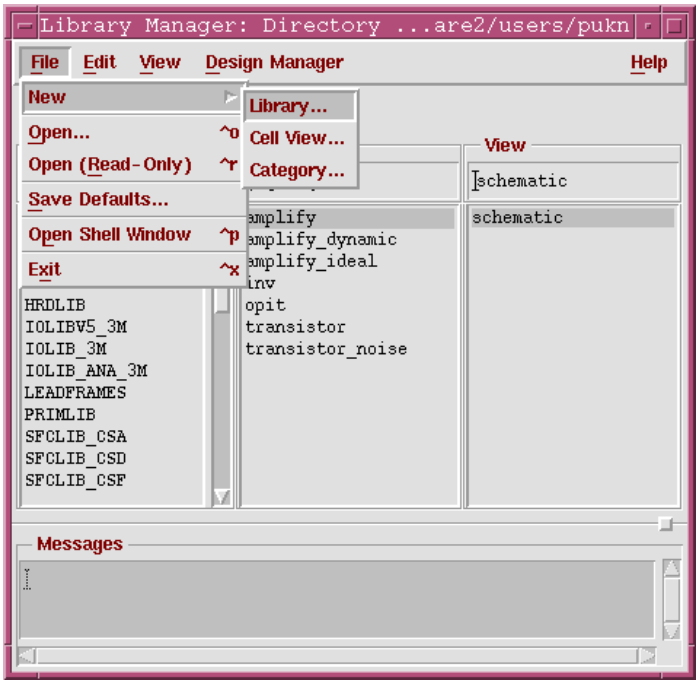
Клетка е най-общото име на проект. Дадена библиотека може да съдържа различен брой проекти (клетки). Всеки проект може да има различни видове представяния, например: схемно (**schematic**), символно (**symbol**), топологично (**layout**) и др.



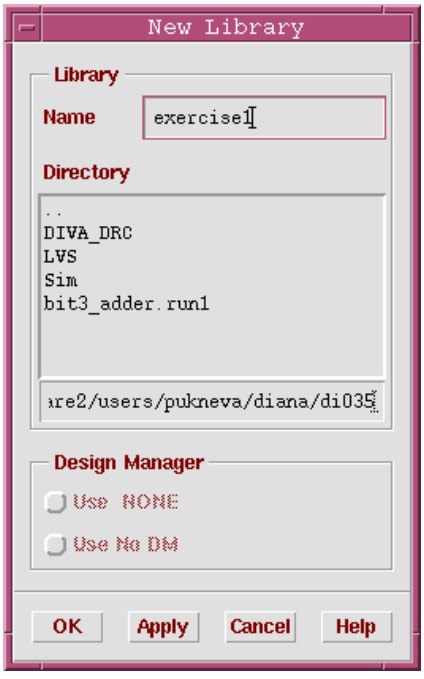
Фиг. 2. Прозорец на библиотечния браузър

4. Създаване на нова библиотека.

След стартиране на **CADENCE**, за да могат да се въвеждат нови проекти, е необходимо първо да се създаде нова библиотека, където ще се съхраняват тези проекти. Това става като от менюто **File** на библиотечния браузър се избере командата **New** ⇒ **Library**, както е показано на фиг. 3.



Фиг. 3. Създаване на нова библиотека чрез библиотечния браузър



Фиг. 4. Прозорец за задаване на име на нова библиотека

Появява се прозорецът **New Library** (фиг. 4). В полето за запис на име (**Name**) се задава име на новата библиотека, например **exercise1** и се натиска бутона **OK**. След това се появява прозорецът за обвързване на библиотеката с определен технологичен файл (**Technology File for New Library**). Менюто предлага възможност за обвързване на новата библиотека с различни технологични файлове. Това е необходимо, за да се проектира дизайна съобразно технологичните правила и изисквания. Избира се обвързване със съществуващ технологичен файл (**Attach to an existing techfile**) и се натиска **OK**. Появява се формата от фиг. 5 (**Attach Design Library to Technology File**), където се избира технологичният файл – в случая за 0.35μm Si CMOS технология **TECH_CSI**.

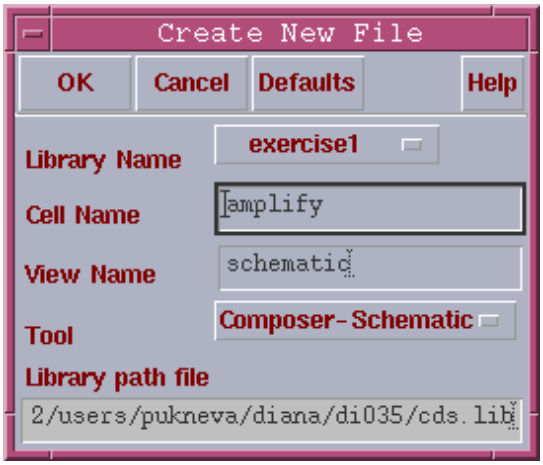
Фиг. 5. Обвързване на новата библиотеката с определен технологичен файл

5. Създаване на нов проект (клетка).

За създаване на нов проект от менюто **File** на **LMW** се избира командата **New** ⇒ **CellView** (фиг. 6). Например, в отворената форма се попълват:

Име на библиотеката: exercise1 (новата библиотека)
Име на клетката: amplify (име на новия проект)
Вид представяне: schematic

Натиска се бутонът **OK,** с което се отваря прозорецът на схемния редактор, в който ще се въведе новият проект.



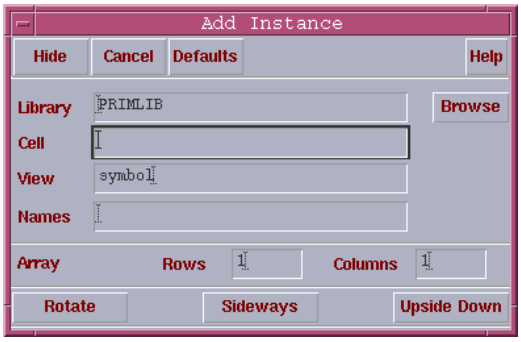
Фиг. 6. Прозорец за създаване на нова клетка

6. Въвеждане на електрическа схема.

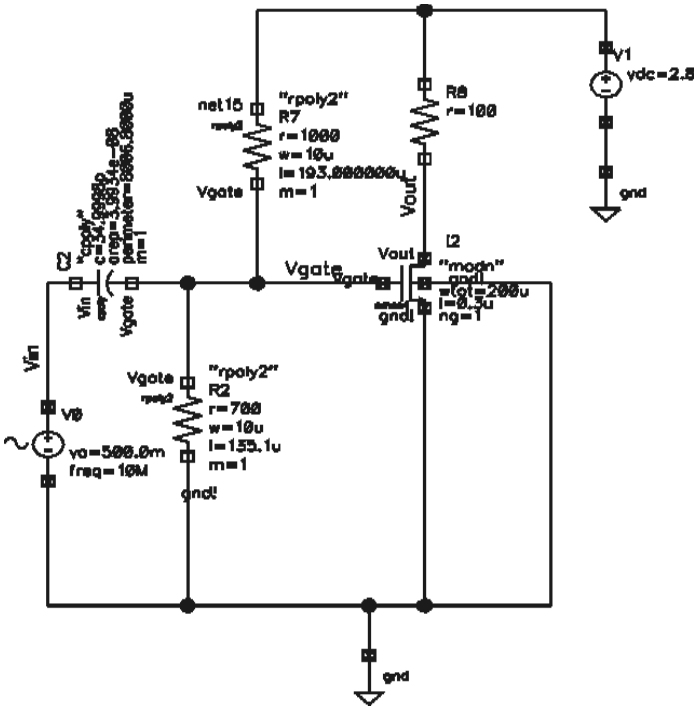
За да се въведе електрическата схема, се добавят и разполагат елементи в схемния редактор на **CADENCE** (**Virtuoso Schematic Composer**). За целта се използва командата **Add** ⇒ **Instance**. Същата команда може да се избере и чрез натискане на 10-ти бутон от лявата страна на редактора за изчертаване на схеми или като се използва бутонът **“i”** от клавиатурата. Независимо от това, кой от трите начина се използва, се отваря диалоговият прозорец за добавяне на елемент (фиг. 7). В него се натиска бутонът **Browse** за избор на библиотека. За използваната технология реалните елементи се намират в библиотеката **PRIMLIB** (AMS 0.35μm CMOS), а идеалните съответно в **analogLib**.

Избира се желаният елемент, например:

Library ⇒ AnalogLib
Cell Name ⇒ vdc
View Name ⇒ symbol



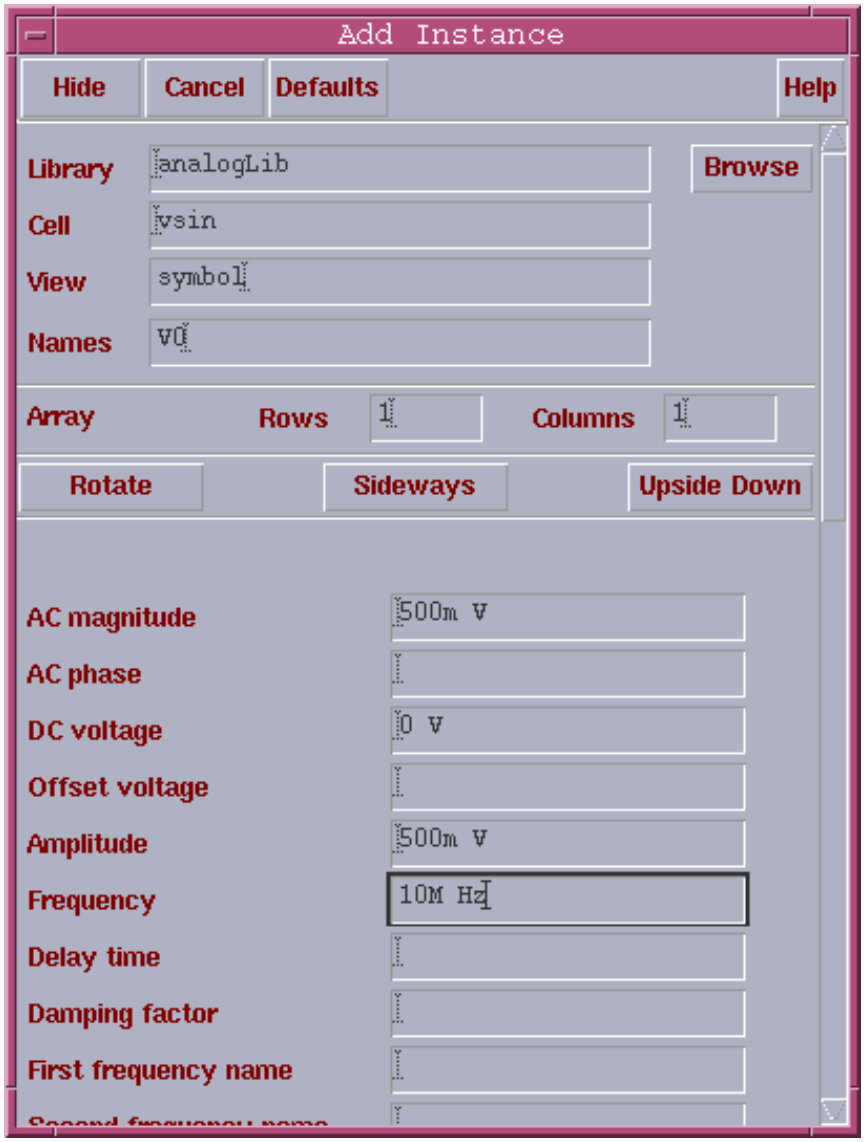
Фиг. 7. Прозорец за добавяне на елемент



Фиг. 8. Примерна схема

Избраният елемент може да се разполага многократно в схемния редактор. За да се прекрати командата, трябва да се натисне **Esc** от клавиатурата. С промяна на името на елемента във формата на командата **Add Instance** може да се продължи с разполагането на други елементи. Когато всички елементи, необходими за дадена схема са разположени, стойностите на техните параметри могат да се променят като се избере 9^{ти} бутон от лявата страна на прозореца на **Virtuoso Schematic Composer** (или “**q**” от клавиатурата). Това е еквивалентно на избиране на командата **Edit** ⇒ **Object Properties**. Имената на

елементите, стойностите на техните параметри и библиотеките, в които се намират са показани в таблица 1, а параметрите на входния източник (**vsin**) са показани на фиг. 9. Схемата, която се използва в това упражнение, е дадена на фиг. 8. Източникът **vsin** и масата (**gnd**) се намират в библиотеката **analogLib**.



Фиг. 9. Задаване параметрите на източника на напрежение **vsin**

Таблица 1.

Име на библиотеката	Име на елемента	Стойности на параметрите на елементите от фиг. 8
PRIMLIB	rpoly2	R7=1000Ω
PRIMLIB	rpoly2	R2=700Ω
analogLib	res	R8=100Ω
PRIMLIB	cpoly	C2 = 34,9998pF
PRIMLIB	nmos4	w = 200μm, l = 0.3μm, N=1
analogLib	vdc	DC voltage=2.8V

7. Изчертаване на връзките между елементите.

След като се разположат всички елементи, те трябва да се свържат помежду си. За изчертаване на връзките се използва командата **Add** ⇒ **Wire (narrow)** – 11 бутон от лявата страна на схемния редактор или "**w**" от клавиатурата. На свързващите проводници могат да се задават имена (**Add** ⇒ **Wire Name** или 13^{ти} бутон в прозореца на редактора (или "**l**" от клавиатурата). След като схемата е изцяло изчертана, тя се запазва с командата **Design** ⇒ **Save**. Може едновременно да се запази и провери за допуснати грешки при разполагането и свързването на елементите – **Design** ⇒ **Check and Save**.

II. Аналогова симулация с помощта на симулатора Spectre.

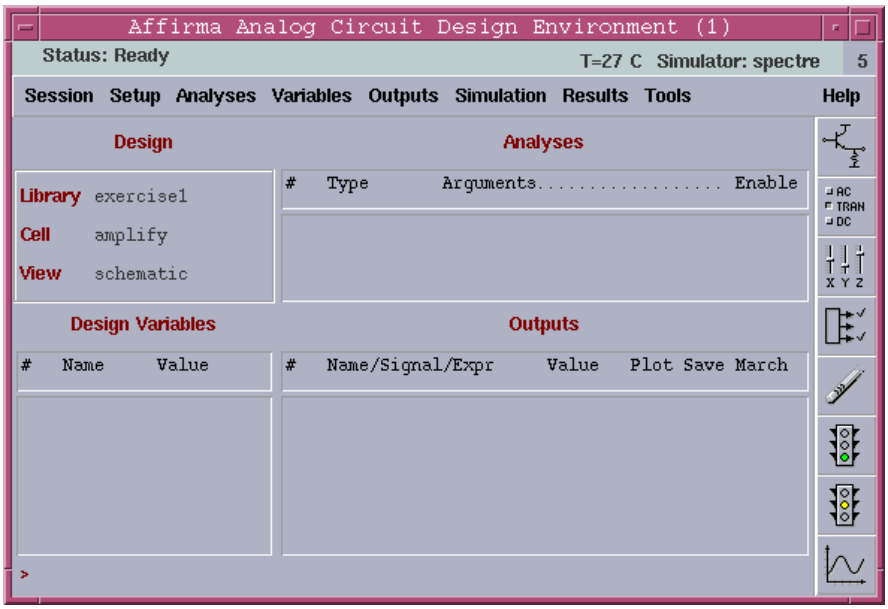
За да се избере и стартира симулация, от менюто **Tools** на схемния редактор се използва командата **Analog Environment,** след което се появява прозорецът на средата за аналогова симулация **Affirma Design Environment** (**ADE**), който е показан на фиг. 10.

Полето "**Design**" дава информация за проекта, който ще се симулира (библиотека, име и вид на представяне на проекта).

Полето "**Analyses**" съдържа списък на избраните анализи.

Полето "**Design variables**" показва списък на променливите, които са зададени в схемата и ще се използват при симулация.

Полето "**Outputs**" дава списък на напрежения/токове, които трябва да бъдат симулирани, визуализирани или записани.



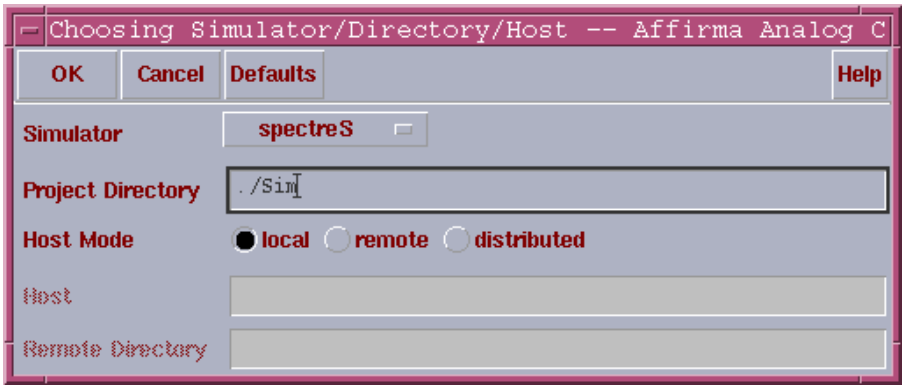
Фиг. 10. Прозорец на средата за аналогова симулация **ADE**

Бутоните, които са разположени от дясната страна в прозореца на **ADE** са помощни и представляват съкратени пътища към някои от основните команди в менюто:

1-ви бутон – Избор на проект (**Choose design**);

2-ри бутон – Избор на анализи (**Choose Analyses**);
3-ти бутон – Задаване и редактиране на променливи (**Edit variables**);
4-ти бутон – Избор на напрежения/токове, които ще се визуализират (**Setup Outputs**);
5-ти бутон – Изтриване (**Delete**);
6-ти бутон – Създаване на нетлист и стартиране на симулация (**Netlist and Run**);
7-ми бутон – Стартиране на симулация (**Run simulation**);
8-ми бутон – Визуализиране на избраните напрежения/токове (**Plot Outputs**).

1. Избор на симулатор.



Фиг. 11. Форма за избор на симулатор

Симулаторът, който се използва по подразбиране в средата за аналогова симулация **ADE**, е **Spectre**. Той може да се смени, като в прозореца на **ADE** се избере командата **Simulator/Directory/Host** от менюто **Setup**. Появява се формата показана на фиг. 11, която позволява да се избере желаният симулатор от падащото меню. Освен това формата дава възможност да се зададе или промени пътя до директорията, в която ще се съхраняват резултатите от симулациите за даден проект.

2. Избор на анализ.

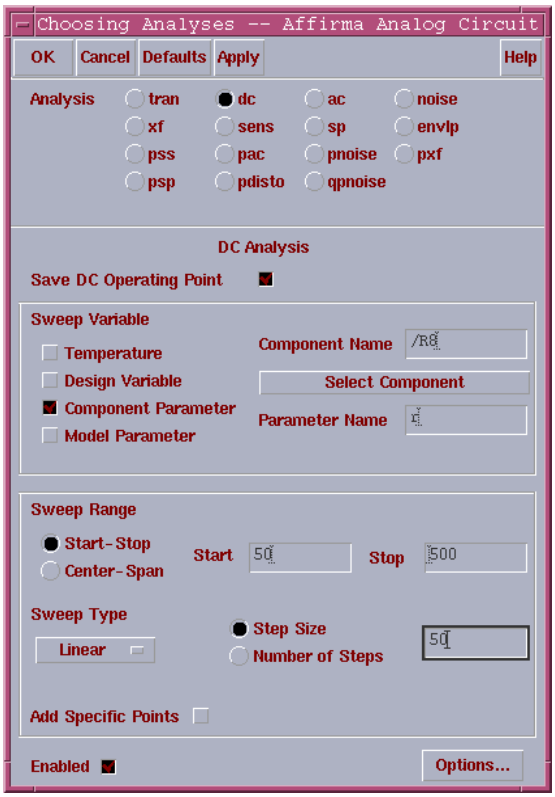
За да се зададе анализ, от лентата с менютата на **ADE** се избира **Analyses** ⇒ **Choose**. Появява се прозорец с всички достъпни анализи. От него може да се избере желаният вид анализ и да се зададат неговите опции. Симулацията се стартира като се натисне бутона със зеления светофар или се избере командата **Simulation** ⇒ **Run**. След като се изпълни симулацията има няколко начина за визуализиране на резултатите. Ако предварително са избрани да се визуализират някои изходи от схемата, те се записват в полето "**Outputs**" и ще се появят автоматично след приключване на симулацията. В противен случай може да се използва менюто **Results** ⇒ **Direct Plot**, или помощното средство - **Calculator**, което се намира в менюто **Tools**.

3. Постояннотоков анализ (dc).

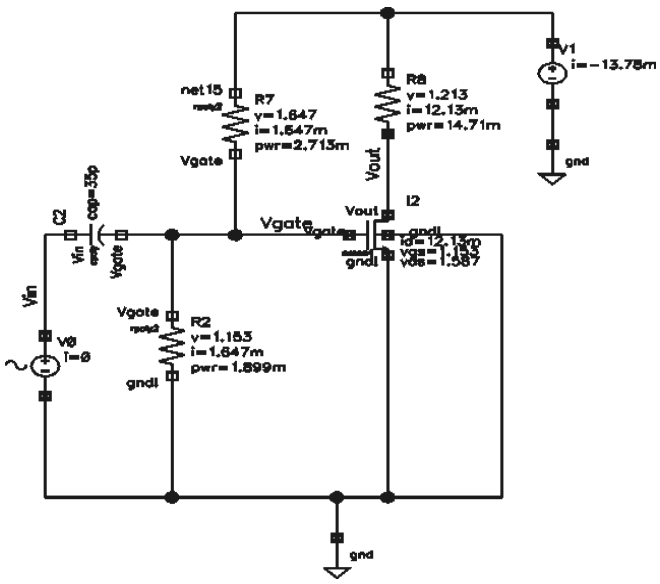
Постояннотоковият анализ се използва за определяне на работна точка или задаване на постояннотокова развивка по температура, промяна на параметър на елемент или моделен параметър.

Например, за да се направи постояннотоков анализ, могат да се използват следните стъпки:

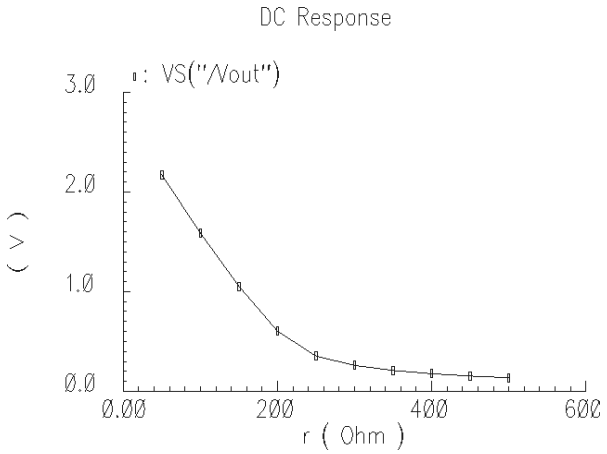
1) Избира се **Analyses** ⇒ **Choose** ⇒ **dc**;
2) Маркира се **Save DC Operating Point**;
3) Селектира се **Component Parameter** и се избира елемента R8 (вж. фиг. 8), чийто параметър **r** ще се променя от 50 до 500Ω, със стъпка 50Ω (фиг. 12).
4) Натиска се бутонът **OK** и се стартира симулацията.
5) За визуализиране на резултатите за работната точка на схемата се използва командата **Results** ⇒ **Annotate** ⇒ **DC Operating Points**. Резултатът от анализа се визуализира в схематичния редактор и е показан на фиг. 13.
6) Резултатите от развивката по параметър на R8 се визуализират като се използва командата **Results** ⇒ **Direct Plot** ⇒ **dc** избира се възелът "**Vout**" и се натиска **Esc** от клавиатурата. Резултатите от този анализ са показани на фиг. 14.



Фиг. 12. Задаване на **dc** анализ с развивка по параметър на елемент

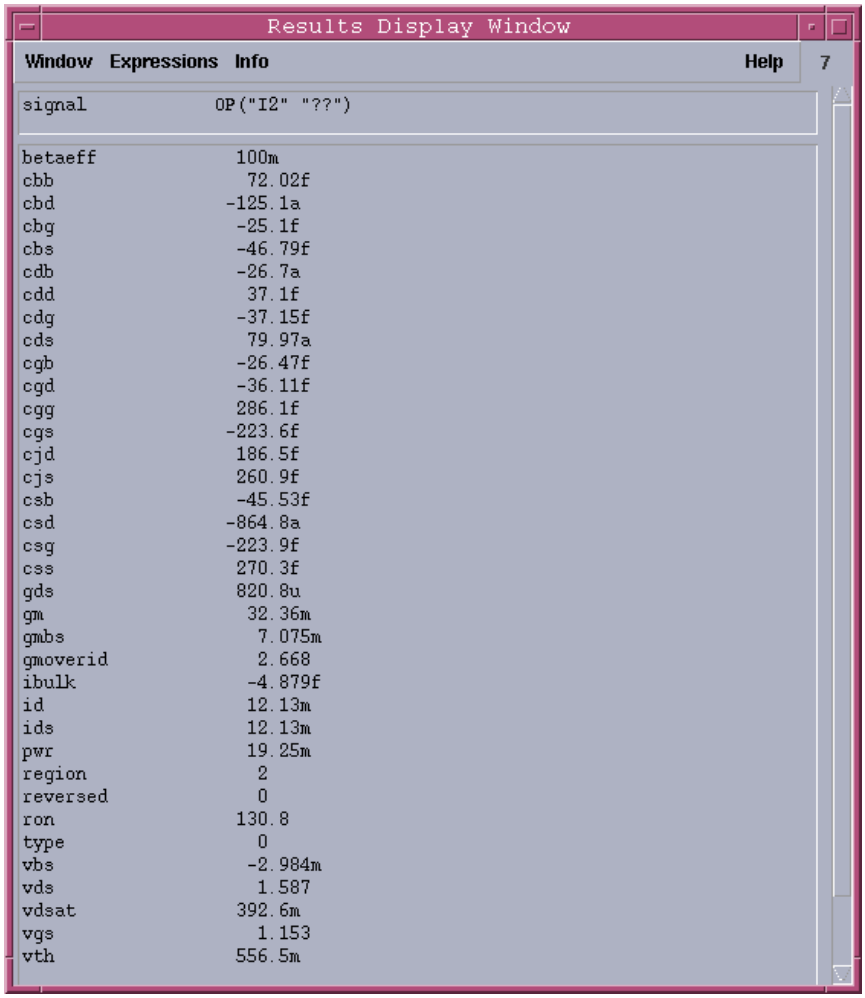


Фиг.13. Постояннотоков режим на схемата от фиг. 8



Фиг. 14. Резултати от **dc** анализ на схемата с R_8 като параметър

При постояннотоковия анализ може да се визуализират и стойностите на параметрите на елемент за дадена работна точка. За целта се задава командата **Results** ⇒ **Print** ⇒ **Operating Point**, след което се избира желаният елемент от схемата. Стойностите на параметрите на NMOS транзистора (**nmos4**) са показани на фиг. 15.



Фиг. 15. Стойности на параметрите на транзистора **nmos4** за дадената работна точка

4. Честотен анализ (ac).

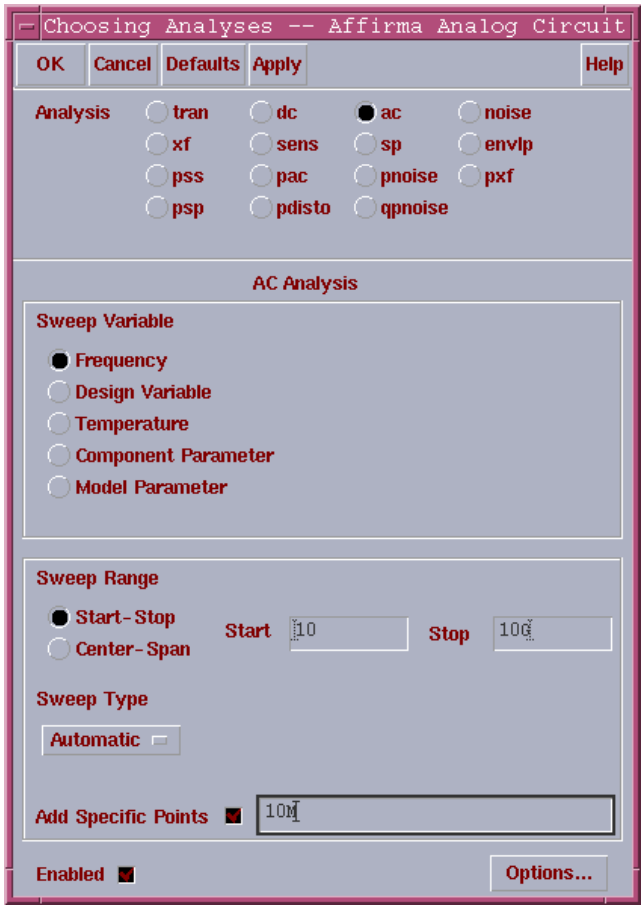
За задаване на честотен анализ се използва **Analyses** ⇒ **Choose** ⇒ **ac** от менюто на **ADE**. Задаването на анализа е показано на фиг. 16.

Стъпките за задаване на **ac** анализ са:

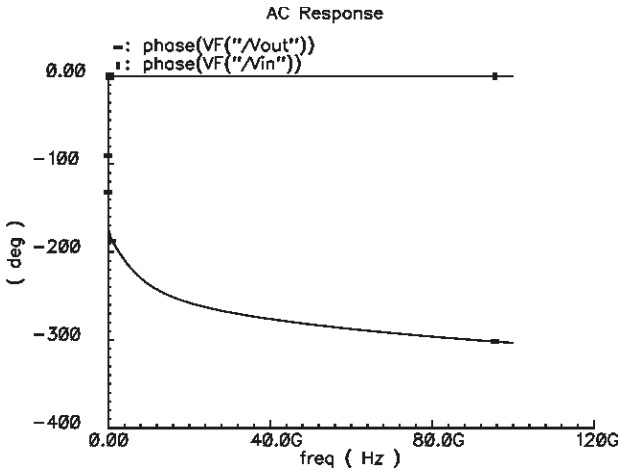
1) Избира се **ac**.
2) Задава се честотата като **Sweep Variable** и обхват на изменение (**Sweep Range**) от 1Hz до 100GHz. Задава се **Sweep Type** - **Automatic**.
3) Натиска се бутонът **OK** и се стартира симулацията.
4) Резултатите се визуализират като се използва:

– **Results** ⇒ **Direct Plot** ⇒ **AC Phase**, посочват се възлите “**Vin**” и “**Vout**” и се натиска **Esc**. Резултатите от анализа са показани на фиг. 17.

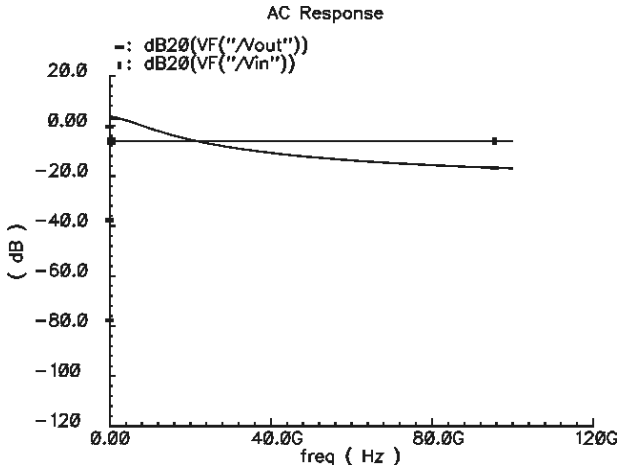
– **Results** ⇒ **Direct Plot** ⇒ **AC db20**, посочват се възлите “**Vin**” и “**Vout**” и се натиска **Esc** (фиг. 18).



Фиг. 16. Прозорец за задаване на **ac** анализ



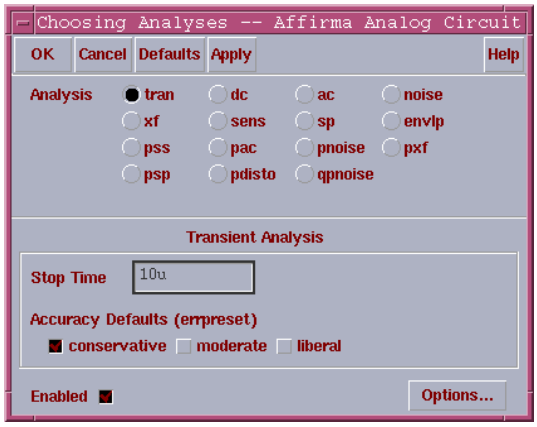
Фиг. 17. Фаза на напреженията във възли “**Vin**” и “**Vout**”



Фиг. 18. Амплитуда в dB на напреженията във възли “**Vin**” и “**Vout**”

5. Времеви анализ (tran).

За задаване на времеви анализ се използва **Analyses** ⇒ **Choose** ⇒ **tran**. Прозорецът за избор е показан на фиг. 19.

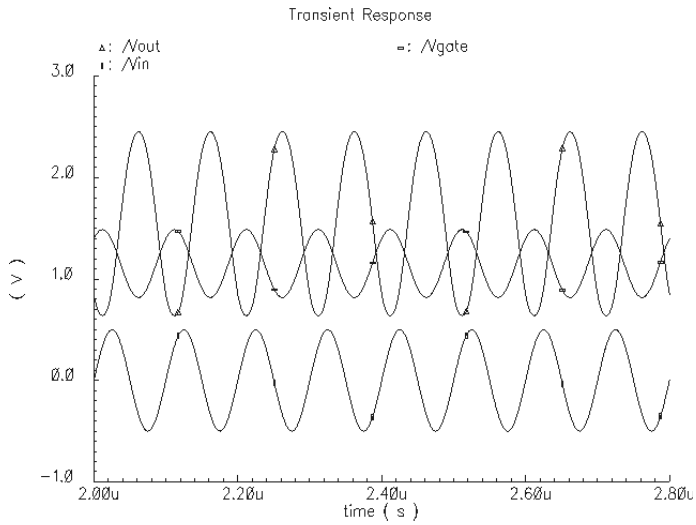


Фиг. 19. Задаване на времеви анализ

Задава се **Stop Time,** с което се определя края на интервала за извършване на симулацията. Отбелязва се точността, с която да се изпълни анализът и се маркира **Enable.** В **Options** могат да се задават допълнителни условия – стъпка на изчисляване, интеграционен метод за пресмятане и др.

За времеви анализ могат да се използват следните стъпки:

1) Избира се анализът **tran**.
2) Задава се **Stop Time** – 10μs (фиг. 19).
3) Натиска се бутонът **OK** и се стартира симулацията.



Фиг. 20. Визуализиране на резултата от времевия анализ на схемата

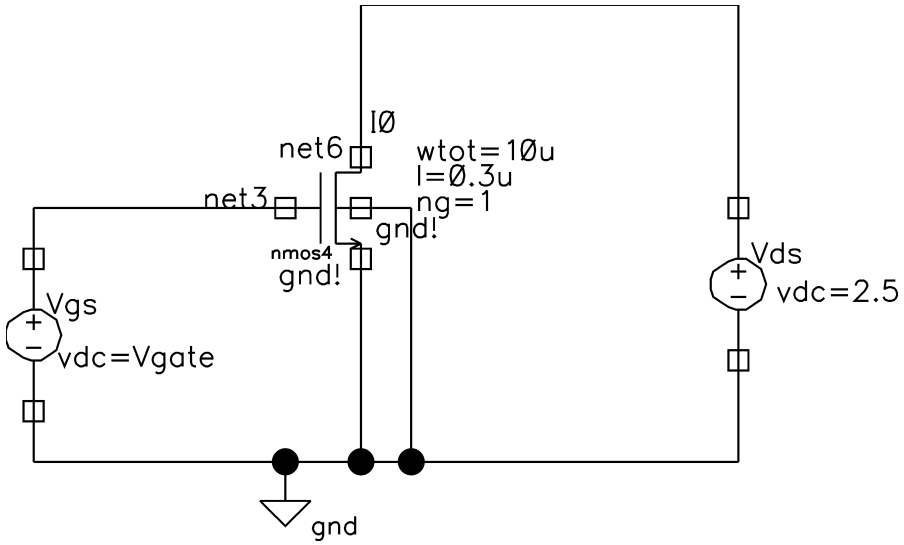
4) Резултатите от симулацията се визуализират чрез:

Results ⇒ **Direct Plot** ⇒ **Transient Signal**, маркират се възлите “**Vin**”, “**Vgate**” и “**Vout**”, след което се натиска **Esc** от клавиатурата. Резултатът от анализа е показан на фиг. 20.

Упражнение № 2

Параметричен и шумов анализ

I. Параметричен анализ (Parametric Analysis).



Фиг. 1. Схема за изследване на изходните характеристики на MOS транзистор

1. Задаване на променлива.

Изчертава се схемата от фиг. 1 в прозореца на схемния редактор (**Virtuoso Schematic Composer**). Транзисторът **nmos4** е реален и се взима от библиотеката, обвързана със съответната технология (**PRIMLIB** за AMS 0.35μm CMOS технология). Стойностите на ширината и дължината на канала на транзистора се оставят по подразбиране – w=10μm и l=0.3μm. Задаващите източници на напрежение (**vdc**), които се използват при симулация, се извикват от библиотеката **analogLib**. В полето **DC voltage** на входния източник, задаващ напрежението гейт-сорс, се присвоява променлива $\mathbf{V_{gate}}$. Тази променлива се копира в прозореца на средата за аналогова симулация **ADE** (**Analog Design Environment**) като от менюто **Variables** се избира командата **Copy from cellview**, а стойността ѝ се задава от командата **Edit** на същото меню.

2. Задаване на постояннотоков анализ (dc).

За задаване на постояннотоков анализ от менюто **Analyses** се избира команда **Choose** и се маркира **dc.** За получаване на изходните характеристики на MOS транзистора е необходима развивка по напрежението дрей-сорс. За целта като **Component parameter** се избира източникът, задаващ напрежението дрей-сорс, и се посочва диапазонът на изменение на стойностите на параметъра му **dc** от 0 до 3.3V.

В системата за автоматизирано проектиране CADENCE по подразбиране се запазват стойностите на напреженията във възлите на симулираната схема. За да могат да се визуализират токовете, те трябва предварително да се запазят като в менюто **Outputs** се избере команда **Save all** и се маркира опцията **Select all DC/Transient terminal currents** (при използване на симулатора SpectreS).

3. Задаване на параметричен анализ (Parametric Analysis).

Формата за задаване на параметричен анализ се извиква от менюто **Tools** чрез команда **Parametric analysis**. Като изменящ се параметър се задава напрежението гейт-сорс чрез променливата **Vgate**. Стойностите й могат да бъдат например от 0 до 2.5V с 5 стъпки на изменение.

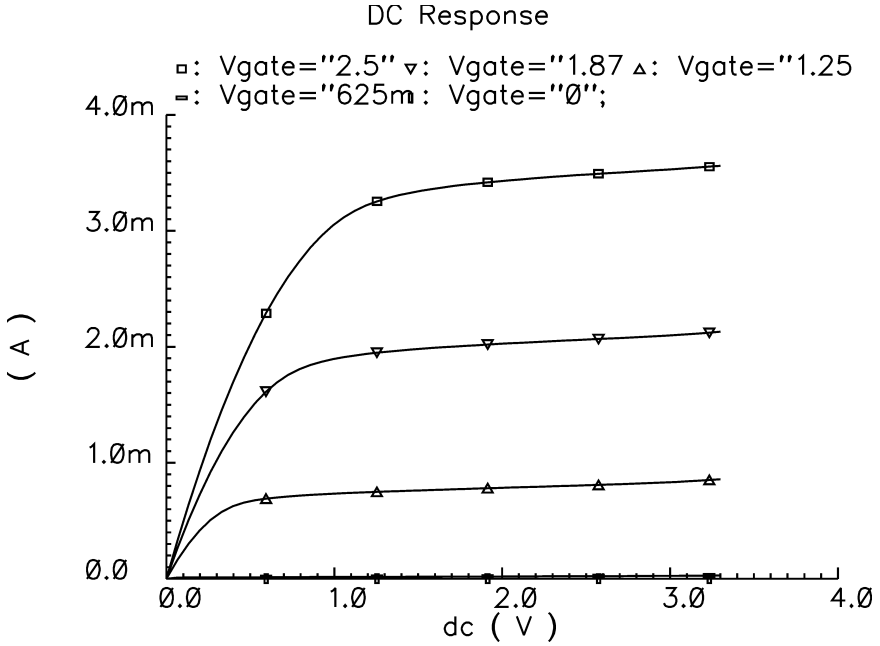
4. Стартиране на симулация.

Симулацията се стартира от формата за параметричен анализ като се избира командата **Start** от менюто **Analyses**.

5. Визуализиране на резултатите.

За визуализиране на изходния ток след приключване на симулацията, от менюто **Results** се избира **Direct Plot** ⇒ **dc** и в прозореца на схематичния редактор се селектира дрейновият пин на транзистора (фиг. 2). Той може да се изобрази графично и ако това е зададено преди стартирането на симулацията (**Outputs** ⇒ **To Be Plotted** ⇒ **Select On Schematic**).

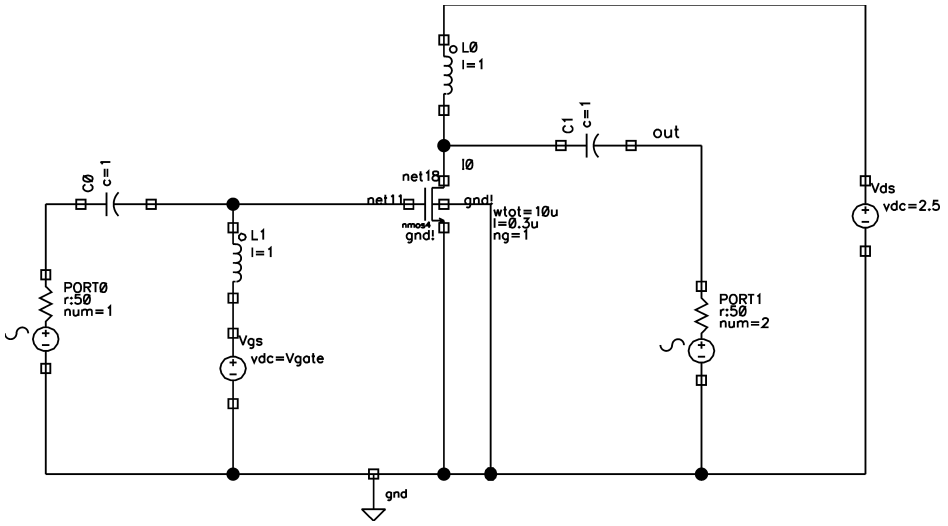
Като променливи при параметричен анализ могат да се задават параметри на транзистора (ширина **w** и дължина на канала **l**), а така също и параметри на други елементи, честота, температура и др.



Фиг. 2. Изходни характеристики на NMOS транзистор (w=10μm, l=0.3μm, брой гейтове=1)

II. Коефициент на шум (Noise Figure)

За изследване на коефициента на шума на транзистор се ползва схемата от фиг. 3. Необходимо е да се поставят на входа и изхода източници, използвани при изследване на шум (портове). Такива са компонентите **psin** от библиотеката **analogLib**. Във формата за задаване на параметри на всеки един от тях се попълва единствено номер на порта (**Port number**), който трябва да бъде цяло положително число, например за PORT0 - 1, а за PORT1 - 2. Бобините (**ind**) и кондензаторите (**cap**) са идеални елементи и се взимат от **analogLib**. Задават им се големи стойности – за капацитета (**mF ÷ F**) и за индуктивността (**mH ÷ H**).



Фиг.3. Схема за определяне на коефициента на шума на NMOS транзистор

1. Задаване на анализ на разпределени параметри (sp).

В прозореца на **ADE** се избира **Analyses** ⇒ **Choose** ⇒ **sp**. Задава се честотната област, в която ще се изследва коефициента на шум и се посочват източниците на шум – входен и изходен порт (фиг. 4).

2. Визуализиране на NF.

От менюто **Results** се избира командата **Direct plot** ⇒ **S-parameter...** Във формата за визуализиране на резултатите от **sp** анализа се маркира параметърът **NF**. Резултатът може да се представи като амплитудна стойност или в dB. Изчертаването на графиката става след натискане на бутона **Plot**.

Освен **NF** симулаторът SpectreS дава възможност да се изследват и други шумови характеристики като: еквивалентен входен и изходен шум, квадратичен входен и изходен шум и др. Тези шумови характеристики се изследват чрез **noise** анализ. Пример за задаване на такъв тип анализ е показан на фиг. 5.

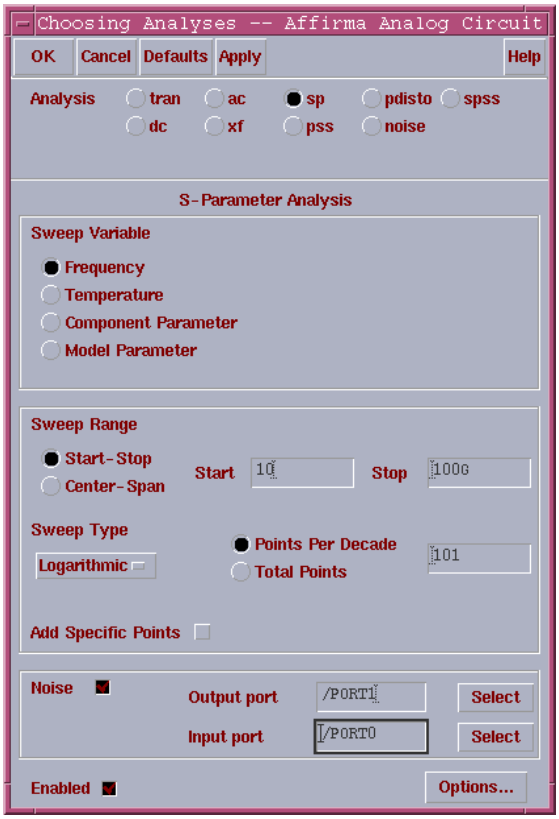
3. Визуализиране на резултатите при изследване на шум и разпределението му върху всички елементи в схемата (фиг. 6).

Print ⇒ Noise Summary

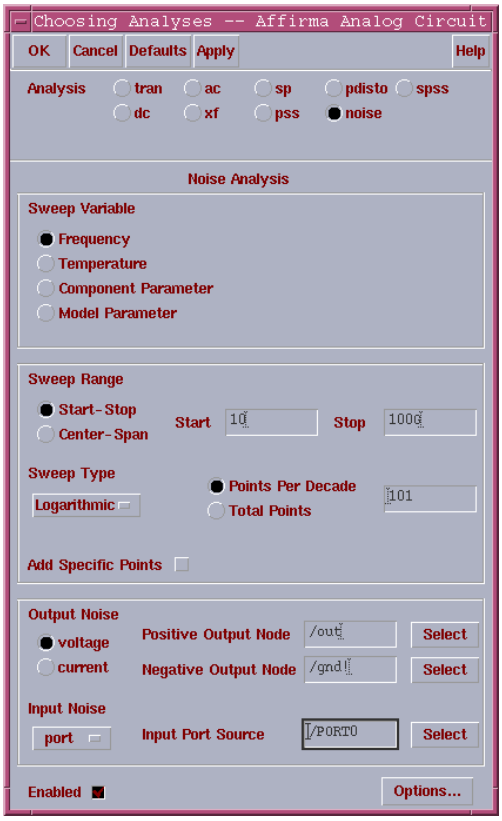
Избира се: **Frequency spot:** 2GHz

Filter ⇒ include: all types

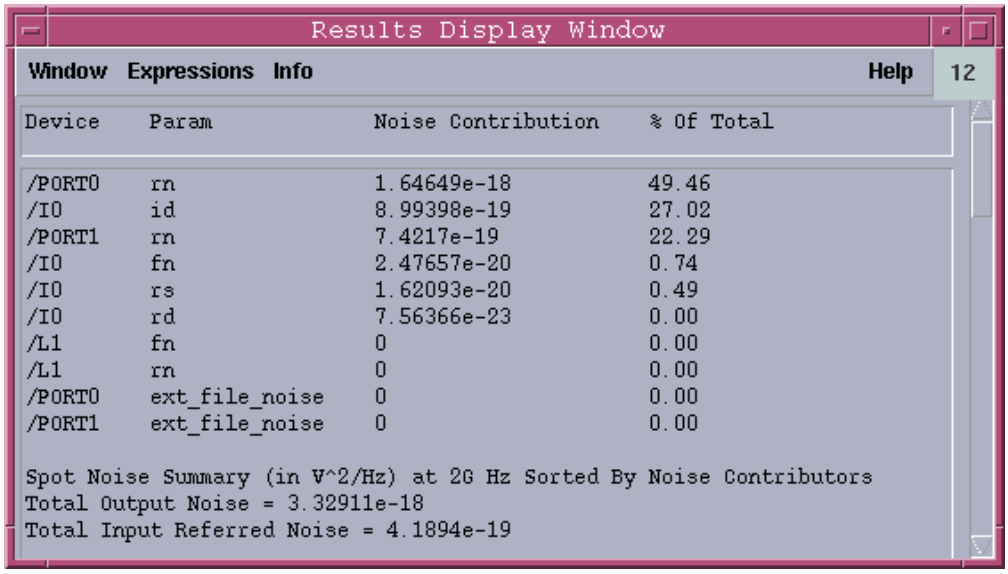
Truncate&Sort: 10



Фиг. 4. Настройка на **sp** анализ

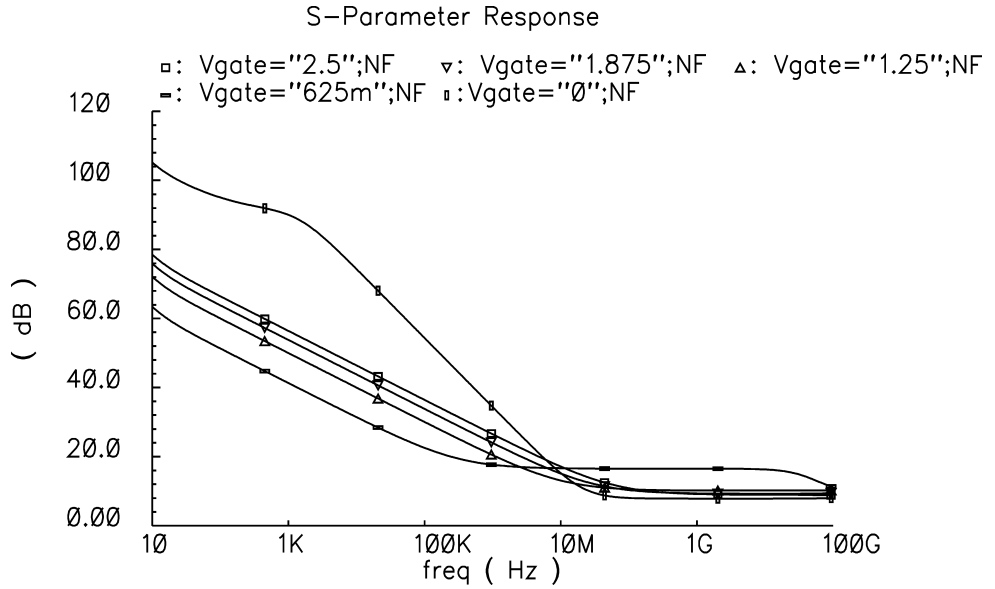


Фиг. 5. Настройка на **noise** анализ

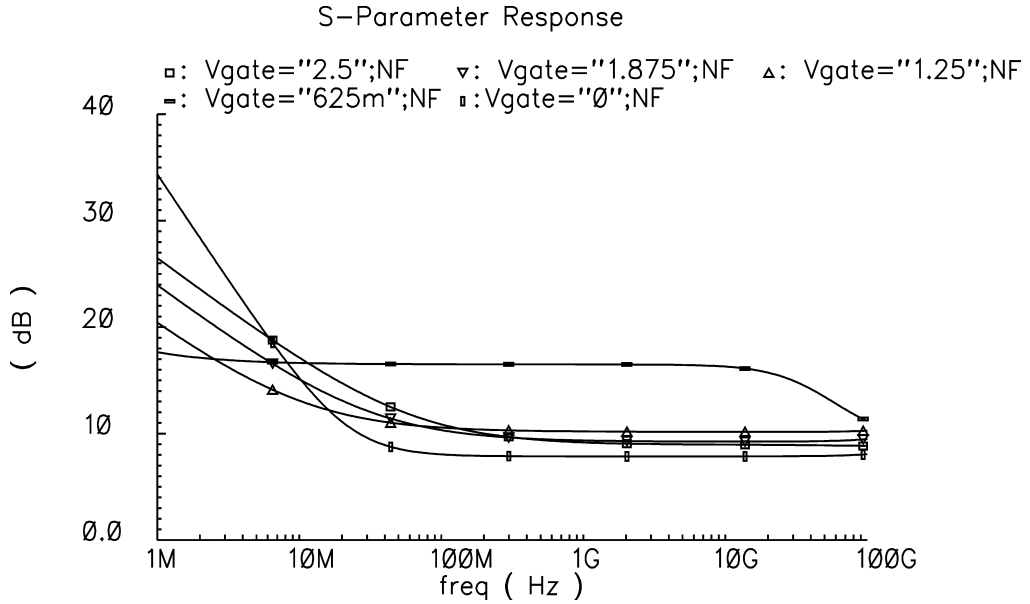


Фиг. 6. Разпределение на шума, генериран от елементите в схемата за изследване на MOS транзистор

За изследване на коефициента на шума при различно гейтово напрежение се стартира параметричен анализ с променлива **Vgate**, както е показано в точка 1. Резултатите от такъв анализ за целия зададен честотен обхват са показани на фиг. 7, а само за високи честоти на фиг. 8.



Фиг. 7. Коефициент на шума на MOS транзистор за целия честотен обхват при различни стойности на променливата **Vgate**



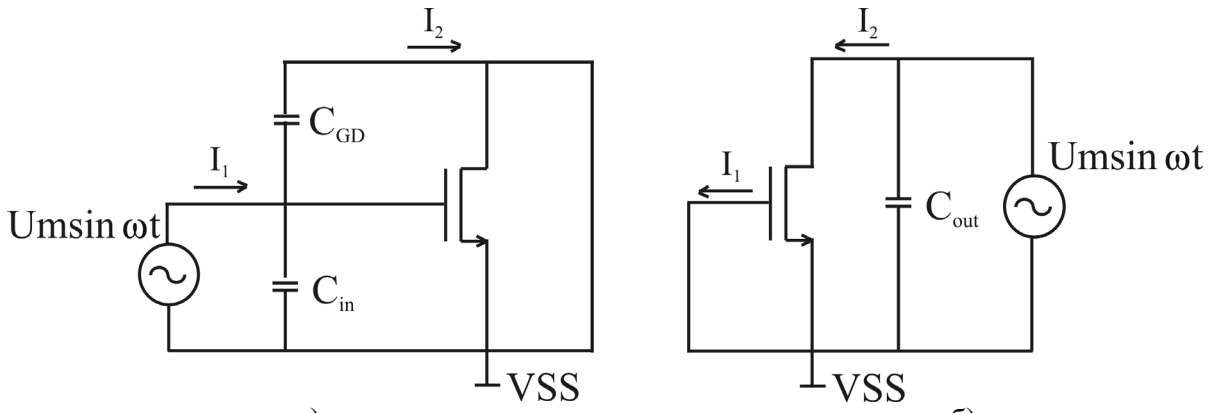
Фиг. 8. Коефициент на шума на MOS транзистор за високи честоти при различни стойности на променливата **Vgate**

Шумовите анализи показват стабилността на схемата по отношение на смущаващи сигнали, които могат да бъдат външни или генерирани от елементите в схемата. Има възможност да се изследва влиянието на параметрите на транзистора, като брой гейтове, дължина **l** и ширина **w** на канала, площ и други параметри върху шумовите характеристики.

Упражнение № 3

Изследване на параметри и характеристики на MOS транзистор

I. Теоретична постановка.



Фиг. 1. Схеми за определяне на входен и изходен капацитет на MOS транзистор

За определяне на входния **Cin** и изходния **Cout** капацитет, се използват схемите от фиг. 1а и б. На входа (гейта) на транзистора се подава сигнал чрез генератор на напрежение при свързан на късо изход (фиг.1а). Избраният постояннотоков режим $U_{DS}=U_{GS}=0$ (Ids=0), премахва влиянието на статичните параметри върху измерването. По аналогичен начин се определя и изходният капацитет с помощта на схемата от фиг.1б. Постояннотоковият режим е същия, но сигналът се подава към изхода (дрейна) на транзистора при даден на късо вход. Капацитетите **Cin** и **Cout** се определят съответно по формули (1) и (2):

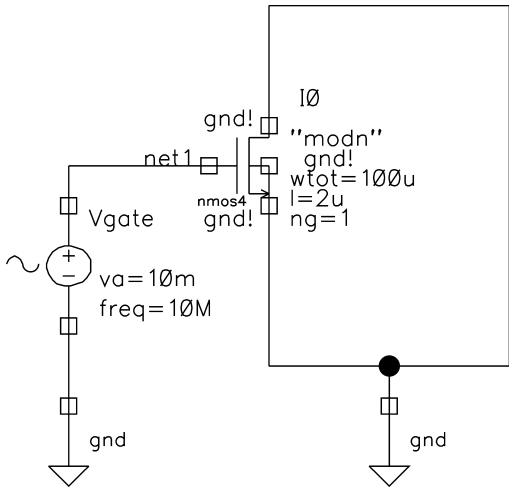
$$Cin = \frac{I1 - I2}{2\pi f Um} \qquad (1)$$

$$Cout = \frac{I1 - I2}{2\pi f Um} \qquad (2)$$

1) Симулационно определяне на Cin и Cout

Изчертава се схемата показана на фиг. 2 в прозореца на **Virtuoso Schematic Composer**.

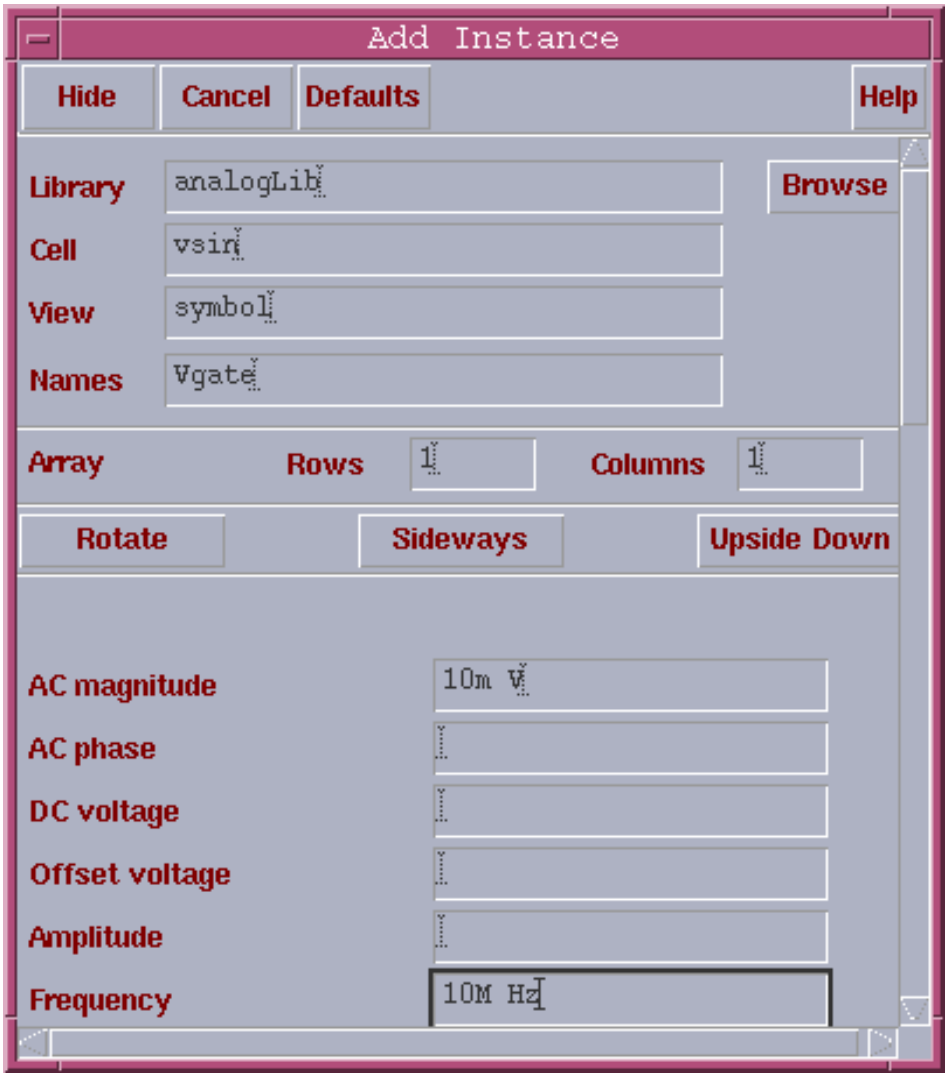
Използваният транзистор е **nmos4** от библиотеката **PRIMLIB** (за 0.35μm CMOS технология). Неговите параметри са:



Фиг. 2. Симулационно изследване на входния капацитет на MOS транзистор **Cin**

Ширина на канала: **w**=100μm
Дължина на канала: **l**=2μm
Брой гейтове: **NG**=1

Източникът на напрежение **vsin**, използван за симулациите, се намира в библиотеката **analogLib**. Стойностите му се задават, както е показано на фиг. 3.

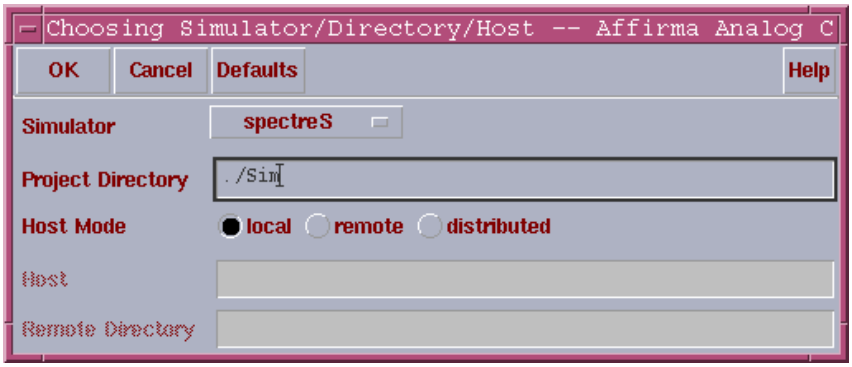


Фиг. 3. Задаване на параметрите на източник **vsin**

II. Симулации.

1. Избор на симулатор.

От прозореца на **Virtuoso Schematic Composer** се избира **Tools** ⇒ **Analog Environment,** след което се появява прозорецът на средата за аналогова симулация **ADE**. За работа със симулатора SpectreS, от прозореца на **ADE** се избира командата **Setup** ⇒ **Simulator/Directory/Host**. Появява се меню за избор на симулатор. В него се посочва името на избрания симулатор и пътя до директория, в която ще се съхраняват резултатите от симулацията (фиг. 4).



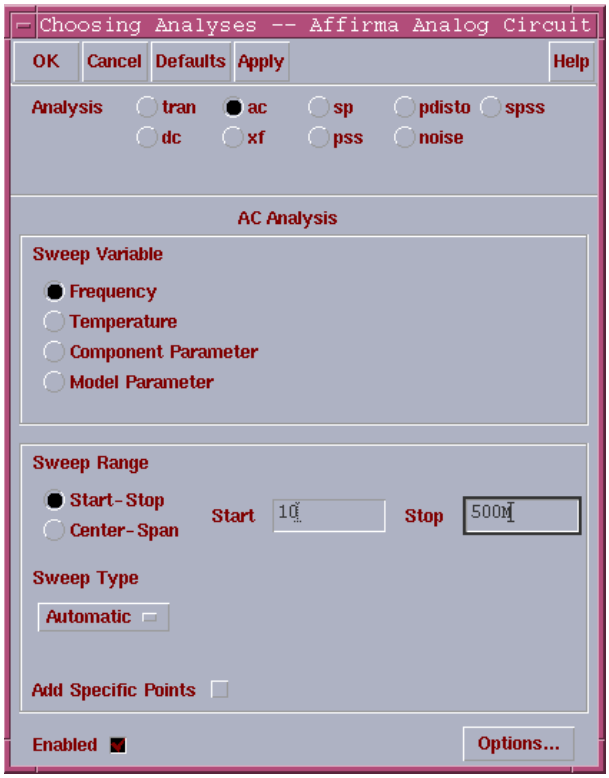
Фиг. 4. Меню за избор на симулатор

2. Задаване на AC анализ.

1) От прозореца на **ADE** се избира **Analyses ⇒ Choose ⇒ ac**. Настройките за честотния анализ са показани на фиг. 5.
2) Задават се следните параметри за анализа:

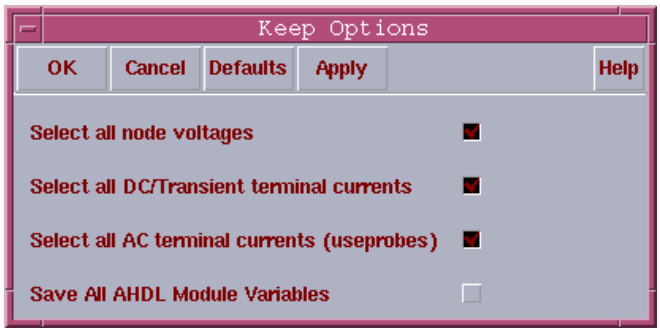
Sweep Variable ⇒ честота
Sweep Range ⇒от 10Hz до 500MHz
Sweep Type ⇒ **Automatic**

3) Потвърждава се с **OK.**
4) От прозореца на **ADE** се избира **Outputs ⇒ Save all ⇒ AC currents** (фиг. 6).
5) Стартира се симулацията.
6) Визуализиране на резултатите от симулацията.



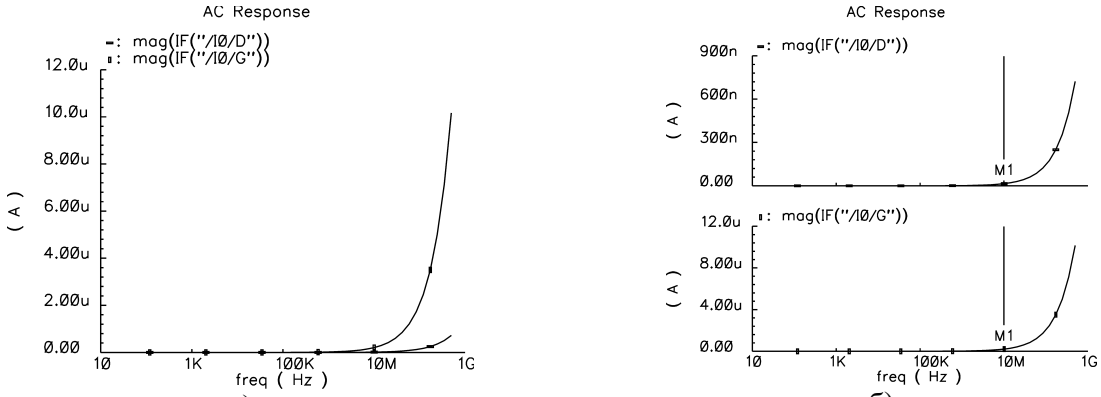
Фиг. 5. Настройки за **ac** анализ

За целта се използва командата **Results ⇒ Direct Plot ⇒ AC Magnitude**. В схемата се селектират токовете в гейта и дрейна на транзистора и се натиска бутонът **Esc** от клавиатурата (фиг. 7a).



Фиг. 6. Запазване на резултатите за стойностите на токовете преди симулация

За визуализиране на графичните резултати в отделни прозорци (фиг. 7b), се избира **Switch Axis Mode** (7^ми бутон от лявата страна на прозореца за визуализиране на резултатите от анализа).

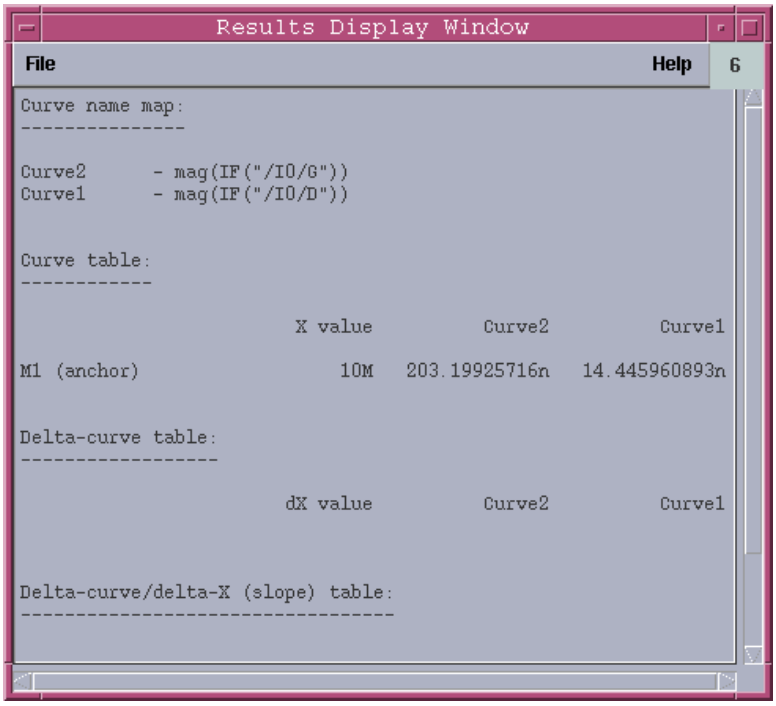


а) б)

Фиг. 7. Амплитуда на дрейновия и гейтовия ток

За отчитане на токовете при дадена честота, от прозореца с резултатите се избира командата **Markers** ⇒ **Vertical Markers**. Задава се желаната честота (например 10MHz) и се натискат последователно бутоните **Apply** и **Display Intercept Data.**

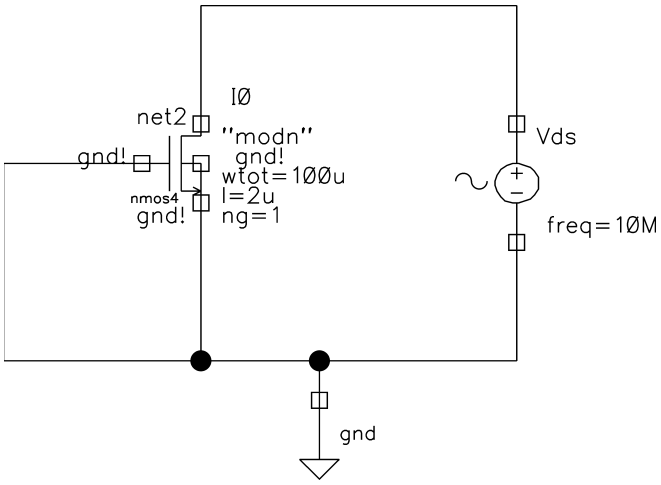
Фиг. 8. Задаване на маркер



Фиг. 9. Визуализиране на резултатите от анализа за схемата от фиг. 2 с помощта на маркер

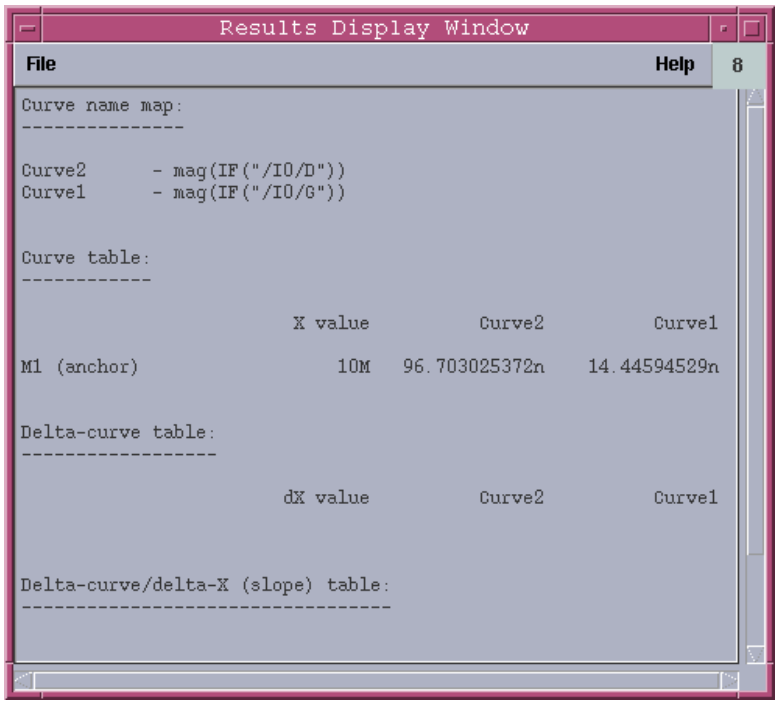
Входният капацитет може да се изчисли чрез израз (1) като се използват числените стойности на токовете, отчетени по описания по-горе начин.

За определяне на изходния капацитет **Cout**, се използва схемата показана на фиг. 10 и по аналогичен начин се повтарят стъпките за извършване на симулация.



Фиг. 10. Симулационно изследване на изходния капацитет **Cout**.

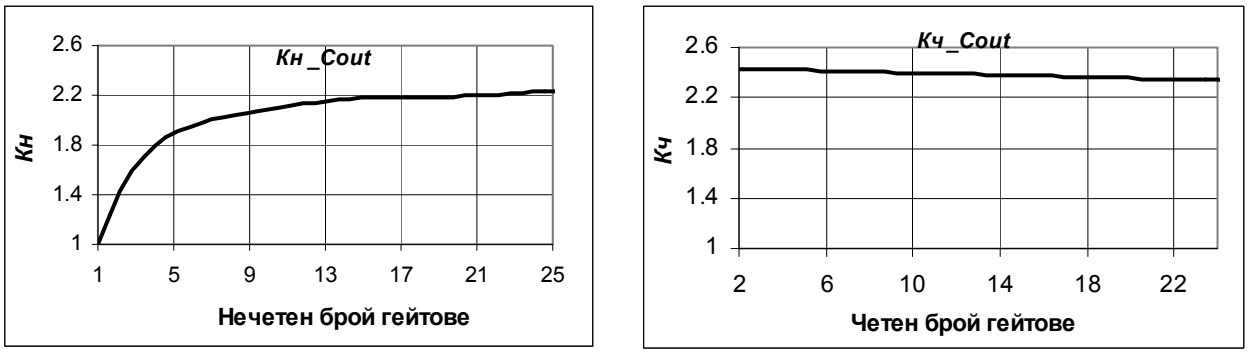
За честота 10MHz, стойностите на дрейновия и гейтовия ток са показани на фиг. 11.



Фиг. 11. Визуализиране на резултатите от анализа за схемата от фиг. 10 с помощта на маркер

Изходният капацитет може да се изчисли чрез формула (2) и отчетените при симулацията стойности на токовете. Той може да се изчисли приблизително и за транзистори с различен брой гейтове

(например 5 и 10) като се използват нормираните характеристики, предложени на фиг. 12а и б.



а) б)

Фиг. 12. Нормирани характеристики за различен брой гейтове.

За построяването на нормираните характеристики са изполвани коефициенти за четен и нечетен брой гейтове $К_н$ и $К_ч$, дефинирани посредством формули (3) и (4):

$$K_н = \frac{C_н}{C_1} \tag{3}$$

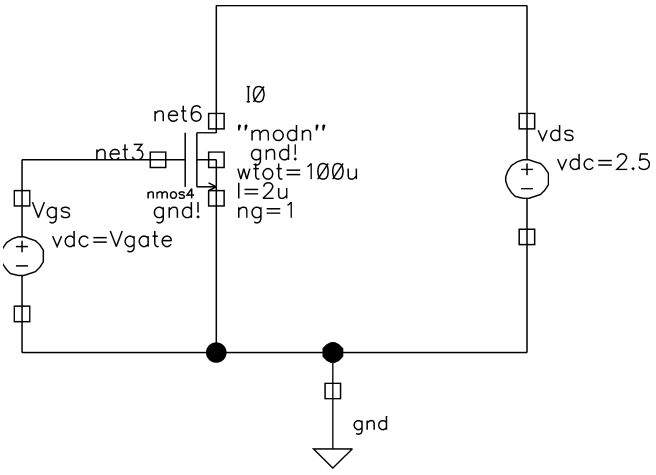
$$K_ч = \frac{C_{н+1}}{C_1} \tag{4}$$

където $C_н$ е капацитета на транзистор с нечетен брой гейтове, $C_{н+1}$ - капацитет на транзистор с четен брой гейтове, C_1 – капацитет на транзистор с един гейт.

Получените по този начин резултати могат да се сравнят с тези от симулацията.

3. Изходна характеристика на MOS транзистор.

В прозореца на **Virtuoso Schematic Composer** се начертава схемата, показана на фиг. 13. Транзисторът **nmos4** има същите параметри като тези от фиг. 2 и фиг. 10.



Фиг. 13. Схема за определяне на изходната характеристика

1) Задаване на променлива.

В прозореца **Edit Object Properties** на източника **Vgs** в полето **DC voltage** се задава име на променливата **Vgate**. Същото име на променлива се задава и в **ADE**, в меню **Variables** ⇒ **Edit.**

2) Задаване на постояннотоков анализ (dc).

В **ADE** се избира команда **Analyses** ⇒ **Choose** ⇒ **dc**. Като **Component parameter** се посочва източникът **Vds** и се задава обхватът на изменение на стойностите му от 0 до 2.5V. За симулацията е необходим дрейновият ток на транзистора **Id**. За неговото визуализиране, както и това на всички желани токове, е задължително те предварително да се запазят. От прозореца на **ADE** се използва: **Outputs ⇒ Save all ⇒ DC currents.**

3) Задаване на параметричен анализ.

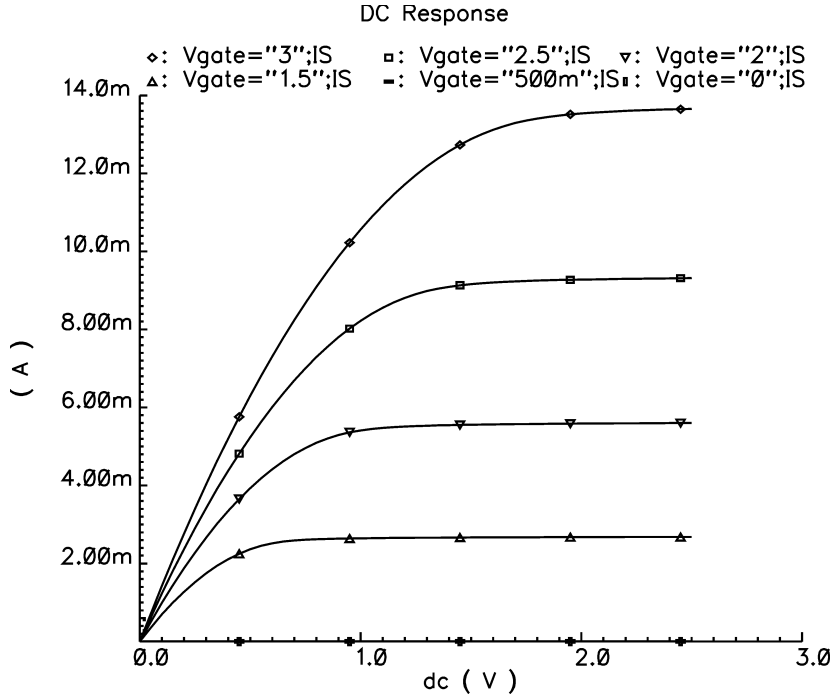
В **ADE** се избира **Tools** ⇒ **Paramertric analysis** и се задава като параметър гейтовото напрежение Vgs чрез променливата Vgate. Стойностите й могат да бъдат например: **Vgate** = 0, 0.5V, 1.5V, 2V, 2.5V и 3V.

4) Стартиране на симулация.

Симулацията се стартира от прозореца за параметричен анализ (**Analysis ⇒ Start**)

5) Визуализация на резултатите от анализа.

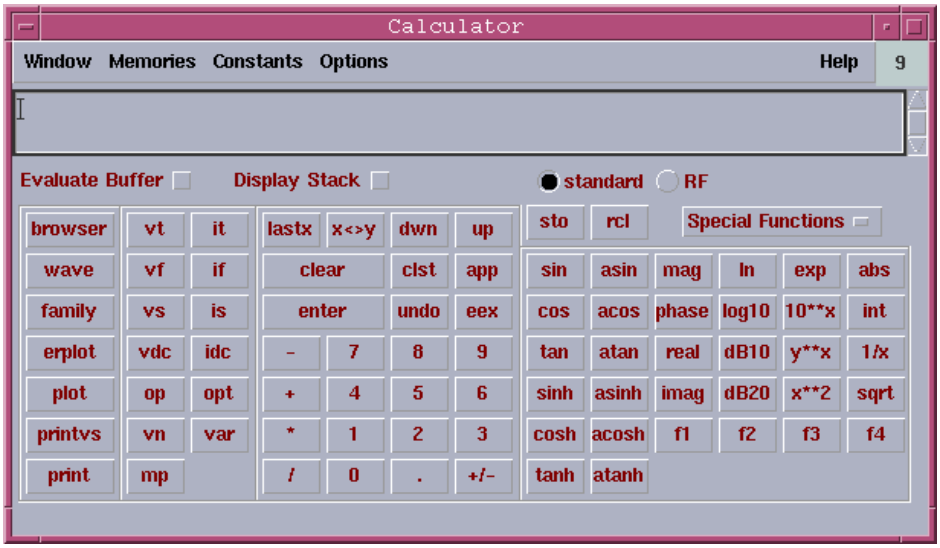
Когато симулацията приключи, се избира дрейновият ток **Id**. Използва се командата **Results ⇒ Direct Plot ⇒ dc.** Той може да се изобрази графично и ако е отбелязан преди пускането на симулацията.



Фиг. 14. Изходни характеристики на NMOS транзистор (w=100μm, l=2μm, брой гейтове=1)

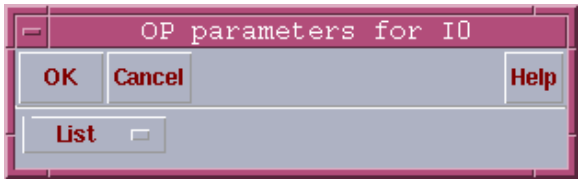
4. Проводимост gm.

За изчисляването на проводимостта се използва помощното средство калкулатор (**Calculator**). Стартира се от менюто **Tools** на **ADE**.



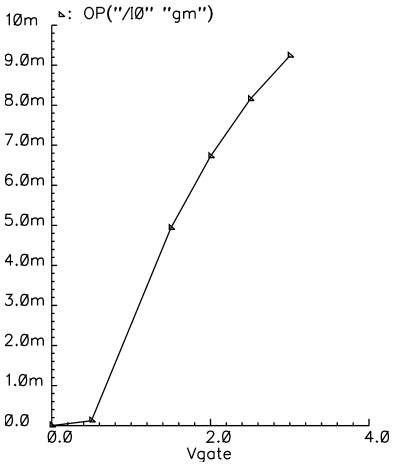
Фиг. 15. Калкулатор

Натиска се бутонът “**op**”, разположен от лявата страна на калкулатора. Появява се прозорецът за визуализиране на параметрите от постояннотоковия анализ. Маркира се транзисторът в схемата и се избира бутонът “**list**” (фиг. 16).

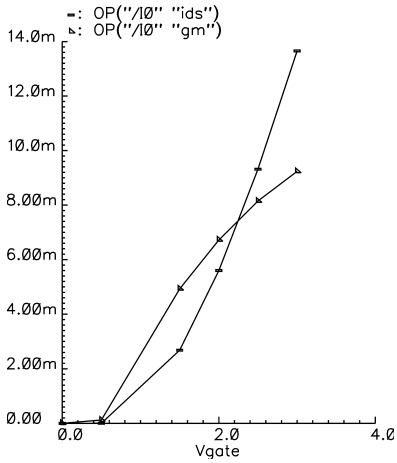


Фиг. 16. Визуализиране на постояннотокови параметри с калкулатор

От списъка с параметри се избира **gm** и натиска бутонът **plot** на калкулатора. Резултатите за проводимостта са показани на фиг. 17.



Фиг. 17. Зависимост на **gm** от напрежението в гейта **Vgate**

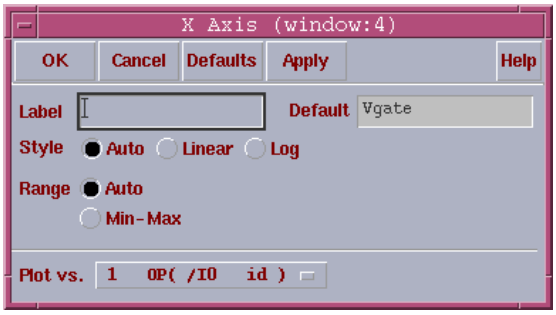


Фиг. 18. Зависимост на **gm** и **Id** от напрежението в гейта **Vgate**

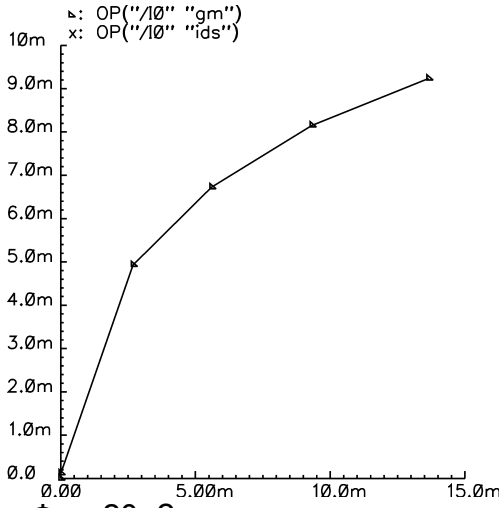
За да се визуализира **gm** като функция от дрейновия ток **Id**, трябва да се изпълнят следните стъпки:

1) Избира се дрейновият ток **Id** от списъка (**list**) и се визуализира, както е показано на фиг. 18.

2) В прозореца за визуализиране на резултатите от анализа се избира **Axes ⇒ X Axis** (фиг. 19).



Фиг. 19. Прозорец за настройки по оста X



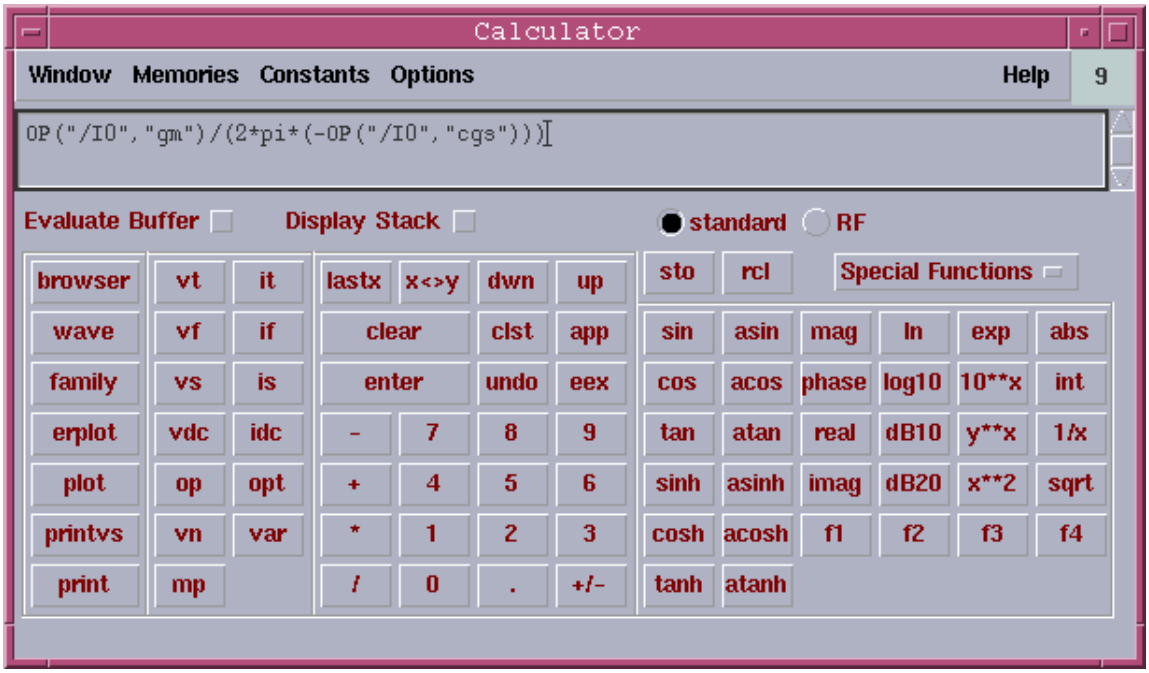
Фиг. 20. Зависимост на **gm** от дрейновия ток Id

3) Натиска се бутонът "**independent variable**" и се избира **Id** (фиг. 19). По този начин по x–оста се явява токът Id (фиг. 20).

5. Транзитна честота ft.

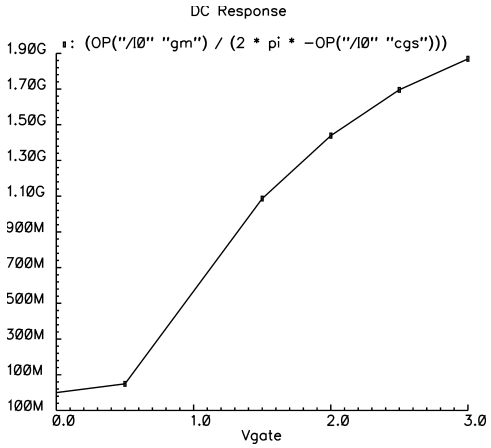
Честотата **ft** може да се изрази като се използва следният израз:

$$ft = \frac{g_m}{2\pi Cgs} \qquad (5)$$

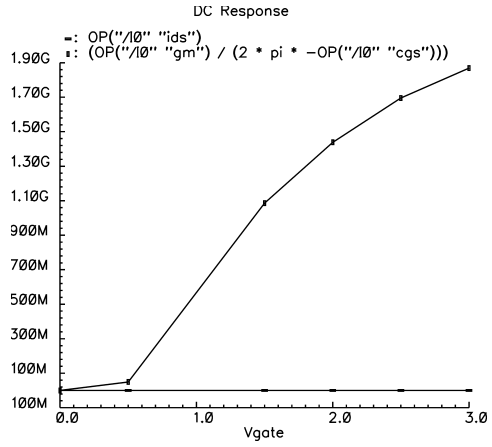


Фиг. 21. Въвеждане на израз (5) в калкулатора

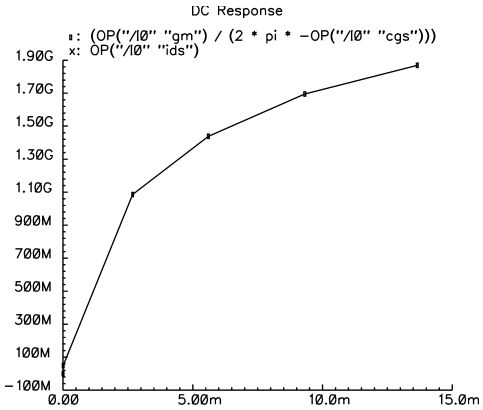
Въвежда се израз (5) в калкулатора (фиг. 21) и се визуализира транзитната честота като функция на дрейновия ток **Id**. Графичните зависимости са показани на фиг. 22, 23 и 24.



Фиг. 22. Зависимост на **ft** от напрежението в гейта **Vgate**



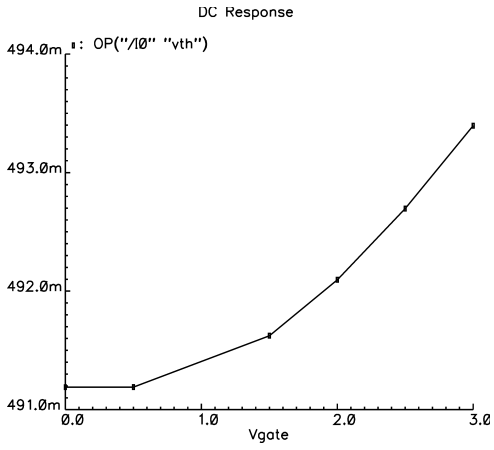
Фиг. 23. Зависимост на **ft** и дрейновия ток Id от напрежението в гейта **Vgate**



Фиг. 24. Зависимост на **ft** от дрейновия ток **Id**

6. Определяне на праговото напрежение Vth.

По аналогичен начин се изразява зависимостта на праговото напрежение **Vth** от стойността на гейтовото напрежение (фиг. 25).



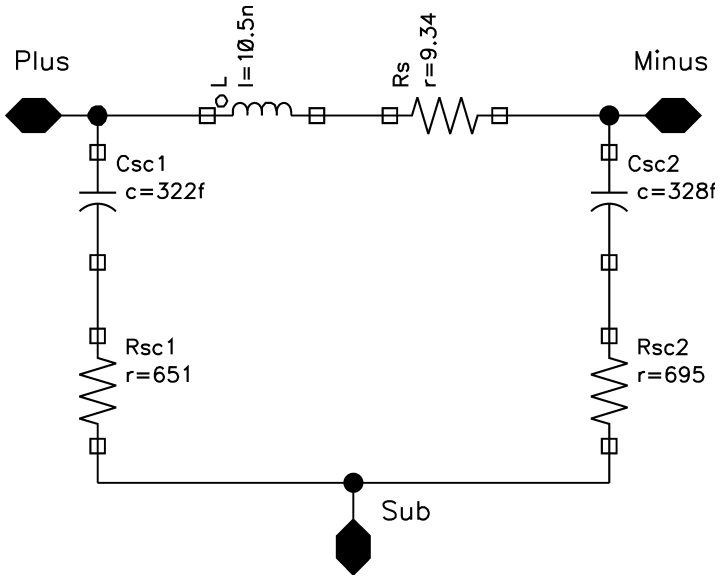
Фиг. 25. Зависимост на праговото напрежение **Vth** от гейтовото напрежение **Vgate**

Упражнение № 4

Създаване на символ

1. Въвеждане на схема.

На фиг.1 е показана теснолентовата еквивалентна схема на монолитна бобина. Всички използвани в нея елементи са идеални.

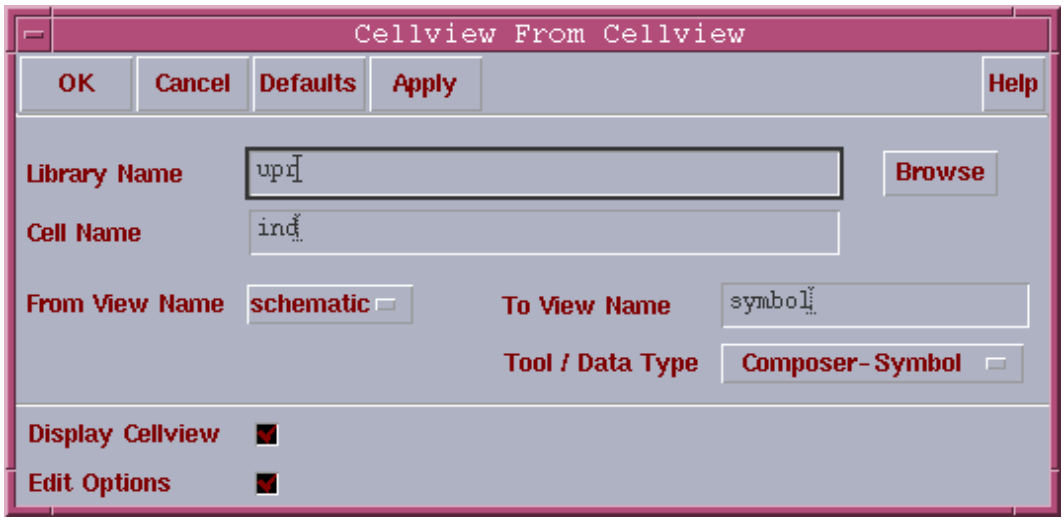


Фиг. 1. Теснолентова еквивалентна схема на бобина

За да бъде възможно създаването на символно представяне от схемно, е необходимо да се поставят пинове за входа, изхода на схемата, а също така и към подложката. Командата, която се използва е **Add ⇒ Pin.**

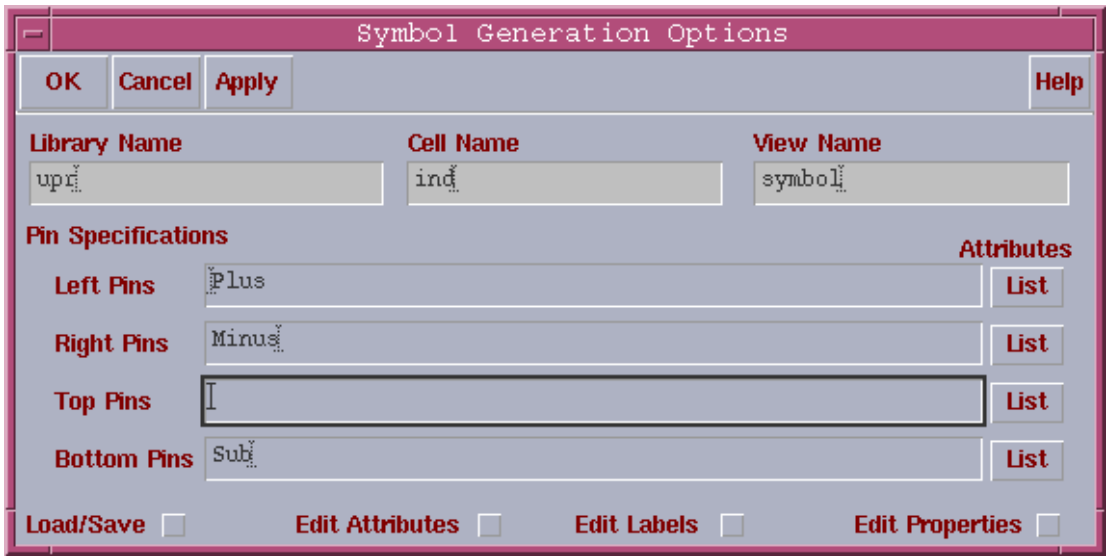
2. Създаване на символ.

От менюто **Design** на схемния редактор се избира командата **Create Cellview ⇒ From Cellview**.



Фиг. 2. Форма за създаване на символ от схема

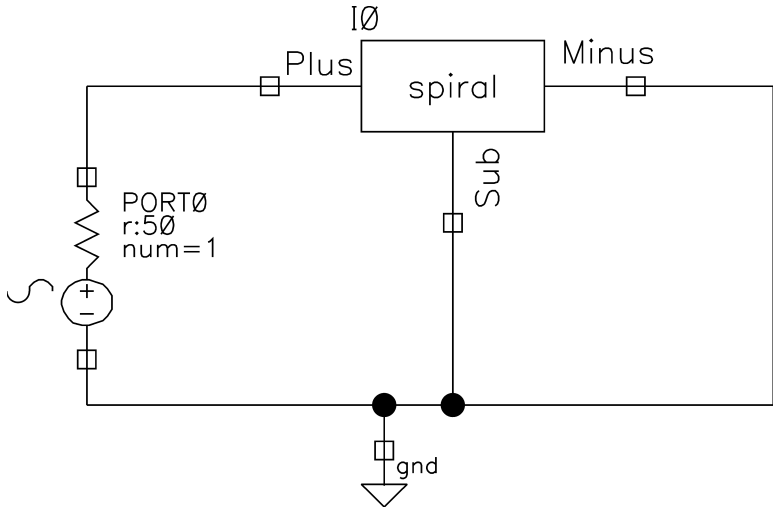
Следващата стъпка е да се обозначи разположението на пиновете в символа:



Фиг. 3. Форма за определяне разположението на пиновете

3. Изследване на качествения фактор (Q) на бобината.

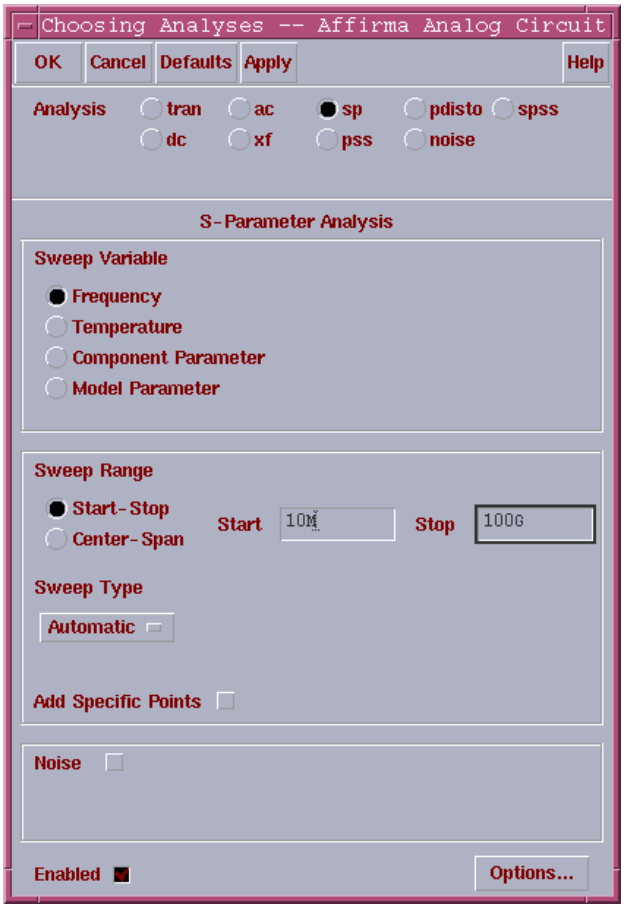
Изследването на Q фактора на бобината се извършва след изчертаване на схемата от фиг. 4. Извиква се създаденият вече символ и се свързва по показания начин. След това се задава **sp** анализ (фиг. 5).



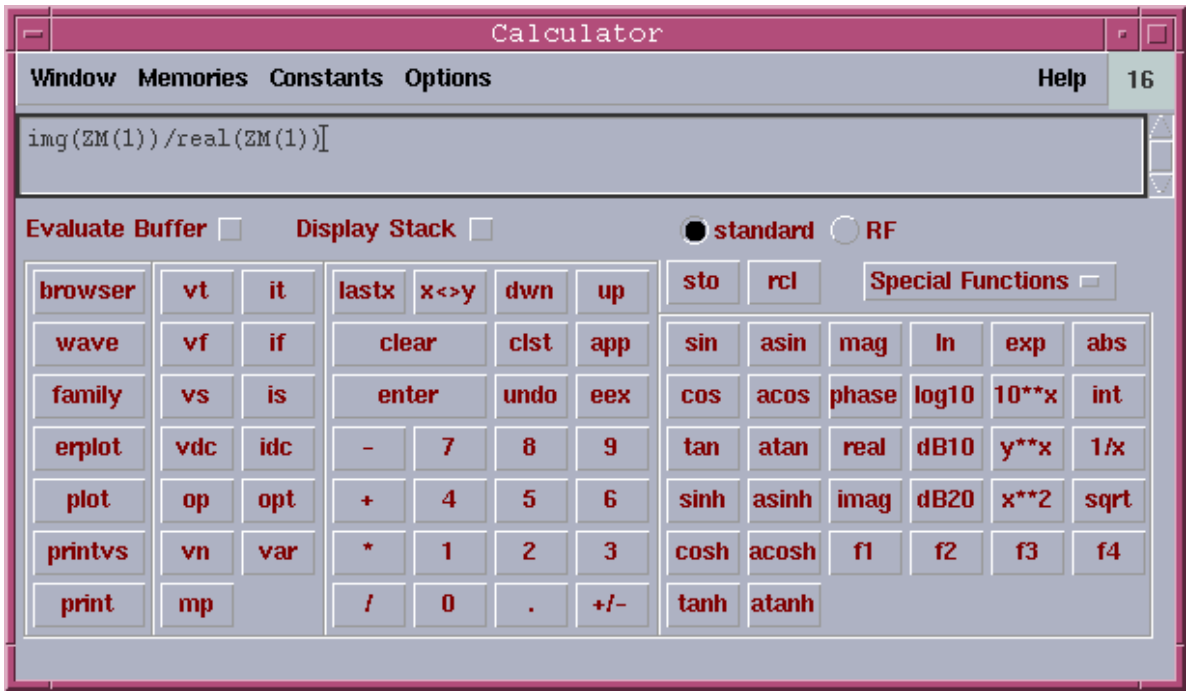
Фиг. 4. Схема за изследване на Q фактор на бобина

4. Визуализиране на резултата.

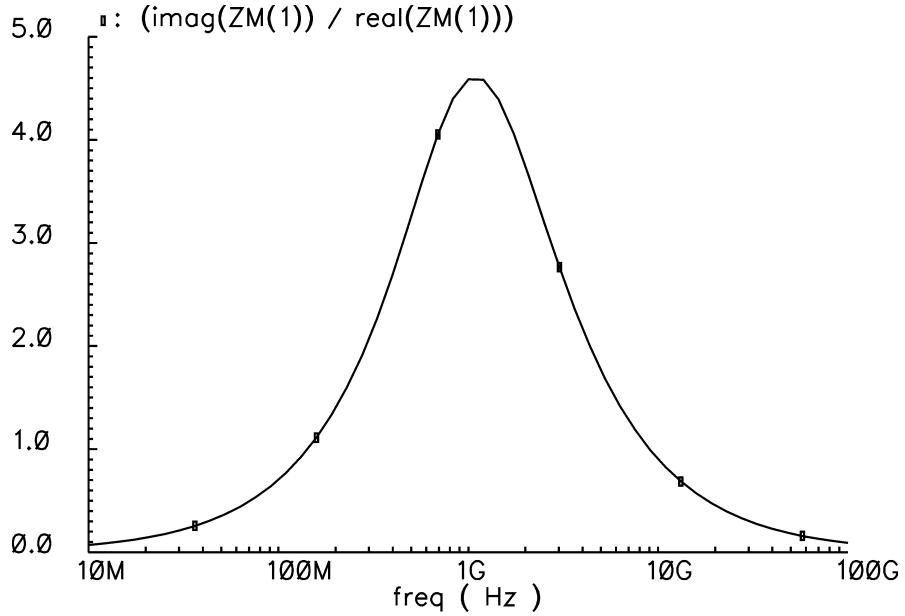
След приключване на симулацията, резултатът от анализа се извежда графично с помощта на модула на CADENCE калкулатор (**Calculator**), който се извиква от менюто **Tools** на средата за аналогова симулация **ADE**. В него се въвежда изразът за изчисляване на Q фактор (фиг. 6) и се визуализира чрез натискане на бутона **Plot**. Графиката на изменение на качествения фактор на бобината е показана на фиг. 7.



Фиг. 5. Задаване на **sp** анализ за изследване на Q фактор на бобина



Фиг. 6. Визуализиране на резултати от симулация с помощта на **Calculator**



Фиг. 7. Зависимост Q фактор на бобина от честотата

С помощта на този анализ може да се изследва влиянието на различните компоненти от еквивалентната схема върху качествения фактор на бобината.

Упражнение № 5

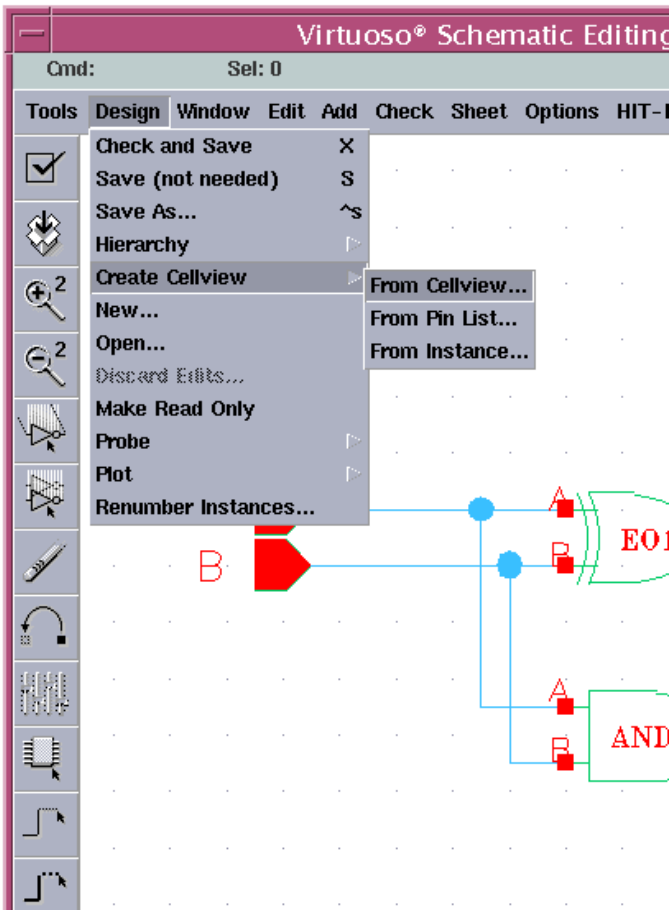
Проектиране и симулация на цифрови схеми

I. Изчертаване на схема.

Целта на упражнението е да се проектира три-битов суматор с входен и изходен пренос. Три-битовият суматор трябва да се състави от отделни едно-битови суматори, всеки от които от своя страна е изграден от полусуматори. За създаването на полусуматорите се използват логически елементи от библиотеката **HRDLIB**, като представянето на клетките е във вид на символ (**symbol**). Тъй като проектът има йерархична структура, трябва едно-битовият суматор и полусуматорът да се изградят като символи. Всеки от входовете и изходите на изградения три-битов суматор трябва да се свърже към входен/изходен порт (**pin**).

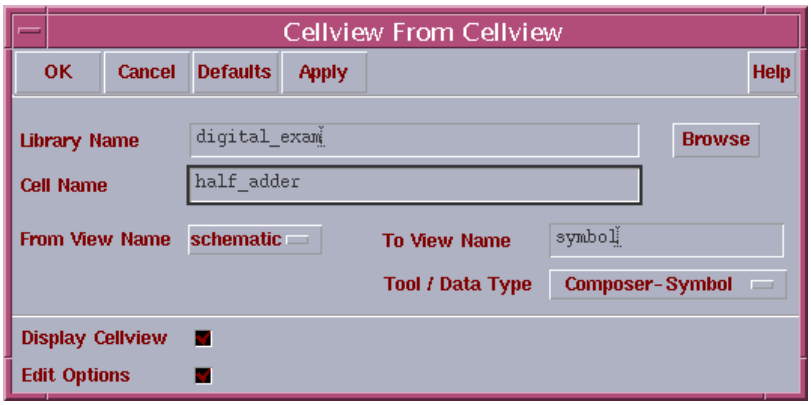
1. Създаване на символ.

След като е направена съответната схема, сложени са съответните входно/изходни портове и е направена проверка за грешки, може да се пристъпи към създаването на символ. За целта се извиква командата: **Design** ⇒ **Create Cellview** ⇒ **From Cellview** (фиг. 1).



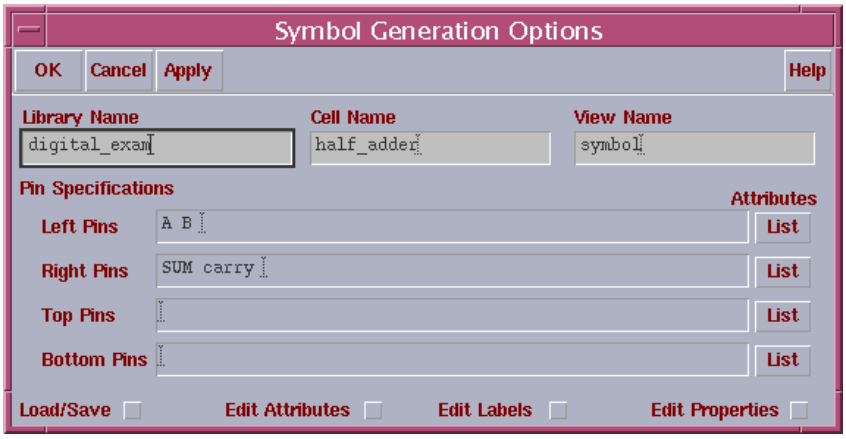
Фиг. 1. Създаване на символ

Появява се прозорец, в който трябва да се укаже името на новата клетка и библиотеката, към която тя ще принадлежи (фиг. 2).



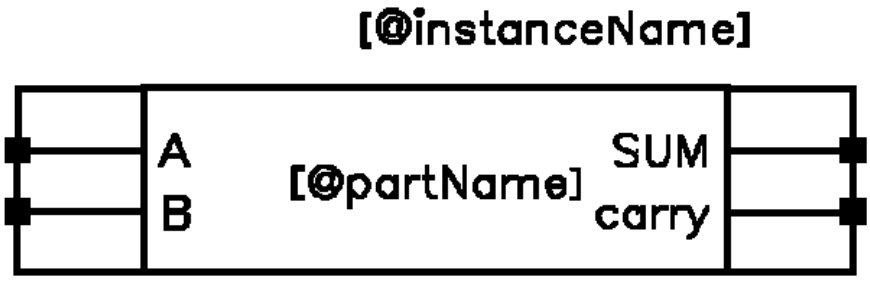
Фиг. 2. Указване на име за новата клетка и библиотеката, към която тя ще принадлежи

След попълване на съответните полета се натиска бутонът **OK**. Появява се прозорецът от фиг. 3, в който трябва да се укаже към коя библиотека принадлежи символът, неговото име, видът на представянето, имената и разположението на входно-изходните пинове и др. След като се попълнят съответните полета с желаната информация се натиска **OK**.



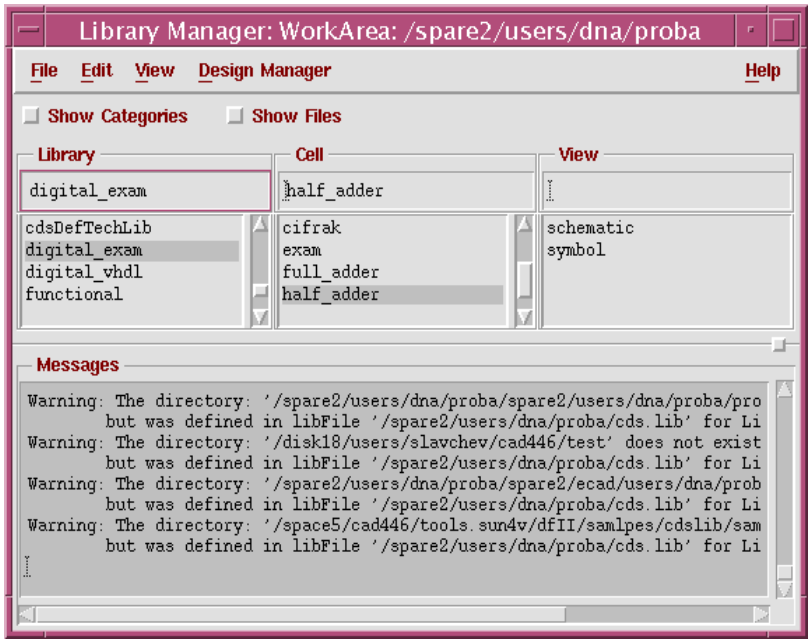
Фиг. 3. Задаване на параметрите на новата клетка

Появява се нов прозорец, в който е показан видът на символа и неговите входно-изходни пинове (фиг. 4).



Фиг. 4. Новосъздаденият символ

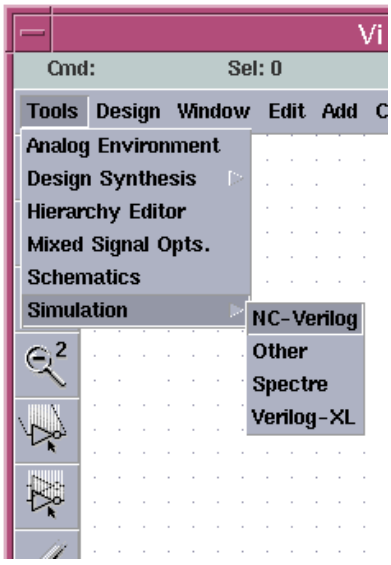
В лявата част на прозореца има поле с бутони, чрез които могат да се извикат разнообразни функции за редактиране, така че символът може да се оформи по желания начин. В случая е препоръчително да той да има формата на трапец (вж. фиг. 13). След като редактирането на символа е приключило, запазват се направените промени и се затваря прозорецът. Новият символ се появява в библиотеката (фиг. 5).



Фиг. 5. Библиотеката с новия символ

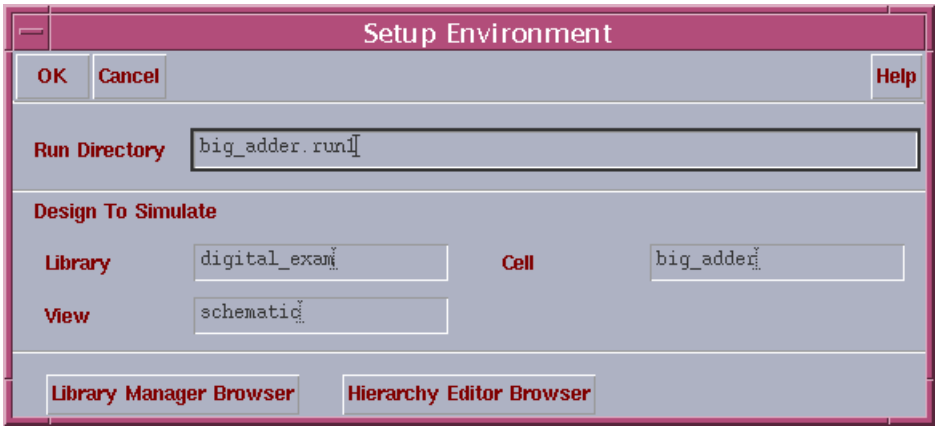
II. Симулация на схемата.

След като схемата е създадена е необходимо тя да се симулира, за да се провери дали функционира правилно. За целта се използва цифров симулатор. Той се извиква от менюто **Tools** ⇒ **Simulations** ⇒ **Verilog-XL** (фиг. 6).



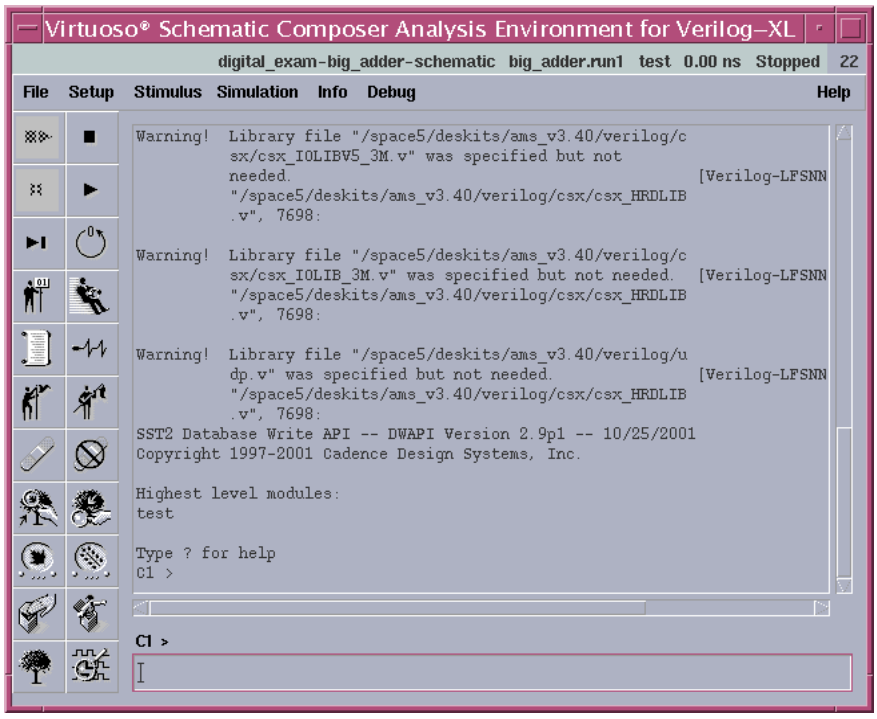
Фиг. 6. Стартиране на цифровия симулатор

Появява се прозорецът, показан на фиг. 7, в който се уточнява името на директорията, в която ще се извършва симулацията и др. Избират се опциите, които са по подразбиране, като се натиска бутонът **OK**.



Фиг. 7. Настройване на симулатора

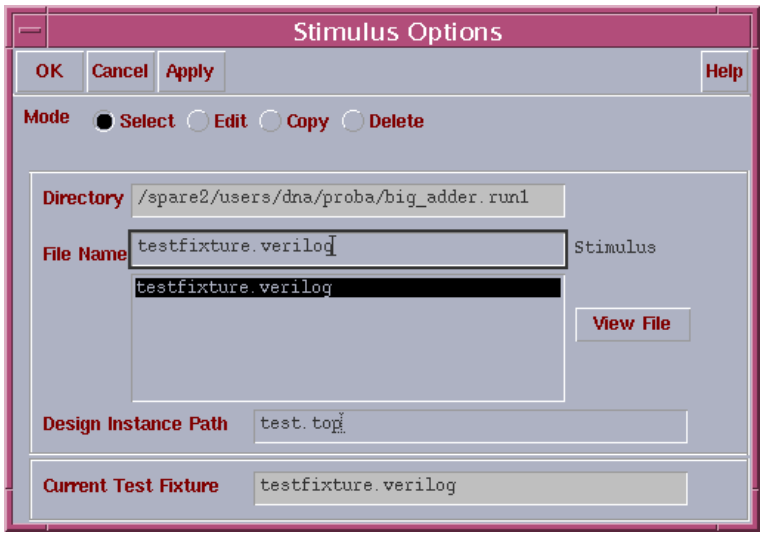
Появява се прозорецът на цифровия симулатор **Verilog-XL**, показан на фиг. 8. От менюто **Simulation** се избира опцията **Start Interactive**.



Фиг. 8. Симулатор **Verilog-XL**

Следващият етап е да се създаде файл, в който се задават подходящи тестови въздействия. Това става, като от менюто **Stimulus** се избере опцията **Stimulus**. Появява се прозорецът, показан на фиг. 9, в който трябва да се укаже името на файла, който ще съдържа тестовите въздействия. По подразбиране неговото име е testfixture.verilog. Избира се опция **Edit** и се натиска бутонът **Apply**.

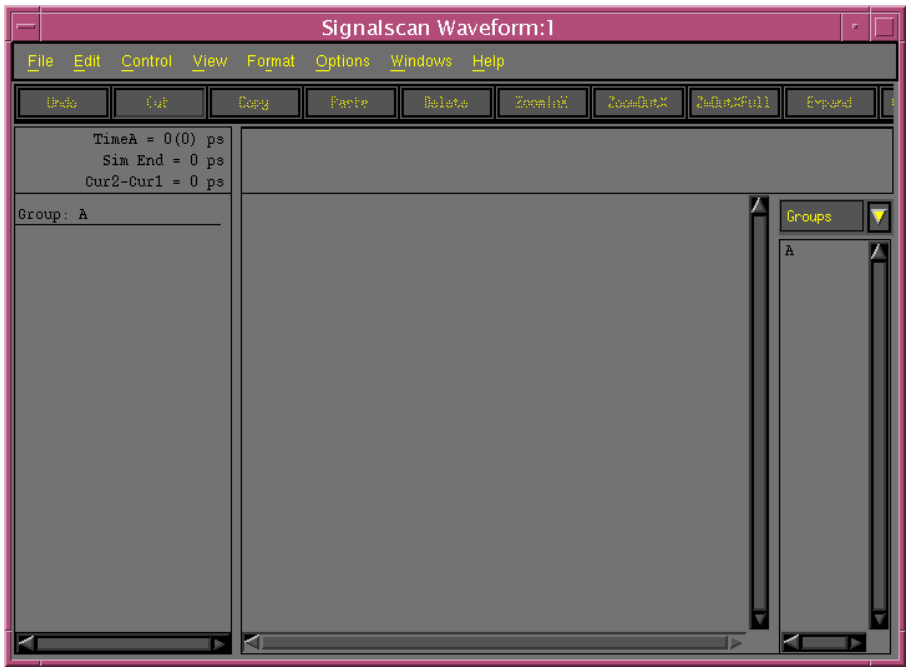
Стартира се текстов редактор, чрез който се задават тестовите въздействия.



Фиг. 9. Редактиране на файла за тестови въздействия

След описване на необходимите тестови вектори, се затваря текстовият редактор и се избира опцията **Select**. Натиска се бутонът **OK**.

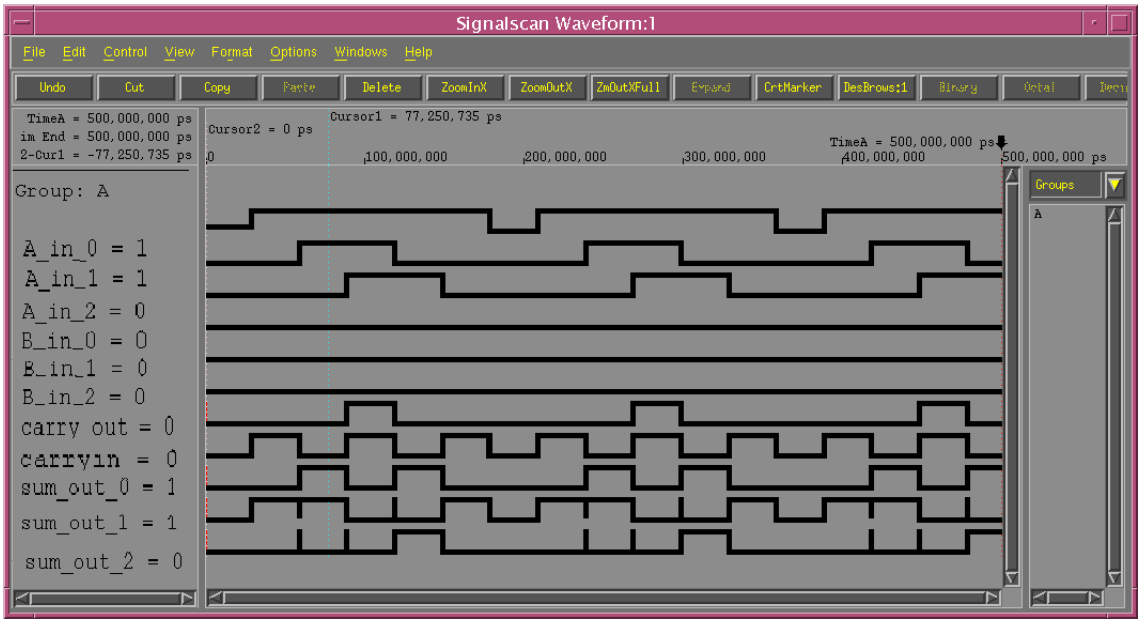
В прозореца на симулатора се натиска отново бутонът **Start Interactive**. Избира се опцията **view waveform**. Отваря се нов прозорец. В него се избира опцията **desBrows,** след което се появява прозорецът от фиг. 10.



Фиг. 10. Прозорец за визуализация на сигналите

От менюто **Instances in current contest** се избира кой модул от йерархията да се наблюдава, а от менюто **modes/variables in current content** се селектират сигналите, които ще се наблюдават. След това се натиска бутонът **add,close**. Отново в менюто на симулатора се натиска

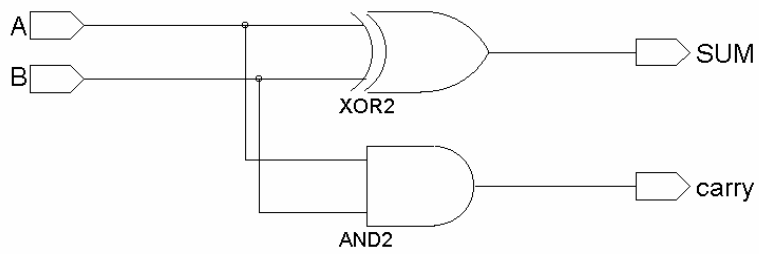
бутонът **continue**. По този начин се стартира симулацията. За да се наблюдават резултатите, се преминава в прозореца **SignalScan waveform** (фиг. 11).



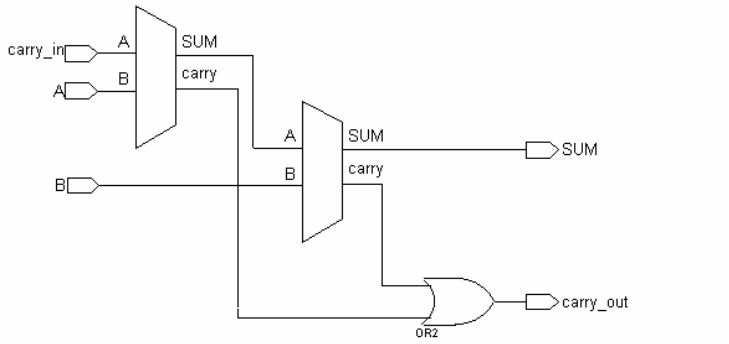
Фиг. 11. Резултати от симулацията

III. Приложение

В приложението са показани схеми на полусуматор (фиг. 12) и суматор (фиг. 13). Те са дадени за улеснение, като не е задължително да се изградят по същия начин. Като допълнение е даден и пример за testbench, написан на езика Verilog.



Фиг. 12. Схема на полусуматор.



Фиг. 13. Схема на суматор.

Примерен testbench:

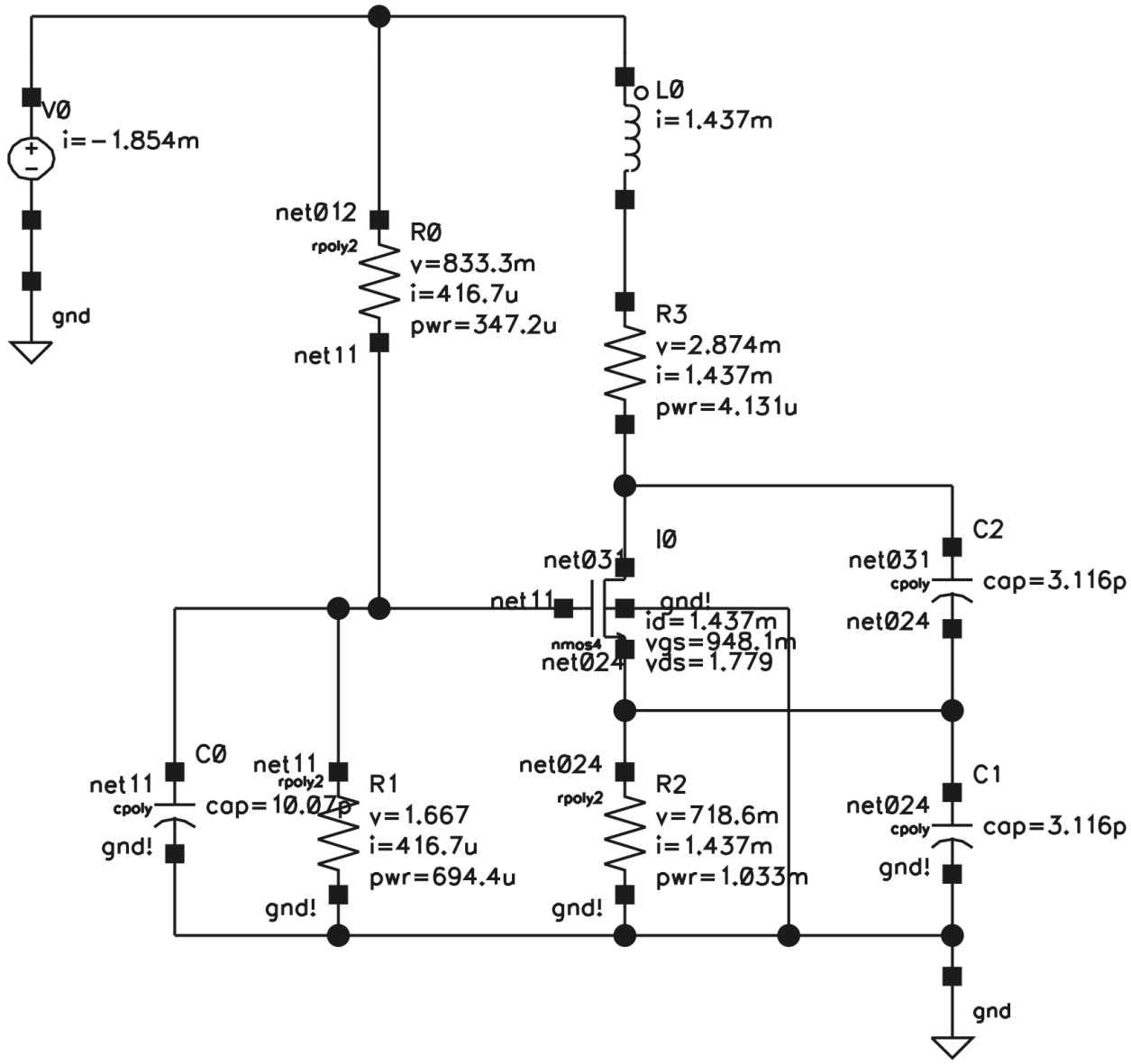
```
// Verilog stimulus file.
// Please do not create a module in this file.


// Default verilog stimulus.

initial
begin

   A_in_0 = 1'b0;
   A_in_1 = 1'b0;
   A_in_2 = 1'b0;
   B_in_0 = 1'b0;
   B_in_1 = 1'b0;
   B_in_2 = 1'b0;
   carryin = 1'b0;
end

initial #500000 $finish;

 always
  begin
   #30000 carryin = 1'b1;
   #30000 carryin = 1'b0;
 end

 always
  begin
   #30000 A_in_0 = 1'b1;
   #30000 A_in_1 = 1'b1;
   #30000 A_in_2 = 1'b1;
   #30000 A_in_1 = 1'b0;
   #30000 A_in_2 = 1'b0;
   #30000 A_in_0 = 1'b0;
 end
```

Упражнение № 6

Проектиране на клетка

1. Въвеждане на схема.

Въвежда се електрическата схема показана на фиг. 1. В нея бобината L0, резисторът R3, захранващият източник V0 и земята са идеални елементи и се взимат от библиотеката **analogLib**. Резисторът R3 замества серийното съпротивление на бобината (вж. упражнение № 2). Всички останали елементи са реални и се намират в библиотеката **PRIMLIB**: резисторите са **rpoly2**, транзисторът е **nmos4**, а кондензаторите – **cpoly**.

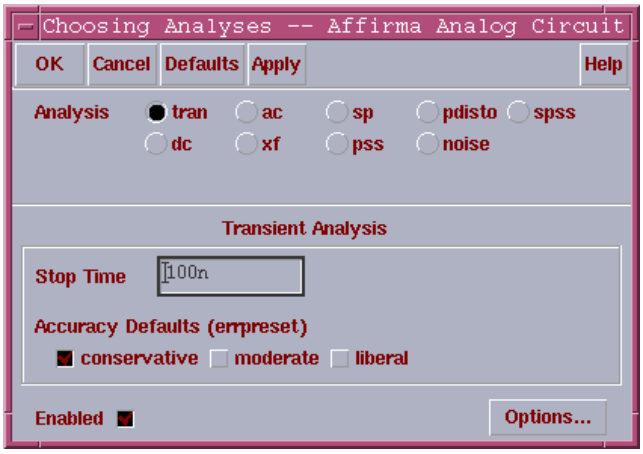


Фиг. 1. Схема на генератор на Колпитц – постояннотокова работна точка

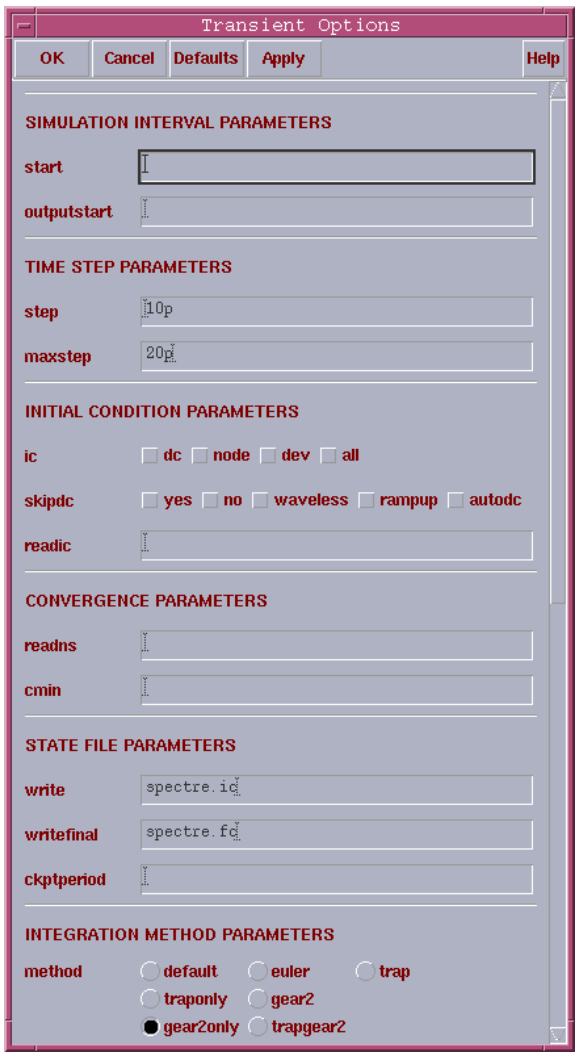
Имената на елементите, използвани в схемата, стойностите на техните параметри и библиотеката, в която се намират са показани в таблица 1:

Таблица 1.

Библиотека	Име на елемент	Стойности на параметрите
analogLib	V0	Vdc=2.5V
PRIMLIB	nmos4	W=100um, number of gates=10
PRIMLIB	R0	2kΩ
PRIMLIB	R1	4kΩ
PRIMLIB	R2	500Ω
PRIMLIB	C1	3.12pF
PRIMLIB	C2	3.12pF
PRIMLIB	C0	10pF
analogLib	L0	5.83nH
analogLib	R_L (R3)	2Ω



Фиг. 2. Прозорец за задаване на времеви анализ



Фиг. 3. Задаване на допълнителни опции на времевия анализ

2. Анализ на схемата.

За проверка на работоспособността на въведената схема се използват постояннотоков анализ на работна точка и времеви анализ. Задаването на анализите става като от менюто на схемния редактор се избира **Analog Environment**. Отваря се прозорецът на средата за симулиране. От него се избира командата **Analyses** ⇒ **Choose** и се задават съответно постояннотоковият анализ на работна точка (**dc** ⇒ **Save DC Operating Point**) и времеви (**tran**). За времевия анализ се задава **Stop Time**, т.е. краят на интервала от време, за което ще се извършва симулацията (фиг. 2). То трябва да е достатъчно, за да може генераторът да започне работа и да достигне установен режим. Задават се и някои допълнителни опции на времевия анализ (фиг. 3). Избира се стъпка (**step**), максимална стъпка (**maxstep**) и интеграционен метод (**integration method**), така че да се изпълнят условията за сходимост и да се улесни работата на симулатора.

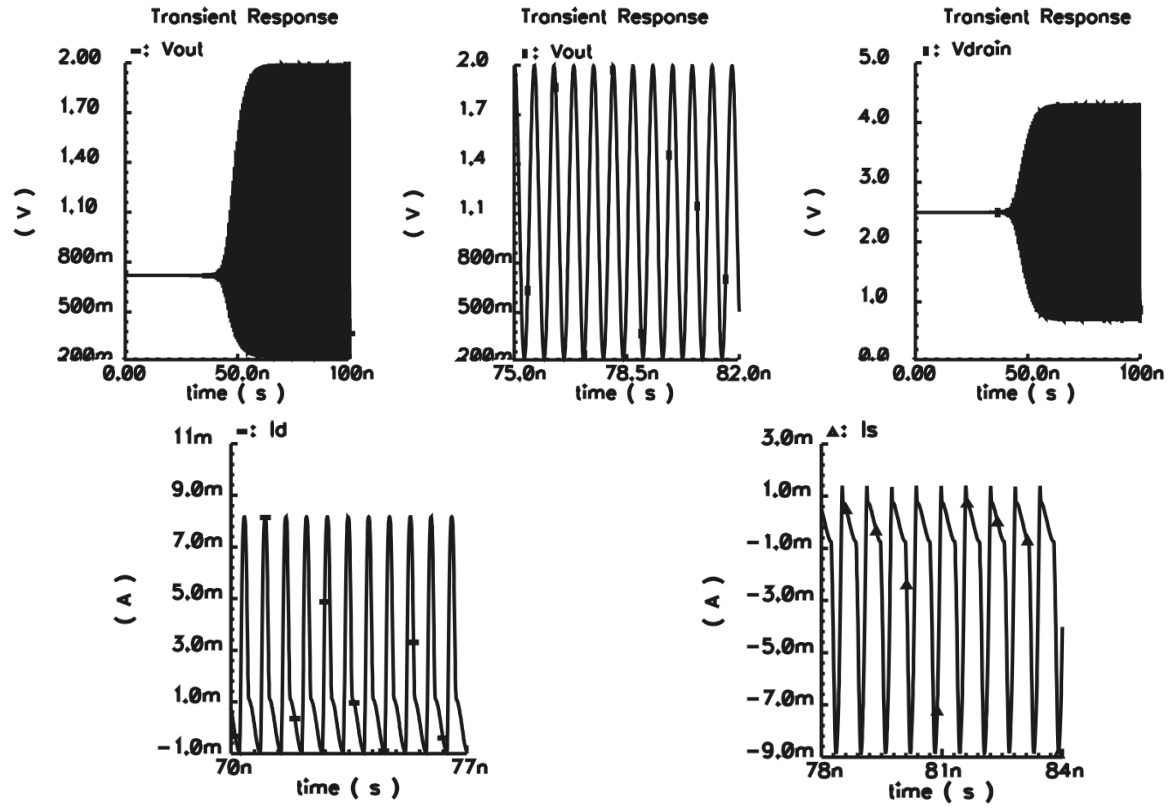
След като симулацията приключи се извеждат резултатите за:

1) Постояннотоковата работна точка (фиг. 1);

Използва се командата **Results** ⇒ **Annotate** ⇒ **DC Operating Points**.

2) Времеви анализ – визуализират се резултатите за токовете и напреженията в дрейна и сорса на транзистора (фиг. 4).

Командата е **Results** ⇒ **Direct Plot** ⇒ **Transient Signal.**



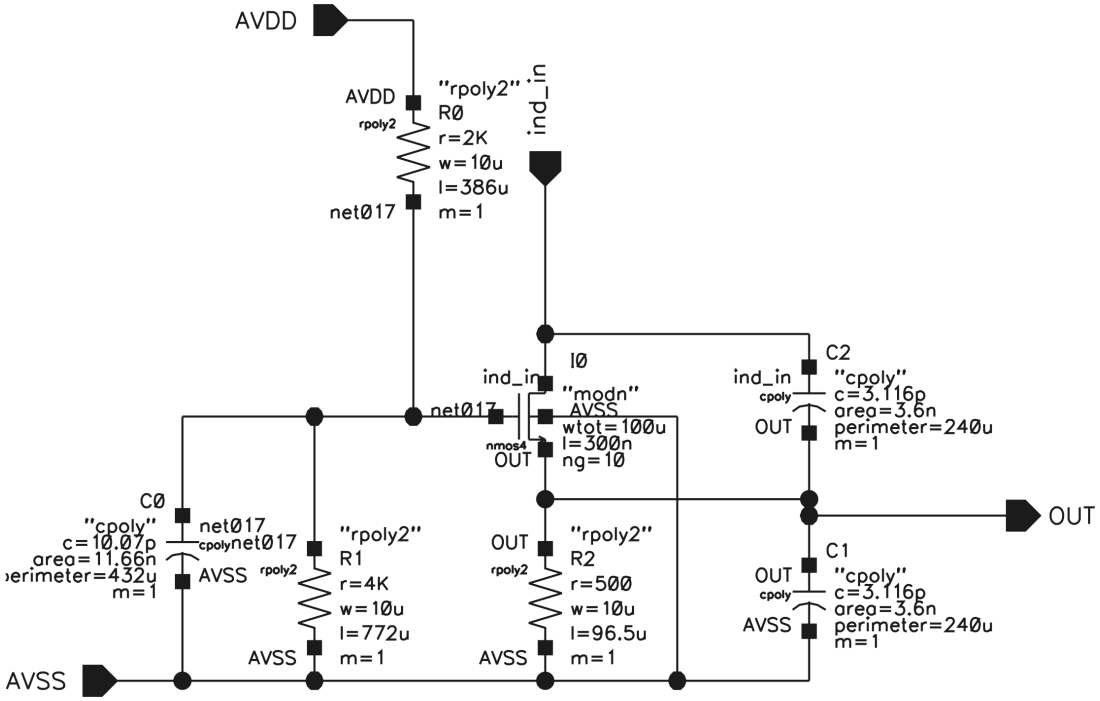
Фиг. 4. Резултати от времевия анализ

3. Подготовка на схемата за създаване на топология.

Преди да се премине към създаване на топология е необходимо да се подготви схемата. Копира се схемата от фиг. 1 в нова клетка от библиотеката, в която се работи. За да се създаде топология, всички елементи в схемата трябва да принадлежат на библиотеката, обвързана с даден технологичен процес. В случая това е библиотеката **PRIMLIB**, която съдържа реалните елементи за 0.35μm CMOS технология на AMS. Идеалните елементи в схемата са бобината, резисторът, който моделира нейното серийно съпротивление, захранващият източник и земята. Всички тези елементи нямат топологично представяне (**layout**) и трябва да се премахнат преди да се прехвърли схемата в топологичния редактор. Разбира се, тези елементи са необходими, за да работи схемата и те ще се добавят външно при симулиране на работата на схемата след създаването на топология (при ресимулация).

Генераторът на Колпитц ще се проектира като стандартна клетка. Следователно, след като се премахнат всички идеални елементи, трябва да се добавят пинове, които ще представляват изводите на схемата. Така създадената клетка може да се използва като градивен елемент в схеми на по-високо йерархично ниво. Когато се проектира чип, вместо пинове се слагат периферни клетки (входно-изходни и захранващи).

Подготвената за топологично проектиране схема е показана на фиг. 5. Добавени са пинове за изход, свързване на външна бобина и захранване.



Фиг. 5. Схема на генератор на Колпитц, подготвена за топологично проектиране

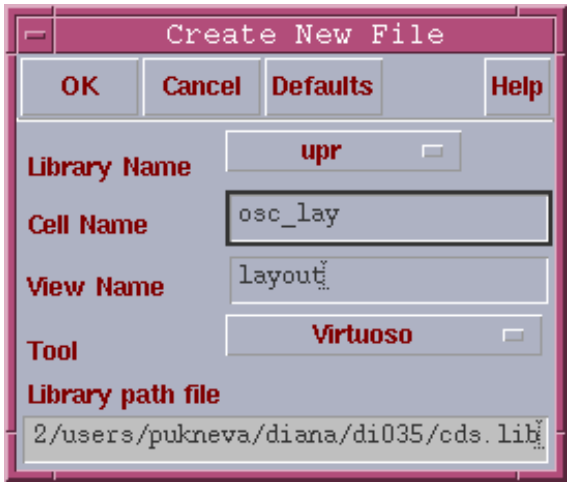
За да може да се направи ресимулация на схема, е необходимо тя да има символно представяне. Затова се прави символ на схемата (вж. упражнение № 4). Създава се нова клетка, която съдържа този символ и всички останали идеални елементи, за да се провери още веднъж функционалността на схемата след направените промени.

4. Създаване на топология.

1) Прехвърляне на елементите от схематично представяне в топологично.

За създаване на топология се използва схемата от фиг. 5. От менюто на схемния редактор се избира командата **Tools** ⇒ **Design Synthesis** ⇒ **Layout XL**.

Отваря се меню за създаване на нов файл с модула на **CADENCE Virtuoso XL**. Полето за име на файл е попълнено автоматично (фиг. 6). Проверява се дали зададеният модул е **Virtuoso** и се натиска бутонът **OK**. Отваря се прозорецът на **Virtuoso XL**. В него се избира командата **Design** ⇒ **Gen from source**.



Фиг. 6. Форма за създаване на ново топологично представяне

Появява се формата **Layout Generation**. Задават се размерите на пиновете и слоят, който ще се използва за тях в топологията. Формата е попълнена с параметри по подразбиране (например на пиновете е зададен минимално допустимият размер за технологията и слой **MET1 pn**). Може да се използват наготово зададените параметри или те да се редактират. След това се натиска бутонът **OK**. Всички елементи от схемата се появяват в прозореца на топологичния редактор. Виждат се само очертанията на елементите. За да се визуализират всички слоеве, се натискат едновременно бутони **Shift** и **f** от клавиатурата.

Със стартирането на **Virtuoso XL** се отваря и един допълнителен прозорец – **LSW** (**Layer Selection Window**). В него са показани всички слоеве, които могат да се използват при създаването на топология. Има възможност да се включват или изключват слоеве от зададения списък. Използва се командата **Edit** ⇒ **Set Valid Layers**.

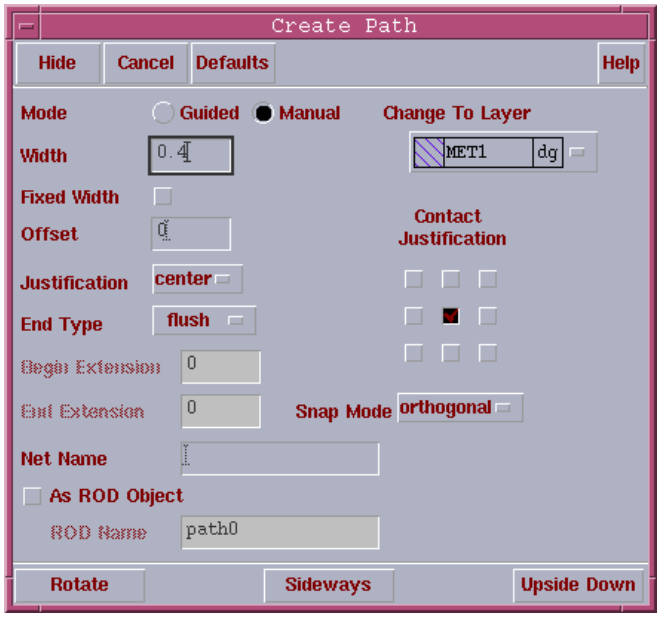
В **LSW** се избира слоят за създаване на всяка една форма, която се изчертава в прозореца на топологичния редактор. Допълнително слоевете е възможно да се правят видими или невидими, да се селектират или не.

2) Разполагане на елементите в схемата.

След като се стартира **Virtuoso XL** и се въведат всички елементи от схемата, те са разположени произволно в работното поле. Всички елементи трябва да се поставят в площта, оградена със слой **prBndry dg**, а пиновете по границите на тази площ. Това може да стане ръчно или автоматично като се използва командата **Place** ⇒ **Pin Placement**. При разполагане на елементите трябва да се следят връзките между тях, така че да се улесни опроводяването. **CADENCE** позволява да се визуализират логическите връзки между елементите. Командата, която се използва е **Connectivity** ⇒ **Show Incomplete Nets**.

Формата на елементите може да се промени, както в схемния редактор – **Virtuoso Schematic Composer** (формата **Edit Properties**), така и в топологичния редактор **Virtuoso Layout XL**. Всички промени, направени в топологията, трябва да бъдат прехвърлени обратно и в схемното представяне. Това става автоматично, като се използва командата **Connectivity** ⇒ **Update** ⇒ **Schematic Parameters**. Елементите, в които има направени промени, трябва да бъдат маркирани преди да се стартира посочената по-горе команда.

За да се провери дали има разлика между двете представяния схемно (**schematic**) и топологично (**layout**), в **Layout XL** се използва **Connectivity** ⇒ **Check** ⇒ **Against Source**.



Фиг. 7. Форма за определяне на параметрите на шина

Разполагането на елементите зависи от особеностите на схемата, правилата за проектиране и ограниченията, свързани с използването на дадена технология.

3) Изчертаване на връзките между елементите.

След като всички елементи се разположат, трябва да се начертаят връзките между тях. Избира се слой за изчертаване от **LSW**. Повечето връзки се чертаят на първи метален слой (**MET1 dg**). Командата за създаване на шини е **Create** ⇒ **Path** или бутон “**p**” от клавиатурата. По подразбиране ширината на шините е минимално допустимата за даден слой. Тя трябва да се промени в зависимост от тока, който трябва да издържи шината. За целта, след като е избрана командата **Create** ⇒ **Path**, се натиска бутонът **F3** от клавиатурата. Появява се формата **Create Path** от

фиг. 7. Освен ширина на шината (**width**), може да се зададе отместване, подравняване и др.

За да се избегне пресичането на шини, се преминава на друг метален слой. Използва се полето **Change To Layer** от формата за създаване на шини.

В таблица 2 са показани съкратените комбинации от бутони на клавиатурата за най-често използваните команди във **Layout XL**.

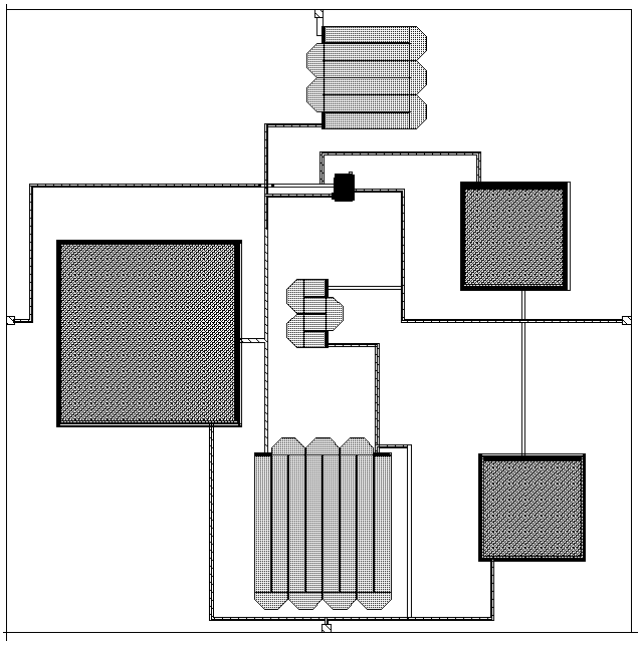
Таблица 2.

Shift-f	Прави видими всички слоеве
Ctrl-f	Прави видими само очертанията на елементите
Shift-z	Намаляване 2 пъти
Ctrl-z	Увеличаване 2 пъти
g	Включване/изключване на привличането към грида
m	Преместване/завъртане на обекти
команда +F3	Допълнителни опции към дадена команда
k	Извикване на линийката
K	Изтриване на линийката
W	Предишен изглед
i	Въвеждане на елемент
q	Извиква прозореца с параметри на елементи
F4	Превключва режимите за селектиране пълен/частичен
u	Връщане една стъпка назад

4) Няколко съвета при създаване на топологията на генератор на Колпитц:

- Обикновено пиновете за захранване и маса се разполагат в противоположни страни на топологията.
- Разстоянието между елементите трябва да е колкото може по-малко и в същото време трябва да е съобразено с изискванията на съответната технология.

Примерна топология е показана на фиг. 8.



Фиг. 8. Топология на генератор на Колпитц

5. Проверка и верификация на топологията.

За проверка и верификация на топология в средата за автоматизирано проектиране на аналогови и смесени (аналогово-цифрови) схеми **CADENCE** се използва продуктът **Diva**. Той съдържа няколко инструмента за физическа верификация,

които позволяват да се откриват и коригират грешки в проекта. Това са: проверка на правилата за проектиране, извличане на елементи от топологията на схемата и сравнение на физическата реализация с изходната схема.

1) Проверка на правилата за проектиране (**DRC**).

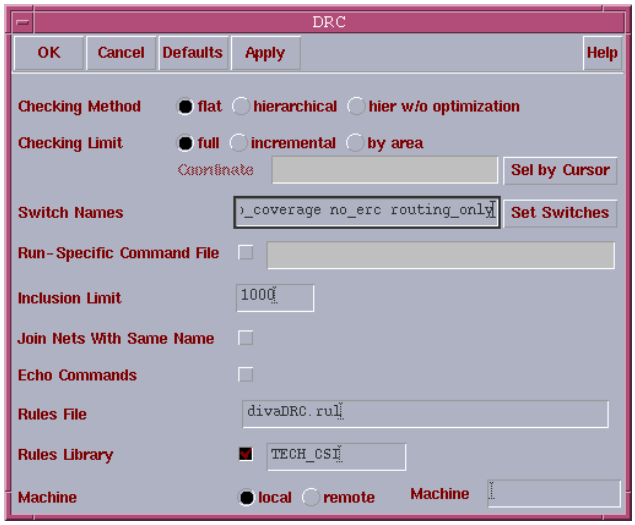
След като се направят всички връзки, топологията трябва да се провери за това дали са изпълнени всички изисквания при проектирането (**Design Rules Check – DRC**). От прозореца на **Layout XL** се избира **Verify ⇒ DRC.**

С **DRC** (фиг. 9) може да се прави проверка само на част или на цялата топология. Ако са допуснати някакви грешки, те трябва да се намерят и отстранят.

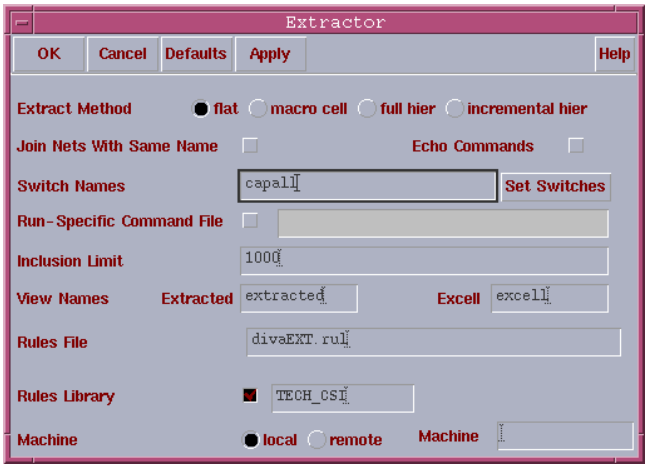
Намирането на грешки става чрез командата:

Verify ⇒ Marker ⇒ Find.

Ако не може да се обясни причината за наличието на някоя грешка се използва командата **Verify ⇒ Marker ⇒ Explain**.



Фиг. 9. Форма за задаване на **DRC**



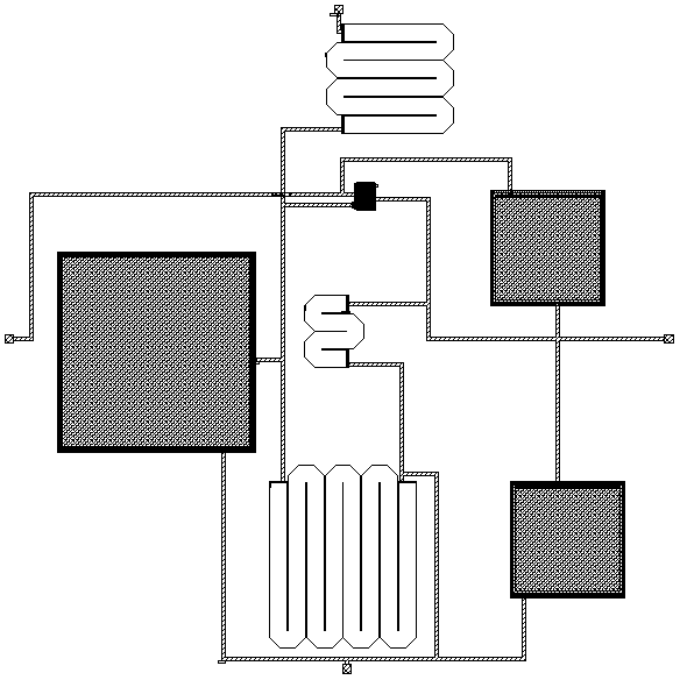
Фиг. 10. Форма за задаване на екстракция

2) Извличане на елементи от топологията на схемата (**Extract**).

Следващата стъпка е да се извлекат схемните и паразитните елементи от топологията. При екстракция се използва процесът на разпознаване на устройства, при което се създава символ за всяко разпознато устройство. Символът се поставя в **extracted** представянето на схемата, което се генерира при всяко стартиране на програмата за екстракция. Необходимо е да са дефинирани правилата за разпознаване, за да могат да се отчетат всякакви паразитни влияния.

Тази програма за верификация осигурява инструменти за измерване на параметрите на разпознатите устройства, прави изчисления с тях и запазва резултата като стойност на параметрите в екстрактнатата верига. Освен това могат да се измерват и паразитите, с

които също могат да се правят изчисления. Резултатът се запазва или като паразитен параметър на някое устройство, или като паразитни устройства между възлите. Стойностите на тези параметри са достъпни за други модули в **CADENCE**, типичен пример за което е програмата, генерираща нетлист. Това е необходимо, за да се извърши ре-симулация. Като паразити се разглеждат устройствата, които съществуват в топологията само като страничен ефект при производството на интегралните схеми, което представлява поредица от маски. Паразитите нормално не се изчертават в схемите. Добър пример за това е паразитният капацитет, образуван при пресичането на два метални слоя. Стойността му се изчислява като се измерва площта, където двата слоя се припокриват.



Фиг. 11. **Extracted** представяне на схемата

За да се стартира програмата за екстракция се използва командата **Verify** ⇒ **Extract**. Задава се допълнително ключът **capall** (в прозореца **Extractor**, показан на фиг. 10 като се натиска бутонът **Set Switches)**. Ако не се избере този ключ, програмата за екстракция ще разпознае само елементите от самата схема, но не и паразитните елементи. Натиска се бутонът **Apply** или **OK**. Ако се появят отново съобщения за грешки, те трябва да се коригират, както е описано по-горе.

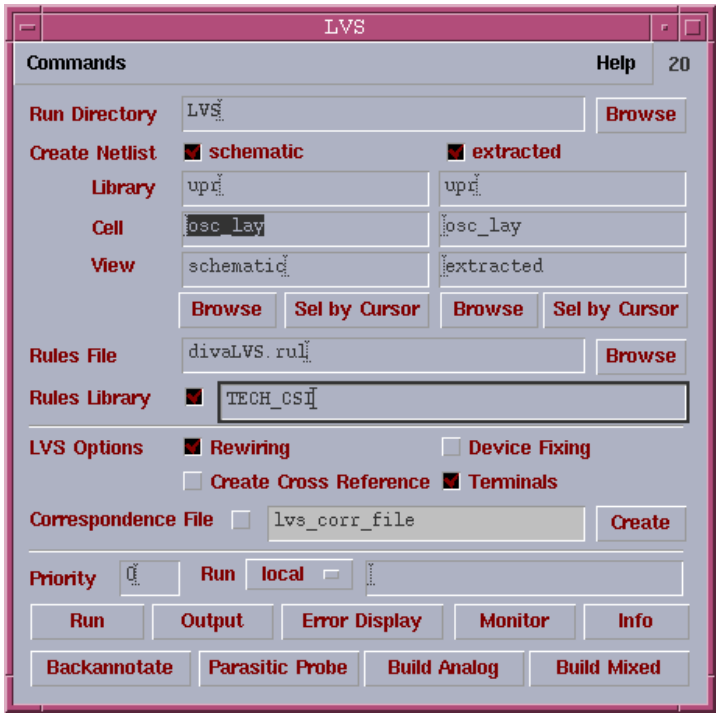
След изпълнението на екстракцията в библиотечния браузър **Library Manager** се появява ново представяне на клетката – **extracted**. То е подобно на топологичното представяне, но за разлика от него са добавени паразитни капацитети (фиг. 11). Това представяне се използва на следващата стъпка от проектирането – ресимулация.

3) Сравнение на физическата реализация с изходната схема (**LVS**).

LVS (**Layout Versus Schematic**) прави сравнение на две представяния на клетка и показва разликите между тях. Най-често се прави сравнение между топологичното представяне с извлечени схемни и паразитни елементи (**extracted**) и схемното представяне (**schematic**), от което е генерирана топологията. Модулът генерира нетлист за всяко

от представянията на схемата и ги сравнява, като използва зададените правила.

Стартирането на **LVS** става от прозореца на **extracted** представянето чрез командата **Verify** ⇒ **LVS**. Отваря се формата показана на фиг. 12. Ако и двете представяния (**schematic** и **extracted**) са отворени при стартиране на командата, формата ще се попълни автоматично. В противен случай името на клетката и представянията трябва да се въведат ръчно от потребителя. Сравнението се стартира с натискането на бутона **Run**.



Фиг. 12. Форма за стартиране на **LVS**

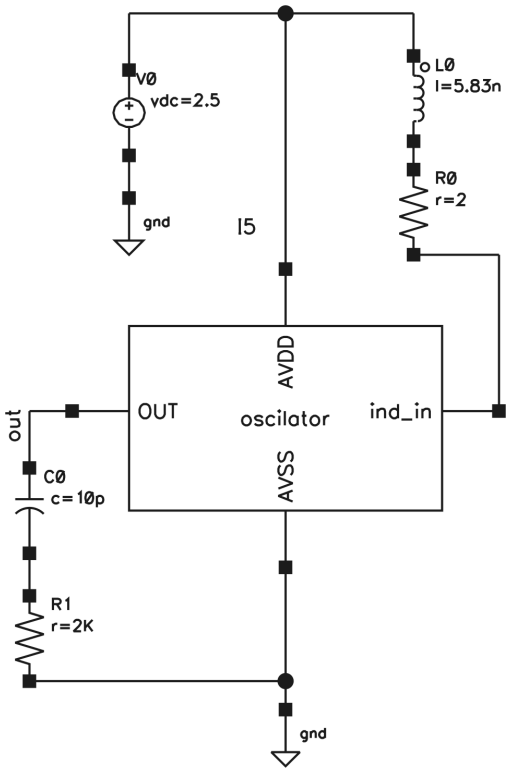
Сравнението може да отнеме няколко минути. Когато приключи, се появява прозорец със съобщението **Analysis Job Succeeded**, което показва че сравнението е направено успешно. Ако сравнението е неуспешно, се натиска бутонът **Info** във формата за **LVS**, за да се отвори лог (**log**) файла. В него са записани всички изпълнени команди при сравнението и може да се види, каква е причината за дадена грешка при изпълнение на програмата.

Ако сравнението е успешно, се натиска бутонът **Output**. Отваря се текстов прозорец с изходните данни от сравнението. Ако в него присъства съобщението "**The net-lists match**", това означава, че създадената топология съответства на схемата. Но ако съдържа "**The net-lists failed to match**", вероятно в топологията са допуснати грешки при опроводяването. Те трябва да се коригират и всички проверки трябва да се стартират отново. За да се открият грешките, се натиска бутонът **Error Display** във формата за **LVS.** Появява се форма за визуализиране на грешки. В нея се натиска бутонът **First** и в полето **Display** се изписва съобщение за първата грешка, а в **extracted** представянето се осветяват сгрешените обекти.

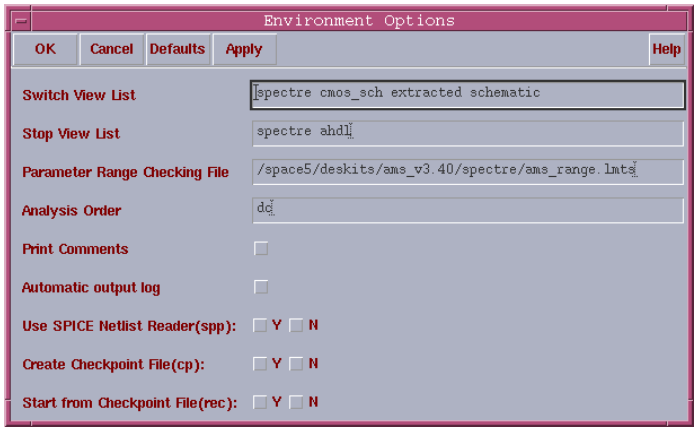
6. Ресимулация.

За ресимулация ще се използва клетката, направена по-рано в упражнението, която съдържа символа на генератора на Колпитц с бобина като външен елемент. Схемата трябва да е подобна на показаната на фиг. 13. Стартира се средата за аналогови симулации

(**Tools** ⇒ **Analog Environment**). За да се използва **extracted** представянето при симулацията (т.е. за да се направи ресимулация) в прозореца на средата за симулация се избира командата **Setup** ⇒ **Environment** (фиг. 14). В полето **Switch View List** се изписва **extracted** преди **schematic**. Стартират се отново анализите, направени на схемата.



Фиг. 13. Схема на генератор на Колпитц, използвана за ресимулация



Фиг. 14. Използване на **extracted** представяне при ресимулация – начин на задаване

Упражнение № 7

Автоматично топологично проектиране на чип

1. Създаване на схемно представяне (schematic).

В прозореца на схемния редактор (**Virtuoso Schematic Composer**) се изчертава електрическата схема на аналогова матрица от взаимосвързани транзистори (фиг. 1). Пиновете в нея се заменят с периферните клетки, изброени по-долу. Всички транзистори в схемата са реални и се взимат от библиотеката, обвързана със съответната технология (**PRIMLIB** за AMS 0.35μm CMOS технология):

1) NMOS транзистори – името на компонента в библиотеката е **nmos4**.
2) PMOS транзистори – името на компонента в библиотеката е **pmos4**.

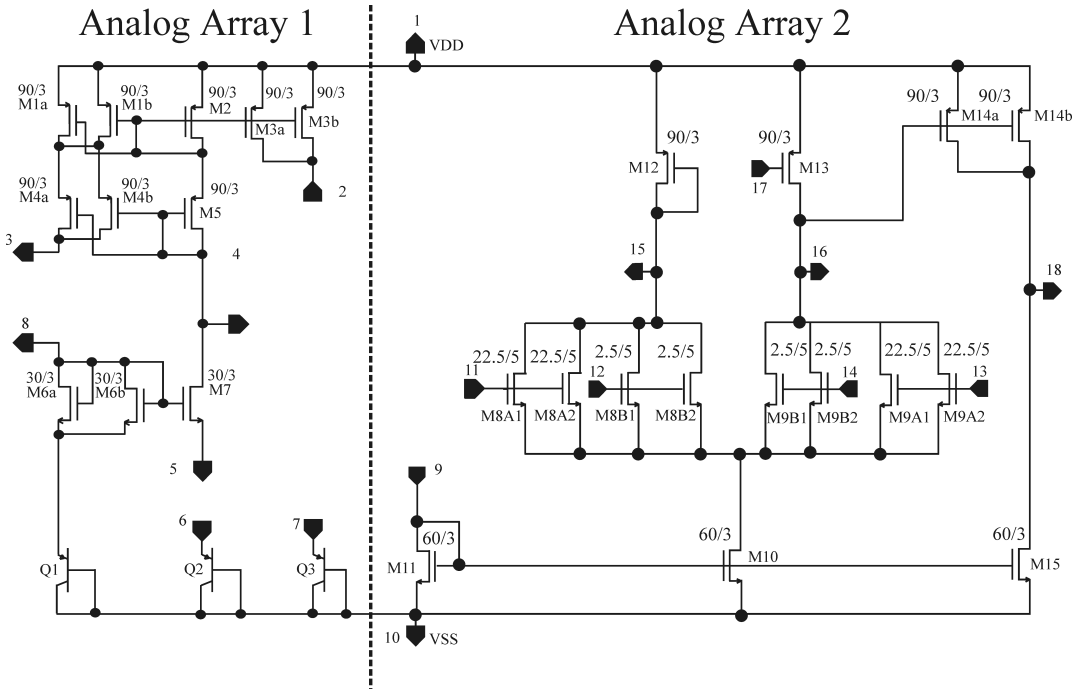
Периферните клетки в случая за AMS 0.35μm CMOS технология се взимат от библиотеката **IOLIB_ANA_3M**.

3) Входно-изходни периферни клетки

- За изходите се използват периферни клетки **APRIOP**, които са с възможно най-ниското съпротивление – възли 2, 3, 4, 5, 6, 7, 8, 9, 15, 16, 18.
- За входовете, свързани към гейтовете на транзисторите, се използват периферни клетки **APRIO1K5P**, които са с възможно най-високото съпротивление – възли 11, 12, 13, 14, 17.

4) Периферни клетки за захранване и маса:

- За VDD – **AVDDALLP** (възел 1).
- За gnd – **AGNDALLP** (възел 10).



Фиг. 1. Схема на аналоговата матрица

На PMOS транзисторите с отношение W/L=90/3 се задава брой гейтове - 5, а на NMOS транзисторите с W/L=60/3 и 30/3 – 3. За NMOS транзисторите с W/L=22.5/5 се задава брой гейтове – 2.

Чрез команда **Check and Save** се проверява схемата за евентуално нарушение на правилата при изчертаването на схемата, след което тя се запазва.

2. Топологично проектиране.

За да се създаде **layout** представяне, от прозореца на схемния редактор **Virtuoso Schematic Composer** се избира:

Tools ⇒ Design Synthesis ⇒ Layout XL

Стартира се формата на модула за топологично проектиране **Virtuoso XL**. В полето **Cell Name** се попълва името на файла, който ще съдържа топологичното представяне на схемата, показана на фиг. 1. Проверява се дали от падащото меню на формата е избран модулът **Virtuoso** и се натиска бутонът **OK**. Появяват се прозорецът на **Virtuoso XL** и прозорецът за селектиране на слоеве **LSW** (**Layer Selection Window**).

В прозореца **Virtuoso XL Layout** се избира:

Design ⇒ Gen from source

Отваря се прозорецът **Layout Generation**, в който се задават размери на пиновете и кои метални слоеве ще се използват в топологията (в случая за пиновете се избира трети метален слой – **MET3 pn**). Параметрите, които се задават в този прозорец, може да се оставят по подразбиране или да се променят от проектанта. Компонентите се появяват като символи (ниво 0). За показване на всички слоеве на елементите се използва комбинация от бутони на клавиатурата **Shift+f**.

LSW показва всички слоеве, които могат да бъдат използвани при топологичното проектиране на съответната схема. Активирането или забраняването на отделните слоеве става чрез използване на командата **Set Valid Layers** от менюто **Edit.**

В прозореца **LSW** се избира слоят, на който ще се чертае. Обектите в топологията могат да се променят от режим видими в режим невидими и обратно, както и от режим избираеми – в неизбираеми.

Virtuoso XL

Когато се стартира **Virtuoso XL (Virtuoso Layout Accelerator)** и всички компоненти са извлечени от схемата (**schematic** представяне), те се разполагат произволно в работното пространство чрез командата за преместване (**m + F3**). Възможно е да се визуализират логическите връзки между тях като от менюто **Connectivity** се избере команда **Show Incomplete Nets**.

Геометрията на елементите може първоначално да бъде променена в прозореца на схемния редактор **Virtuoso Schematic Composer** (форма **Edit Object Properties**) или по време на топологичното проектиране в прозореца на топологичния редактор. Ако промените се направят в топологичното представяне е необходимо да се нанесат и в схемното представяне на съответния проект. За целта се селектират всички елементи и в прозореца на топологичния редактор **Virtuoso XL Layout** от менюто **Connectivity** се избира командата **Update ⇒ Schematic Parameters.**

За да се провери дали има разлика в параметрите на елементите в **schematic** и **layout** представянето, от менюто **Connectivity** се избира командата **Check ⇒ Against Source**.

Когато се разполагат периферните клетки в площта на чипа, е необходимо да се спазват следните правила:

- Периферните клетки се разполагат в квадрат или в две срещуположни страни, ако формата на чипа е правоъгълна.
- Периферните клетки се разполагат плътно по границата на чипа (слой **PrBndry dg**). Разстоянията между тях трябва да са еднакви.
- В случая за AMS 0.35μm CMOS технология, съгласно технологичните правила минималното разстояние между периферните клетки е 15μm.

В ъглите на чипа може да се поставят помощни клетки, които улесняват свързването на периферните клетки. Такива са клетките **CORNERP** от библиотеката **IOLIB_3M** (категория **Glue_Cells**).

Разполагат се всички други елементи в ядрото, което се образува в центъра на чипа.

Разположението на елементите се избира според схемните особености, правилата при проектиране и технологичните правила за топологията.

3. Автоматично опроводяване с Cadence Chip Assembly Router.

Когато се опроводява даденият проект като чип в Cadence Chip Assembly Router**, слоят** POLY1 **е забранен за използване. За да се реализират връзките с гейтовете на транзисторите е необходимо да се поставят контакти** metal1-poly1 **(команда** Create ⇒ Contact**… и се избира** P1_C**).**

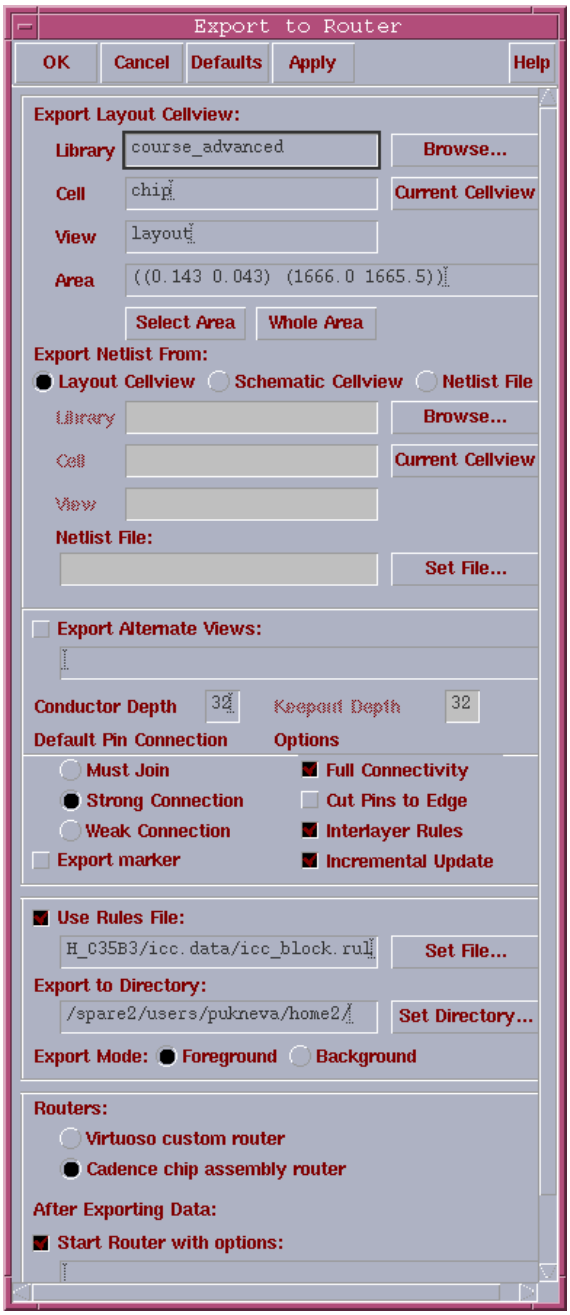
След като е изпълнена тази стъпка, може да се стартира **Cadence Chip Assembly Router.**

Избира се транслатор и се проверяват съответните изисквания. От лентата с менютата се избира **Route ⇒ Export to Router**. Попълва се формата **Export to Router**, както е показано на фиг. 2. В полето **Use Rules File** се натиска бутонът **Set file** и се избира следният път:

space5/deskits/ams_v3.50/artist/HK_C35/TECH_C35B3/icc.data/icc_block.rul

В полето **Export to Directory** се избира настоящата работна директория. Натиска се бутонът за избор на **Cadence Chip Assembly Router**. Всички останали опции могат да бъдат оставени по подразбиране и се натиска **OK**.

Появява се прозорецът на **Cadence Chip Assembly Router**. Когато проектът се отвори в модула, е необходимо да се провери за евентуалното наличие на грешки при стартирането или предупреждения, за да е сигурно че проектът е транслиран правилно.



Фиг. 2. **Export to Router** форма

1) При отваряне на проекта ауторутерът проверява за наличието на необходимата му информация. Проверява за несъответствие в нетлиста, картата на компонентите и настройките на грида. Сигнализира за пинове, които са извън грида, по-малки от минималната ширина на даден слой, недостъпни и т.н. Отваря се прозорец, в който се изписват грешките и предупрежденията. Тези съобщения могат да бъдат показани по всяко време на работата с това приложение чрез командата **Report** ⇒ **Specify** ⇒ **Startup Errors**. Важно е всяко съобщение да се проверява внимателно. Несъответствие в данните за проекта или проблеми с транслатора са причина за повечето грешки при стартирането на модула за автоматично опроводяване. Може да се наложи редактиране на тези данни и ретранслиране на проекта. Грешките, появили се при стартирането, трябва да бъдат отстранени преди да започне самото опроводяване.

За затваряне на формата, изписваща информацията за грешки и предупреждения, се натиска бутонът **Close**.

2) Ауторутерът предоставя няколко възможности, които спомагат за идентифициране и локализиране на грешките и несъответствията, които могат да създадат проблеми.

- Правилата за разполагане се проверяват чрез командата **Rules** ⇒ **Check** ⇒ **Placement**.
- Правилата за опроводяване се проверяват чрез командата **Rules** ⇒ **Check** ⇒ **Route**.

В менюто **Setup and Check** се задава типът на правилата, за чието изпълнение да се следи – **Rules** ⇒ **Check** ⇒ **Setup and Check**. Командата **Check** осветява обектите с грешки и ги показва в изходния файл.

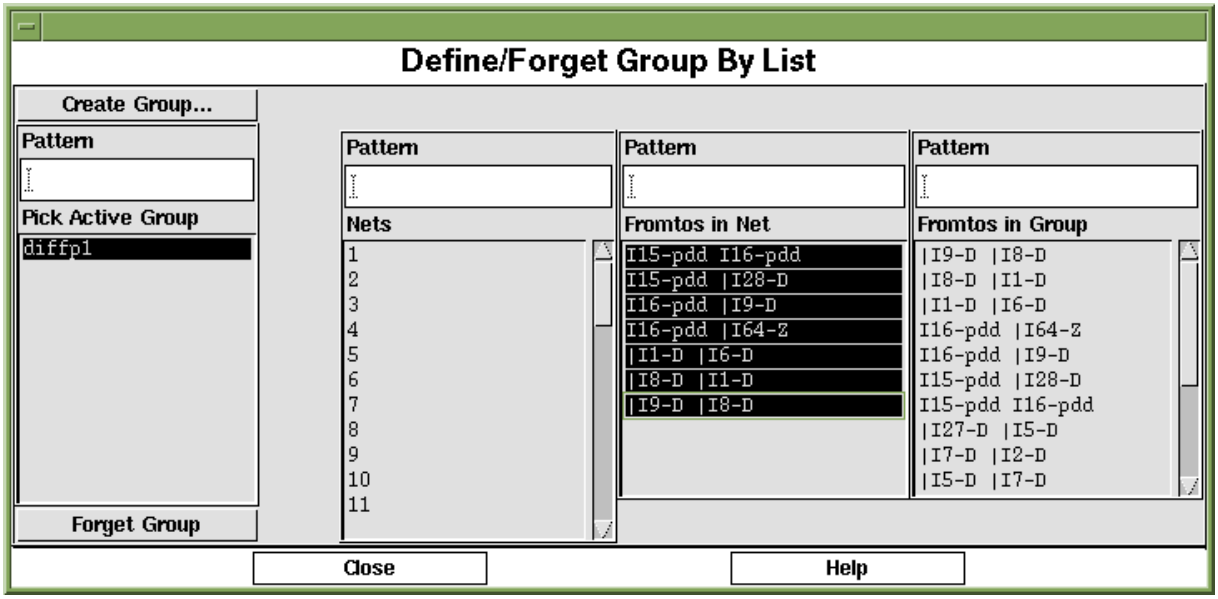
Задаване на правила за опроводяване.

Има възможност за задаване на правила за определени възли, част от възел между два пина (***fromto***), и цели части от проект. За прилагане на едно и също правило към свързани възли може да се създаде клас от възли (**class**) и да се приложи правилото за всички възли в този клас. Обикновено се създават класове от възли за тактови или захранващи сигнали, за критични информационни шини и възли, които изискват специфично опроводяване.

Типовете групирания, които могат да се дефинират при задаване на правила, са следните:

- **class** – група от възли
- **group** – група от **fromtos**

1) За да се дефинират групи от възли, се използва командата **Define** ⇒ **Group** ⇒ **Define/Forget by List**, при изпълнението на която се отваря формата, показана на фиг. 3.



Фиг. 3. Форма за дефиниране на група

Във формата **Define/Forget by List** се натиска бутонът **Create Group** и се въвежда **Group ID**. Тогава се избират **Nets** и **Fromtos in Nets**. Полето **Fromtos in Group** се попълва автоматично.

2) По аналогичен начин се дефинират и класове:

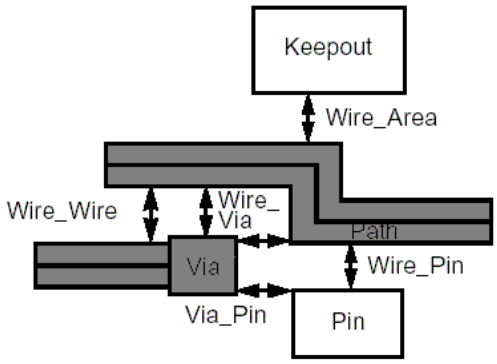
Define ⇒ Class ⇒ Define/Forget by List

Във формата **Define/Forget by List** се натиска бутонът **Create Class** и се въвежда **Class I**. След това се селектират възлите (**Nets**), които трябва да се включат в съответния клас и се натиска бутонът **Close** за затваряне на диалоговия прозорец.

3) Въпреки че ауторутерът е без грид, има възможност да се зададе производственият грид, за да се спазват правилата за проектиране. Обикновено производственият грид, използван при проектирането, се прехвърля от технологичния файл при въвеждането на проекта в модула за автоматично опроводяване. Ако оригиналният производствен файл не съдържа размерите на грида, те могат да бъдат зададени чрез командата **Define - Design Grids**. Има опция за задаване на различна големина на грида за всеки слой за опроводяване и определяне дали грида се спазва спрямо краищата или центровете на проводниците.

Контролиране на ширини и разстояния.

Ауторутерът прилага правилата за проектиране и свойствата на форми като пинове, проходи и проводници. Типовете правила за спазване на определени разстояния са **Wire-to-Wire** (проводник-проводник), **Wire-to-Pin** (проводник-пин), **Wire-to-Via** (проводник-проход), **Wire-to-Area** (проводник-област), **Via-to-Via** (проход-проход), **Via-to-Pin** (проход-пин), **Via-to-Area** (проход-област), **Pin-to-Pin** (пин-пин), **Area-to-Area** (област-област) и са показани на фиг. 4.



Фиг. 4. Правила, отнасящи се до разстояния

Ширината на проводниците, разстоянията и слоевете за опроводяване се задават чрез командата **Rules ⇒ Layer ⇒ Width/Clearance….** Може да се зададат ширина и разстояния за всеки метален слой. Също така има опция за задаване на определени ширини и разстояния на конкретен възел или **fromto**, класове възли и т.н. Отделно може да се зададе и ширината на металните слоеве за възли, обединени в един клас:

Rules ⇒ Class Layer ⇒ Width/Clearance…

Създаване на каналите за захранване и маса

Ширината и разположението на каналите на захранване и маса са много важни при опроводяването на чип, особено за цифровите схеми.

При опроводяване на захранващите сигнали рутерът може да опроводи пръстени около всички индивидуални блокове или всеки избран блок. При него автоматично се създават захранващи канали около и между индивидуалните блокове. Задава се посоката за всеки слой на каналите в **Layer Panel** и се избира първи и втори слой за опроводяване чрез **AutoroutePower** ⇒ **Route** ⇒ **Ring Components**.

Автоматично опроводяване

Първо трябва да се извърши глобално опроводяване. При него се определят пътищата за всеки неопроводен възел на проект от високо ниво. Глобалният рутер (**Global Router**) създава двумерна матрица от клетки за глобално опроводяване (**gcells**) за улесняване на детайлното опроводяване.

Глобалният рутер следва посоката на локалния слой и критичните правила и отделя всички неопроводени връзки в **gcells**, през които трябва да мине група от връзки. Това позволява на глобалния рутер да направи оптимизация на високо ниво и да намали времето за изпълнение на детайлното опроводяване.

Командата за стартиране на глобалния рутер е:

Autoroute ⇒ **Global Route** ⇒ **Global Router…**

Използват се опциите, зададени по подразбиране във формата **Global Router**. В терминалния прозорец може да се провери общата дължина на шините за опроводяване.

Създаване на входно-изходен пръстен

Опроводяването на входно-изходен пръстен представлява свързване на всички странични портове на периферните клетки, където е възможно, с ширината и на слоя за тези портове до образуването на пръстен. Това се прилага за всички междуклетъчни връзки като захранване, клетки за тестване и т.н.

Избират се всички входно-изходни клетки чрез **Select - Components** и се стартира опроводяване на пръстена на входно-изходните клетки чрез командата **Autoroute** ⇒ **Power Route** ⇒ **Route**. Активира се опцията **As I/O Ring**.

Защита и запазване на захранващото опроводяване

Автоматично може да се защитят опроводените канали чрез **Autoroute** ⇒ **Power Route** ⇒ **Route**. Активира се опцията **Protect Trunk**. Ако е необходимо да се редактират направените от ауторутера проводници и многоъгълници, те не трябва да се защитават, за да могат да бъдат премествани или редактирани интерактивно. Преди стартиране

на командата **route** или **clean** тези проводници отново трябва да се защитят.

Запазването става чрез запис на информацията чрез команда **File - Write.**

Детайлно опроводяване

Същинското опроводяване се извършва от детайлния рутер (**Detailed Router**). Той се стартира след глобалното опроводяване. По време на детайлното опроводяване ауторутерът използва пътищата, предложени от глобалното опроводяване. Използването на глобалното преди детайлното опроводяване намалява като цяло времето за опроводяване.

Детайлното опроводяване се стартира чрез командата **Autoroute** ⇒ **Detail Route** ⇒ **Detail Router…** Използват се опциите по подразбиране във формата на детайлния рутер.

В прозореца на модула за автоматично опроводяване се виждат итерациите по време на самото опроводяване. Ако след завършването резултатите не са задоволителни, се избира команда **Autoroute** ⇒ **Clean…**

Използват се 3 до 5 итерации за оптимизиране на опроводяването. Итерациите за изчистване намаляват броя на проходите и подобряват качеството на самото опроводяване. Рутерът се опитва да оптимизира връзките и опроводи наново някои възли. Оптимизация може да се извърши и ръчно, както в модула за автоматично опроводяване, така и в топологичния редактор на **CADENCE**.

След завършване на опроводяването трябва да се експортира информацията за него във **Virtuoso XL**. Използва се:

File ⇒ Write ⇒ Session

Избират се следните бутони: **Force Include Placement**, **Include Global Paths**,и **Merge Straight Paths**. Натиска се **OK** и се премества курсора в прозореца на **Virtuoso XL**, където автоматично се прехвърля опроводяването.

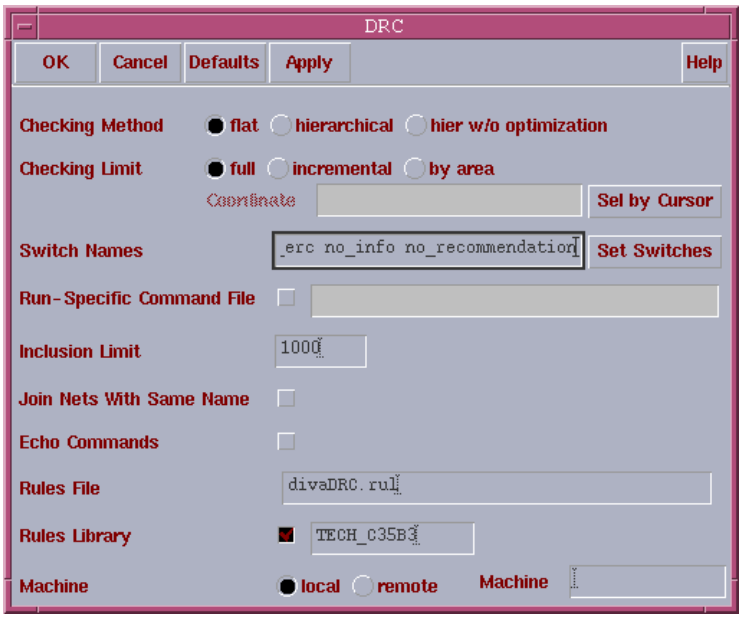
Проверява се дали всички връзки са завършени (**Connectivity** ⇒ **Show Incomplete Nets**). Ако има незавършени такива, те се откриват и опроводяват ръчно.

Следващата стъпка е стартиране на **DRC** (**Design Rules Check**) верификация, за да се провери дали са спазени правилата за проектиране. От прозореца на **Virtuoso XL** се избира командата **Verify** ⇒ **DRC**. Задават се опциите, както е показано на фиг. 5. Използват се следните проверки: **no_erc**, **no_info**, **no_recommendations**, **no_coverage**.

DRC позволява да се проверява част от проекта или цялата топология. При наличие на грешки те трябва да се открият и отстранят (**Verify ⇒ Marker ⇒ Find**).

Ако не е ясен типа на грешката се избира опция **Verify ⇒ Marker ⇒ Explain.**

Следващата стъпка е извличане на паразитни компоненти от топологията. В прозореца на **Virtuoso** се избира **Verify ⇒ Extract**. Използва се ключът **capall** (в **Extractor** прозореца се натиска бутонът **Set Switches**, избира се **capall** и се натиска бутонът **OK**). Натиска се **Apply** или **OK** за стартиране на екстракцията.



Фиг. 5. Форма за DRC верификация

Ако има някакви грешки, те трябва да се открият и отстранят.

Упражнение № 8

Проектиране на мултиплексор

I. Цел на упражнението.

Това упражнение представлява въведение в топологичните редактори **Virtuoso Layout Editor**, **Virtuoso Layout Accelerator (Layout XL)** и интерактивните продукти за верификация на **Assura**. В него целта е да се създаде топология на мултиплексор, като се използват различни команди в топологичния редактор **Virtuoso Layout Editor**. Преди да се започне със самото упражнение, трябва да се създаде нова директория, която ще бъде използвана за стартиране на **CADENCE** и в която ще се съхраняват всички проекти (за повече информация вж. Упражнение №1). Начинът, по който ще се стартира **CADENCE**, зависи от технологията, която ще се използва. В случая се използва AMS 0.35μm Si CMOS три метална технология и **CADENCE** се стартира по следния начин:

source ~radonov/scripts/.ams350
ams_cds -tech c35b3 -mode fb

В упражнението често се използват следните съкращения:
CIW – **C**ommand **I**nterpreter **W**indow;
LMW – **L**ibrary **M**anager **W**indow;
VLEW – **V**irtuoso **L**ayout **E**ditor **W**indow;
LSW – **L**ayer **S**election **W**indow.

II. Проектиране на инвертор.

1. Създаване на схемно и символно представяне на инвертор.

Преди да се създаде топологично представяне на инвертор е необходимо първо да се генерират другите две негови представяния: схемно и символно. Всички тези представяния на клетката трябва да се запазят в съществуваща библиотека. Ако такава липсва се започва от създаване на библиотека.

Създаване на нова библиотека:

В **LMW** се избира **File** ⇒ **New** ⇒ **Library**

Попълва се име на новата библиотека в полето **Library Name** на появилата се форма и се обвързва новосъздадената библиотека с подходящ технологичен файл (в случая **TECH_C35B3**);

Създаване на схематично представяне на инвертор.

1) Маркира се създадената библиотека и в **LMW** се избира:
File ⇒ **New** ⇒ **Cellview**
2) Във форма за създаване на нова клетка се попълва:

Library: <ваша_библиотека>
Cell Name: Inv
View Name: schematic

Натиска се бутонът **OK** и след няколко секунди се появява прозорецът на схемния редактор. В него ще се създаде схемата на инвертора, показана на фиг. 1. В таблица 1 са изброени необходимите елементите, както и от коя библиотека трябва да бъдат взети:

Таблица 1.

Библиотека	Име на елемента	Вид представяне
PRIMLIB	nmos4	symbol
PRIMLIB	pmos4	symbol
analogLib	vdd	symbol
analogLib	gnd	symbol

За добавяне и разполагане на елементи в прозореца на схемния редактор (**Virtuoso Schematic Composer**) се използват следните команди:

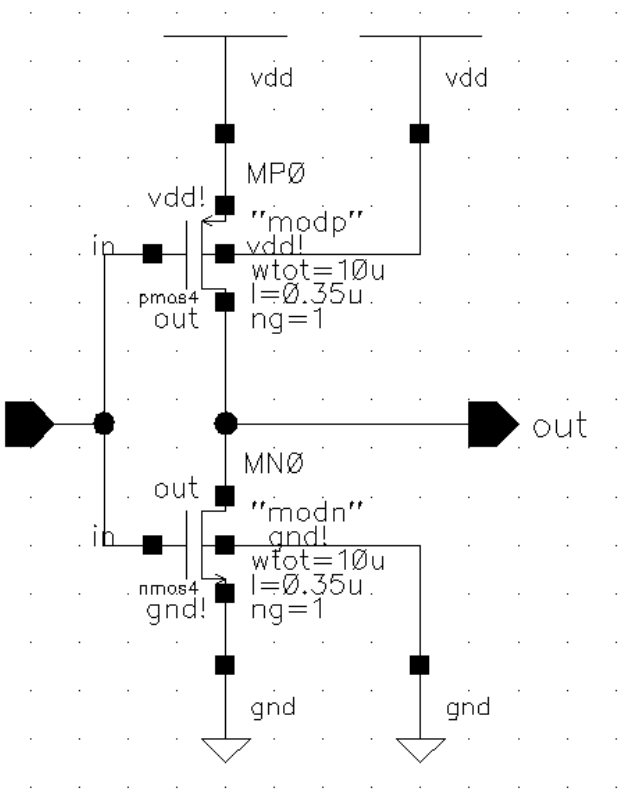
3) За добавяне на елемент: **Add** ⇒ **Instance** (бутон "**i**" от клавиатурата).

Може да се използва бутонът **Browse** за избиране на желаните елементи, включени в схемата на инвертора и библиотеката, в която те се намират. Елементите могат да се поставят многократно в работната площ на схемния редактор. Като се променя името на елемента в **Add Instance** формата могат да се разположат всички елементи. За да се прекъсне добавянето на елементи се натиска бутонът **Esc** от клавиатурата. За транзисторите от схемата се използват параметрите по подразбиране.

4) За добавяне на пинове: **Add** ⇒ **Pin** (или бутон "**p**" от клавиатурата).

Във формата за добавяне на пинове се попълва име на пина и се задава посоката му: входен, изходен, входно-изходен.

5) След като се разположат всички елементи, се изчертават връзките между тях чрез командата **Add** ⇒ **Wire** (бутон "**w**" от клавиатурата)



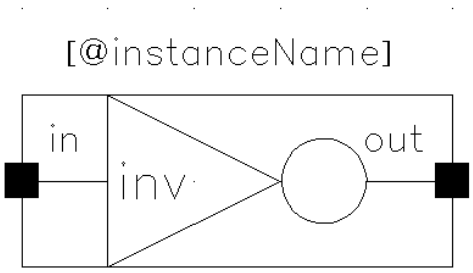
Фиг. 1. Схема на инвертор

След като схемата на инвертора е начертана, тя трябва да се запази. Използва се командата **Design ⇒ Check and Save**.

Създаване на символно представяне на инвертора.

1) От прозореца на схематичния редактор се избира:
Design ⇒ CreateCellview ⇒ FromCellView
2) Появява се попълнена форма за създаване на символ. Оставят се настройките по подразбиране и се натиска бутонът **OK**.
3) Появява се форма, в която трябва да се посочи мястото на пиновете: за входния – ляво; за изходния – дясно.

След няколко секунди се появява прозорецът на символния редактор, в който може да се оформи символът на инвертора. По подразбиране формата на всички автоматично генерирани символи е правоъгълна. За промяна на формата на символа се използва командата **Add ⇒ Shape**. Пример за символ на инвертор е показан на фиг. 2.



Фиг. 2. Символно представяне на инвертор

4) Името на символа може да се промени като се селектира полето "**[@partName]**" и се натисне "**q**" от клавиатурата, след което се задава име в полето **Label**.
5) Запазва се символното представяне и се затваря символният редактор.

След като са създадени две от необходимите представяния на инвертора – схемно и символно предстои проектирането на неговата топология.

2. Създаване на топология на инвертор.

В **LMW** се избира библиотеката, която съдържа схемното и символното представяне на инвертора и се задава:
File ⇒ New ⇒ Cellview
Във формата за създаване на нова клетка се попълва:
Library: <ваша_библиотека>
Cell Name: Inv
View Name: layout

След натискане на бутона **OK** се отварят топологичният редактор (**VLEW**) и прозорецът с наличните слоевете (**LSW**). Работното поле на **VLEW** е разделено на 4 части. За работна област се избира 1-ви квадрант на полето (горния десен квадрант). За целта се използва командата **Window ⇒ Pan** (бутонът **Tab** от клавиатурата) и се натиска левият бутон на мишката в горния десен ъгъл на прозореца.

Добавяне на N- и PMOS транзистори

1) За добавяне на N- и PMOS транзистори се използва командата **Create** ⇒ **Instance** (бутон "**i**" от клавиатурата).
2) Във формата се попълва:

Library: PRIMLIB
Cell Name: nmos4
Cell View: layout

3) Премества се курсорът във **VLEW**
4) С натискане на десен бутон на мишката, транзисторът се завърта на 90 градуса. Премества се курсорът на координати X=4.3, Y=3 и се натиска левият бутон на мишката, за да се постави елементът.

X и Y координатите се показват в командната лента в горния ляв ъгъл на VLEW.

Shft+f **или** Ctrl+f **се използват, за да се покажат или всички нива от йерархията (всички слоеве) или само ниво 0!**

5) С натискане на бутона **Esc** от клавиатурата се прекратява текущата операция.
6) По аналогичен начин се добавя и разполага **PMOS** транзисторът (**pmos4**). Координатите му са X=4.3, Y=16.8.

Свързване на входовете и изходите.

За да се създаде инвертор, е необходимо да се направят връзките между N- и PMOS транзистора.

- Избор на слой от LSW.

Всеки слой има свое предназначение, например: възел/**net** (nt), пин/**pin** (pn), изчертаване/**drawing** (dg). При създаването на повечето топологии се използват слоеве с **drawing** предназначение.

За да се избере първи метален слой, се натиска левият бутон на мишката върху **MET1 dg** в **LSW**. Слоят **MET1** става по-тъмен и се появява над списъка със слоевете в **LSW**. Така се показва текущият активен слой, който ще се използва от топологичния редактор – **VLE.**

1. Свързване на изходите.

1) За свързване на изходите се избира **Create** ⇒ **Rectangle** (бутон "**r**" от клавиатурата).

В информационния ред на **CIW** и **VLEW** се изписва:

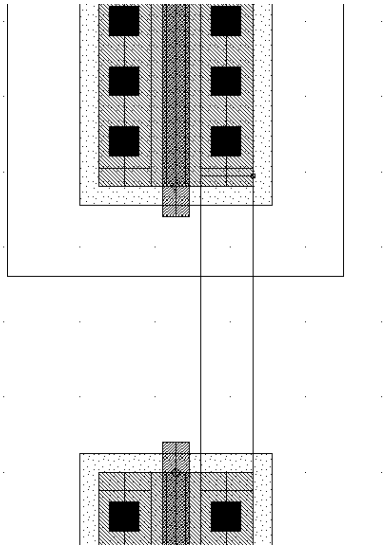
Point at the first corner of the rectangle

2) Позиционира се курсора на мишката в горния ляв ъгъл на дрейновия контакт на NMOS транзистора (фиг. 3) и се натиска с левия бутон. Информационният ред подсказва следващата стъпка – посочване на противоположния ъгъл, за да се изчертае четириъгълникът:

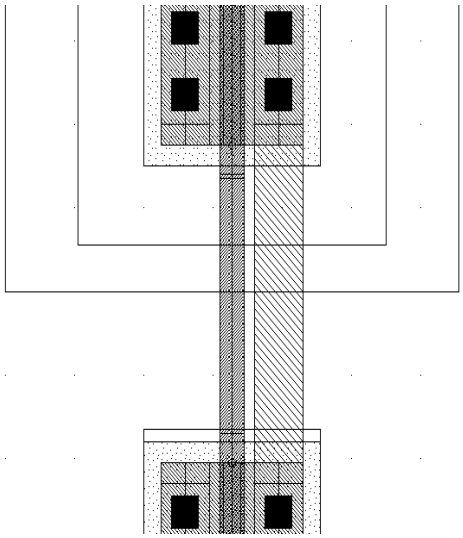
Point at the opposite corner of the rectangle

В случая това е долният десен ъгъл на дрейновия контакт на PMOS транзистора (фиг. 3).

3) Натиска се **Esc** от клавиатурата. Така изходите на двата транзистора са свързани и се прекратява командата.



Фиг. 3. Свързване на изходите



Фиг. 4. Свързване на входовете

2. Свързване на входовете.

За свързване на входовете на слой **POLY1** на транзисторите се използва командата за създаване на шина. Формата на шината се определя от нейната ширина, вид и начин на центриране.

1) В **LSW**, се избира слой **POLY1 dg**.
2) Избира се командата **Create** ⇒ **Path** (бутон "**p**" от клавиатурата).

За да се отворят допълнителните опции на командата, се натиска **F3** от клавиатурата. Появява се формата за създаване на шина (**Create Path**). Използва се подравняване в дясно, а останалите параметри се оставят по подразбиране.

3) Информационният ред подканва да се избере началото на шината. В случая това е горният десен край на гейта на NMOS транзистора (фиг. 4):

Point at the first point of the path

4) Шината се завършва в долния десен край на гейта на PMOS транзистора. Това става с двукратно натискане на ляв бутон на мишката в крайната точка или с еднократно натискане на ляв бутон на мишката и **Enter** от клавиатурата (фиг. 4).
5) Натиска се бутонът "**o**" от клавиатурата, за да се създаде контакт между слоя **POLY1** на гейта и **MET1**. Този контакт е необходим при свързването на инвертора в схема на мултиплексор. Избира се контакт **P1_C**.
6) Контактът се поставя на координати X=2.7, Y=14.6.
7) Натиска се бутонът **Esc**.

С това изчертаването на входните и изходните връзки между двата MOS транзистора в инвертора е завършено.

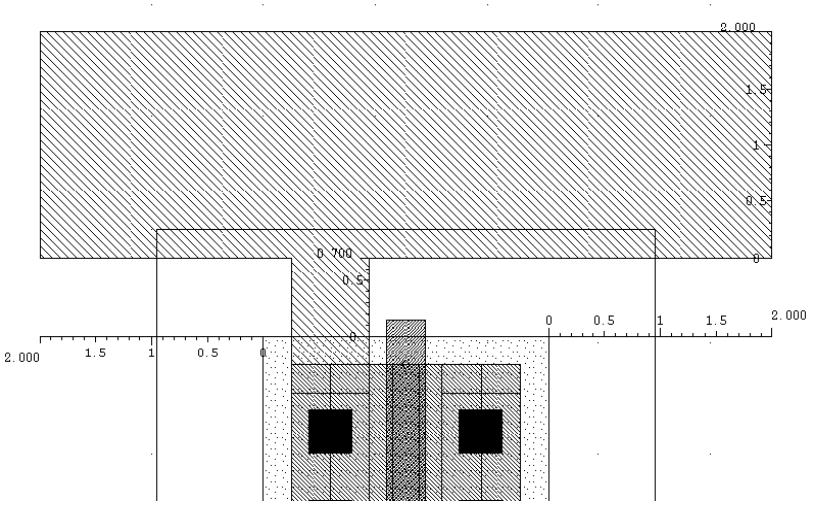
Създаване на връзки за захранване и маса.

За реализиране на тези връзки се използва командата за създаване на многоъгълник. Многоъгълникът се определя от множество точки. Той трябва да бъде затворена форма, т.е. първата и последната точка трябва да съвпадат. Топологичният редактор може автоматично да затваря многоъгълниците.

Тъй като ще се създава топология на мултиплексор, като се използват инвертор и NAND като стандарти клетки, за улеснение може всички клетки (инвертор и NAND) да се създадат с еднаква височина. Затова е необходимо шините за захранване и земя да имат еднаква ширина.

3. Добавяне на връзки към захранване.

Многоъгълникът, който ще се използва като шина за захранване, трябва да има размерите посочени на фиг. 5.



Фиг. 5. Размери на шината за захранване

1) Избира се горният край на PMOS транзистора (фиг. 5) като се огражда желаната област чрез натискане и влачене на десен бутон на мишката.
2) Използва се линийката (ruler) или бутон “**k**” от клавиатурата, за да се начертаят водещите линии – по една линия от 2 микрона от ляво и от дясно на **PPLUS** слоя на PMOS транзистора (вж. фиг. 5) и една от 0.7 микрона от горната страна на слоя. Продължава се линийката от 0.7 микрона с още една с дължина 2 микрона като това ще бъде ширината на захранващата шина.
3) В **LSW** се избира **MET1 dg**.

4) Използва се командата **Create** ⇒ **Polygon** (бутон “**P**” от клавиатурата) и се натиска **F3** за достъп до допълнителните опции на командата. В отворената форма се променя параметърът **Snap Mode** на **L90Xfirst**.

Параметърът **Snap Mode** контролира начина, по който елементите ще се прикрепват и движат по грида, когато се изчертава многоъгълникът. Опцията **L90** създава два сегмента, разположени на 90 градуса един от друг между всеки две точки, които се въвеждат, а **Xfirst** определя първия сегмент да бъде успореден на оста X.

5) Във **VLEW** многоъгълникът се изчертава от горния ляв ъгъл на сорса на PMOS транзистора и се продължава като се следват начертаните с линийката размери (фиг. 5).
6) Завършването на многоъгълника става с двукратно натискане на левия бутон на мишката и натискане на **Esc** от клавиатурата за прекратяване на командата.

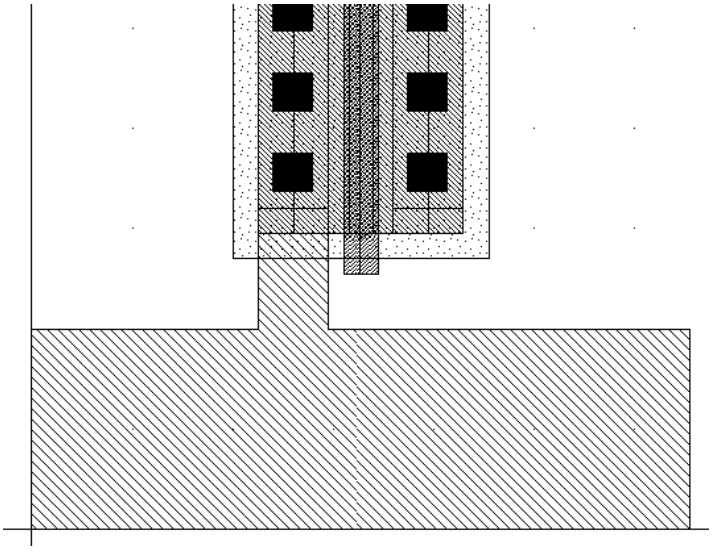
Ако се допусне грешка при изчертаване, не е необходимо да се започва отново. Възможно е връщане произволен брой точки назад като се натиска бутонът Backspace **от клавиатурата.**

За изтриване на начертаните линии с размери се натискат бутоните Shift **и** k **от клавиатурата едновременно**.

4. Добавяне на връзки към маса

За да се създаде връзката към маса, се използва командата за копиране, тъй като тя може да бъде с размери еднакви на тази за захранването.

1) Избира се **Edit** ⇒ **Copy** (бутон “**c**” от клавиатурата) и се натиска **F3** за достъп до допълнителните опции на командата.
2) Избира се вече начертаният многоъгълник на първи метален слой.
3) Във формата на командата за копиране се натиска бутонът **Upside down**. Копието на многоъгълника се обръща спрямо оста X.
4) Премества се курсорът в долния ъгъл на сорса на NMOS транзистора, с което се премества и многоъгълника (фиг. 6).
5) Копието на многоъгълника се разполага като се натисне левият бутон на мишката. С бутонът **Esc** от клавиатурата се прекратява командата за копиране.



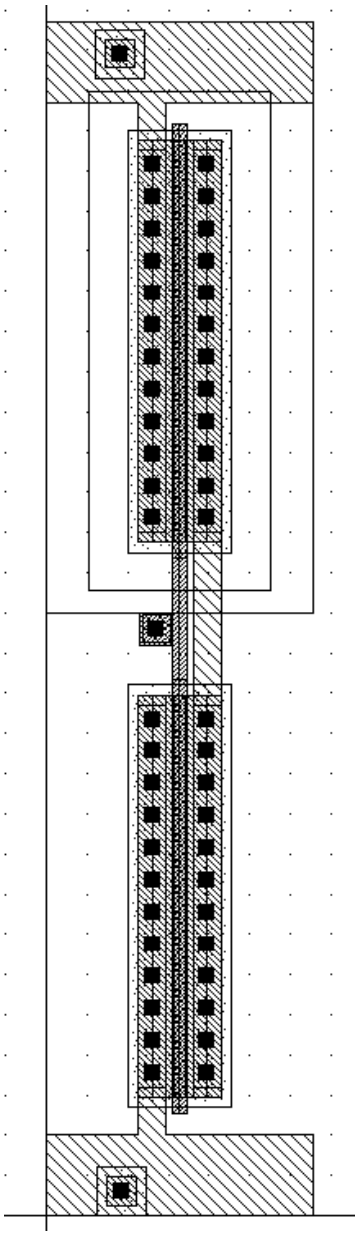
Фиг. 6. Шина за маса на инвертора

Добавяне на N-джоб и контакти към него и подложката.

Следващата стъпка при създаването на топологията на инвертора е да се уголеми N-джобът на PMOS транзистора като се начертае допълнителен правоъгълник на слой **NTUB dg**.

5. Добавяне на N-джоб

1) В **LSW** се селектира слоят **NTUB dg**.
2) Стартира се командата **Create Rectangle** като се натиска бутонът "**r**" от клавиатурата.
3) Началната точка на правоъгълника е горният ляв ъгъл на шината за захранване, а за крайна точка се избира пресечната точка на края на N-джоба на PMOS транзистора и десният ъгъл на шината за захранване (фиг. 7).
4) Изчертава се правоъгълникът и се прекратява командата.



Фиг. 7. Топология на инвертора

След като се създаде даден обект във VLE, той може да се редактира. За да се направи това, обектът първо се селектира. Има два режима на селектиране: пълен (full**) и частичен (**partial**). При първия се избира цял обект. Във втория режим може да се избира само една страна или ъгъл на обект. За преминаване от един режим в друг се използва** F4 **от клавиатурата. Текущият режим се изписва в горния ляв ъгъл на прозореца на** VLE**.**

Full mode: (**F**) Select:0
Partial mode: (**P**) Select:0

6. Добавяне на контакти към джоба и подложката.

Подложката на PMOS транзистора трябва да се свърже към захранването. За да се направи това, е необходимо да се постави подходящ контакт върху захранващата шина.

1) Избира се командата **Create** ⇒ **Contact** (бутон "**o**" от клавиатурата).
2) Във формата **Create Contact** се избира контактът **ND_C**. Използват се неговите параметри по подразбиране.
3) Контактът се поставя върху захранващата шина, както е показано на фиг. 7.

По аналогичен начин се добавя контакт към подложката на NMOS транзистора. Използва се контакт **PD_C** и се поставя върху шината за земя.

С тази команда топологията на инвертора е завършена. Остава да се направи проверка дали са спазени всички правила за физическо проектиране.

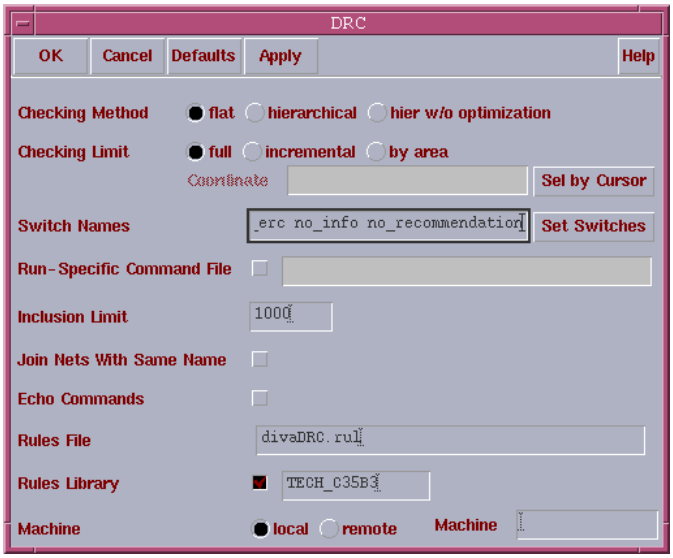
3. Проверка на правилата за физическо проектиране.

Интерактивният продукт за проверка на правилата за физическо проектиране – **Design Rule Checker (DRC)** е част от интерактивните продукти за проверка на **Assura**. Той използва правила за проверка, зададени във файла **divaDRC.rul**.

DRC маркира откритите грешки като създава многоъгълници около тях. Тези многоъгълници (маркери на грешки) се създават на специален "мигащ" слой, запазен за тях. DRC премахва маркерите за грешки автоматично след поправка на грешките и повторно пускане на DRC.

Стартиране на DRC

1) В прозореца на **VLE** се избира командата **Verify** ⇒ **DRC** и се появява формата за проверка на правилата за физическо проектиране (фиг. 8).



Фиг. 8. Форма за стартиране на DRC проверка

2) Натиска се бутонът **Set Switches** и се избират следните ключове – **grid, no_erc, no_info, no_recommendations, no_coverage** (фиг. 8). За избирането на повече от един ключ се натиска и задържа, докато приключи селекцията, бутонът **Ctrl** от клавиатурата. Ключовете определят кои проверки да бъдат изпълнени и кои не.
3) С натискане на бутонът **OK** се стартира проверката.

Ако са следвани точно всички стъпки, не би трябвало да има грешки. В **CIW** прозореца се изписва:

Total errors found: 0

Ако все пак възникнат някакви грешки, се виждат “мигащи” многоъгълници, които маркират всяка грешка.

За да бъдат открити и разпознати грешките, може да се използват следните команди:

Verify -> Markers -> Find

Verify -> Markers ->Explain.

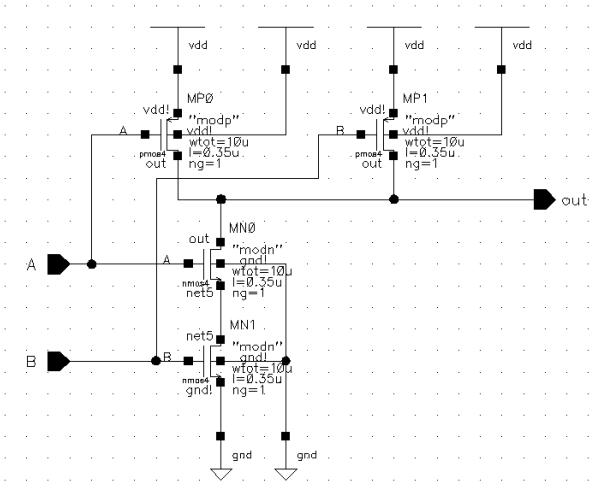
Изтриването на маркерите става чрез командата **Verify** ⇒ **Markers** ⇒ **Delete All**.

С това проектирането на топологията на инвертора е завършена и топологичното представяне трябва да се запази. За целта се използва командата **Design** ⇒ **Save**, или се натиска бутонът **F2** от клавиатурата.

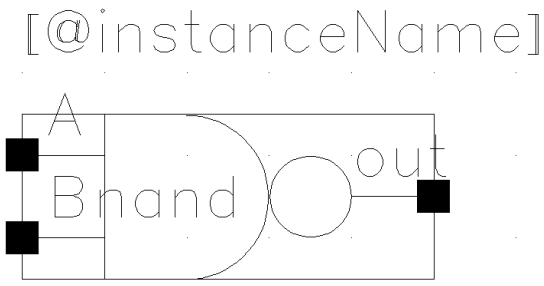
Затваря се прозорецът на **VLE**.

III. Проектиране на NAND.

1. Създаване на схемно и символно представяне на NAND.



Фиг. 9. Схемно представяне на NAND



Фиг. 10. Символно представяне на NAND

Аналогично на създаването на схемата и символа на инвертора, се създават тези представяния и за клетката на NAND. Те трябва да се направят в същата библиотека, в която е създаден и инверторът. Схемата на NAND-а е показана на фиг. 9.

След това се създава и символното представяне на NAND. Използва се вече известната команда **Design** ⇒ **Create Cell View** ⇒ **From Cell View**.

Променя се формата на символа, както е показано на фиг. 10. Запазват се двете представяния и се затварят техните прозорци.

2. Създаване на топология на NAND.

1) В **LMW** се избира командата **File** ⇒ **New** ⇒ **Cell View**.

2) Попълва се формата:
Library: <вашата_библиотека>
Cell Name: NAND
View Name: layout
3) Натиска се бутонът **OK**.

Отваря се прозорецът на **VLE** и отново се избира първи квадрант от работната площ.

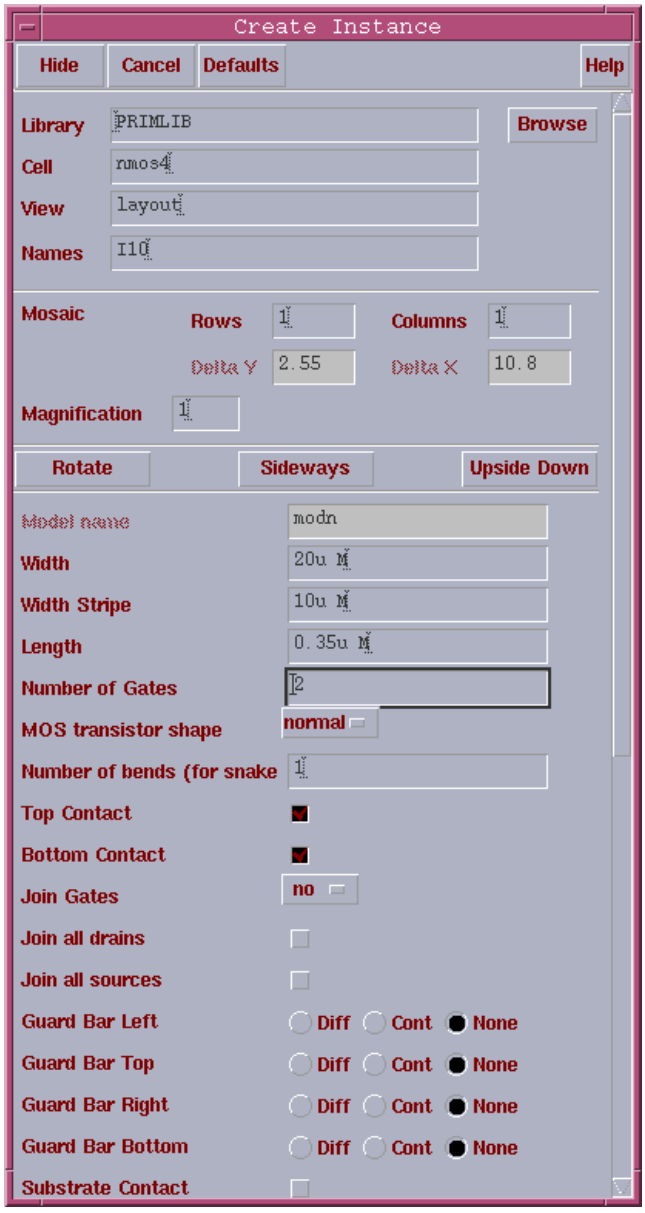
Добавяне на N- и PMOS транзистори.

Използва се командата за добавяне на елементи **Create** ⇒ **Instance**, за да бъдат добавени N- и PMOS транзисторите.

1) Избира се **nmos4** от библиотеката **PRIMLIB**. Този път обаче, се променят някои от параметрите на транзистора. Задава се ширина на канала (**width**) - 20μm и брой гейтове (**Number of Gates**) - 2 (фиг. 11). Изключват се следните параметри:

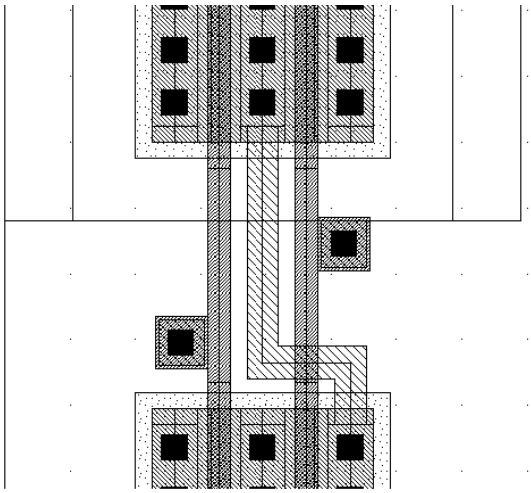
Join all gates **no**
Substrate contacts **-**
Join all drains **-**
Join all sources **-**

2) Преди да бъде поставен транзисторът на координати X=5.7 и У=3, той трябва да се завърти на 90 градуса (едно натискане на десния бутон на мишката).
3) Добавя се и PMOS транзистор (**pmos4**). Неговите параметри са същите като на **nmos4** (фиг. 11).
4) Поставя се елементът на координати X=5.7, У=17. Не трябва да се забравя да се завърти транзисторът.
5) Командата за добавяне на елемент се прекратява като се натисне бутонът **Esc** от клавиатурата.



Фиг. 11. Параметри на транзисторът

Свързване на входовете и изходите на NAND.



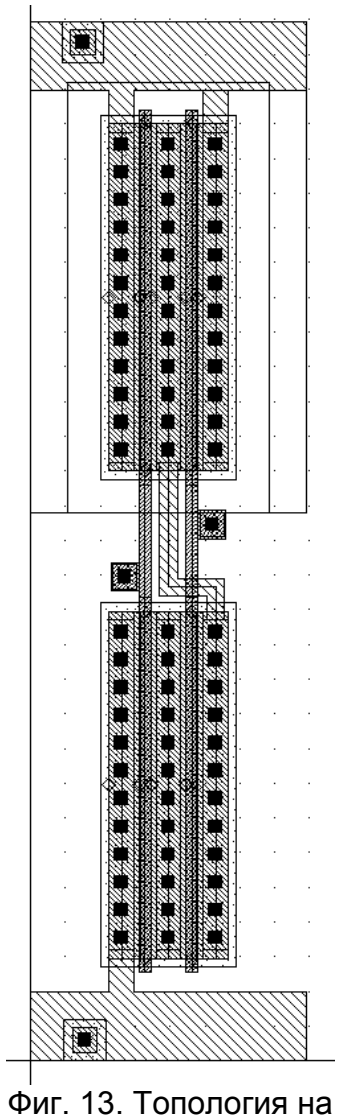
Фиг. 12. Свързване на изходите и входовете на NAND

- Свързване на входовете.

1) В **LSW** се избира слоят **POLY1 dg**.
2) Натиска се бутонът “**p**” от клавиатурата, за да се стартира командата за създаване на шини.
3) Свързват се гейтовете на N- и PMOS транзисторите (вж. за сравнение схемата на фиг. 12).
4) Добавят се контактите между слоевете **POLY1-MET1**.

- Свързване на изходите.

1) Използва се линийката (бутон “**k**” от клавиатурата), за да се начертае линия с дължина 1 микрон от горния край на общия контакт за NMOS транзисторите и още една линия със същата дължина от средата на десния контакт на дрейна на NMOS транзисторите. Тези линии се използват като отправна точка при свързването на изходите.
2) В **LSW** се избира слоят **MET1 dg**.
3) Стартира се командата за създаване на шини (бутон “**p**” от клавиатурата) и се свързват изходите, като се започне от общия дрейн за PMOS транзисторите.
4) Натиска се левият бутон на мишката, когато се достигне първата линия, за да се смени посоката и завие надясно (фиг. 12).
5) По аналогичен начин, използвайки втората линийка, се завършва шината до десния дрейн на NMOS транзистора, както е показано (фиг. 12).
6) Прекратява се командата **Create Path** с бутона **Esc** от клавиатурата.



Фиг. 13. Топология на NAND

Добавяне на шини за захранване и маса.

- Добавяне на шини за захранване.

Добавянето на шина за захранване става по същия начин както при инвертора (използва се командата за създаване на многоъгълник). Шината за захранване трябва да излиза с по 2 микрона от двете страни **PPLUS** слоя на PMOS транзисторите и с 0.7 микрона от горния край на този слой. Ширината на шината е 2 микрона (фиг. 13).

Не трябва да се забравя да се свърже и сорса на втория PMOS транзистор към захранващата шина (бутон "r" **от клавиатурата). Използва се командата** Create Rectangle **(бутон** "r"**).**

Линийките се изтриват след изчертаването на многоъгълника отново с команда shift **+** k**.**

- Добавяне на шини за маса.

Копира се шината за захранване и се завърта по хоризонталната ос (както при създаването на шина за маса на инвертора). Поставя се копираната шина, така че да се допре до левия контакт на сорса на NMOS транзистора (фиг. 13).

Добавяне на N-джоб.

1) Избира се слоят **NTUB dg** в **LSW**.
2) В прозореца на **VLE** се натиска бутонът "**r**".
3) Начертава се правоъгълник, подобно на този на инвертора (фиг. 13).

Добавяне на контакти към подложката и джоба.

Вече е позната командата за добавяне на контакти. Използва се контактът **ND_C** за свързване на подложките на PMOS транзисторите към захранване и **PD_C** за подложките на NMOS транзисторите (фиг. 13).

3. Проверка на топологията на NAND.

Прави се проверка на правилата за физическо проектиране **DRC**, както при инвертора. След това се запазва и затваря топологията на NAND-а.

IV. Проектиране на мултиплексор

В тази част на упражнението се прави йерархичен проект на мултиплексор. Йерархичният проект съдържа елементи, които от своя страна представляват отделни схеми или подсхеми. Целта е да се създаде мултиплексор на базата на направените по-рано клетки на инвертор и NAND.

1. Създаване на схема на мултиплексор.

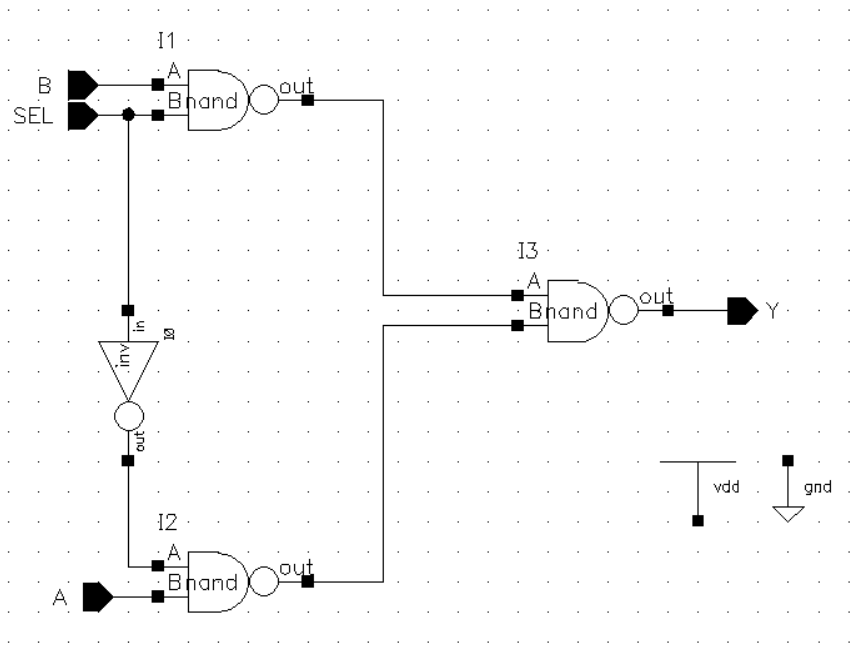
В библиотеката, в която се намират инверторът и NAND, се създава клетка, която съдържа схемата на мултиплексор (MUX), показана на

фиг. 14, като се използват символните представяния на инвертор и NAND.

В таблица 2 могат да се видят всички елементи, който са необходими за начертаването на схемата на мултиплексор, заедно с имената на библиотеките и вида представяне, което трябва да се използва. След като се създаде, схемата се запазва и се минимизира прозорецът на схемния редактор.

Таблица 2.

Име на библиотека	Име на елемент	Представяне
<вашата_библиотека>	NAND	symbol
<вашата_библиотека>	NAND	symbol
<вашата_библиотека>	NAND	symbol
<вашата_библиотека>	INV	symbol
analogLib	vdd	symbol
analogLib	gnd	symbol



Фиг. 14. Схема на мултиплексор

2. Създаване на топология на мултиплексор.

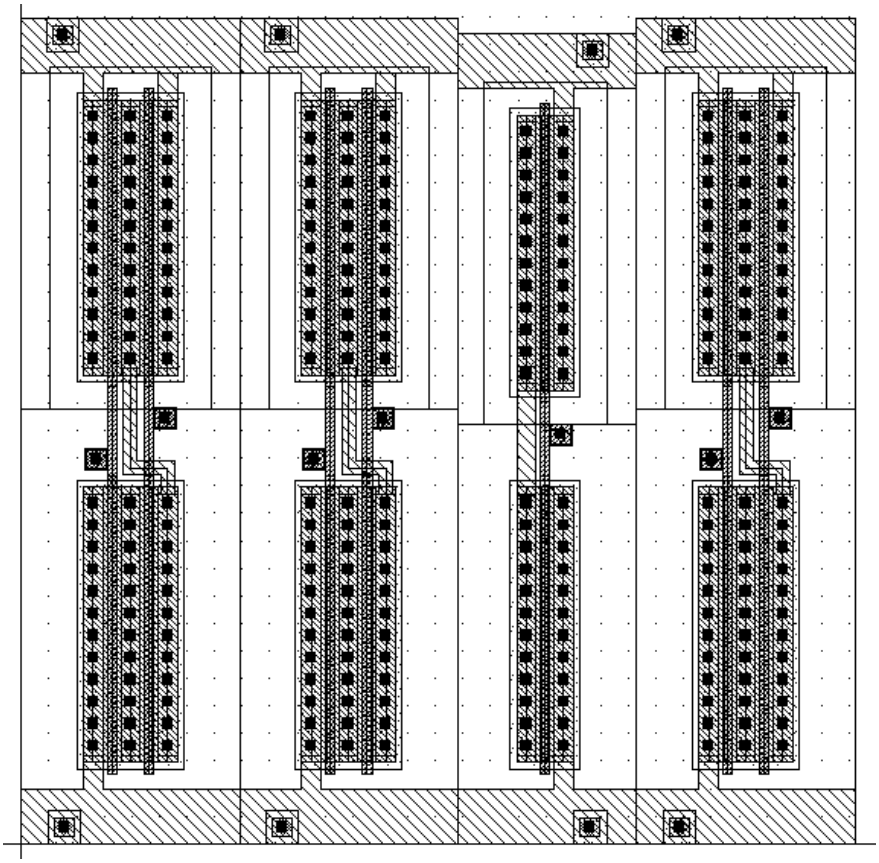
1) За да се създаде топологичното представяне на мултиплексора, в **LMW** се използва командата **File** ⇒ **New** ⇒ **CellView**.
2) В **VLEW** се извиква командата за добавяне на елементи (бутон "**i**" от клавиатурата) и се добавя един NAND. Разполага се в началото на първи квадрант (координати X =0, У = 0).
3) Поставя се отдясно до него втори NAND.
4) Натиска се бутонът "**f**", за да се видят двата NAND-а максимално уголемени на екрана.
5) Добавя се до тях един инвертор. Инверторът трябва да се завърти по оста Y, за да се улесни свързването.

Има два начина да се завърти инверторът. Първо във формата Add Instance **на командата, когато се добавя инверторът може да се натисне бутонът sideways**. **Другия начин е, когато елементът вече е разположен, той се селектира и се натиска бутонът** ‘q’ **от клавиатурата. В прозореца на формата** Edit Instance Properties **се променя полето** Rotation **на ‘**MY**’ и се натиска** OK**.**

6) Натиска се бутонът **Esc** от клавиатурата, за да се прекрати текущата команда.

За да се завърши топологията на мултиплексора, е необходимо да се добави още един NAND и да се начертаят връзките между елементите. Копира се един NAND и се добавя от дясно на инвертора.

7) Забелязва се, че инверторът е по-малък от елементите NAND (фиг. 15). За да се коригира това, е необходимо да се разтегне горната му половина. Тази редакция не може да се направи направо в топологията на мултиплексора. Може да се отвори клетката на инвертора в отделен прозорец, но така няма да може да се наблюдава как ще бъде разположен инверторът спрямо другите клетки. Най-добрият вариант в случая е да се използва командата **Edit in Place**. Тя позволява да се редактира клетката на инвертора на място вътре в топологията на мултиплексора.



Фиг. 15. Топология на мултиплексор

Редактиране на инвертора.

За да се отвори клетката на инвертора в режим на редактиране, се използва командата **Edit in Place**.

1) Избира се командата **Design** ⇒ **Hierarchy** ⇒ **Edit in Place**.

Топологичният редактор издава насочващо съобщение да се посочи формата, която ще се редактира.

2) Натиска се веднъж с левия бутон на мишката върху захранващата шина или шината за маса на инвертора на **MET1**.

Сега клетката на инвертора е отворена за редактиране. Въпреки че, може и да не се забелязва някаква промяна в топологичния редактор, ако се погледне лентата с името на **VLE** може да се види, че там е изписано името на инвертора, а не на мултиплексора.

3) Избира се командата **Window** ⇒ **Fit Edit**.

Във **VLE** топологията на инвертора се уголемява максимално, а очертанията на клетката му са осветени. Това потвърждава, че настоящият режим е за редактиране на клетката на инвертора, като все още се вижда как той е разположен в общата топология на мултиплексора.

- Разтягане на инвертора.

1) За да се разтегне горната част на инвертора, се използва командата **Edit** ⇒ **Stretch** (бутон *"***s***"* от клавиатурата).
2) Посочва се площта, която трябва да се разтегне. Натиска се левият бутон на мишката върху горния ляв ъгъл на шината за захранване и се очертава, без да се отпуска бутонът, един правоъгълник до долния десен ъгъл на N-джоба, след което се отпуска бутонът на мишката. Границите на площта, която ще се разтяга се осветяват.
3) В командния ред на **CIW** се появява съобщението:

Point at the reference point for the stretch.

Топологичният редактор често пита за отправна точка (**reference point**), когато изпълнява команди за редактиране. Това е началната точка, от която започва изпълнението на командата.

4) В случая за отправна точка се избира горната страна на многоъгълника като се натисне върху него с левия бутон на мишката. Премества се курсорът нагоре, докато границите на клетката на инвертора не се изравнят с тези на NAND клетките и отново се натиска левият бутон на мишката, за да се прекрати командата. Така всички клетки са разположени правилно, но инверторът все още е маркиран, което показва, че все още се редактира неговата топология, а не тази на мултиплексора.
5) Натиска се бутонът **Esc** от клавиатурата, за да се прекрати командата **Stretch**.

Връщане обратно в топологията на мултиплексора.

Докато се използва командата **Edit in Place** за една клетка, не може да се редактира топологията около нея (в случая, другите елементи на мултиплексора). След като се приключи с разтягането на инвертора, трябва да се премине отново едно ниво нагоре към топологията на мултиплексора.

1) Използва се командата **Design** ⇒ **Hierarchy** ⇒ **Return**.

При изпълнението й се проверява дали направените промени са запазени или не. Тъй като те не са запазени, се появява диалогов прозорец, в който се пита дали промените да бъдат запазени в топологията на инвертора.
2) Натиска се бутонът **yes**.

Създаване на връзките между елементите в схемата.

Ако четирите елемента са поставени правилно, шините за захранване и земя вече са създадени. Остава да се направят само връзките между елементите в мултиплексора. Могат да се начертаят както е показано на фиг. 16.

Докато се опроводява схемата е важно да се спазват правилата за проектиране като минимална ширина на шината, минимално разстояние между две шини, припокриване и др. Не трябва да се пресичат шини на един и същи слой.

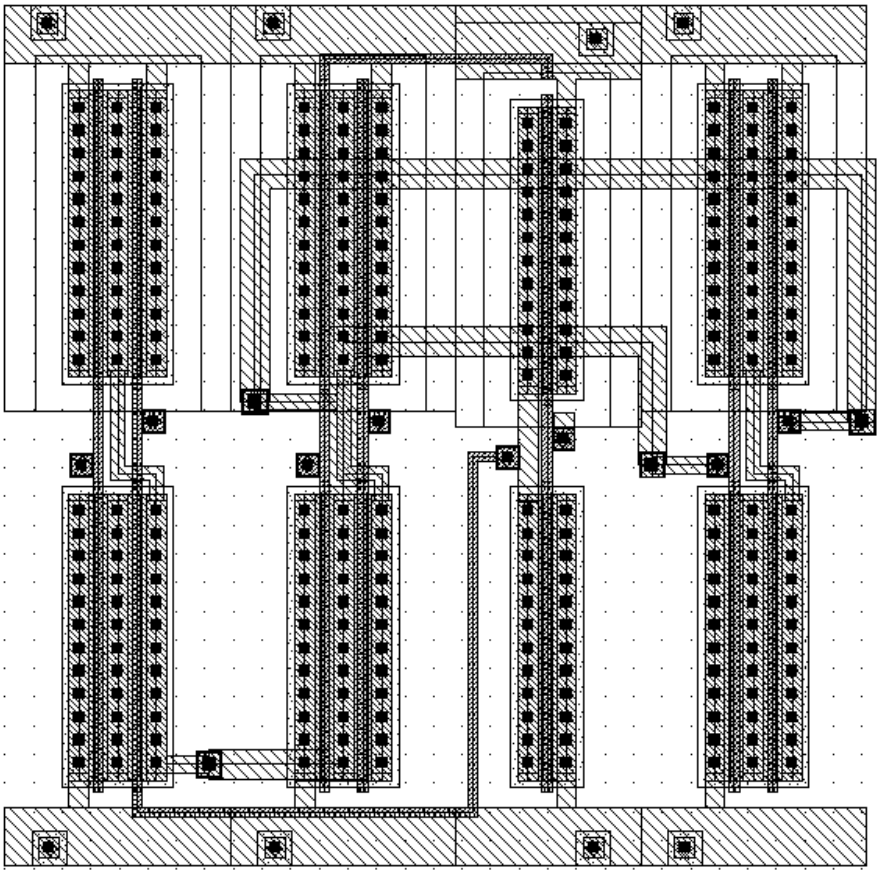
Начинът, по който курсорът се прикрепва към обектите и се движи се нарича гравитация (gravity**). По подразбиране гравитацията е включена. Това е полезно при създаването на елементи и устройства, но за по-лесно начертаване на свързващите шини е необходимо тя да се изключи. Това може да се направи или от менюто** Options **на VLE или чрез бутонът "**g**" от клавиатурата**.

Създаване на изводите (пиновете) на мултиплексора.

След като са начертани всички връзки, трябва във **VLE** към топологията да се добави информация, която се използва от други модули и инструменти на CADENCE. Трябва да се добавят пинове и имена на възли, които ще се използват например при откриване на грешки при стартиране на сравнението между схемата и топологията на мултиплексора – **Layout Versus Schematic** (**LVS**). CADENCE използва пиновете за:

- Да се определят връзките между йерархичните нива. Пинът показва къде дадена клетка може да се свърже към шина или друг компонент.
- Да се определят посоките на достъп при опроводяване.
- При **LVS** се проверява дали съществува конфликт между наименованията, поставени от потребителя, и възлите определени за пиновете.

В клетката на мултиплексора трябва да се създадат шест пина, които имат различни параметри (име, тип на пина, посока за достъп). Те са обобщени в таблица 3.



Фиг. 16. Топология на мултиплексора

Таблица 3.

	Име на пин	Тип (входен/изходен)	Посока за достъп	Слой
1.	vdd!	input/output	top, left, right	met1 dg
2.	gnd!	input/output	bottom, left, right	met1 dg
3.	A	input	top, bottom	met1 dg
4.	B	input	bottom	met1 dg
5.	SEL	input	bottom	met1 dg
6.	Y	output	right	met1 dg

За създаване на пинове:

1) В **LSW**, се избира слоят **MET1 dg**, след което във **VLE** се избира командата **Create** $\Rightarrow$ **Pin.**

Отваря се прозорец с формата за създаване на символен пин (**Creating Symbolic Pin**). В топологията на мултиплексора ще се използват пинове с определена форма, а не символи, и за целта във формата се натиска бутонът **Shape pin**.

2) След това в полето за имена на изводите се въвеждат названията на пиновете, като за разделител се използва пауза:
   vdd! gnd! A B SEL Y
3) Избира се опцията **Display Pin Name**, за да се визуализира името на конкретния пин на екрана.

4) Пиновете се разполагат един по един, като всеки път преди да се постави пин се задава посоката за достъп (вж. таблица 3).
5) Пинът за захранване например, ще има правоъгълна форма и ще съвпада с шината за захранване на мултиплексора. Затова изчертаването започва от горния ляв ъгъл и завършва в долния десен ъгъл на шината. След което се появява името на пина (**vdd!**) близо до курсора. Премества се курсора, така че **vdd!** да се пада в лявата страна на правоъгълника и се натиска още веднъж левият бутон на мишката, за да се разположи надписът. Във формата **Creating Shape Pin** се задават посоките за достъп за следващия пин (**gnd!**) и той се създава по аналогичен начин.
6) Създават се трите входни пина **A**, **B** и **SEL**. Мястото, където трябва да се разположат се вижда на фиг. 16.
7) Особеност на изходния пин (**Y**) е, че неговият правоъгълник покрива целия контакт на дрейна на последния NMOS транзистор.

Натиска се бутонът **Esc** от клавиатурата, за да се прекрати командата за поставяне на пинове. Запазва се клетката (бутон **F2** от клавиатурата).

Създаване на защитен пръстен (Guard Ring)

- Стартиране на **Layout Accelerator** (**Layout XL**).

За стартирането на топологичния редактор **Layout XL** във **VLEW** се избира командата **Tools** ⇒ **Layout XL**. Появява се форма за задаване на връзки **Define Connectivity Reference**. Натиска се бутонът **Cancel**, за да се затвори формата. Заглавната лента на прозореца се променя на **Virtuoso XL Layout**.

- За **Multipart Path**

За създаване на защитен пръстен около мултиплексора се използва командата **Multipart Path**. **Multipart Path** е ROD обект, който е съставен от една или повече части от ниво 0 на йерархията и използва един или повече различни слоеве. Състои се от една основна част (**master path**) и една или няколко подчасти (**subparts**). Основната част е от една страна обикновена шина на даден слой, а от друга - задаваща за подчастите. Всички подчасти се основават на нея.

- Преместване на мултиплексора.

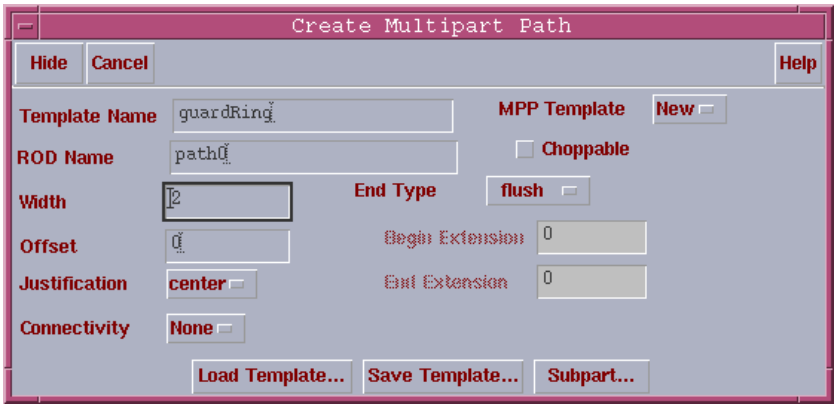
Преди да се създаде защитният пръстен, трябва да се направи място за него. Затова мултиплексорът трябва да се премести надясно и нагоре.

1) Избира се командата **Edit** ⇒ **Select** ⇒ **Select All**, за да се изберат всички елементи на мултиплексора.
2) Извиква се командата **Edit** ⇒ **Move**, и се посочват координатите X=0 и Y=0 като отправна точка на преместването.
3) Посочват се координатите X=8 и Y=2. Така мултиплексорът се премества надясно.

Създаване на защитен пръстен чрез Multipart Path.

Най-лесният начин за създаване на **Multipart Path (MPP)** е да се използва формата **Create Multipart Path** за създаване на шаблон, който може да се запази във файл и да се използва многократно. Защитният пръстен, който ще бъде направен в тази част от упражнението, ще се състои от една основна част (**master**), две подчасти (**enclosure subparts**), и група от подчасти (**subrectangles**).

- Създаване на шаблон за защитен пръстен.



Фиг. 17. Създаване на **Multipart Path**

1) Избира се слоят **PPLUS** в **LSW**. Това е дифузионен p+ слой, който ще се използва за основа (**master**).
2) В прозореца на **Layout XL** се избира командата **Create** ⇒ **Multipart Path**.

Появява се формата за създаване на **Multipart Path**, която е показана на фиг. 17.

3) Въвеждат се зададените в таблица 4 стойности на параметрите в съответните полета на формата. По време на попълването на формата в командния прозорец на **CADENCE CIW** се изписват предупреждения. Те не пречат на работата, затова могат да се игнорират.

Таблица 4.

MPP Template	New
ROD Name	guardRing
Choppable	off
Width	2
End Type	flush
Offset	0
Begin Extension	0
End Extension	0
Justification	center
Connectivity	None

4) Следващата стъпка е да се създаде първата подчаст (**enclosure subpath**) на слой **MET1**, като се натисне бутонът **Subpart** и се избере

Enclosure Subpath. Появява се формата от фиг. 18, където се въвеждат стойностите от таблици 5 до 7.

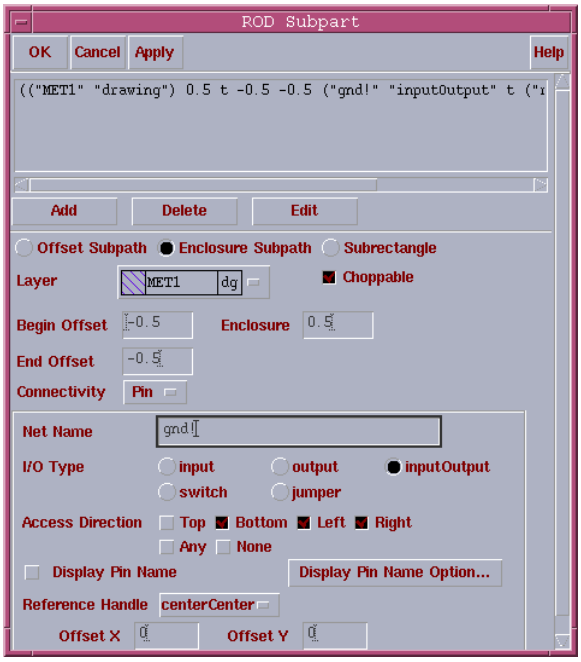
Таблица 5.

Layer	metal1
Choppable	on
Begin Offset	-0.5
Enclosure	0.5
End Offset	-0.5
Connectivity	Pin

Когато за връзка се избере **Pin**, се появява ново поле, в което се въвеждат следните стойности на параметрите:

Таблица 6.

Net Name	gnd!
I/O Type	inputOutput
Access Direction	Bottom, Left, Right
Display Pin Name	No
Reference Handle	centerCenter
Offset X	0
Offset Y	0



Фиг. 18. Форма на **Enclosure Subpath**

5) Натиска се бутонът **Add**, за да се запазят въведените данни.
6) Въвеждат се стойностите на параметрите от таблица 7:

Таблица 7.

Layer	*DIFF*
Choppable	*on*
Begin Offset	-0.5
Enclosure	0.5

End Offset	-0.5
Connectivity	*None*

7) Натиска се бутонът **Apply**, за да се запишат тези данни в шаблона без да се затваря формата.
8) Следващата стъпка е създаване на контакти като **subrectangles**. Във формата **ROD Subpart** се натиска бутонът **Subrectangle***.* Въвеждат се стойностите на параметрите дадени в таблица 8.

Таблица 8.

Layer	*CONT*
Choppable	*on*
Begin Offset	*-0.7*
Width	*0.4*

Със задаването на начално отместване (**Begin Offset**) и ширина (**Width**), системата автоматично присвоява стойности на параметрите **End Offset**, **Length**, и **Space**.

9) Натиска се бутонът **Add**, за да се запазят въведените данни.
10) Натиска се бутонът **OK**, за да се запазят въведените данни в шаблона и да се затвори формата.

- Запазване на шаблона.

1) Във формата **Create Multipart Path** се натиска бутонът **Save Template.**
2) Записва се **guardRing** в полето за име на шаблона.
3) Натиска се бутонът **OK.**

Запазването на шаблона се налага, защото след като се въведе последната точка на защитния пръстен въведените във формата данни се изтриват и не могат да се възстановят.

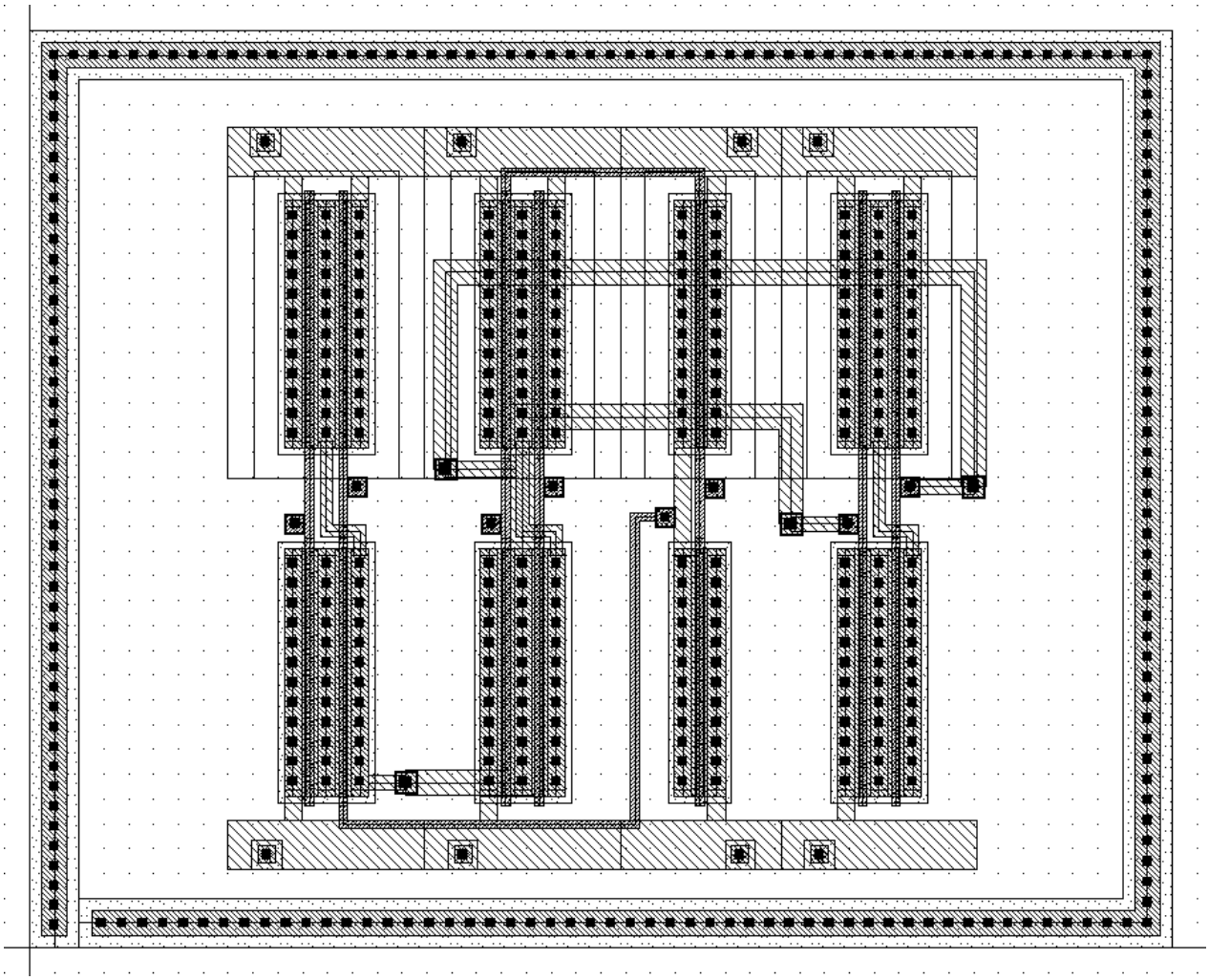
- Изчертаване на защитния пръстен

След като шаблонът е направен и запазен може да се премине към самото изчертаване на защитния пръстен около мултиплексора.

В прозореца на топологичния редактор се въвеждат посочените по-долу точки като се натиска по един път левият бутон на мишката, а за последната точка два пъти.

Първа точка: X=2, Y=1
Втора точка: X=45, Y=1
Трета точка: X=45, Y=36
Четвърта точка: X=1, Y=36
Пета точка: X=1, Y=0

Топологията на мултиплексора със защитен пръстен е показана на фиг. 19.



Фиг. 19. Топология на мултиплексора със защитен пръстен

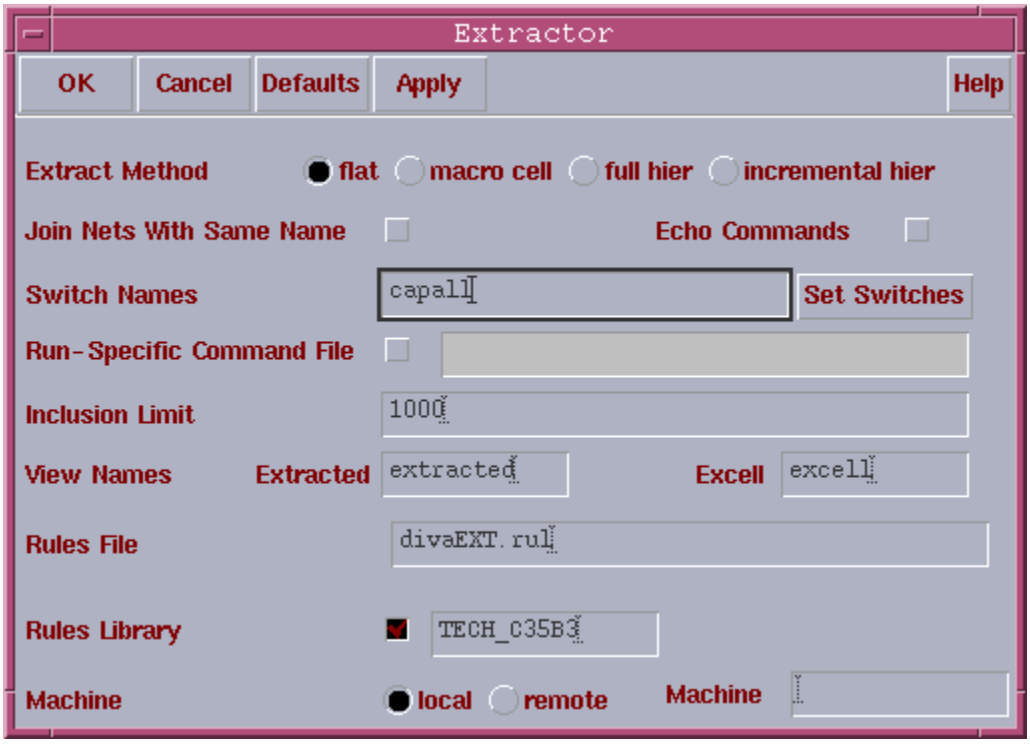
V. Проверки на топологията на мултиплексора

1. Стартиране на DRC

Проверката на правилата за проектиране се прави, както при инвертора и NAND. Задават се необходимите ключове и се стартира. Особеност в случая е, че трябва да се зададе опцията **Join Nets With Same Name**. Ако се получат съобщения за грешки в командния прозорец (**CIW**), те трябва да се коригират и да се стартира наново проверката.

2. Разпознаване на елементите и връзките между тях в топологията. Извличане на паразитни елементи.

Разпознаването на елементите и връзките между тях в топологията е необходимо, за да може да се сравни със схемното представяне. За тази верификация трябва да се пусне програмата за екстракция (**Extract**). Тя използва правила за разпознаване на устройства, дефинирани в **divaEXT.rul** файл и задава електрическите връзки между тях. Програмата създава представяне, наречено **extracted**.



Фиг. 20. Форма на програмата за екстракция

1) В прозореца на **VLE** се избира командата **Verify** ⇒ **Extract**. В **Extract** формата се задава ключът **capall** (така се задава и извличането на паразитни елементи от топологията), както е показано на фиг. 20. Отново се включва опцията **Join Nets With Same Name**.
2) Стартира се програмата за екстракция.

След като приключи екстракцията в командния прозорец **CIW** се изписва съобщението:

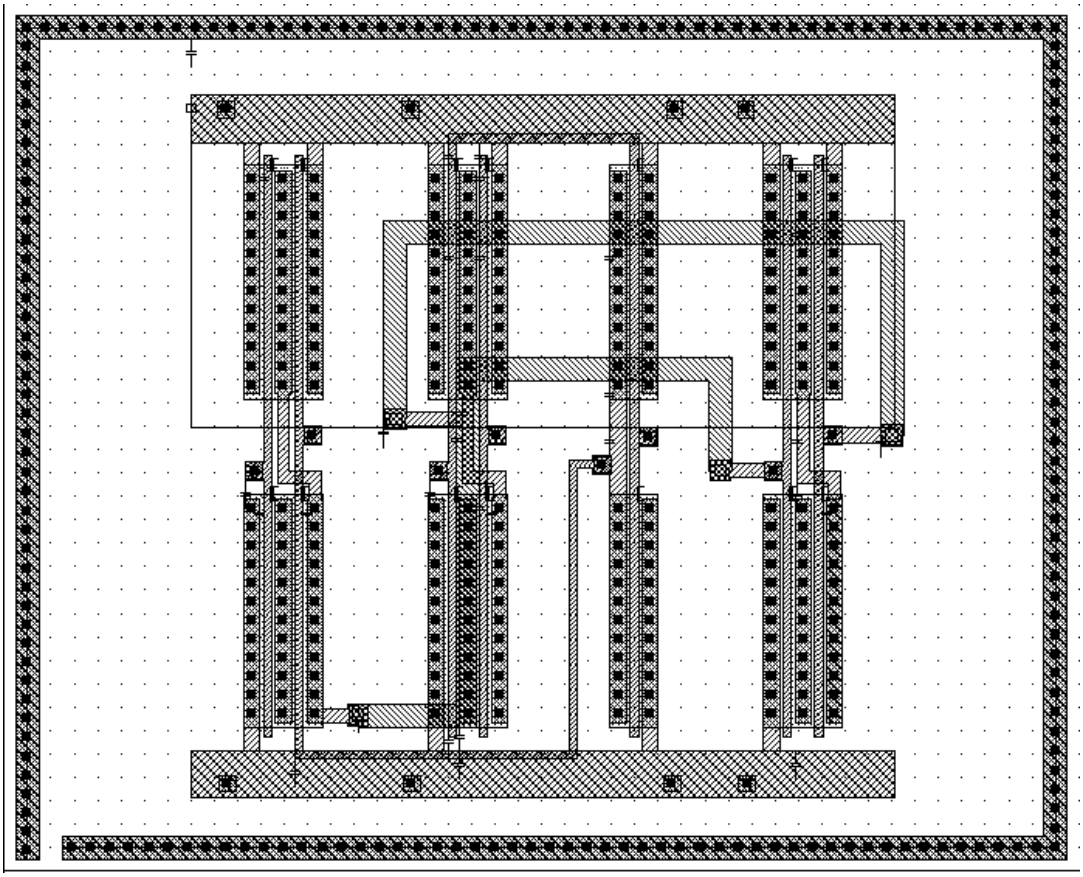
saving rep <име_на_вашата_библиотека>/mux/extracted

Това показва, че **extracted** представянето е създадено. То прилича на топологичното представяне, но не съвпада с него.

3) Отворя се **extracted** представянето на мултиплексора. То трябва да изглежда като това показано на фиг. 21. На всеки гейт, има сложен по един символ на транзистор, който показва, че там има разпознато устройство – транзистор.
4) Натискат се едновременно бутоните **Ctrl** и **f** от клавиатурата, за да се види само ниво 0.

Символите вече не се виждат и може да се прочете надписа в областта на всеки гейт. Транзисторите са означени или с **nmos4**, или с **pmos4**.

5) Натискат се едновременно бутоните **Shift** и **f**, за да се виждат отново всички нива.



Фиг. 21. **Extracted** представяне на мултиплексор

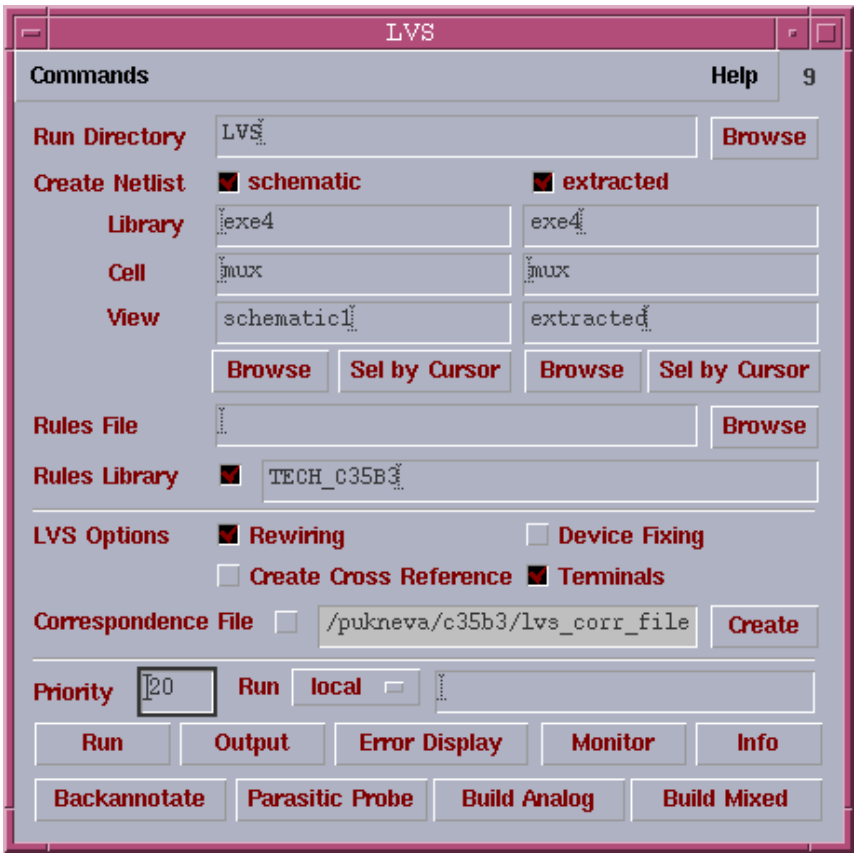
3. Сравнение на топологията със схемата (LVS)

Програмата **LVS** дава възможност да се направи сравнение между схематично и топологично (физическо) представяне и да се провери за грешки при опроводяването. **LVS** използва схематично и **extracted** представяне. Стартира се от прозореца на **extracted** представянето чрез командата **Verify** ⇒ **LVS**. Появява се прозорецът на формата за **LVS**. В нея се задават опциите показани на фиг. 22.

1) Задава се приоритет 20. По подразбиране той е 0, но ако се зададе такъв висок приоритет, това ще забави всички други процеси на системата.
2) Стартира се програмата **LVS**. Появява се прозорец с форма за запазване **Save Cell View**. Натиска се бутонът **OK,** за да се запази топологията на мултиплексора.
3) Програмата за **LVS** се стартира в скрит режим (**background**) и може да отнеме няколко минути. Когато приключи, се появява прозорец със съобщението **Analysis Job Succeeded**, което показва че сравнението е направено успешно.

Ако сравнението е неуспешно, натиска се бутонът **Info** във формата за **LVS**, за да се отвори лог (**log**) файла. Той съдържа всички

изпълнени команди при сравнението и в него може да се види каква е причината за дадена грешка при изпълнение на програмата.



Фиг. 22. Форма за стартиране на **LVS**

4) Ако сравнението е успешно:

Натиска се бутонът **Output** във формата за **LVS**. Отваря се текстов прозорец с изходните данни от сравнението. Преглежда се информацията, която съдържа файлът. Ако в него бъде открит ред:

The net-lists match.

Това означава, че топологията, която е създадена, съответства на схемата. Но ако съдържа:

The net-lists failed to match.

Топологията на мултиплексора не съответства на схемата. Вероятно са допуснати грешки при опроводяването. Тези грешки трябва да се коригират (вж. упражнение №6) и всички проверки трябва да се стартират отново.

Упражнение № 9

Създаване и редактиране на ROD обекти

Няколко думи за ROD обектите.

ROD или **R**elative **O**bject **D**esign (проектиране на обекти, които са свързани помежду си) позволява да се създават обекти и дефинират връзки между тях на високо ниво на абстракция, така че да се обърне внимание на специфичните особености на даден проект. **ROD** автоматично управлява сложни операции при преминаването през различни нива на йерархия и опростява изчисленията, които са необходими за създаването, разполагането и подравняването на геометрии.

На всеки обект от базата данни, който има име като елемент, топологично представяне на клетка или форма, автоматично се присвоява **ROD** информация, която се съхранява в **ROD** обект. **ROD** обектът също е обект от базата данни, но съществува по отношение на друг обект от базата данни, който носи определено име и с който е свързан. Всеки **ROD** обект има уникален номер (**ID**), с който може да се идентифицира.

ROD обектите, свързани с определена форма, елемент или представяне на клетка съдържат следната информация:

- hierarchical name (йерархично име);
- cellview ID (ID номер на представянето);
- database ID (ID номер на базата данни);
- transformation information – rotation, magnification, and offset (информация за преобразуване – ротация, увеличаване и отместване);
- alignment information, if any (информация за подравняване, ако съществува);
- number of segments /for shapes/ (брой сегменти);
- names and values of user-defined handles, if any (Имена и стойности на зададени от потребителя отправни точки за преобразуване на обекта – “хендъли”, ако такива съществуват);
- names of system-defined handles (имена на дефинирани от системата хендъли).

1. Създаване на правоъгълник като ROD обект.

Създаването на **ROD** обекти може да стане по два начина – чрез въвеждането на команди в управляващия прозорец на CADENCE (**CIW**) или графично като се използват възможностите на топологичния редактор (**Virtuoso Layout Editor - VLE**) и командата **Create Rectangle**. Когато се използва **VLE** за създаване на правоъгълник, първата стъпка е да се създаде ново топологично представяне, а ако е необходимо и нова

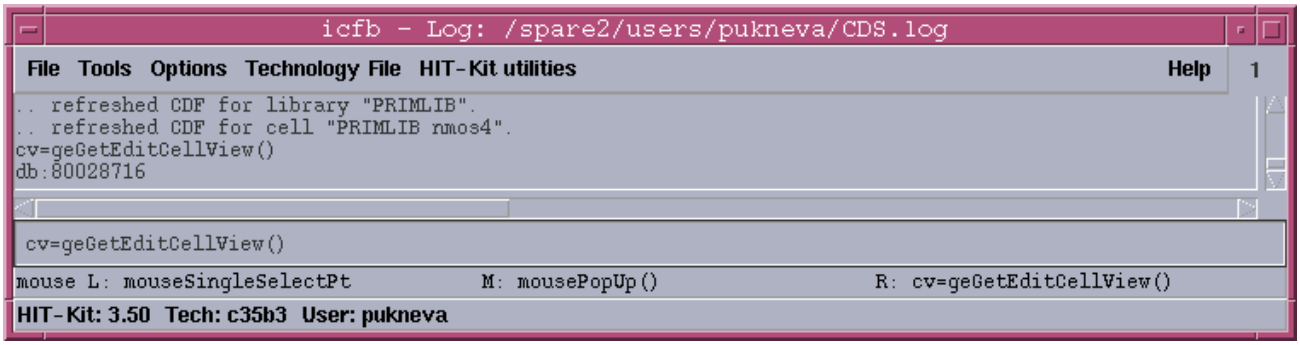
библиотека.

1) В прозореца на библиотечния браузър (**Library Manager Window** - **LMW**) се избира командата **File** ⇒ **New** ⇒ **Cellview**.
2) Задава се име на клетката (например **rod**) и вида на представянето й и се натиска бутонът **OK**:

Cell Name rod
View Name layout

3) ID номерът на новосъздадената клетка може да бъде видян като се напише следната команда в CIW прозореца (фиг. 1).

cv =geGetEditCellView()



Фиг. 1. Въвеждане на команди в **CIW** прозореца

4) Избира се слоят **POLY1** в прозореца за избор на слоеве (**LSW**).
5) Във **VLE** се избира командата **Create** ⇒ **Rectangle**. Появява се формата за създаване на правоъгълник.
6) Включва се опцията **As ROD Object**, която позволява редактиране на полето **ROD Name**.
7) Записва се **rect** в полето **ROD Name**.
8) Натиска се с левия бутон на мишката последователно на координати X=3, Y=11 и X=9, Y=9, за да се начертае правоъгълникът.
9) Натиска се бутонът **Esc**, за да се затвори формата за създаване на правоъгълник.

За да се създаде същия правоъгълник с команди в **CIW**, се използва програмният език на **CADENCE – SKILL.** Трябва да се въведе следното (това е само пример и не е необходимо да се въвежда):

```
rect =rodCreateRect(
?name “rect”
?cvId geGetEditCellView()
?layer “poly1”
?bBox list (3:11 9:9)
)
```

Изследване на ROD правоъгълника.

Информация за всеки **ROD** обект може да се получи като се отвори **Edit Properties** формата или се зададат някои команди в **CIW**.

Извеждане на информация за ROD обекта в Edit Properties формата.

Всички промени, който се правят с правоъгълника се записват във формата **Edit Properties**. За да се отвори тя:

1) Маркира се правоъгълникът.
2) Избира се командата **Edit** ⇒ **Properties**.

Появява се формата **Edit Properties**. В нея трябва да се види името и X и Y координатите, които са зададени във формата за създаване на правоъгълник.

3) Маркира се **ROD** в най-горната част на формата. Появяват се полетата, свързани с **ROD** обектите. Разглежда се внимателно информацията за правоъгълника, която те съдържат.
4) Сравняват се стойностите на хендълите **upperLeft** и **lowerRight** в полето за системни хендъли.

Стойностите трябва да са същите като координатите в полетата **Attribute**.

Извеждане на информация за ROD обекта в CIW.

За да бъде изведена информацията за **ROD** правоъгълника в **CIW**:

1) Проверява се ID номерът на **ROD** правоъгълника, като се напишат следните команди в CIW:

```
rect=rodGetObj("rect" geGetEditCellView())
rect~>??
```

Системата извежда списък с имена на параметри на правоъгълника и техните стойности.

2) Тази информация трябва да изглежда като дадената по-долу. Само ID номерът на правоъгълника, представянето и ID номерата на базата данни може да са различни.

```
("rodObj:50533216" name "rect" cvId db:60933164
      dbId db:60933372 transform
  ((0.0 0.0) "R0" 1.0) align
  nil numSegments 4 userHandleNames nil
  systemHandleNames
  ("width" "length" "lowerLeft" "lowerCenter" "lowerRight"
       "centerLeft" "centerCenter" "centerRight" "upperLeft" "upperCenter"
       "upperRight" "length0" "start0" "mid0" "end0"
       "length1" "start1" "mid1" "end1" "length2"
       "start2" "mid2" "end2" "length3" "start3"
       "mid3" "end3" "lengthLast" "startLast" "midLast"
       "endLast"  ))
```

Редактиране на ROD правоъгълника.

Предстои промяна на размерите на правоъгълника и проверка на резултата във формата **Edit Properties**.

1) Във **VLE** се размаркира правоъгълникът като се натисне левият бутон на мишката върху празно поле от работната площ.

2) Избира се командата **Edit** ⇒ **Stretch**. Появява се формата за разтегляне. Натиска се с левия бутон на мишката върху дясната страна на правоъгълника, за да започне разтеглянето. Като се премести курсора се очертава формата, която ще се получи след разтеглянето.
3) Премества се курсорът на координати X=11, Y=10 и се натиска бутонът **Enter** от клавиатурата за завършване.
4) Натиска се **Cancel**, за да бъде затворена формата.
5) Маркира се правоъгълникът. Избира се командата **Edit** ⇒ **Properties** и се отваря формата. Стойността за полето **Right** трябва да е променена от 9 на 11.
6) Размаркира се правоъгълника като се натисне левият бутон на мишката върху празно поле от работната площ.

2. Създаване на многоъгълник като ROD обект.

В тази част от упражнението се създава **ROD** многоъгълник. След което ще се разучат и редактират параметрите му, като се използва формата **Edit Properties**, в LSW се избира слоят **MET1**.

1) Във **VLE** се избира командата **Create** ⇒ **Polygon**. Появява се формата за създаване на многоъгълници. В нея се активира опцията **As ROD Object**, която позволява редактиране на полето **ROD Name**.
2) В полето ***ROD Name*** се записва **polygon**.

За да се начертае многоъгълникът, може да се въведат точките или направо в командния прозорец **CIW**, или с натискане на левия бутон на мишката в графичния прозорец.

За създаване на многоъгълника трябва да се направи едно от следните две неща:

– За да се въвеждат точки в **CIW** се премества курсорът в командния ред на **CIW** и се натиска веднъж с левия бутон на мишката, след което се въвеждат последователно X и Y координатите, както е показано по-долу. След въвеждането на всяка двойка координати се натиска бутонът **Enter** от клавиатурата:

11:11
11:7
17:7
17:9
13:9
13:11

– За да се създаде многоъгълника във **VLE**, се натиска последователно левият бутон на мишката на координатите изписани по-долу. След като се въведе последната точка (X=13, Y=11), или се натиска два пъти левия бутон на мишката, или един път заедно с **Enter** от клавиатурата.

First click: X = 11, Y = 11

Second click: X = 11, Y = 7
Third click: X = 17, Y = 7
Fourth click: X = 17, Y = 9
Fifth click: X = 13, Y = 9
Sixth click: X = 13, Y=11

3) Натиска се бутонът **Cancel,** за да се затвори формата за създаване на многоъгълници.

За да се създаде същият многоъгълник чрез команди на **SKILL** се записва следното в **CIW** прозореца. (Не е необходимо да се въвежда. Това е просто пример.)

```
polygon=rodCreatePolygon(
?name "polygon"
?cvId geGetEditCellView()
?layer "metal1"
?pts list(11:11 11:7 17:7 17:9 13:9 13:11)
)
```

Извеждане на информация за ROD многоъгълника.

По аналогичен начин, както при разглеждането на **ROD** правоъгълника, може да се получи информация и за многоъгълника.

1) Извежда се ID номера на **ROD** многоъгълника като се запише следното в **CIW**.

```
polygon=rodGetObj( "polygon" geGetEditCellView ( ) )
polygon~>??
```

Системата извежда списък с имена на параметри на многоъгълника и техните стойности.

2) Изведената информация трябва да е подобна на тази:

```
("rodObj:50533228" name "polygon" cvId db:60933164
   dbId db:60933624 transform
   ((0.0 0.0) "R0" 1.0) align
   nil numSegments 6 userHandleNames nil
   systemHandleNames
   ("width" "length" "lowerLeft" "lowerCenter" "lowerRight"
      "centerLeft" "centerCenter" "centerRight" "upperLeft" "upperCenter"
      "upperRight" "length0" "start0" "mid0" "end0"
      "length1" "start1" "mid1" "end1" "length2"
      "start2" "mid2" "end2" "length3" "start3"
      "mid3" "end3" "length4" "start4" "mid4"
      "end4" "length5" "start5" "mid5" "end5"
      "lengthLast" "startLast" "midLast" "endLast"
   ) )
```

Работа с хендъли на ROD обекти.

Вече е известно как да се създават **ROD** обекти – правоъгълници и многоъгълници и как да се извеждат и променят техните параметри. Важен параметър на **ROD** обекта са неговите хендъли. Те се използват за

съхраняване на координати на точки, изчисления и друга информация. В полето **ROD** на формата **Edit Properties**, може да се видят имената и стойностите на всички хендъли.

1) Избира се многоъгълникът.
2) Маркира се опцията **ROD**.
3) Използва се диаграмата от фиг. 2, за да се определи кои точки съответстват на стойностите на системните хендъли ***start0***, ***start3***, и ***start5***.

Фиг. 2. Диаграма на системните хендъли в един многоъгълник

Редактиране на ROD многоъгълник

Формата на многоъгълника може да се променя като се редактират стойностите на точките, зададени като координати във формата **Edit Properties**.

1) Във формата се избира параметърът **Attribute**.
2) В полето **Points** се променя 11:11 на 11:13 и 13:11 на 13:13.
3) Натиска се бутонът **Apply**. Формата на многоъгълника се променя в съответствие с новите точки.
4) Маркира се опцията **ROD**. Стойностите за **start0** и **start5** трябва да са променени.
5) Размаркира се многоъгълникът.

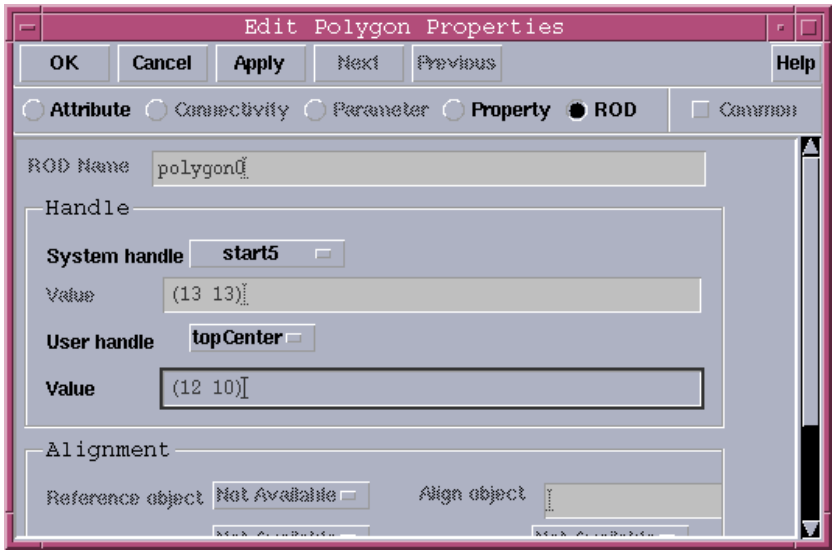
Създаване на хендъли от потребителя.

Когато се дефинира хендъл, се задава име и му се присвоява стойност. Стойностите на хендълите дефинирани от потребителя се съхраняват в базата данни.

1) За създаване на собствен хендъл в **CIW** се записва:

```
rodCreateHandle(
?name "topCenter"
?type "point"
?value 12:10
?rodObj polygon
)
```

2) Маркира се отново многоъгълникът. При разглеждане на параметрите му се забелязва, че в полето **User handle** се е появил новосъзданият хендъл със зададената стойност (фиг. 3).



Фиг. 3. Форма за редактиране на параметри на многоъгълника

3) Размаркира се многоъгълника.

3. Подравняване на ROD обекти. Подравняване на многоъгълник и правоъгълник.

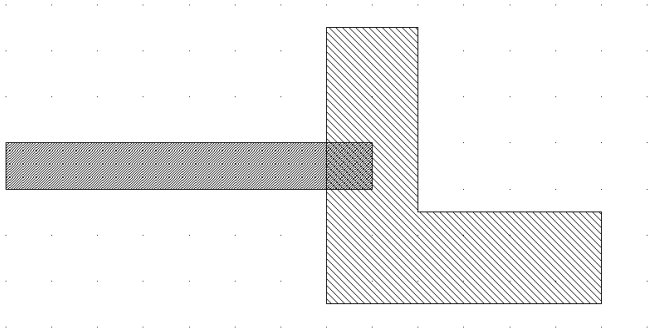
Важна особеност на **ROD** обектите е способността им да задават точка на подравняване за един обект, спрямо която може да се разположи друг обект. Това се нарича “свързано подравняване” (relative alignment). Обикновено подравняването става като се посочат хендъли за двата обекта. Също така може да се задава и разстояние между два обекта, спрямо оста X, Y или и двете. Подравняването между двата обекта се запазва, докато се работи, с който и да е от обектите, или при запазване и затваряне на топологичния редактор.

В тази част от упражнението се свързват и подравняват правоъгълникът и многоъгълникът, създадени по-рано. Като отправен обект се използва правоъгълникът, а отправна точка е хендълът **centerRight**. Обектът, който ще се подравни, е многоъгълника, като се използва неговия хендъл **topCenter**. Това всъщност е създадения в стъпките по-горе хендъл.

1) В **CIW** се въвеждат следните команди:

```
rodAlign(
?alignObj polygon
?alignHandle "topCenter"
?refObj rect
?refHandle "centerRight"
)
```

Така правоъгълникът и многоъгълникът са подравнени спрямо хендълите **centerRight** (за правоъгълника) и дефинирания хендъл **topCenter** (фиг. 4).



Фиг. 4. Подравняване на правоъгълника и многоъгълника

2) Селектира се многоъгълникът.
3) В полето **Alignment** на формата **Edit Properties** информацията трябва да съвпада с въведената в **CIW** прозореца.
4) Маркира се правоъгълникът.

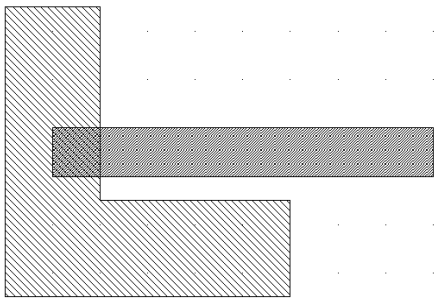
Редактиране на подравнени обекти

За да се провери как подравнени обекти остават свързани помежду си при работа с един от тях, се променя стойността на хендъла, с който е подравнен правоъгълникът. Премества се многоъгълникът и се разтяга правоъгълникът.

Промяна на хендъла, с който е подравнен правоъгълника.

Промяна на хендъла, спрямо който е подравнен правоъгълника, може да стане като се зададе друга стойност в полето **Align handle** във формата за редактиране на параметрите на правоъгълника.

1) Маркира се правоъгълникът.
2) В полето **Alignment** на **Edit Properties** формата се променя полето **Align handle** от **centerRight** на **centerLeft**.

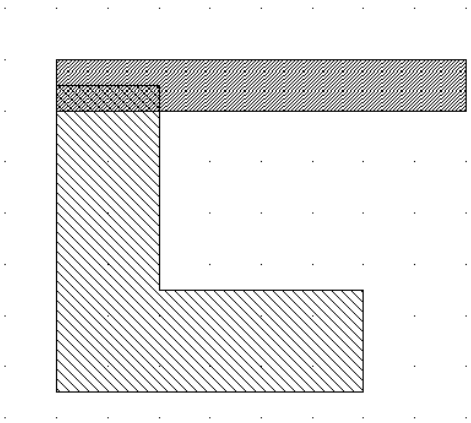


Фиг. 5. Подравняване на многоъгълника и правоъгълника

3) Натиска се бутонът **Apply**.

Формата показва, че хендълът **centerLeft** на правоъгълника и отделно дефинираният **topCenter** на многоъгълника са подравнени, както е показано на фиг. 5.

4) Променя се полето **Reference Handle** от **topCenter** на **start0**.
5) Натиска се бутонът **Apply**.



Фиг. 6. Подравняване на правоъгълника и многоъгълника

Хендълът **centerLeft** на правоъгълника и **start0** на многоъгълника се подравняват, както е показано на фиг. 6.

6) Размаркира се правоъгълникът като се натисне с левия бутон на мишката някъде в празното пространство на работната площ.

Преместване на подравнени обекти.

За да се премести многоъгълникът, трябва да се направи следното.

1) Избира се командата **Edit** ⇒ **Move**.
2) Селектира се многоъгълникът.
3) Премества се многоъгълникът някъде в работната площ. Правоъгълникът остава подравнен с него.
4) Натиска се бутонът **Cancel**.
5) Размаркират се всички обекти като се натисне с левия бутон на мишката някъде в празното пространство на работната площ.

Разтягане на подравнени обекти

1) За да се разтегне правоъгълникът, се избира командата **Edit** ⇒ **Stretch**.
2) Натиснка се с левия бутон на мишката върху лявата страна на правоъгълника.
3) Премества се курсорът малко наляво и се натиска бутонът **Enter** от клавиатурата.

Правоъгълникът се разтяга наляво, а многоъгълникът се премества, за да остане подравнен спрямо хендъла **centerLeft** на правоъгълника.

4) Затваря се прозорецът.

Упражнение № 10

Среда за автоматизирано проектиране на цифрови ИС с висока степен на интеграция

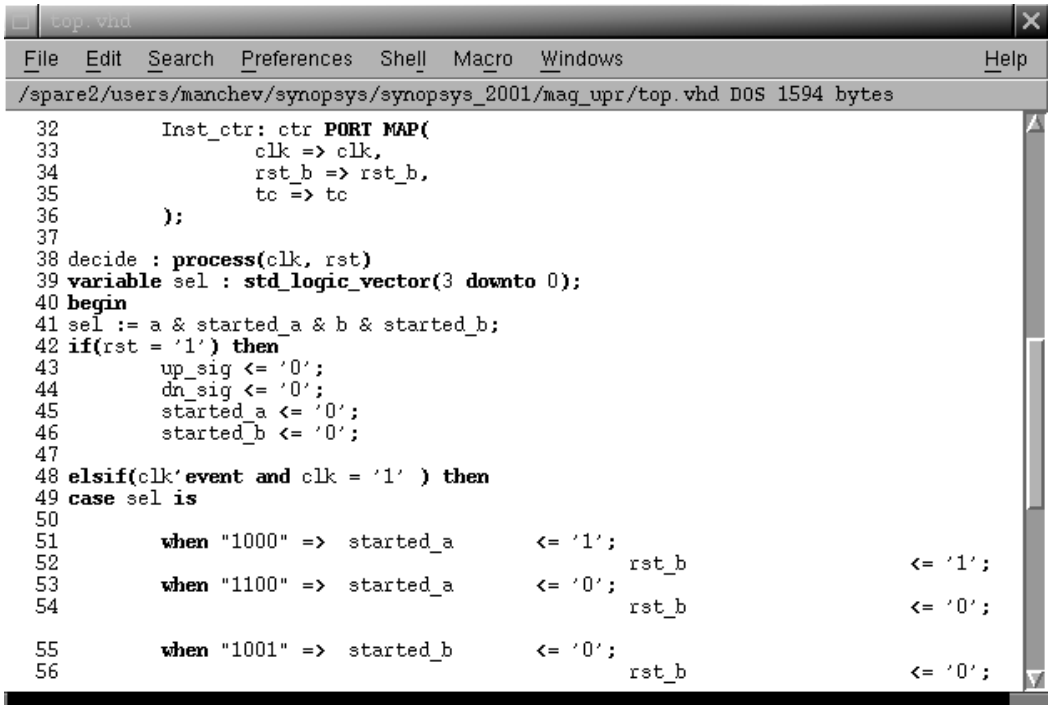
1. Подготовка на работната среда.

За включване в системата са необходими зададени потребителско име и парола. Стартира се *терминал*, в който се прави работна директория (**mkdir lab№**), след което се стартира конфигурационния файл synopsys_cfg.csh (**source ./synopsys_cfg.csh**).

2. Въвеждане на VHDL код.

Използва се текстов редактор – ***NEdit***. За да се стартира, в терминала се записва **nc2&**. В неговия прозорец се въвежда кода (фиг. 1). ***NEdit*** разпознава синтаксиса на VHDL и оцветява ключовите думи.

Файлът трябва да се записва често!

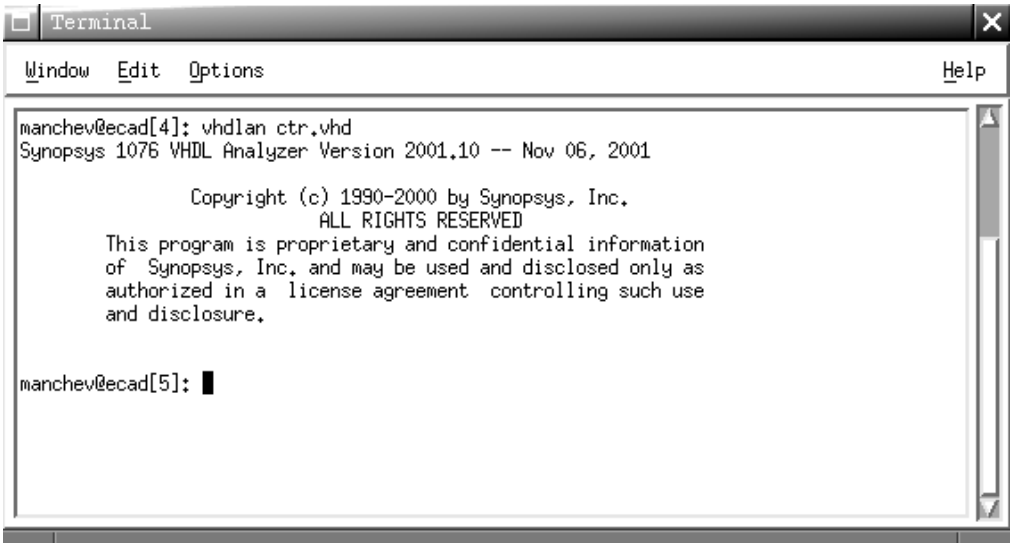


Фиг. 1. Текстов редактор NEdit

3. Анализ на VHDL код.

Тази процедура превежда VHDL описанието във формат, удобен за симулатора, използвайки инструмента ***VHDL Analyzer***.

1) Зареждат се конфигурационните файлове;
2) В работната директория, се създава поддиректория “WORK”;
3) Стартира се *VHDL Analyzer (vhdlan* <filename. vhd> *)*.



Фиг. 2. Терминален прозорец

Ако кодът не съдържа синтактични грешки, прозорецът на терминала ще изглежда подобно на този от фиг. 2. В противен случай е необходимо да се прегледа кода в текстовия редактор и да се поправят грешките, след което се анализира отново.

4. Симулация на проекта чрез симулатор *Scirocco*.

В тази стъпка, трябва да се симулира VHDL кода, след като е анализиран, с цел верификация на задачите заложени в заданието. За изпълнението на тази задача, трябва да се създаде специален модул, написан на VHDL – *test bench*. Ако по време на симулацията симулаторът изведе съобщение за грешка, трябва да се провери кода на проекта и да се направят съответните корекции. След това отново се анализира проекта (виж ***Анализ на VHDL кода***), и се ресимулира. За да се стартира симулатора, трябва да се следват следните етапи:

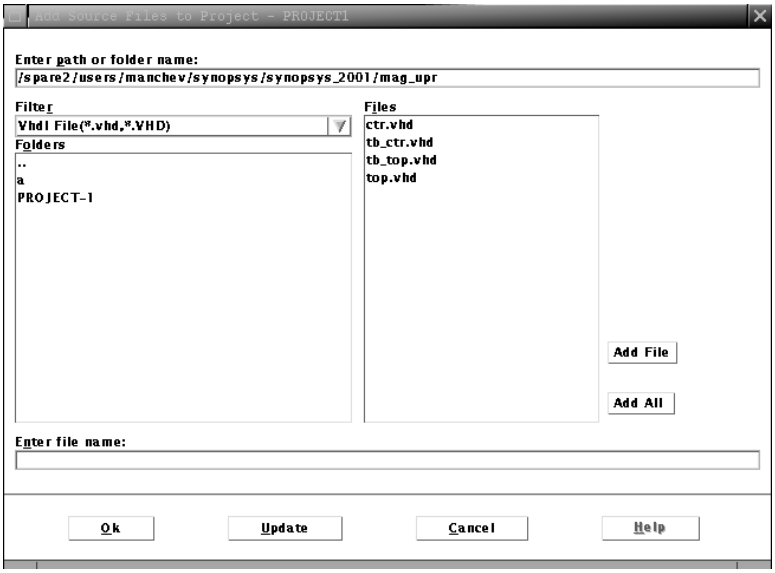
1) В прозореца на терминала се записва **scirocco&,** за извикване на симулатора. На екрана се появява малък прозорец, показан на (фиг.3).

Фиг. 3. Лента с бутони на VirSim

2) Натиска се бутона **Project** с левия бутон на мишката.
3) В появилия се прозорец от менюто **File** ⇨ **New** ⇨ **Workspace**, се създава ново работно поле (*Workspace*) (фиг. 4). В работното поле е организиран целия проект, като дървовидна структура, със съдържащите се в него файлове (йерархия), библиотеки и т.н.

Фиг. 4. Работно поле

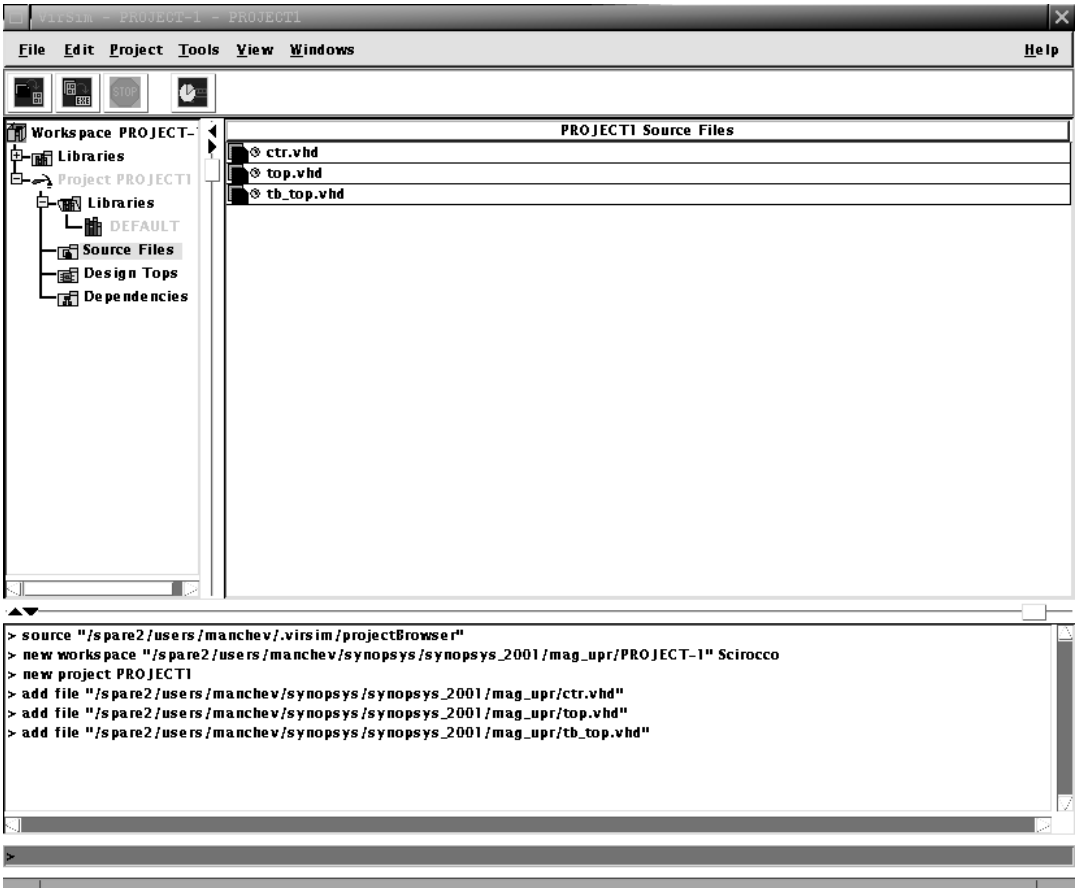
4) От менюто се избира **Project** и се щраква с десен бутон на мишката. Появява се меню, от което се избират файловете от проекта. *Test bench* модулът трябва да е поставен на най-високото ниво от йерархията (фиг. 5). Файловете, които ще се вмъкват в работното поле трябва да са в работната директория.



Фиг. 5. Добавяне на файлове към проекта

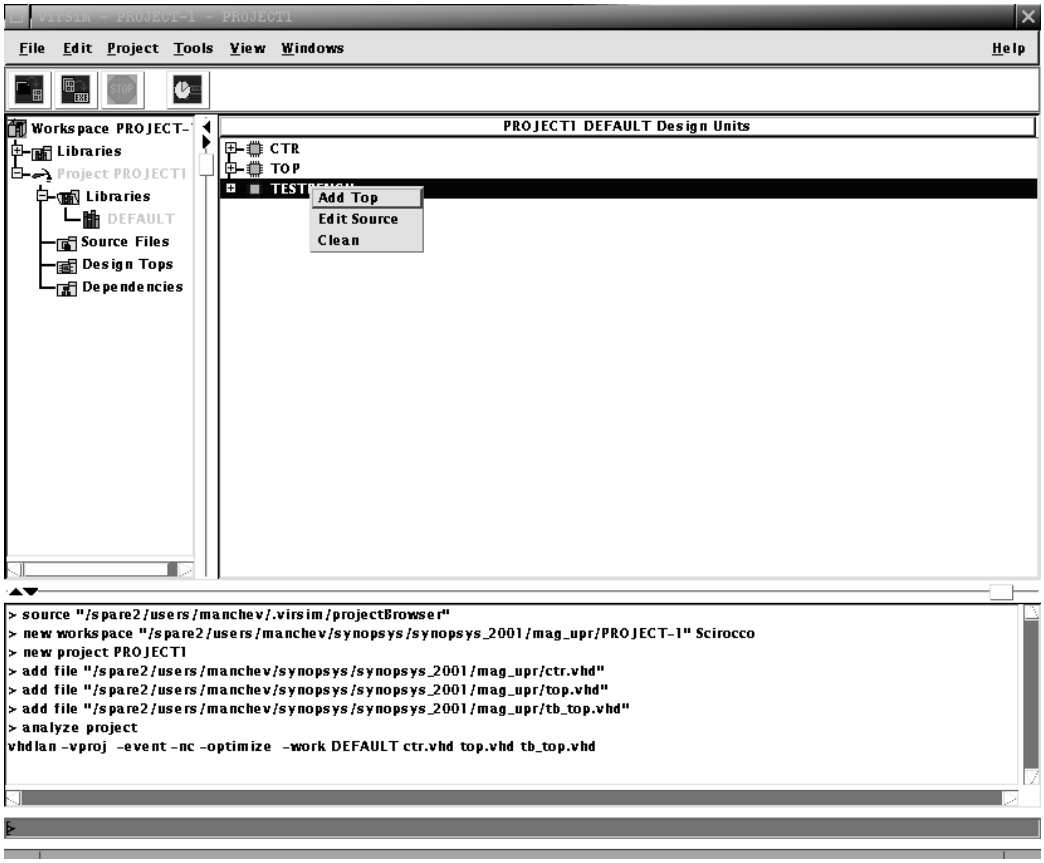
Добре е да се спазват нивата на йерархия и да се изберат първо файлове с модули от най-ниското ниво и така към върха на структурата!

5) За да се видят файловете, трябва да се щракне с левия бутон на мишката върху клона **Source files.** Файловете се появяват вляво от дървото в главния панел (фиг. 6).

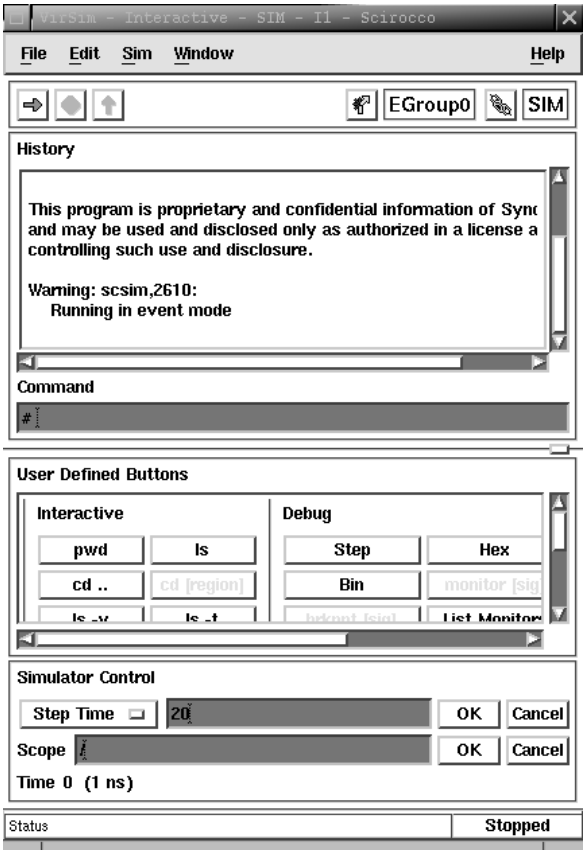


Фиг. 6. Текущи файлове в проекта

6) Файловете трябва да са подредени според йерархията на проекта. Избира се най-високото ниво (*top module*), например *top.vhd* и след това се натиска бутона **Analyze** на лентата с инструменти. Симулаторът анализира структурата на проекта и неговите под модули.
7) Важно е да се укажете на симулатора, кой от файловете заема най-високото йерархично ниво. За симулацията този файл е тестовия модул - *test bench***.** От дървото на проекта се избира клона **Libraries** $\Rightarrow$ **DEFAULT** . В главния панел се появяват файловете. Избира се *test bench VHDL* модула и се декларира като *TOP,* посредством **Add Top** (фиг. 7).
8) След като най-високото йерархично ниво е избрано, трябва да се натисне бутона **Elaborate** на лентата с инструменти. Проверява се за допуснати грешки в командния прозорец. Ако няма може да се пристъпи към следващата стъпка.
9) Натиска се бутона **Simulate** на лентата с инструменти. Появява се прозореца на симулатора - Interactive (фиг. 8).



Фиг. 7. Задаване на най – високо йерархично ниво

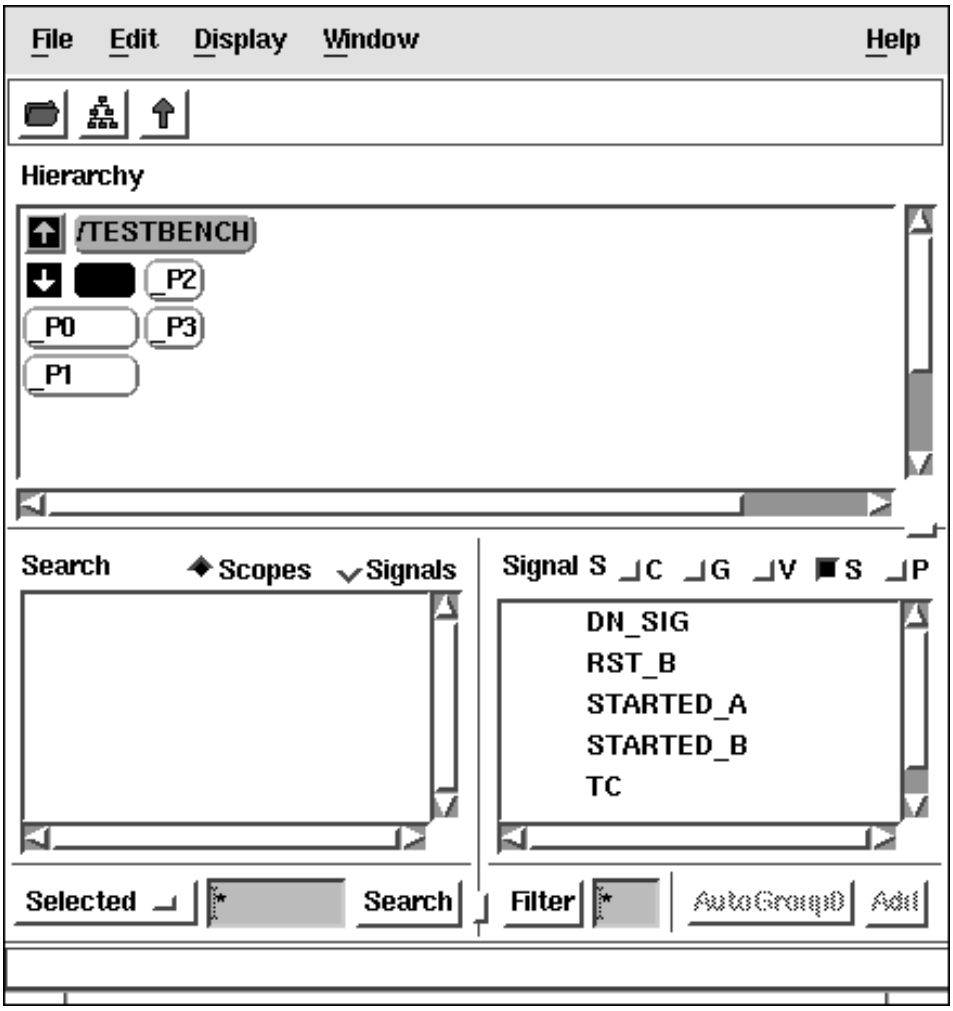


Фиг. 8. Основен прозорец на симулатора

10) Натиска се бутона **Hierarchy** от основната лента с бутони (фиг. 9).

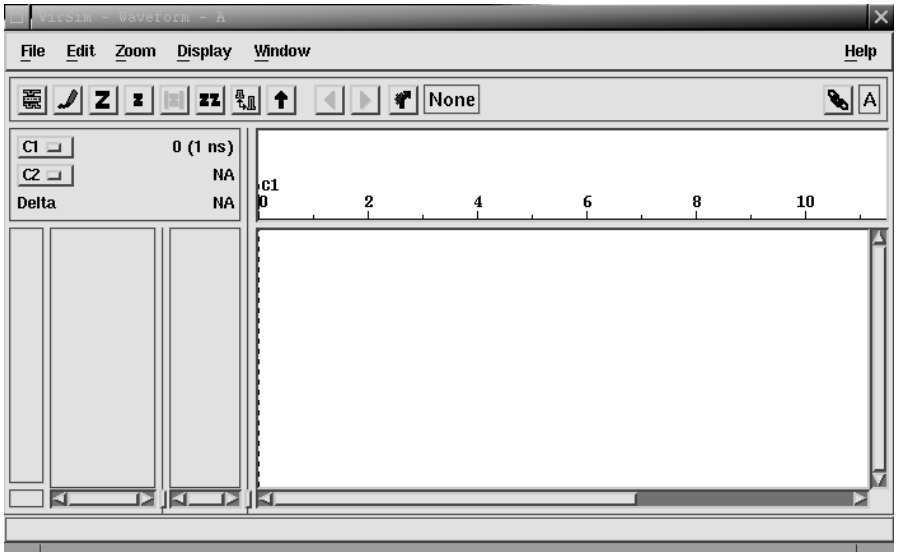
Фиг. 9. Основна лента с бутони на VirSim

11) В основния прозорец се избира най-високото ниво – *top level*. В полето **Signal** са показани входните и изходните сигнали определени от тестовия модул. Полетата **C, G, V** и **P** се размаркират, и се оставя маркирано само **S** *(signals)* полето (фиг. 10). Сигналите от йерархичните нива се наблюдават, като се натисне зелената стрелка до наименованието на модула.



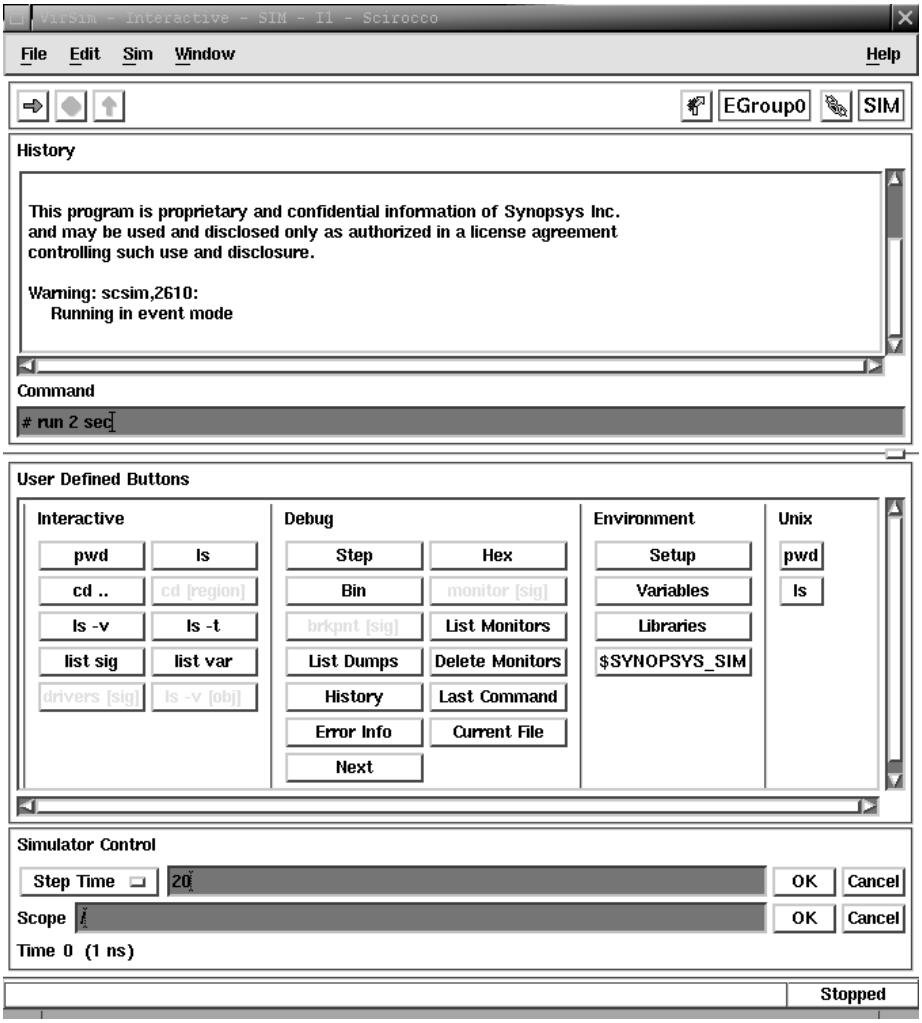
Фиг. 10. Избор на сигнали за визуализация

12) Избира се **Waveform** от главната лента. Появява се графичен редактор, където може да се наблюдават времедиаграмите на сигналите (фиг. 11).



Фиг. 11. Waveform редактор

13) За да се видят самите времедиаграми, трябва да се “довлачат” от прозореца **signals window**.



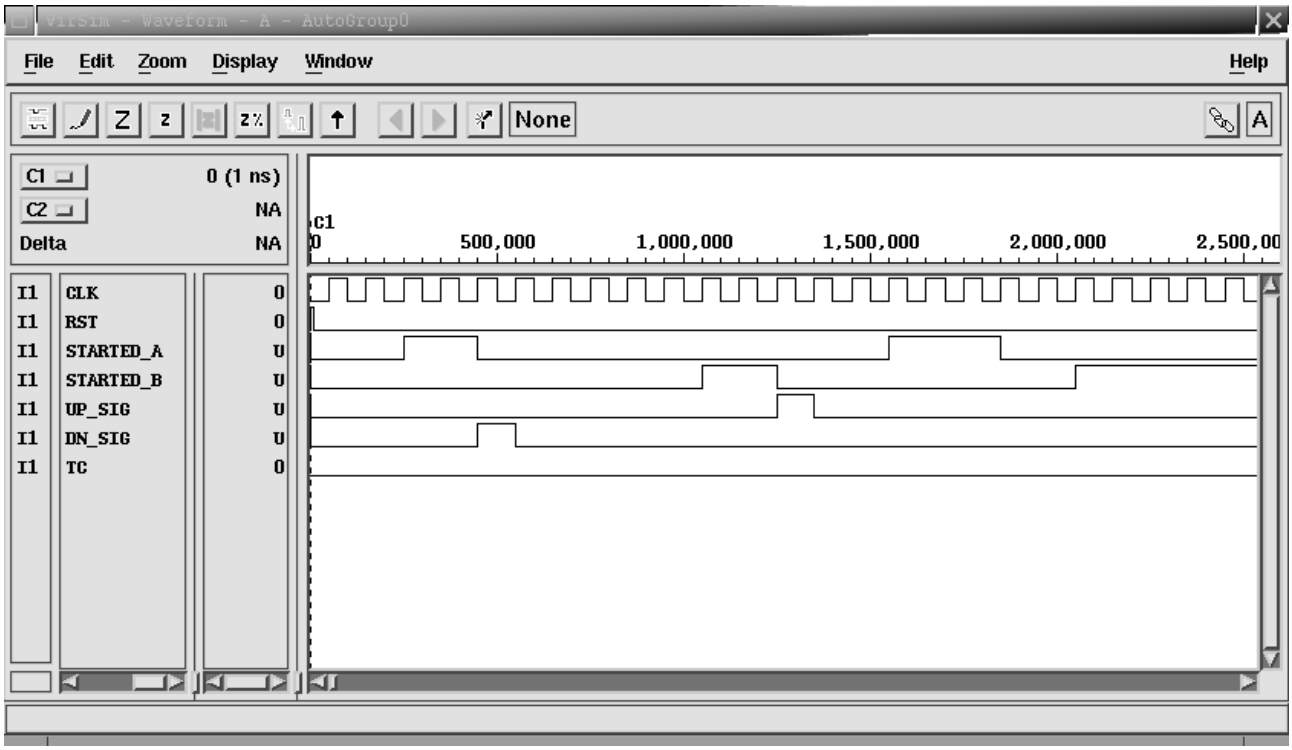
Фиг. 12. Задаване на време на симулацията

- Избират се сигнали;

- Задържа се ***средния бутон на мишката;***
- Завлича се избрания сигнал в прозореца на **waveform editor**.

Сигналът е готов за симулиране и ще бъде изобразен на екрана след като завърши симулацията.

14) В средата на симулатора (***interactive environment***) (фиг. 8), в командния ред ***(Command line)*** се записва времето за симулация например **run 2 sec**, където 2 е времето за симулация зададено в секунди (s) (фиг. 12).
15) След като е въведено времето за симулация, се натиска **Enter**. Получените резултати са показани като времедиаграми в графичния редактор (**waveform editor)** (фиг. 13).

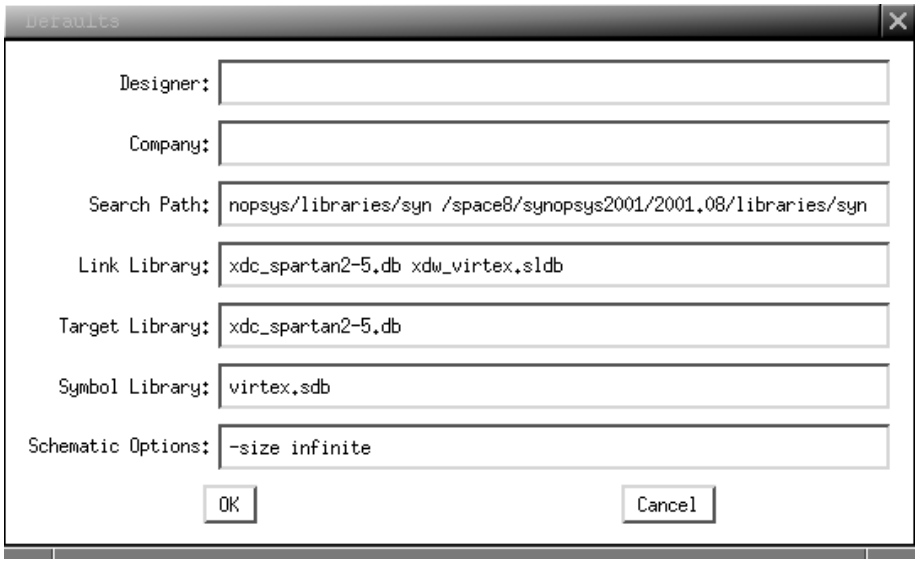


Фиг. 13. Времедиаграми от функционална симулация

5. Синтез на проекта с инструмента Design Analyzer

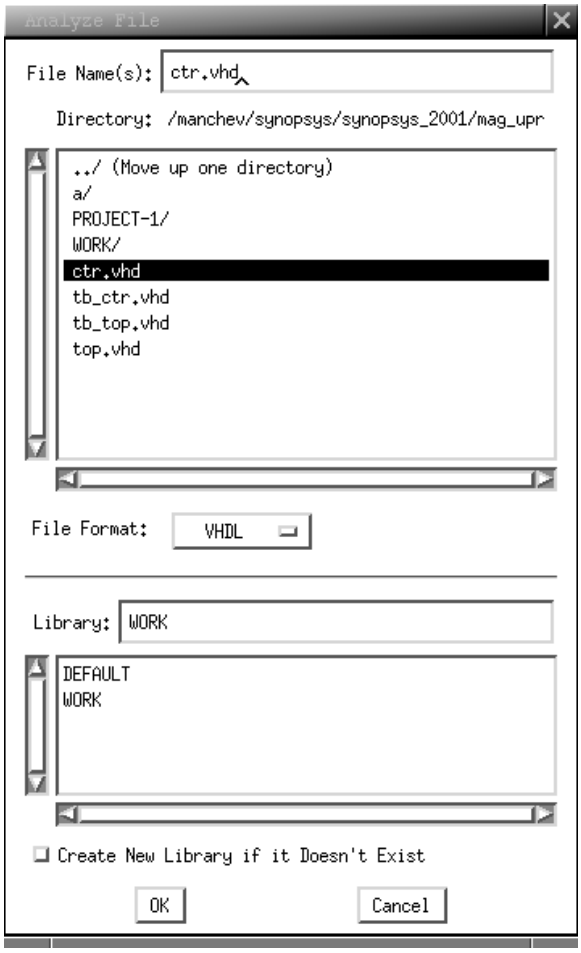
Когато проекта е симулиран и е проверена неговата работоспособност и точност посредством ***Scirocco***, той трябва да бъде синтезиран. Това става посредством ***Design Analyzer***. За изпълнението на синтеза трябва да се спази следната последователност:

1) Подготвят се конфигурационните файлове на ***Synopsys***.
2) Извиква се синтезатора чрез командата **des_ana&** в прозореца на терминала.
3) Проверява се какви библиотеки са заредени за синтезатора от менюто **Setup ⇨ Defaults,** в главния прозорец на синтезатора (фиг. 14).

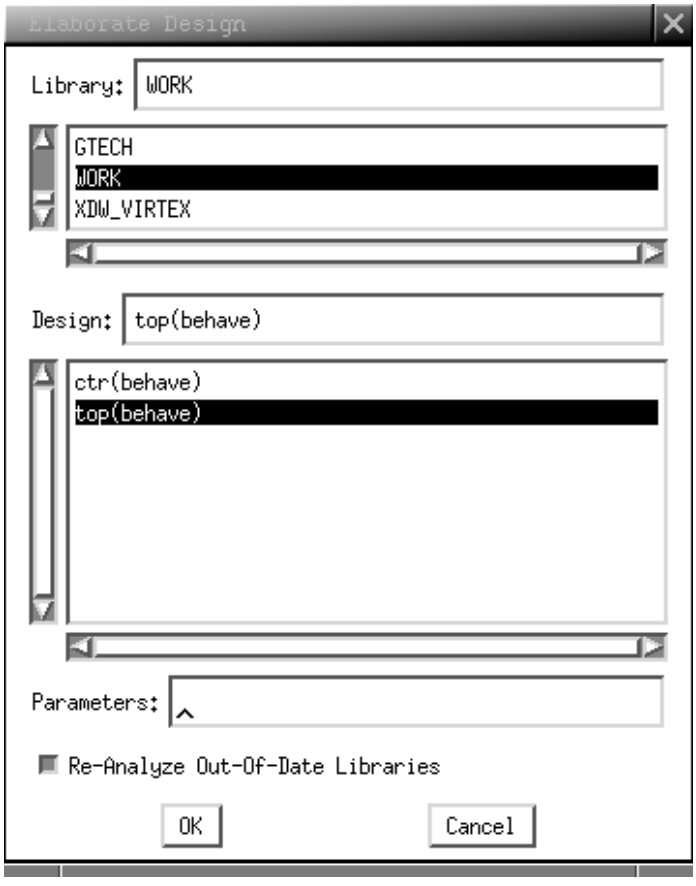


Фиг. 14. Предварително заредени библиотеки

4) Анализира се проекта (фиг. 15), като от меню **File** се избира **Analyze**.



Фиг. 15. Избор на файлове за анализ

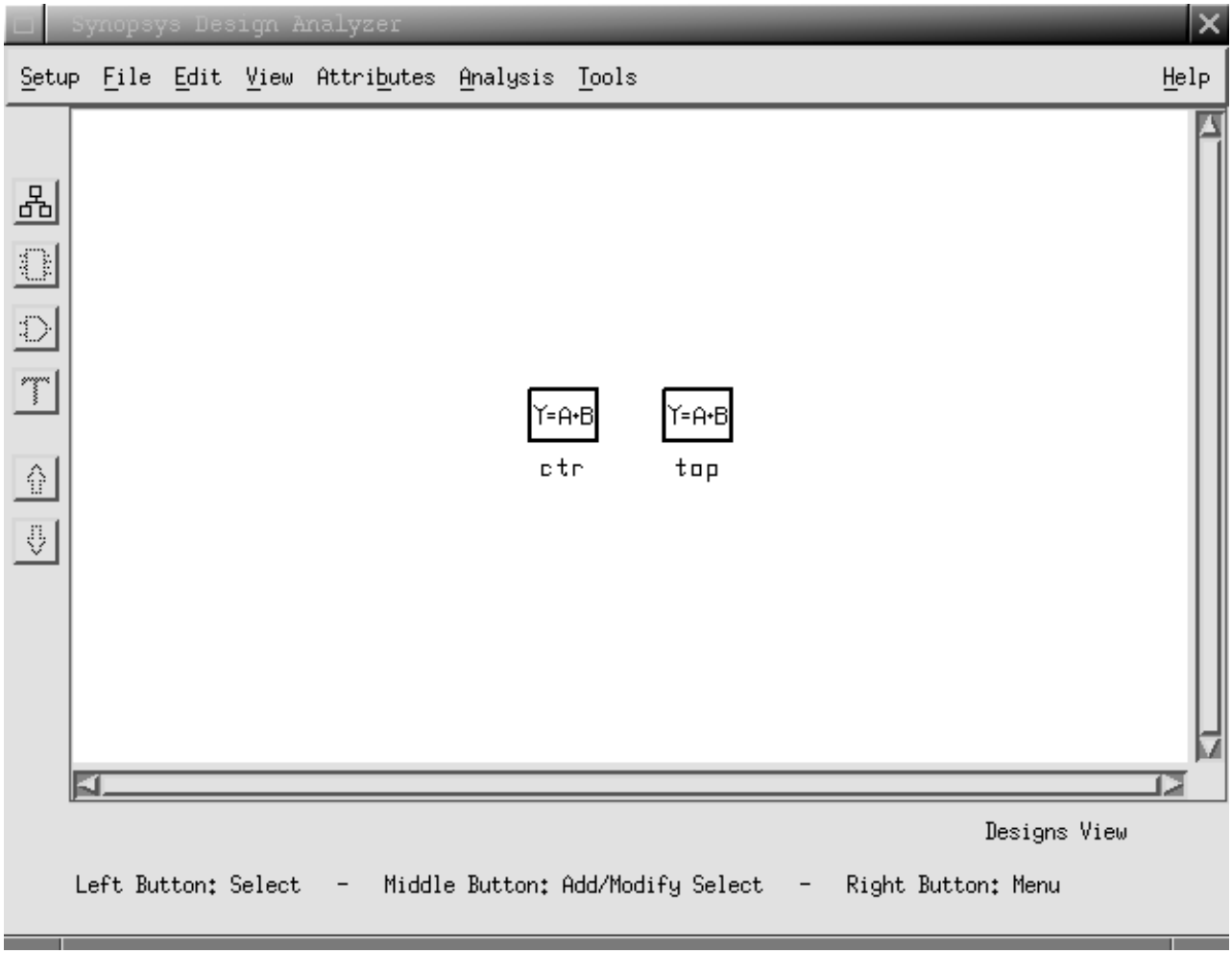


Фиг. 16. Избор на файлове за синтез

Полето Create new library if it doesn't exist трябва да е отбелязано!

5) След структурния анализ на проекта се налага операцията изработване *(Elaborate)*. Това става като се избере **Elaborate** от менюто **File**. В полето **library** се избира **DEFAULT** и после в полето **design field** се избира **top модула** на дадения проект (фиг. 16).

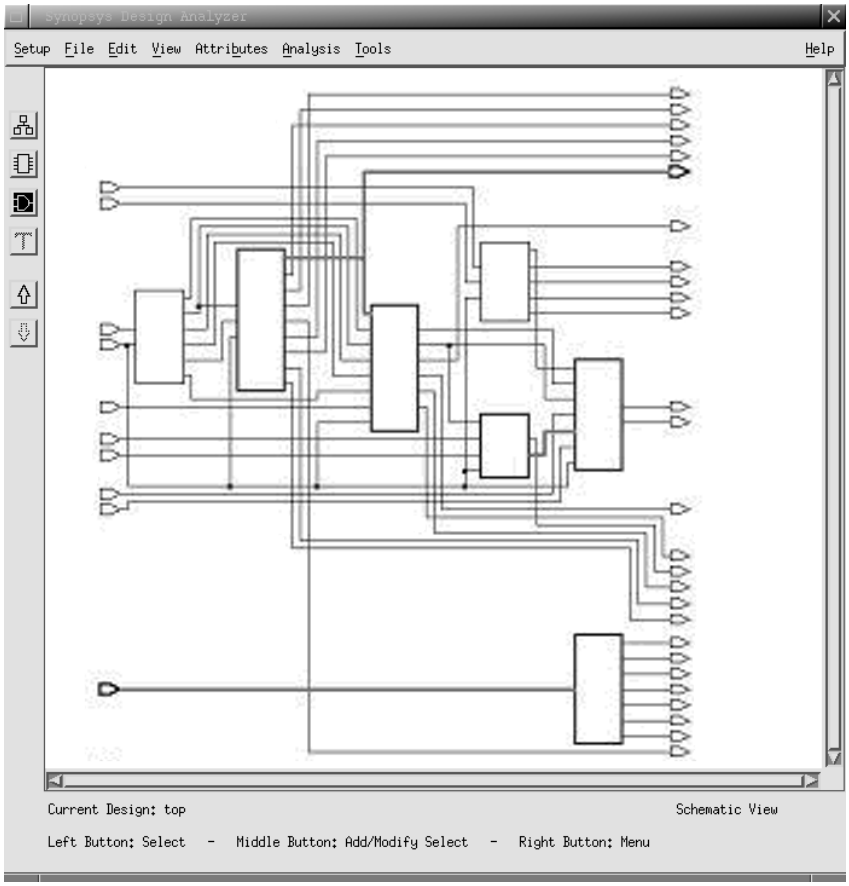
6) След приключване на операцията изработване (*Elaboration*), проектът е синтезиран и резултатите са налице в главния прозорец на ***Design Analyzer*** (фиг. 17).



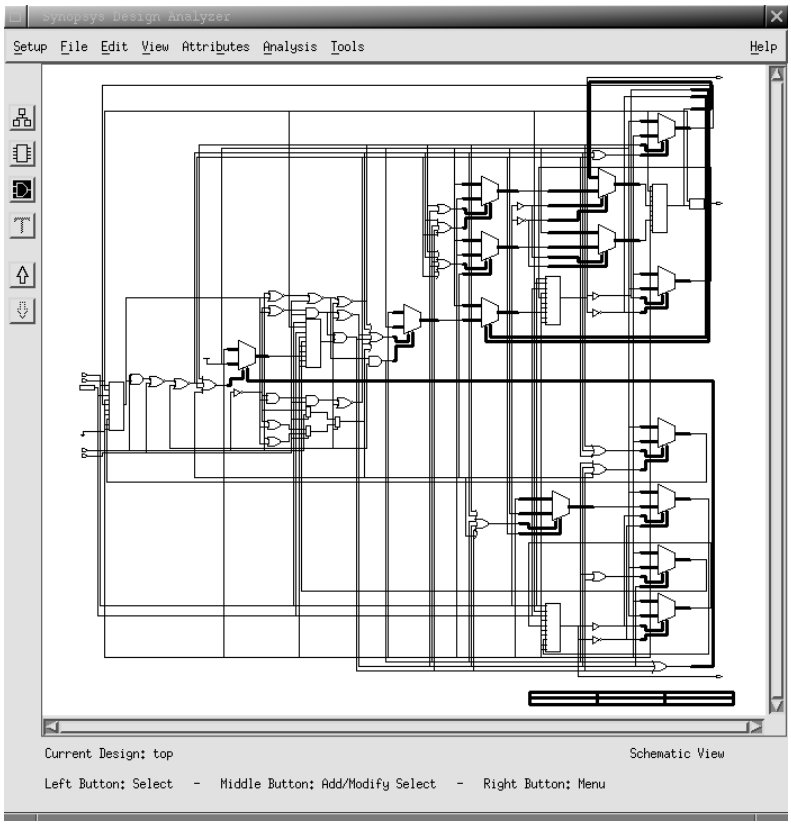
Фиг. 17. Основен прозорец на Design Analyzer

Структурата на блоковете може да се изследва, като се щракне два пъти с левия бутон на мишката върху него. На фиг. 18 е показана примерна структура на йерархичен проект.

Структурата на модулите не е създадена с реални елементи, а с техни модели. Например, на фиг. 19 е даден примерен синтез на краен автомат. За да се заменят тези модели, трябва да се извърши операцията оптимизация (*Design Optimization*).

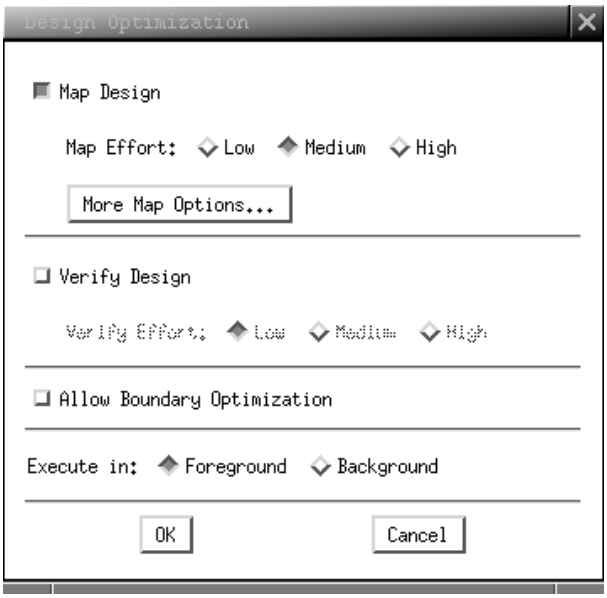


Фиг. 18. Примерен проект



Фиг. 19. Примерен модел на краен автомат

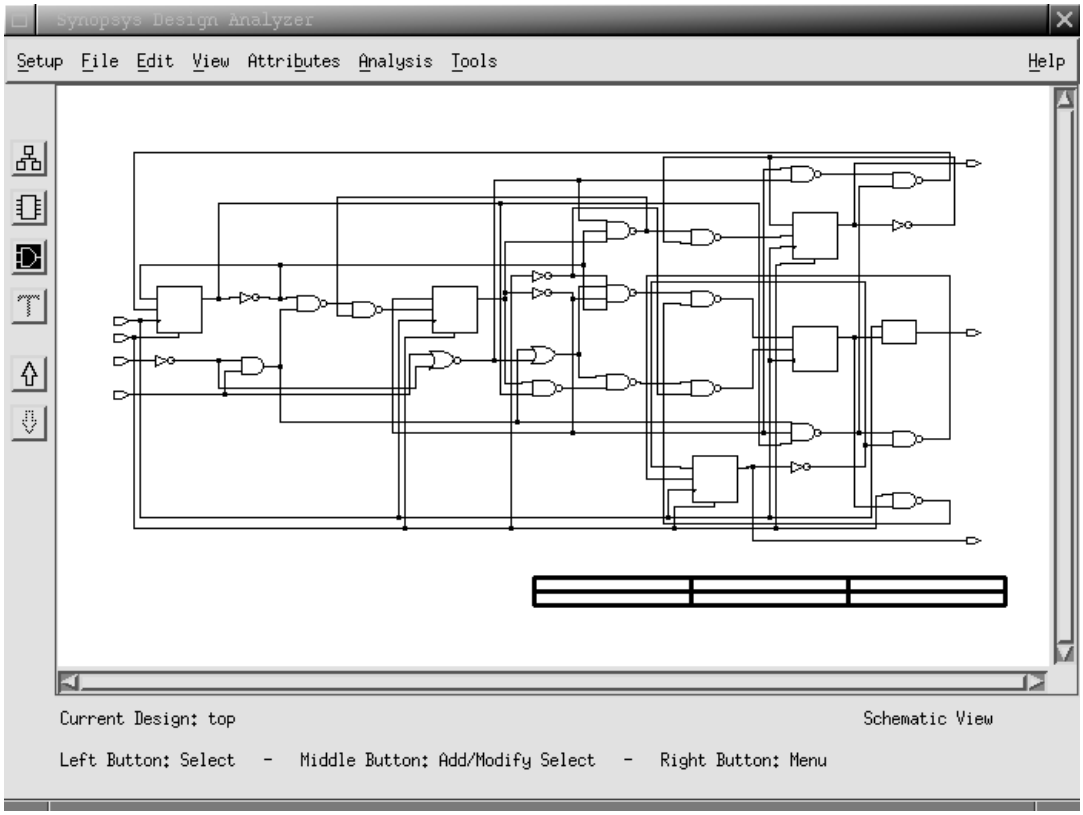
7) Оптимизация на проекта (*Design Optimization*) се извиква от менюто **Tools** (фиг. 20).



Фиг. 20. Оптимизация на проекта

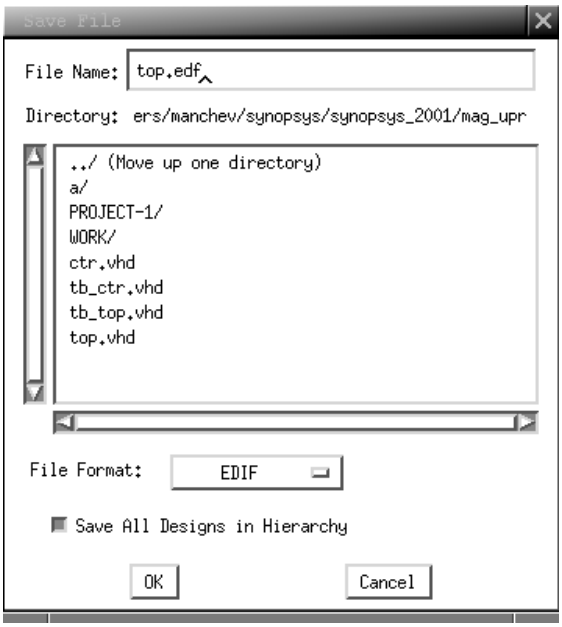
Оптимизацията е последната стъпка от процеса на синтез, когато временните клетки на ***SYNOPSYS*** се обръщат в примитиви от избраната технология. В случая тя е *XILINX FPGA*.

Примерен синтез на краен автомат след прилагане на операцията *Оптимизация* е показан на фиг. 21.



Фиг. 21. Синтез на краен автомат с оптимизация

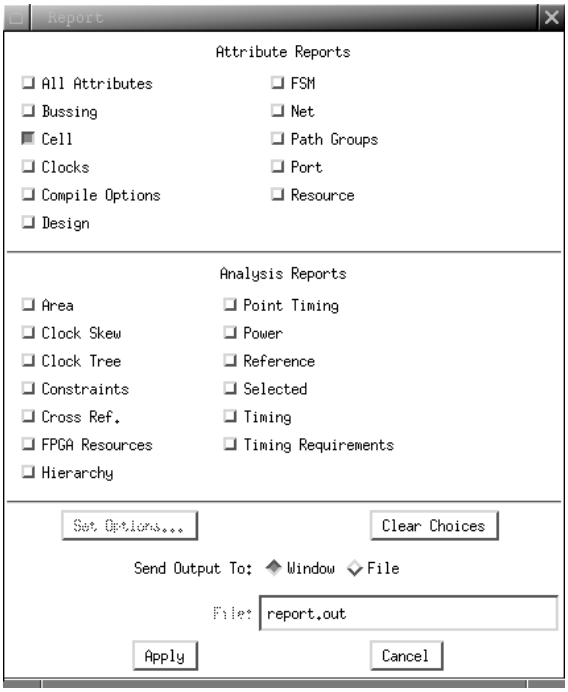
След като проекта е синтезиран, той се записва в EDIF формат, както е показано на фиг. 22.



Фиг. 22. Запазване на файл

Форматът EDIF е подходящ за изпълнение на следващите операции от процеса на проектиране – имплементация и верификация. Те се извършват, използвайки стандартни системи за въвеждане на схеми и редактиране на топология, като *CADENCE Design Framework II* и *Silicon Ensemble*.

След синтезиране на схемата се генерира отчет за използвания ресурс (фиг. 23).

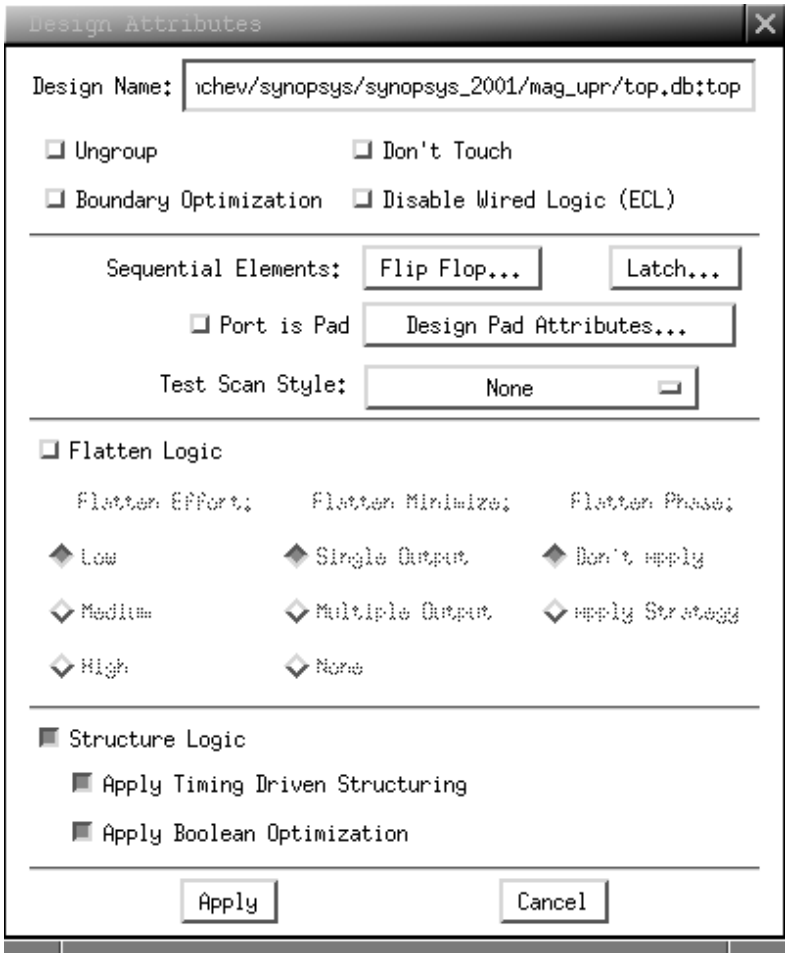


Фиг. 23. Настройки за изработване на отчет

8) Задаване на оптимизационни ограничения (*Design Constraints*). За да бъде съобразен синтеза на проекта с предварително поставени изисквания към схематичната реализация и честотно-времевите параметри, е необходимо да се зададат така наречените *оптимизационни ограничения.* От менюто **Attributes**, се избира **Optimization Directives**, след което **Design.**

Като ограничения могат да се задават (фиг. 24): гранична оптимизация (*Boundary Optimization*), декомпозиция на проекта (*Ungroup*), тип на запомнящите елементи, сработващи по фронт и по ниво (*Flip-Flop & Latch*), минимизиране на логическите йерархични нива (*Flatten Logic*), структурна логическа и времева оптимизация (*Structure Logic*).

За да се приложат желаните оптимизации, е необходимо да се избере **Apply** и повторно да се премине през операцията *Design Optimization*.



Фиг. 24. Задаване на оптимизационни ограничения

Прилагането на оптимизационни ограничения трябва да е предварително съгласувано със структурата на проекта. В противен случай, съществува реален риск от влошаване параметрите на схемата.

Упражнение № 11

Проектиране на декодер от BCD към 7-SEG код

1. Цел.

Да се придобие практически опит за разработване на поведенчески VHDL модели на комбинационни логически схеми.

2. Теоретично въведение.

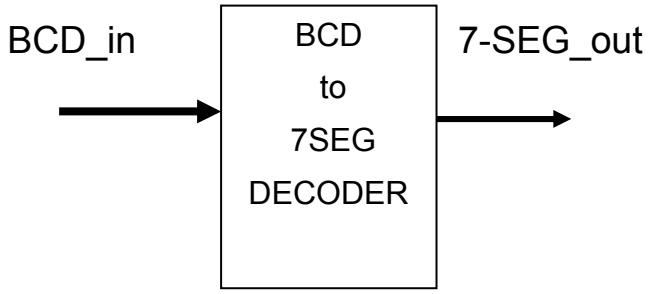
Комбинационни логически схеми са такива, при които състоянието на изходите им зависи само от текущото състояние на техните входове.

Дешифраторите (Decoder) са комбинационни логически схеми, които активират по един изход за определена комбинация на входните променливи. При n входа максималния възможен брой изходи е $m = 2^n$.

3. Задание.

1) Да се реализира дешифратор BCD към 7-SEG като се използва езика за хардуерно описание VHDL и top-down подход на поведенческо описание.
2) Да се опишат тестови вектори и да се направят симулации на получения модел. Да се сравнят с дадените в точка 4, за да се провери коректността на работата на схемата.
3) Да се синтезира схемата и получения файл да се запише с разширение .db.

- Блоков вид на декодера



Фиг. 1. Блоков вид

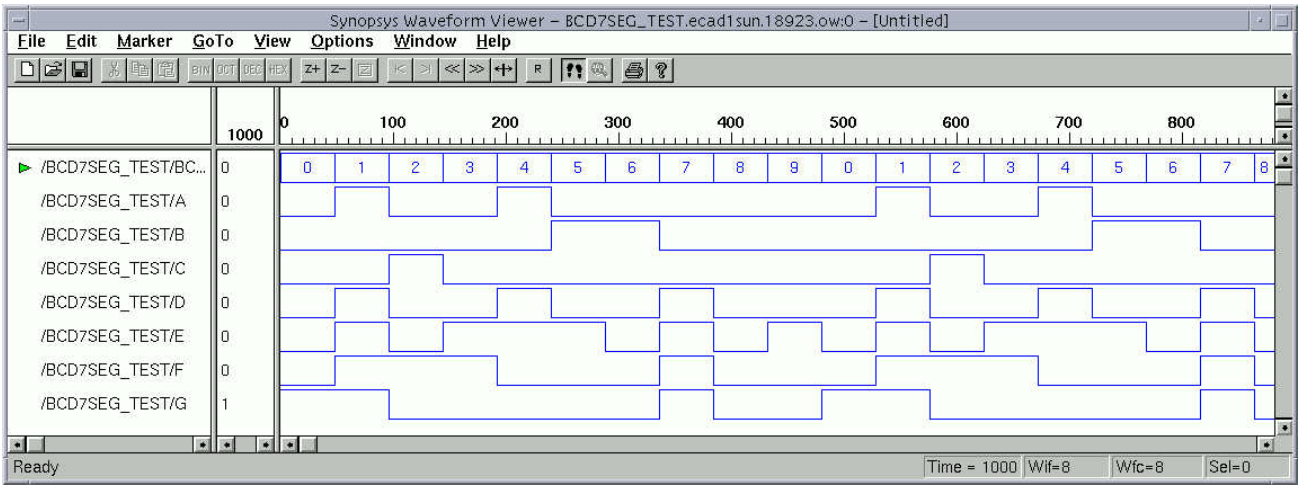
- Интерфейсна част на модула

Таблица 1. Интерфейсна част на модула.

Сигнал	Тип	Посока	Функция
BCD_in	std_logic_vector (3 downto 0)	вход	входно състояние
7SEG_out	std_logic_vector (6 downto 0)	изход	декодирано състояние

Прилагането на оператор CASE, би улеснил значително задачата.

4. Симулационни резултати.



Фиг. 2. Времедиаграми

Упражнение № 12

Проектиране на делител на честота

1. Цел.

Да се придобие практически опит за разработване на поведенчески VHDL модели на комбинационни логически схеми.

2. Теоретично въведение.

Комбинационни логически схеми са такива, при които състоянието на изходите им зависи само от текущото състояние на техните входове.

Делителите на честота служат за увеличаване на периода (съответно намаляване на честотата) на даден входен сигнал необходим брой пъти.

3. Задание.

1) Да се реализира делител x1, x10, x100, x1000, като се използва езика за хардуерно описание VHDL и top-down подход за поведенческо описание.
2) Да се придържа към блоковия вид на делителя и описанията на интерфейсите дадени по-долу.
3) Да се опишат тестови вектори и да се направят симулации на получения модел. Да се сравнят с дадените в точка 4, за да се провери коректността на работата на схемата.
4) Да се синтезира схемата и получения файл да се запише с разширение .db.

Препоръчва се описанието да е изградено с операторите IF и CASE!

- Блоков вид на делителя на честота

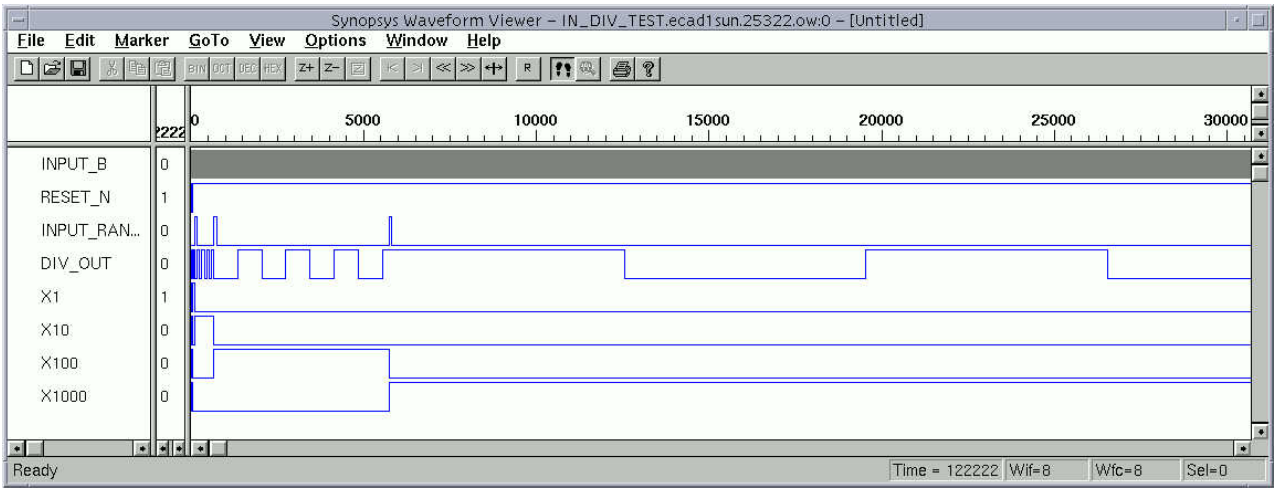
Фиг. 1. Блоков вид

- Интерфейсната част на модула е дадена в таблица 1.

Таблица 1. Интерфейсна част на модула.

Сигнал	Тип	Посока	Функция
INPUT_B	std_logic	вход	входна честота
RESET_N	std_logic	вход	първоначално установяване
INPUT_RANGE	std_logic	вход	избор на коефициент
OUTPUT_DIV	std_logic	изход	разделена честота
X1	std_logic	изход	дели на 1
X10	std_logic	изход	дели на 10
X100	std_logic	изход	дели на 100
X1000	std_logic	изход	дели на 1000

4. Симулационни резултати.



Фиг. 2. Времедиаграми

Упражнение № 13

Проектиране на драйвер за динамична индикация

1. Цел.

Да се придобие практически опит за разработване на поведенчески VHDL модели на комбинационни логически схеми.

2. Теоретично въведение.

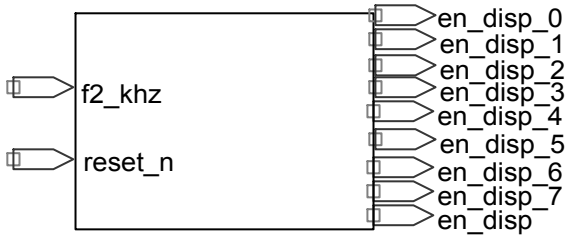
Комбинационни логически схеми са такива, при които състоянието на изходите им зависи само от текущото състояние на техните входове.

Драйверите за динамична индикация служат за управлението на сегментите при различни видове светодиоди или LCD дисплеи.

3. Задание.

1) Да се реализира драйвер за динамична индикация, като се използва езика за хардуерно описание VHDL и top-down подход на поведенческо описание.
2) Да се опишат тестови вектори и да се направят симулации на получения модел. Да се сравнят с дадените в точка 4, за да се провери коректността на работата на схемата
3) Да се синтезира схемата и получения файл да се запише с разширение .db.

- Блоков вид



Фиг. 1. Блоков вид.

- Интерфейсна част на модула

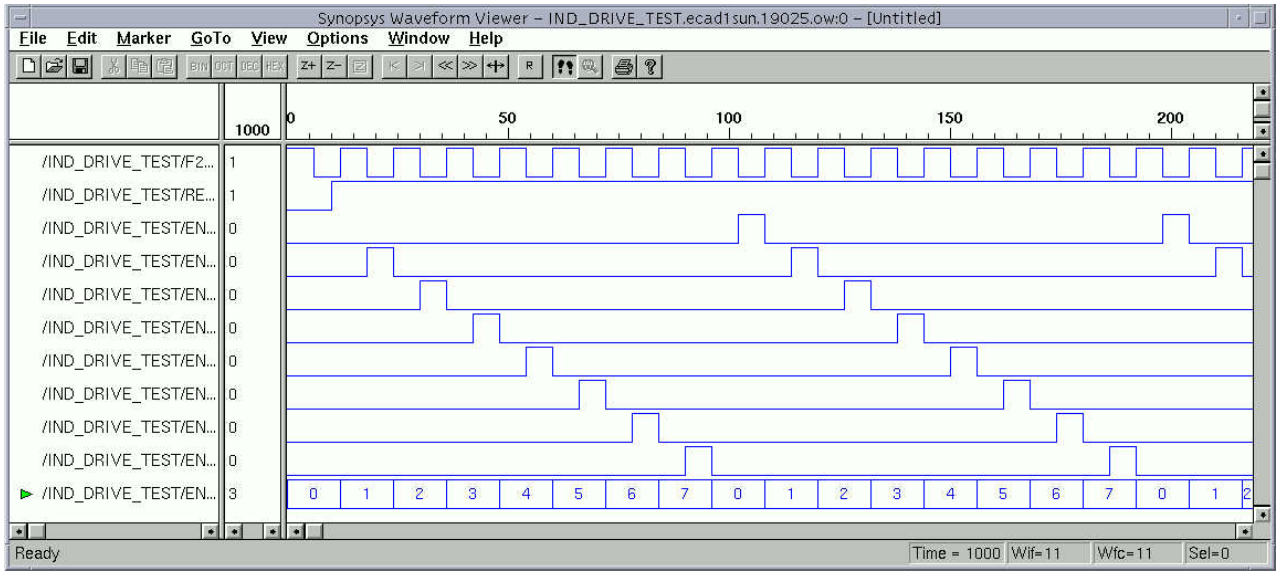
Таблица 1. Интерфейсна част на модула.

Сигнал	Тип	Посока	Функция
F2khz	std_logic	вход	входна честота
RESET_N	std_logic	вход	първоначално установяване
EN_DISP0	std_logic	изход	сегмент 1
EN_DISP1	std_logic	изход	сегмент 2
EN_DISP2	std_logic	изход	сегмент 3
EN_DISP3	std_logic	изход	сегмент 4
EN_DISP4	std_logic	изход	сегмент 5

EN_DISP5	std_logic	изход	сегмент 6
EN_DISP6	std_logic	изход	сегмент 7
EN_DISP7	std_logic	изход	сегмент 8
EN_DISP	std_logic_vector (2 downto 0)	изход	изходна шина

Препоръчва се описанието да е изградено с операторите IF и CASE!

4. Симулационни резултати.



Фиг. 2. Времедиаграми

Упражнение № 14

Проектиране на модул за задаване на режим на работа

1. Цел.

Да се придобие практически опит за разработване на поведенчески VHDL модели на комбинационни логически схеми.

2. Теоретично въведение.

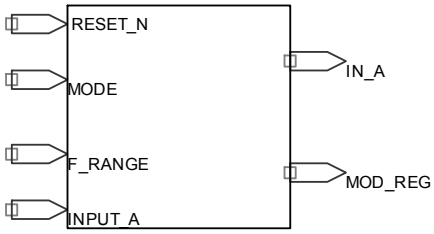
Комбинационни логически схеми са такива, при които състоянието на изходите им зависи само от текущото състояние на техните входове.

В случая, блокът управлява режимите на работа на разработвания мултимер (измерване на честота, измерване на приод, разлика между две честоти, хронометър, брояч и калибровка) Също така и мултиплексира към изход IN_A входове F_RANGE и INPUT_A. Режимите се превключват серийно от бутон, чиито сигнал се приема на вход mode.

3. Задание.

1) Да се реализира модул “mode”, като се използва езика за хардуерно описание VHDL и top-down подход на поведенческо описание.
2) Да се опишат тестови вектори и да се направят симулации на получения модел. Да се сравнят с дадените в точка 4, за да се провери коректността на работата на схемата.
3) Да се синтезира схемата и получения файл да се запише с разширение .db.

- Блоков вид на модула



Фиг. 1. Блоков вид

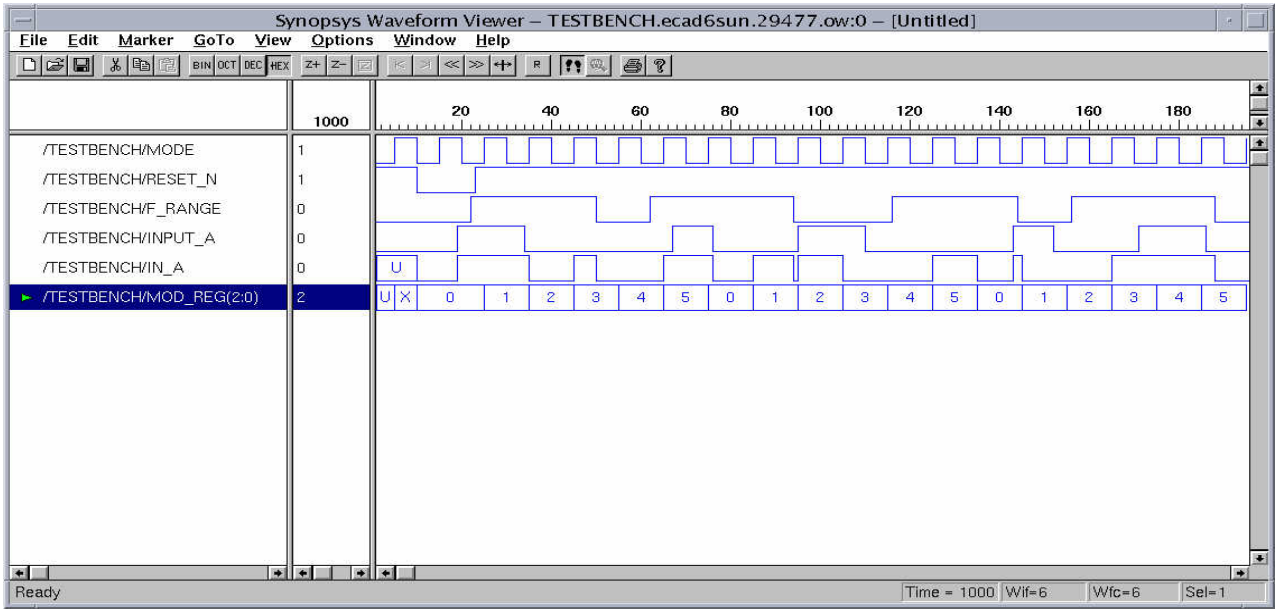
- Интерфейсна част на модула

Таблица 1. Интерфейсна част на модула.

Сигнал	Тип	Посока	Функция
MODE	std_logic	вход	Сигнал за смяна на режима
RESET_N	std_logic	вход	първоначално установяване
F_RANGE	std_logic	вход	текуща еталонна честота
INPUT_A	std_logic	вход	вход за измерваната честота
IN_A	std_logic	изход	Изход към BCD брояча
MODE_REG	std_logic_vector (2 downto 0)	изход	Изход за указване на състоянието

Препоръчва се описанието да е изградено с операторите IF и CASE!

4. Симулационни резултати.



Фиг. 2. Времедиаграми

Упражнение № 15

Проектиране на BCD брояч

1. Цел.

Да се придобие практически опит за разработване на поведенчески VHDL модели на последователностни логически схеми.

2. Теоретично въведение.

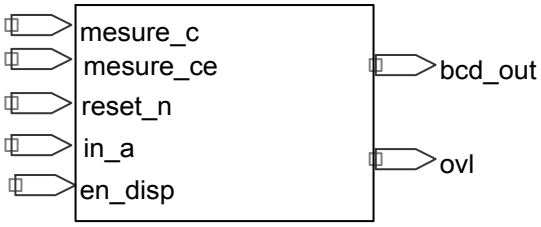
Последователностните логически схеми притежават свойството да запомнят въздействието на входните сигнали. Затова те се наричат още “устройства с памет”.

Броячът (counter) представлява последователностна логическа схема с много устойчиви състояния всяко, от които съответства на броя на постъпилите импулси. Той се състои от тригерни клетки (елементи памет) всяка, от които съхранява по един разред на числото, съответстващо на изброените импулси.

3. Задание.

1) Да се реализира BCD брояч като се използва езика за хардуерно описание VHDL и top-down подход на поведенческо описание.
2) Да се опише тестов вектор и да се направят симулации на получения модел. Да се сравнят с дадените в точка 4, за да се провери коректността на работата на схемата.
3) Да се синтезира схемата и получения файл да се запише с разширение .db.

- Блоков вид на брояча



Фиг. 1. Блоков вид

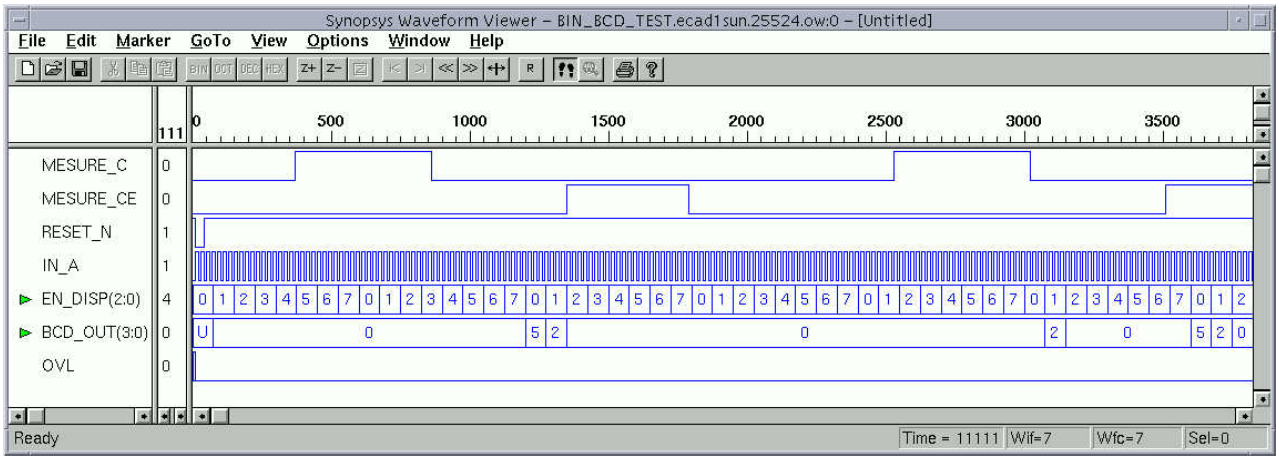
- Интерфейсна част на модула

Таблица 1. Интерфейсна част на модула.

Сигнал	Тип	Посока	Функция
MESURE_C	std_logic	вход	функционален вход
MESURE_CE	std_logic	вход	функционален вход
RESET_N	std_logic	вход	първоначално установяване
IN_A	std_logic	вход	тактова честота

EN_DISP	std_logic_vector(2 downto 0)	вход	установяване на сегмент
BCD_OUT	std_logic_vector(3 downto 0)	изход	изход на брояча
OVL	std_logic	изход	препълване

4. Симулационни резултати.



Фиг. 2. Времедиаграми

Упражнение № 16

Проектиране на модул за задаване на обхват на работа

1. Цел.

Да се придобие практически опит за разработване на поведенчески VHDL модели на последователностни логически схеми.

2. Теоретично въведение.

Последователностните логически схеми притежават свойството да запомнят въздействието на входните сигнали. Затова те се наричат още “устройства с памет”.

Основната задача на блок RANGE е да управлява текущия обхват на работа на честотомер.

Режима на работа се задава серийно чрез сигнал от бутон, пристигащ на вход IN_CLK_RANGE.

Командват се:

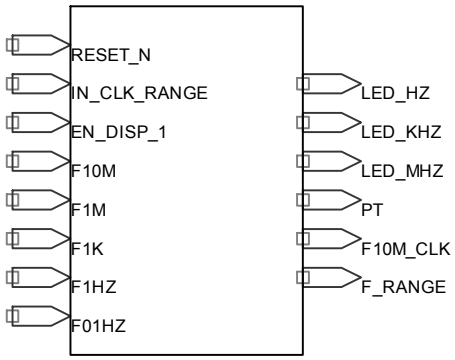
1) LED индикация показваща текущ обхват (MHz, kHz, Hz);
2) Сигнал за управление на десетична точка.

Също така в зависимост от обхвата към изход се мултиплексира F_RANGE - еталонна честота от 1MHz, 1kHz, 1Hz или 0,1Hz.

3. Задание.

1) Да се реализира модул “mode”, като се използва езика за хардуерно описание VHDL и top-down подход на поведенческо описание.
2) Да се опише тестов вектор и да се направят симулации на получения модел. Да се сравнят с дадените в точка 4, за да се провери коректността на работата на схемата.
3) Да се синтезира схемата и получения файл да се запише с разширение .db.

- Блоков вид на модула



Фиг. 1. Блоков вид

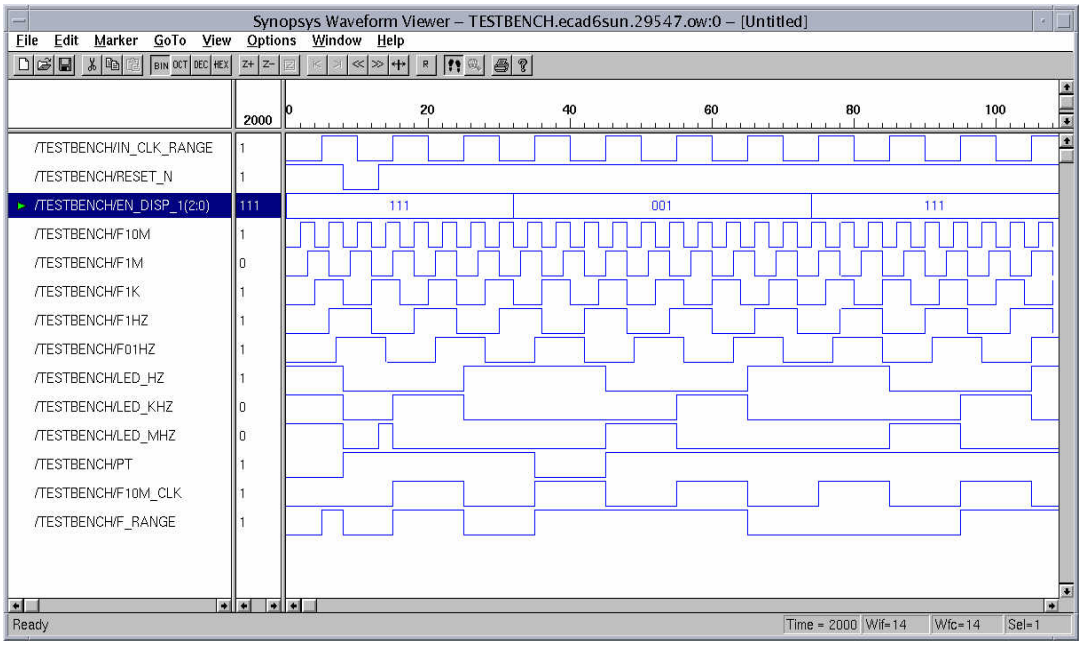
- Интерфейсна част на модула

Таблица 1. Интерфейсна част на модула.

Сигнал	Тип	Посока	Функция
IN_CLK_RANGE	std_logic	вход	Сигнал за смяна на обхвата
EN_DISP_1	std_logic_vector (2 downto 0)	вход	Разрешение за изобразяване на точка на дисплея
RESET_N	std_logic	вход	Първоначално установяване
F10M	std_logic	вход	Вход за честота 10MHz
F1M	std_logic	вход	Вход за честота 1 MHz
F1K	std_logic	вход	Вход за честота 1 KHz
F1HZ	std_logic	вход	Вход за честота 1 Hz
F01HZ	std_logic	вход	Вход за честота 01 Hz
LED_HZ	std_logic	изход	Индикация на режим на измерване Hz
LED_KHZ	std_logic	изход	Индикация на режим на измерванеKHz
LED_MHZ	std_logic	изход	Индикация на режим на измерванеMHz
PT	std_logic	изход	Индикация на точка/акт. ниво – лог. “0”/
F10M_CLK	std_logic	изход	Изход за честота 10MHz
F_RANGE	std_logic	изход	Изход за еталонна честотоа

Да се използват IF и CASE оператори!

4. Симулационни резултати.



Фиг. 2. Времедиаграми

Упражнение № 17

Проектиране на делител на честота

1. Цел.

Да се придобие практически опит за разработване на поведенчески VHDL модели на комбинационни логически схеми.

2. Теоретично въведение.

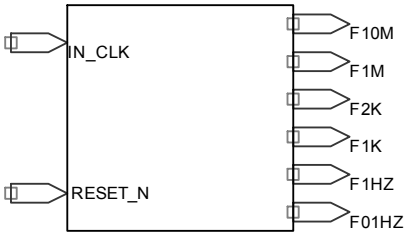
Комбинационни логически схеми са такива, при които състоянието на изходите им зависи само от текущото състояние на техните входове.

Делителите на честота служат за увеличаване на периода (съответно намаляване на честотата) на даден входен сигнал необходим брой пъти.

3. Задание.

1) Да се реализира делител на чесота, делящ еталонна честота 20MHz с изходи за 10MHz, 1MHz, 2kHz, 1kHz, 1Hz и 0.1Hz, като се използва езика за хардуерно описание VHDL и top-down подход на поведенческо описание.
2) Да се опишат тестови вектори и да се направят симулации на получения модел. Да се сравнят с дадените в точка 4, за да се провери коректността на работата на схемата.
3) Да се синтезира схемата и получения файл да се запише с разширение .db.

- Блоков вид на делителя на честота



Фиг. 1. Блоков вид

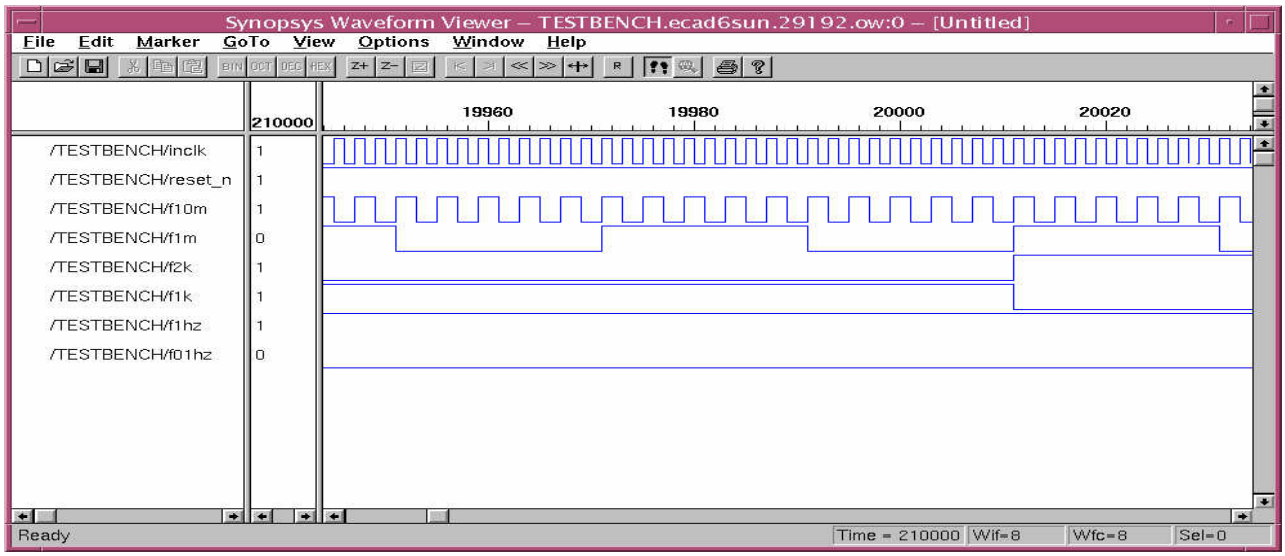
- Интерфейсна част на модула

Таблица 1. Интерфейсна част на модула.

Сигнал	Тип	Посока	Функция
IN_CLK	std_logic	вход	входна честота
RESET_N	std_logic	вход	първоначално установяване
F10M	std_logic	изход	изход 10MHz
F1M	std_logic	изход	изход 1MHz
F2K	std_logic	изход	изход 2kHz
F1k	std_logic	изход	изход 1kHz
F1Hz	std_logic	изход	изход 1Hz
F01Hz	std_logic	изход	изход 0,1Hz

Да се използва каскадно свързване на процесите!

4. Симулационни резултати.



Фиг. 2. Времедиаграми

Упражнение № 18

Проектиране на универсален брояч

1. Цел.

Да се придобие практически опит за разработване на поведенчески VHDL модели на комбинационни логически схеми.

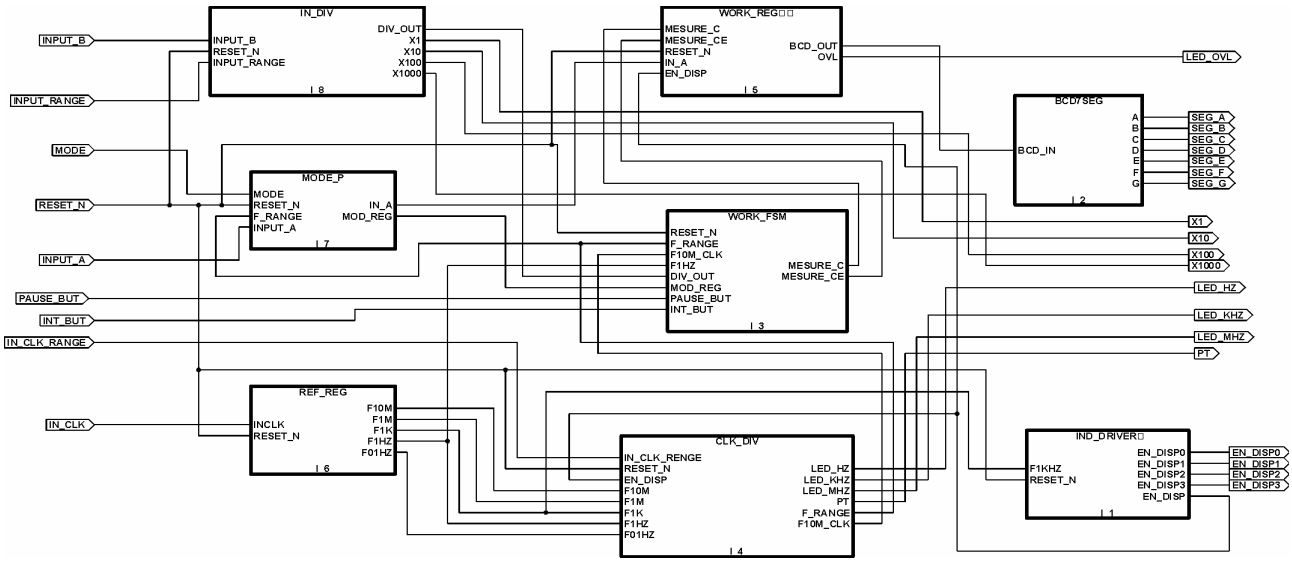
2. Теоретично въведение.

Най-горното ниво на проекта служи за обединяване на множество модули описани отделно в една обща система. Всеки от описаните модули може да бъде свързан както със други модули, така и с входа или изхода на цялата система.

3. Задание.

1) Да се реализира универсален мултимер, като се използва езика за хардуерно описание VHDL, предварително реализираните модули и top-down подход на поведенческо описание.
2) Да се опишат тестови вектори и да се направят симулации на получения модел.
3) Да се синтезира схемата и получения файл да се запише с разширение .db.

- Блоков вид на универсалния брояч



Фиг. 1. Блоков вид.

- Интерфейсна част на модула

Таблица 1. Интерфейсна част на модула.

Сигнал	**Тип**	**Посока**	**Функция**
INPUT_B	std_logic	вход	входна честота
INPUT_A	std_logic		входна честота
RESET_N	std_logic	вход	първоначално установяване
INPUT_RANGE	std_logic	вход	избор на коефициент
MODE	std_logic	вход	разделена честота
PAUSE_BUT	std_logic	вход	пауза
IN_BUT	std_logic	вход	прог. вх. делител
IN_CLK_RANGE	std_logic	вход	задава работна честота
IN_CLK	std_logic	вход	тактов сигнал
LED_OVL	std_logic	изход	препълване
SEG_A-G	std_logic_vector (6 downto 0)	изход	Сегменти от A до G
X1,10,100,1000	std_logic_vector (3 downto 0)	изход	индикация за коефициент на делене
LED_hz	std_logic	изход	честота Hz
LED_khz	std_logic	изход	честота Khz
LED_Mhz	std_logic	изход	честота Mhz
PT	std_logic	изход	положение на точката
EN_DISP1-4	std_logic_vector (3 downto 0)	изход	разряд на индикацията

Упражнение № 19

Среда за автоматизирано проектиране с FPGA и CPLD ИС

1. Създаване на нов проект.

 Стартира се средата за работа ***Project Navigator.***

Това е основният графичен интерфейс на средата ISE. Той е разделен на 5 основни полета, организирани така, че да позволяват лесно контролиране на компонентите на проекта. Те са:

- лента с инструменти *(Toolbar)*;
- йерархия на проектните файлове *(Sources in Project)*;
- задачи за съответния файл *(Process window)*;
- конзолен прозорец *(Transcript Window)*;
- текстов редактор *(HDL Editor/ File Viewer)*.

Създаването на нов проект става като се избере: **File ⇨ New project.** В полето **Project Name** се задава името на проекта. ISE прави директория със същото име на мястото, указано в полето **Project Location.**

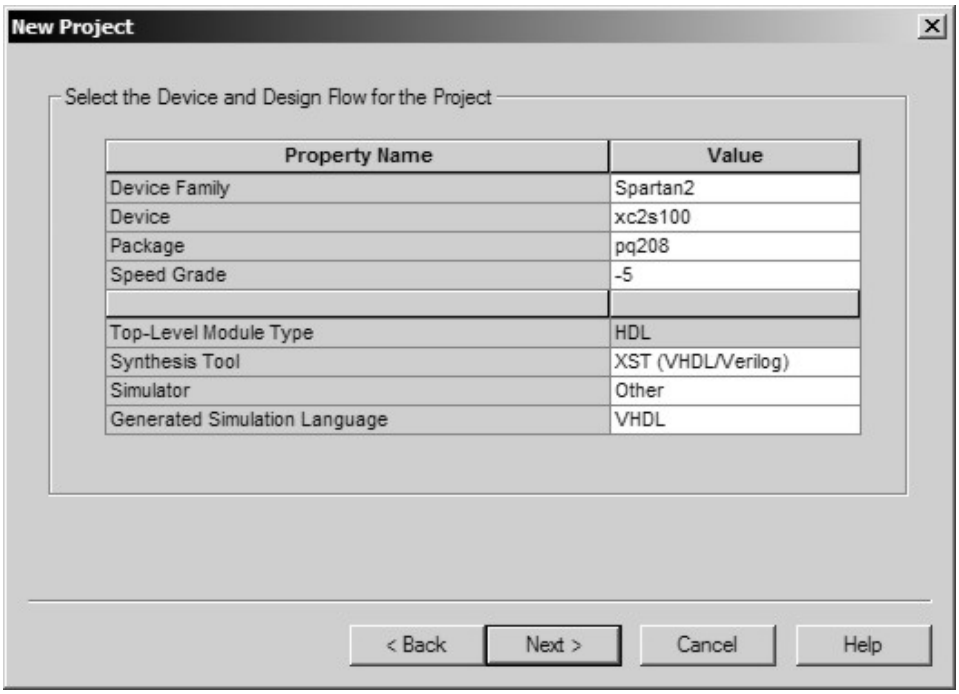
В полето **Top-level module type** се указва желаният вариант за представяне на проекта. Възможностите са:

- **HDL** – представяне посредством език за описание на хардуер (Verilog или VHDL);
- **Schematic** – за въвеждане на модула с най-високо йерархично ниво се използва вградения схематичен редактор ***Xilinx ECS***;
- **EDIF** – използване на стандартен **E**lectronic **D**esign **I**nterchange **F**ormat (edif) файл, генериран от друг софтуерен продукт.

В следващия прозорец (фиг. 1) се задават следните параметри:

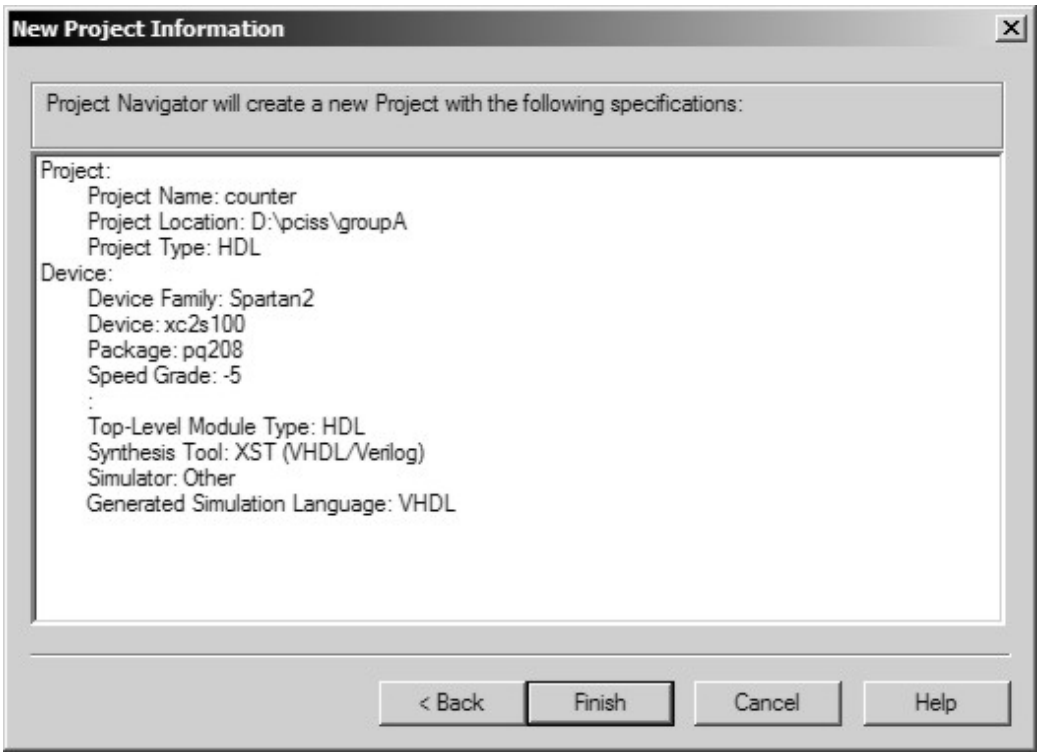
Таблица 1. Параметри на нов проект.

Параметър	Описание
Device Family	Фамилия интегрални схеми (пр. *Spartan2*)
Device	Конкретно устройство от фамилията (пр.*xc2s100*)
Package	Корпус (пр. *pq208*)
Speed grade	Мин. закъснение от извод до извод (пр. *- 5*)
Top-level module	Вариант за представяне на проекта
Synthesis tool	Синтезатор (пр. *XST*)
Simulator	Програма – симулатор (пр. *ModelSim*)
Generated simulation language	Използван език за симулацията (пр. *VHDL*)



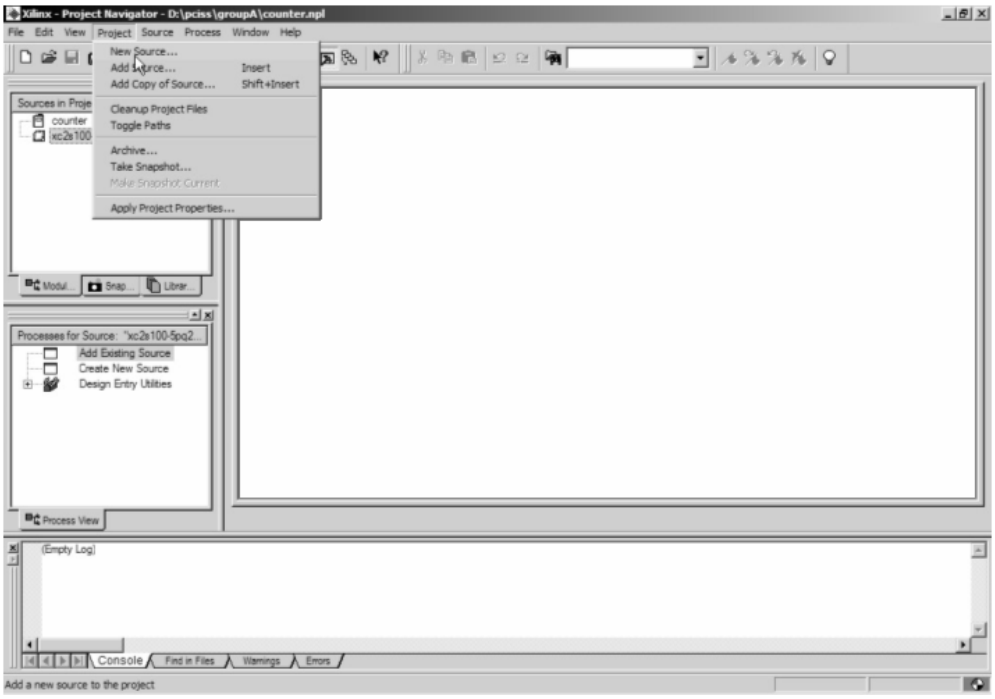
Фиг. 1. Параметри на нов проект

Следващите два прозореца се прескачат с натискане на бутона **Next>**. На екрана се визуализира прозорецът *New Project Information.* В него са изброени зададените параметри на новия проект (име, път, тип на представяне, целево устройство и др.)



Фиг. 2. Статистика за новия проект

Натиска се бутона **Finish**, за да се отвори основния прозорец на средата (фиг. 3).



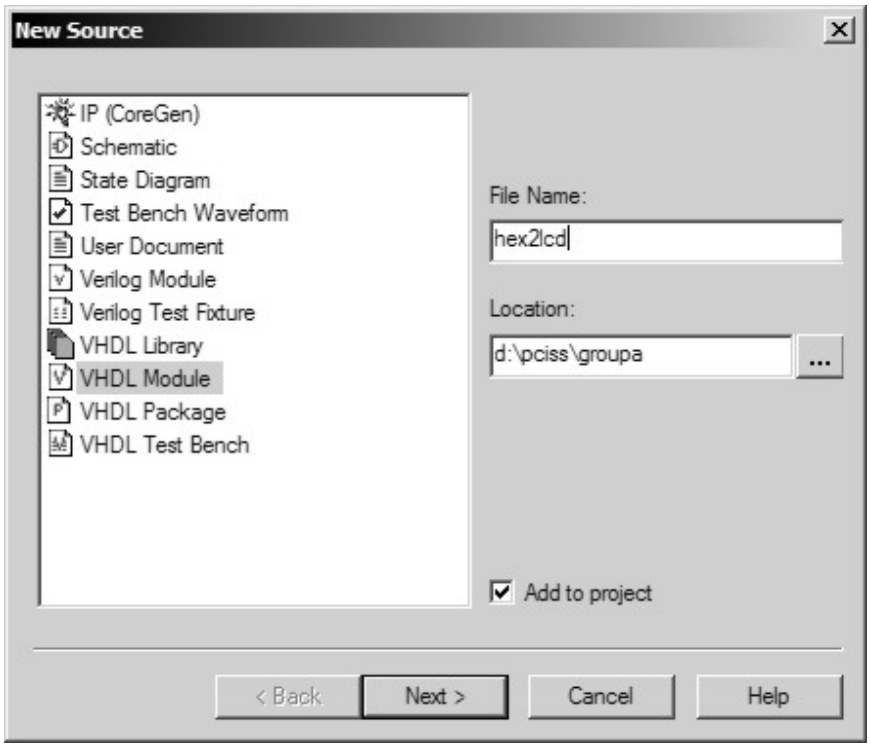
Фиг. 3. Добавяне на нов файл към проекта

Нов файл се създава от менюто **Project** ⇨ **New Source**. Отваря се прозорец *New Source* (фиг. 4), в който са изброени следните възможности за създаване на файл:

Таблица 2. Типове файлове.

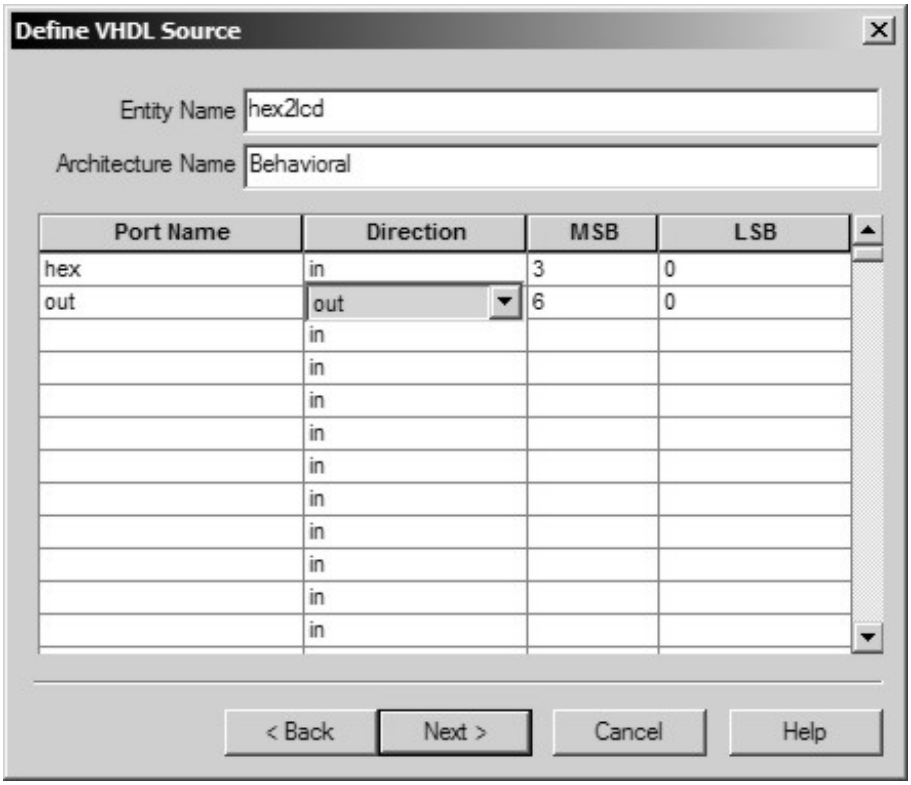
Тип файл	**Описание**
IP (CoreGen)	Файл, съдържащ IP core
Schematic	Принципна електрическа схема
State Diagram	Диаграма на краен автомат
Test Bench Waveform	Графично зададени тестови последователности
User Document	Текстов документ
Verilog Module	Модул, написан на Verilog
Verilog Test Fixture	Описание на тестов модул на Verilog
VHDL Library	Библиотека с конструкции на VHDL
VHDL Module	Модул написан на VHDL
VHDL Package	VHDL модул съдържащ пакети
VHDL Test Bench	Описание на тестов модул на VHDL

Задава се име на файла и път до него.
Отметката **Add to project** трябва да е активирана!



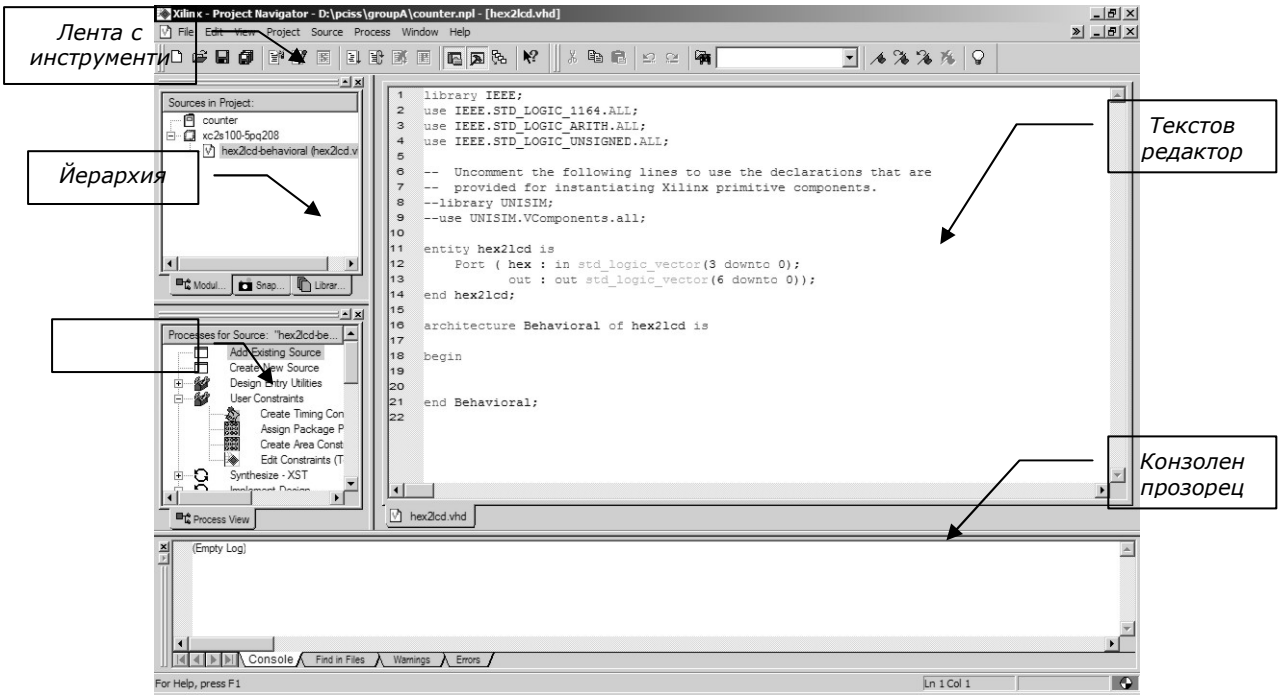
Фиг. 4. Избор на типа на новия файл

В прозореца *Define VHDL Source* (фиг. 5) се задават имена на *entity* и съответстващата *архитектура*. В полето **Port Name** се посочват имена на портовете, а посоката и дължината им – съответно в **Direction** и **MSB/LSB**.



Фиг. 5. Определяне на интерфейсната част на модула

Натиска се бутона **Next>** два пъти. ISE генерира шаблон и отваря текстов редактор за VHDL/Verilog или стартира съответното специализирано приложение (фиг. 6).



Фиг. 6. Работно поле

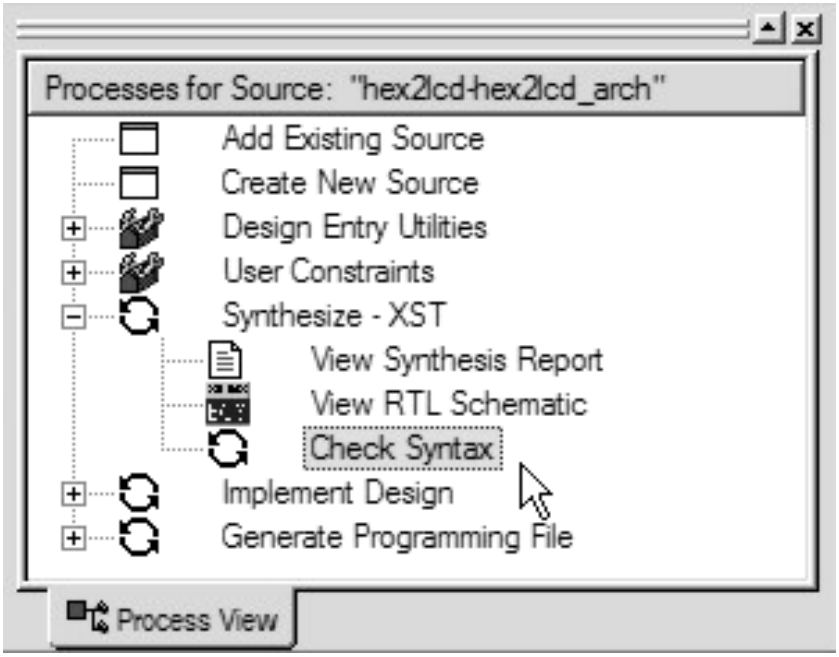
Компоненти на средата **Project Navigator**:

- Лента с инструменти *(Toolbar)* – използва се, за създаване на нов проект, файл, или за настройка на средата;
- Йерархия на проектните файлове *(Sources in Project)* – в този панел са подредени йерархично всички създадени файлове в проекта и задачи свързани с организацията му;
- Задачи за съответния файл *(Process window)* – дава достъп до специфични задачи свързани с избрания в *Sources in Project*, файл;
- Конзолен прозорец *(Transcript Window)* - в него се визуализират съобщения на средата за състояние, за грешки и обобщение на извършените задачи;
- Текстов редактор *(HDL Editor/ File Viewer)* – тук се създава HDL код или се преглеждат текстови документи, генерирани от средата (отчети от синтез или имплементация и др.).

След запознаване със средата се въвежда HDL описанието.

Файлът трябва да се записва често!

Кодът се проверява посредством командата **Check syntax**, която се намира в Process *window* в подменю ***Synthesize – XST*** (фиг. 7).

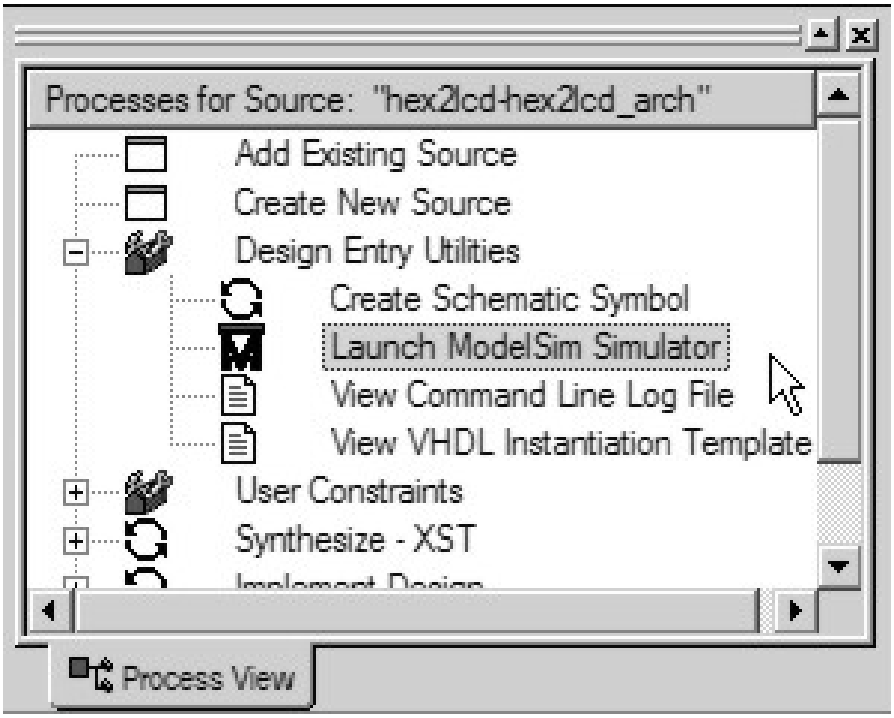


Фиг. 7. Проверка на HDL синтаксиса

В прозореца *Transcript Window* може да бъде проследен процеса на работа.

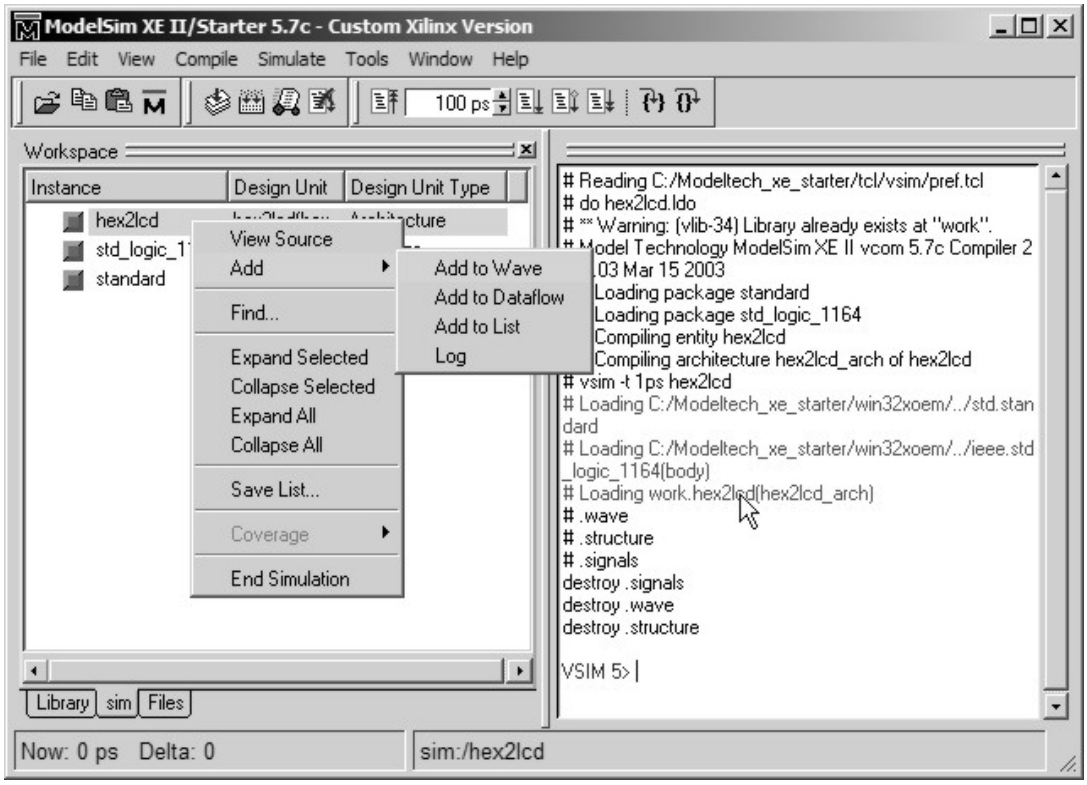
След изчистване на кода от грешки се пристъпва към следващия етап от проектирането – функционална симулация. В средата ISE няма вграден симулатор. Затова се използва външен симулатор – ModelSim.

За да се стартира ModelSim, в *Process window* се избира **Launch ModelSim Simulator**, както е показано на фиг. 8:



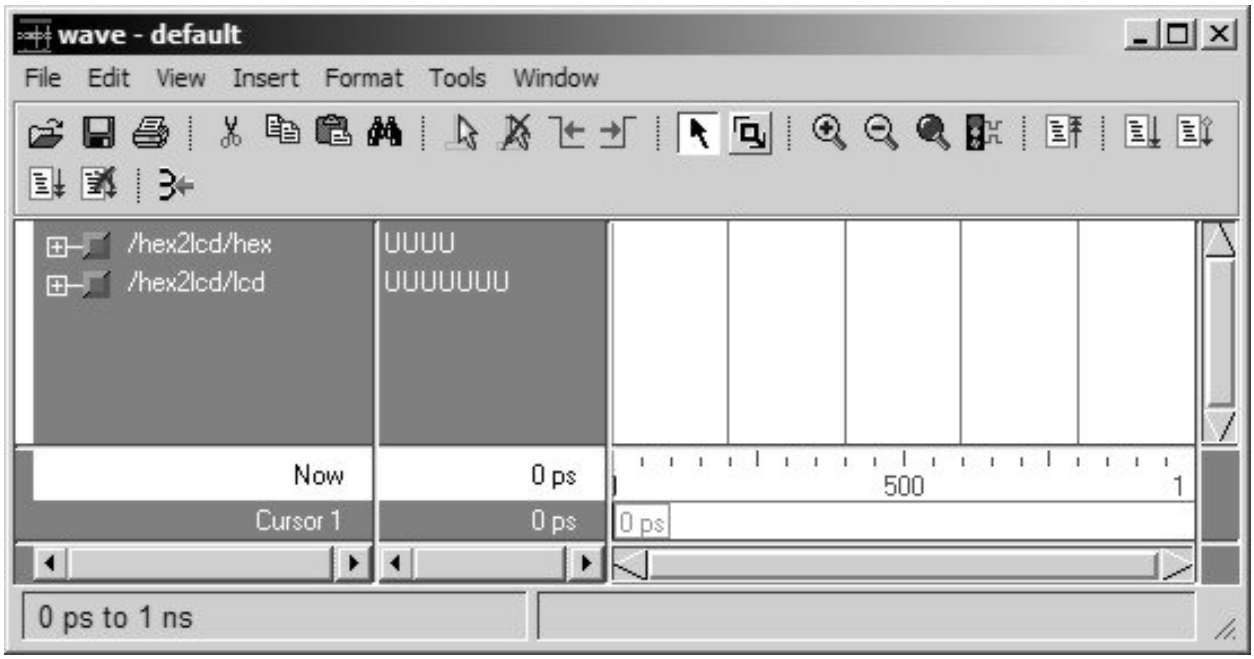
Фиг. 8. Стартиране на ModelSim

2. Симулация с ModelSim.



Фиг. 9. Работно поле на симулатора ModelSim

След стартиране от средата ISE, ModelSim автоматично компилира файла и от своя страна стартира симулатора. В прозореца **sim** се избира името на *entity*, което ще се симулира. Натиска се с десен бутон върху контекстното меню и се избира **Add ⇨ Add to Wave.** В резултат на тази операция се отваря прозореца *Wave* (фиг. 10).



Фиг. 10. Прозорец Wave

Необходимо е да са зададени входни въздействия, за да се изследва реакцията на изходите!

В средата на ModelSim входните въздействия се задават с помощта на командата **force**, която както всички команди, се задава в командния ред. Синтаксиса на командата **force** (в най–простия и вид) е:
force <item_name> <value> [<time>][, <value> <time> ...], където:

<item_name> - име на сигнала, който се променя;
<value> - стойност, която сигнала приема;
<time> - момент на присвояване на стойността <value>.

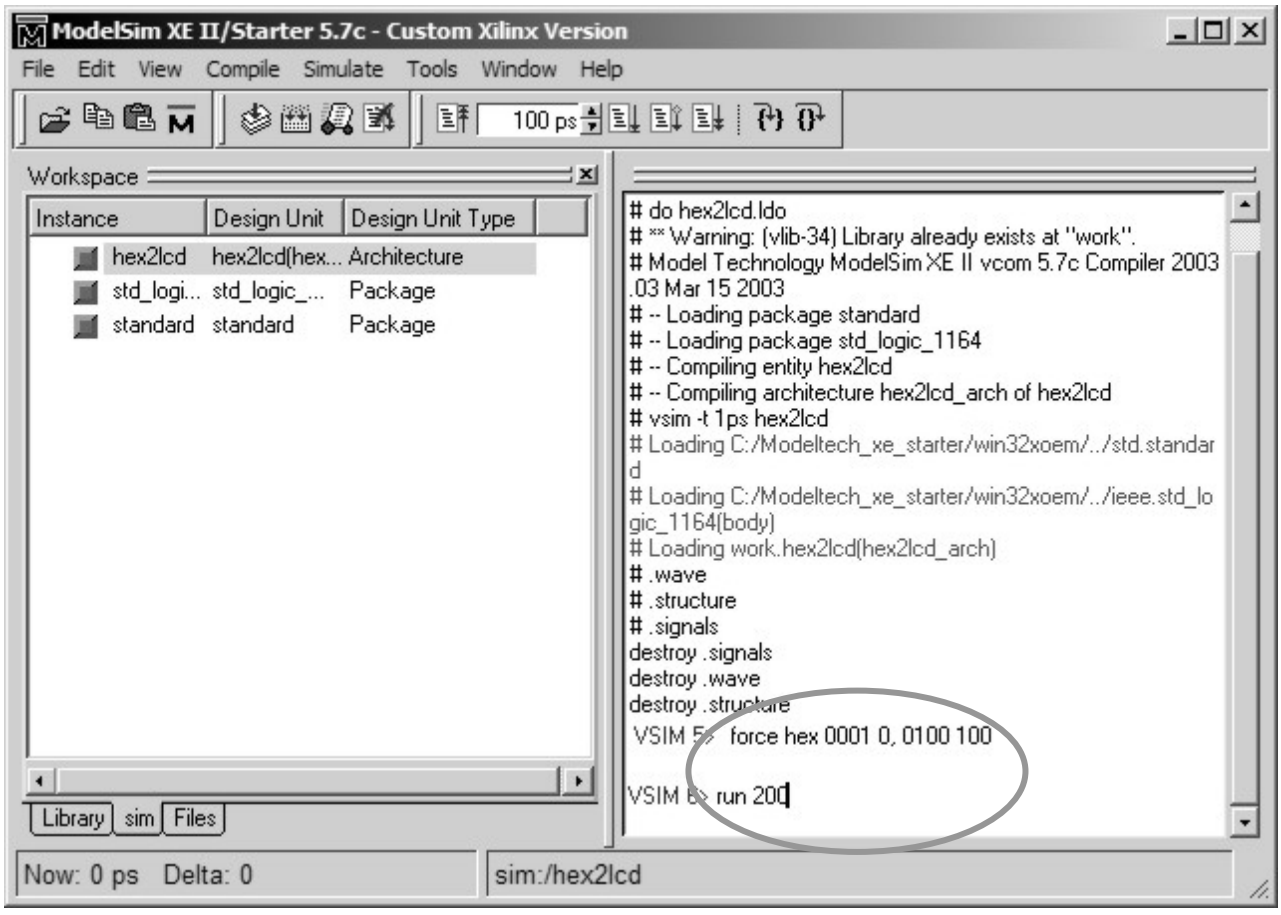
За изпълнение на симулацията се използва командата **run.** Синтаксиса на командата **run** (в най – простия и вид) е:

run [<timesteps> [<time_units>]], където:

<timesteps> - брой времеви единици;
<time_units> - размерност.

Пример (фиг. 11):

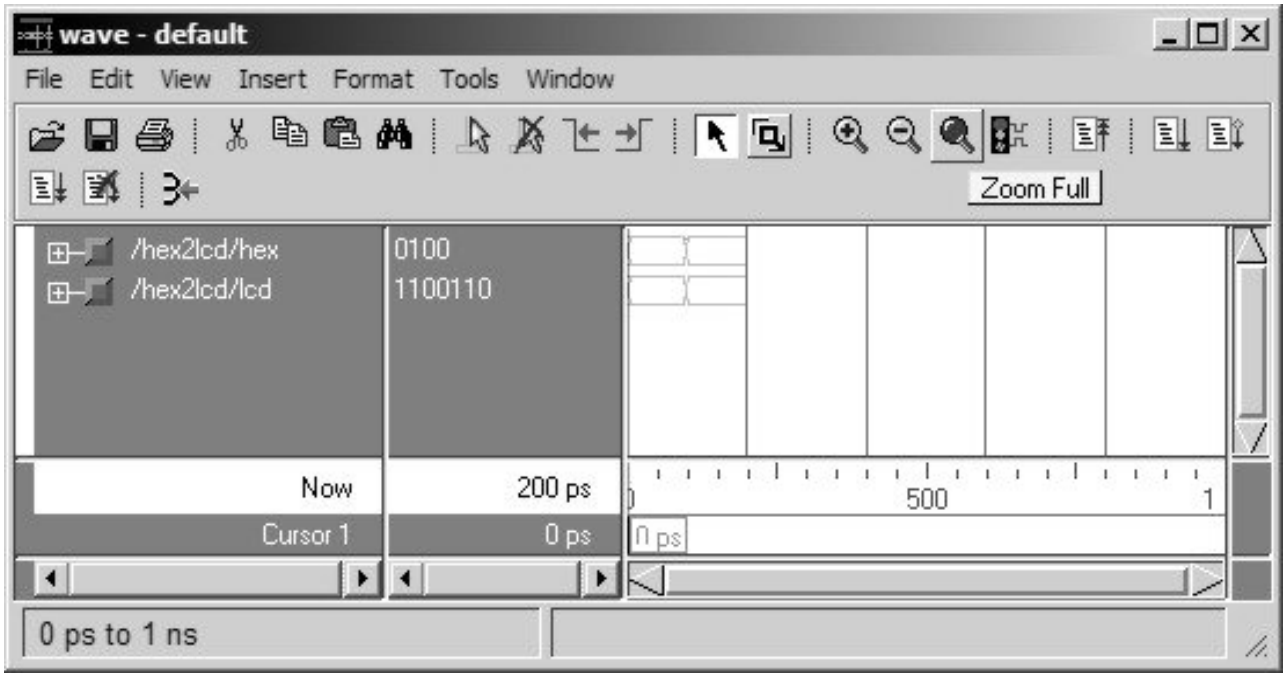
force hex 0001 0, 0100 100
run 200



Фиг. 11. Примерна симулационна постановка

Пълно описание на командите може да бъде намерено в ModelSim User Guide.

В прозореца *Wave* се изчертават времедиаграми, според зададените въздействия и поведението на HDL модела (фиг. 12).



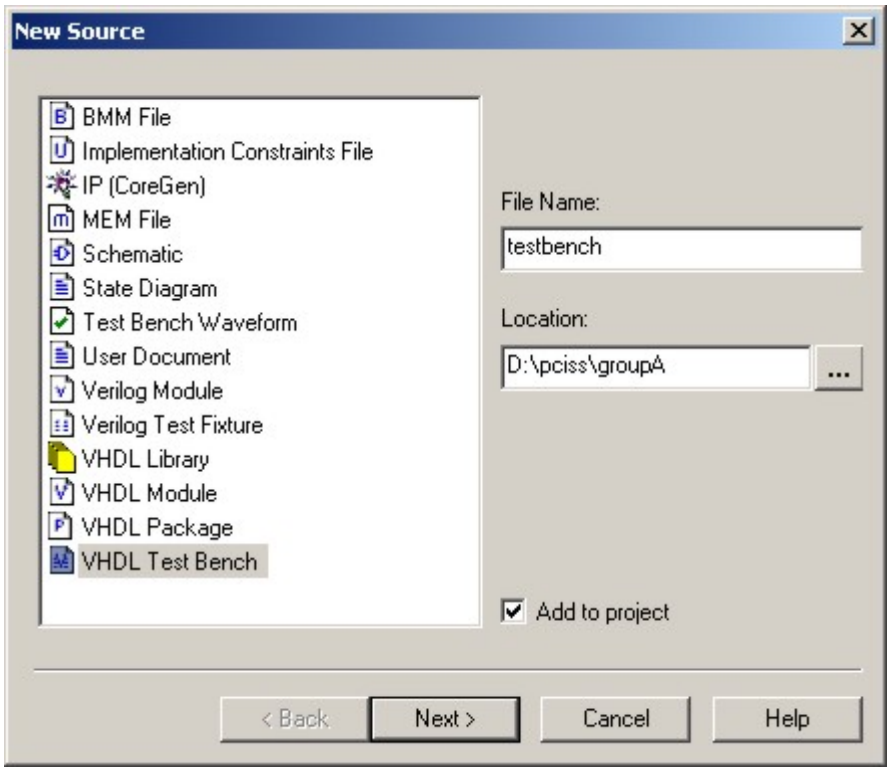
Фиг. 12. Времедиаграми в прозореца Wave.

Посредством бутоните **Zoom**, може да се увеличава или намалява размера на времедиаграмите.

3. Симулация на *test bench* модул.

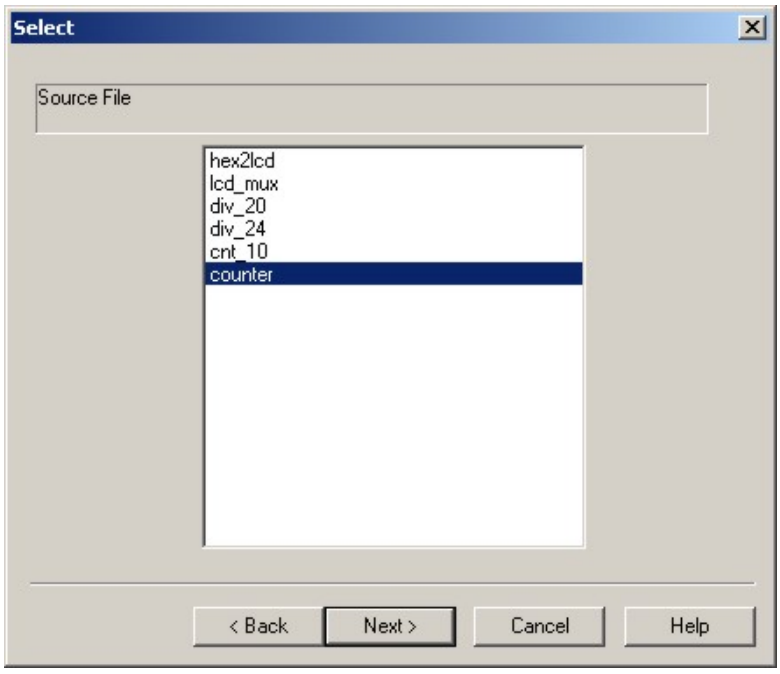
За да се провери дали даден VHDL код е коректно описан, е необходимо да се направи функционална симулация. За целта могат да се зададат тестови въздействия ръчно в средата на симулатора. Друг начин за задаване на тестови въздействия е чрез самия език VHDL. Създава се т. нар. тестова постановка *(test bench).* Важна особеност е, че не се задават входно-изходни портове. Така модулът представлява затворена система, съдържаща тествания компонент, генератор на входни въздействия и средства за визуализация на изходната реакция.

В ***ISE WebPack*** е заложена възможността за създаване на тестови модули. За създаване на нов *test* bench се добавя нов файл към проекта и се избира опцията **“VHDL Test bench”,** както е показано на фиг. 13. Препоръчително е името на файла, съдържащ тестовия модул да бъде подобно на тествания модул.



Фиг. 13. Създаване на нов test bench

ISE WebPack ще пита към кой файл от йерархията на текущия проект да насочи тестовия модул (фиг. 14).

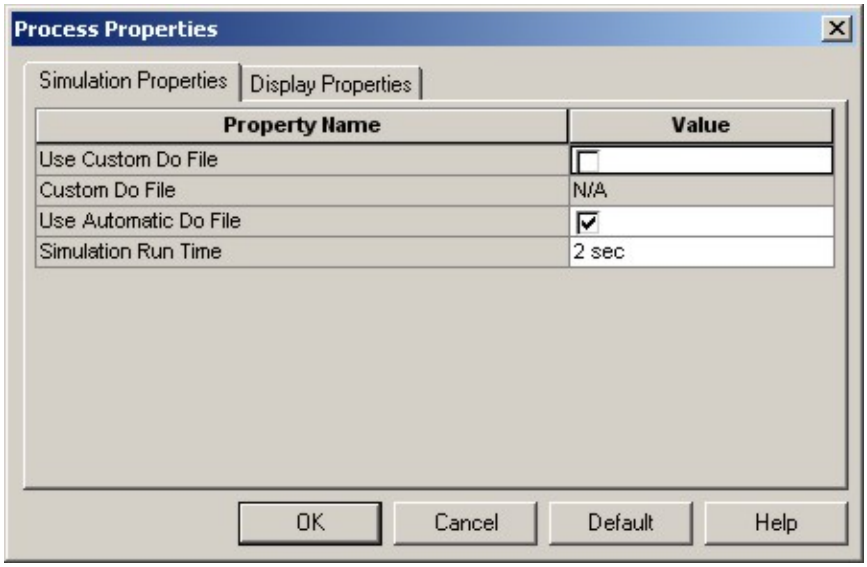


Фиг. 14. Избор на модул за тестване

Така в прозореца *Sources in Project* се добавя файл, съдържащ шаблон за тестов модул, генериран на базата на избрания в предходната стъпка файл. Този шаблон съдържа вмъкнат компонент за тест и сигнали, съответстващи на входно-изходните му портове.

Посредством тези сигнали се задават входни въздействия и се наблюдават изходните реакции на тествания модул.

При маркиран тестов модул, в прозореца *Process* се избира **Simulate Behavioral Model** и менюто **Process ⇨ Properties**, за да бъдат извикани настройките за симулацията (фиг. 15). Времето за симулация може да се зададе в полето **Simulation Run time.**



Фиг. 15. Настройки за симулация

Симулаторът ***ModelSim*** е напълно интегриран в средата ***ISE WebPack.*** Всички файлове, необходими за симулация на *test bench* се генерират автоматично с настройки по подразбиране. Процеса на симулация се стартира с **Process ⇨ Run**. След стартиране на ***ModelSim*** автоматично се визуализират резултатите от симулацията.

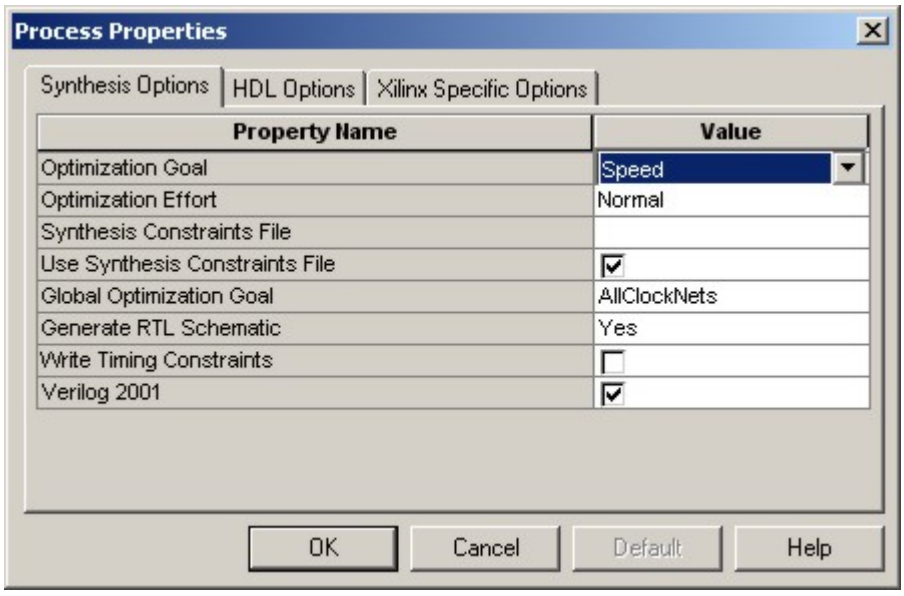
Менютата в Process window са зависими от избрания в прозореца Sources in Project файл! Когато се изпълняват процесите на синтез и имплементация трябва да е избран файлът, съдържащ HDL модула, а при симулация – съответния файл с test bench.

4. Имплементиране във FPGA.

След като проектът е симулиран успешно, се преминава към етап *“Синтез” (Synthesize)*, където HDL – описанието се превръща в *netlist* (ngc) на ниво логически елементи, съдържащ схемата, която трябва да се имплементира.

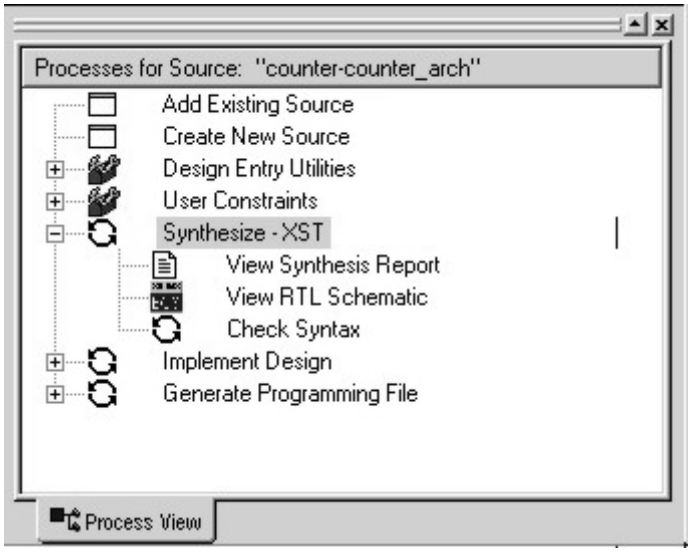
За да се синтезира конкретен файл, той трябва да бъде избран от прозореца *Sources in Project*. Когато синтезирания проект се състои от няколко йерархични нива, се избира файлът, разположен най-високо в йерархията. ***XST*** ще синтезира останалите автоматично. От прозореца *Process window* изберете **Synthesize.** Както и при симулация, настройките на синтезатора могат да се променят от менюто **Process ⇨**

Properties (фиг. 16). Те са специфични и зависят от конкретната реализация. За подробности по всички полета е предвиден бутона **Help**.



Фиг. 16. Настройки на синтезатора

За стартиране на процеса на синтез се щраква с десен бутон върху **Synthesize** в прозореца *Process window* и се избира **Run**.



Фиг. 17. Стартиране на синтезатора XST

След завършване на процеса, средата отбелязва резултата с графичен знак, съответно:

✔ - Процесът е завършен успешно.

! - Открити са не фатални грешки *(Warnings)*. Синтезирана е схема.

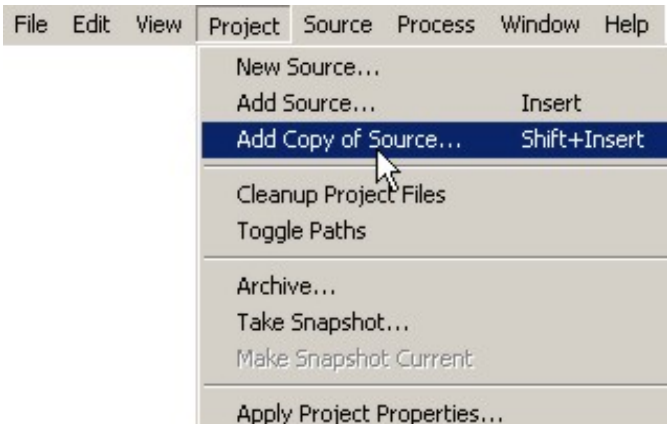
✘ - Открити са фатални грешки *(Errors)*. Процесът е прекратен, схема не е синтезирана.

- Промяна на данните след последното преминаване през процеса.

Синтезираната логическа схема се визуализира, като в прозореца *Process* се избира **Synthesize – XST ⇨ View RTL Schematic**. След завършване на синтеза автоматично се генерира отчет за извършената операция. За да бъде прегледан, се щраква два пъти на **Synthesize – XST ⇨ View Synthesis Report**.

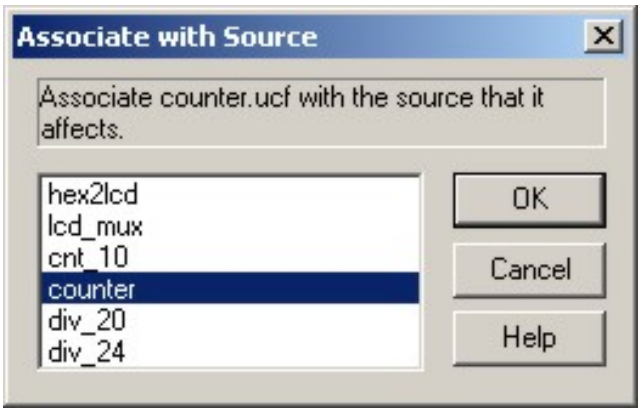
Следващата фаза от имплементацията използва създадения *netlist* и допълнителни условия *(constraints file)* с помощта, на които синтезатора пресъздава проекта с наличните в програмируемата матрица ресурси. Допълнителните условия могат да се отнасят към заемана площ, времеви параметри или желани входно-изходни връзки на чипа.

Първата стъпка е операцията *Translate*. Тя проверява съвместимостта на зададените ограничения (*constraints*) и генерирания *netlist* с избраната архитектура. За добавяне на готов *constraints* – файл се избира **Project ⇨ Add Copy of Source** (фиг. 18).



Фиг. 18. Добавяне на локално копие на файл към проекта

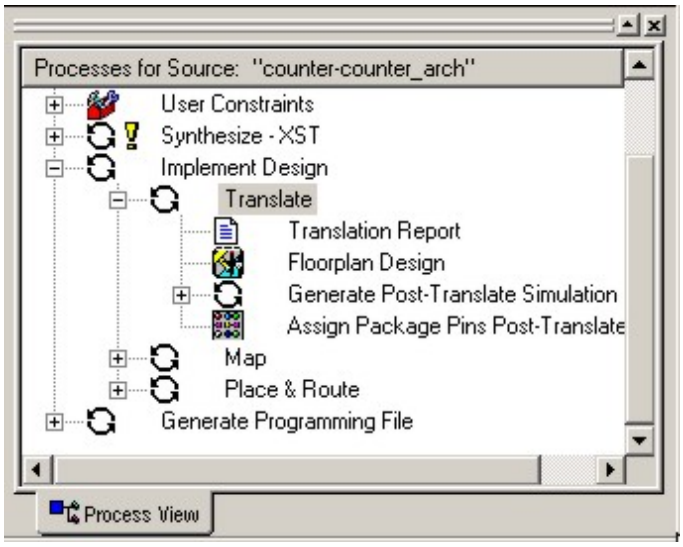
Избира се желаният файл с разширение **.ucf,** след което средата изисква да се укаже модул, към който да отнесе файлът (фиг. 19).



Фиг. 19. Свързване на ucf файл с модул от проекта

В прозореца *Sources in Project* е добавен файлът с *user constraints*.

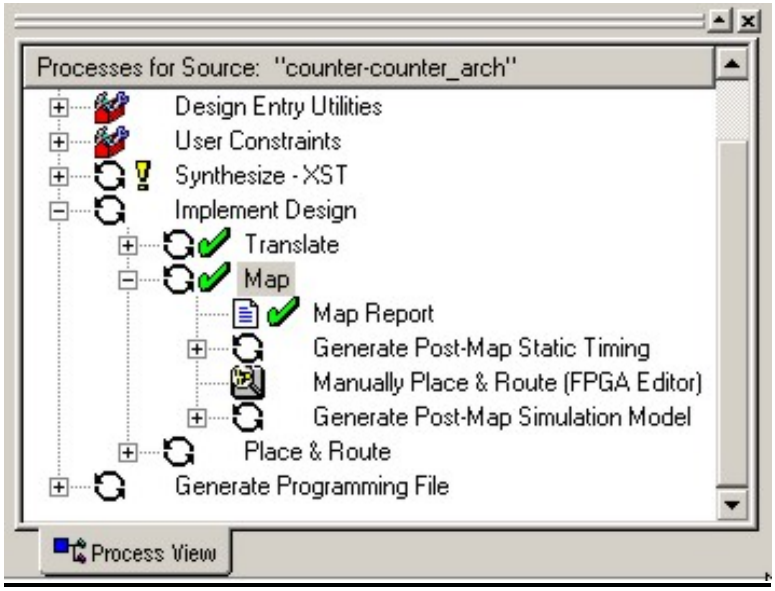
За стартиране на операцията *Translate* се маркира процеса в прозореца *Process window* (фиг. 20) и се избира **Process** ⇨ **Run** от лентата с менюта. Резултата от операцията се отразява със съответните знаци както при процеса *Synthesize*.



Фиг. 20. Стартиране на процеса Translate

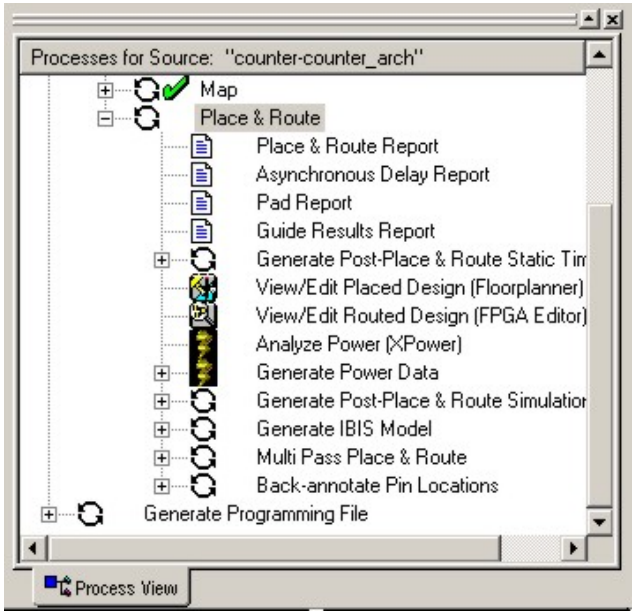
Следващата стъпка от имплементацията е процесът *Map*, при който ресурсите на устройството се разпределят спрямо конкретния проект. Процесът е технологично зависим. Като входни данни разработваният проект приема ограниченията от **ucf** и технологична библиотека за съответното устройство. На този етап се извършва и оптимизация на синтезираната схема в зависимост от желаните резултати (скорост/заета площ).

Резултата от операцията се записва в отчет, който се намира в *Map Report* файл и се отразява с индикатор в *Process window* (фиг. 21).



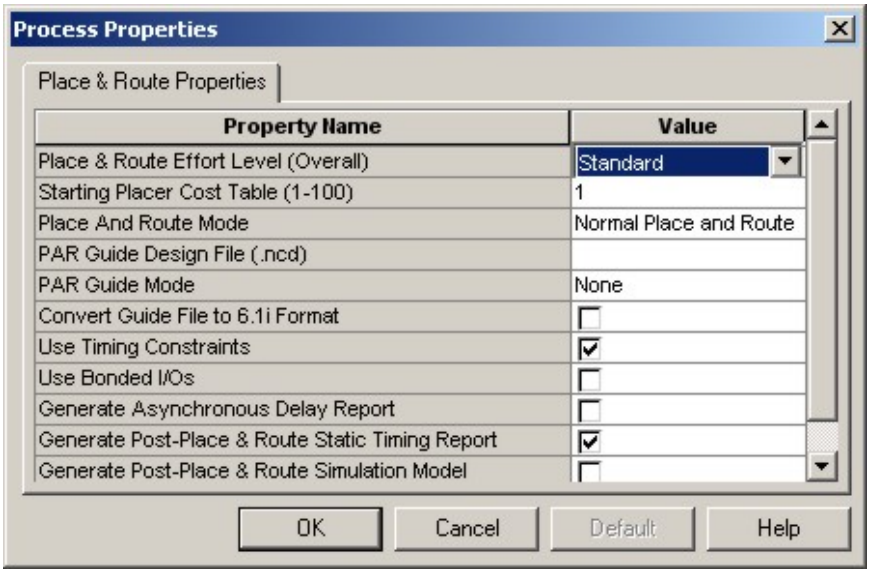
Фиг. 21. Стартиране на процеса Map

Конкретното разположение и свързване на логическите клетки се извършва при процеса *Place&Route*, с който завършва етапа на имплементация. Избират се най-подходящите разположения и свързващи магистрали за съответната схема с оглед спазване на заданието и ограниченията и се генерира HDL модел за ресимулация на проекта, в който са отразени всички закъснения на тригери, комбинационни елементи и свързващи шини. Процеса се стартира от *Process window* (фиг. 22) и състоянието му се отразява с цветни индикатори и в отчет (*Place&Route Report)*.



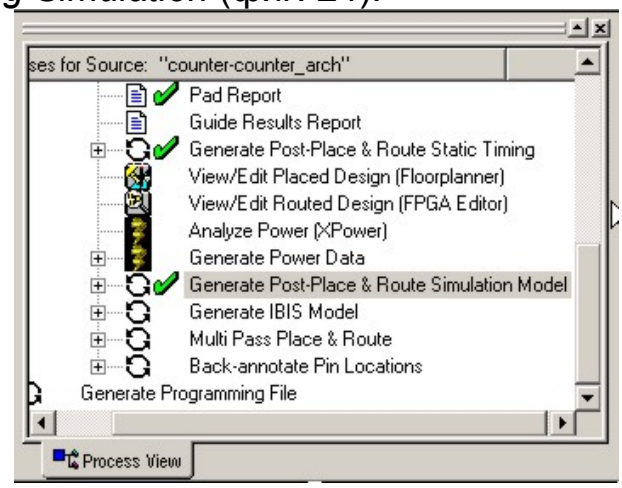
Фиг. 22. Стартиране на процеса Place & Route

Параметри на процеса се задават от **Process ⇨ Properties** (фиг. 23).



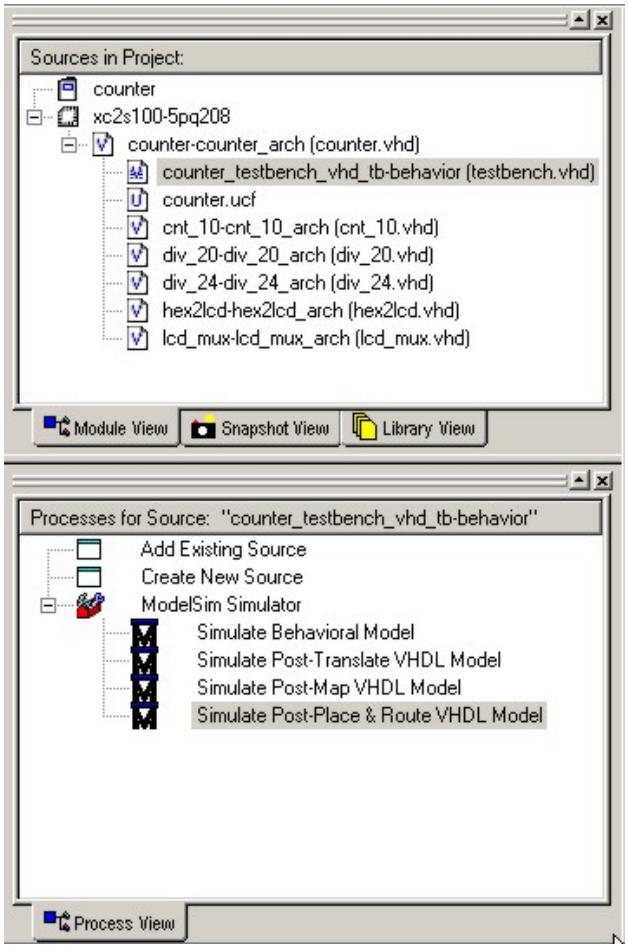
Фиг. 23. Задаване на параметри на процеса PAR

След завършване на *Place&Route* трябва да се генерира модел за ре-симулация – *Timing Simulation* (фиг. 24).



Фиг. 24. Генериране на модел за ресимулация

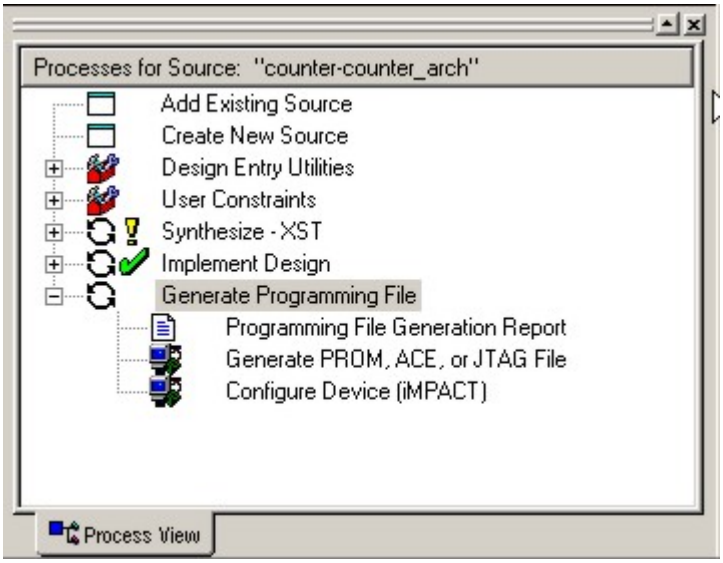
Този модел ще бъде използван при симулация на поведението на реалната схема, заложена във FPGA – устройството. За да се симулира модела, в Sources *in Project* се избира тестовия модул и се стартира симулатора със Simulate **Post-Place & Route VHDL Model** от *Process window* (фиг. 25).



Фиг. 25. Стартиране на ресимулация

5. Конфигуриране на устройството.

Ако схемата на описания проект е готова и отговаря на изискванията за работа с избраната програмируема матрица, последната стъпка към “оживяването” й е зареждането в избрания чип. Изготвя се **.bit** файл, който се използва от програмата ***iMPACT*****,** за да се прехвърли конфигурацията в чипа. Изготвянето на файла се осъществява с командата **Generate Programming File** от *Process window* (фиг. 26).



Фиг. 26. Стартиране на процеса на конфигуриране

Важен параметър на този процес е видът на тактовия сигнал на системата за програмиране. Той се задава от Properties (**Process ⇨ Properties**). Променливата **Startup Clock** трябва да има стойност **JTAG Clock**. Щраква се два пъти върху **Generate Programming File**, за да се стартира процеса и изготви **.bit** файл.

Упражнение № 20

Проектиране на цифров дешифратор с VHDL

1. Въведение.

В настоящото упражнение трябва да се проектира цифров дешифратор, който да преобразува 4-битово число, кодирано с двоично – десетичен код, в 7-битово число, кодирано в 7-сегментен код. Този проблем се среща често при практическата реализация на устройства, извеждащи информация посредством светодиодни или LCD индикатори. За реализацията на заданието се използва професионален софтуер – Xilinx ***ISE*** на фирмата Xilinx (www.xilinx.com), а за функционална симулация – симулатор ***ModelSim*** на фирмата Mentor Graphics (www.mentor.com).

Последователност на работа:

1) Съставяне на VHDL модел на дешифратора;
2) Проверка на синтаксиса и корекции на грешки;
3) Симулация на VHDL модела;
4) Снемане на симулационни резултати.

2. Цели на упражнението.

След изпълнението на поставените задачи, студентите ще могат:

- Да създават VHDL модел;
- Да изпълняват елементарни симулации на VHDL модел, посредством средата на ***ModelSim***;
- Да работят със средите ***ISE WebPack*** и ***ModelSim***.

3. Задачи за изпълнение.

1) Запознаване със средата за проектиране ***ISE WebPack 6.1***.
2) Да се проектира цифров дешифратор със следните характеристики:
   - **Вход:** четирибитово BCD число от тип *std_logic_vector*;
   - **Изход:** двоична комбинация, представяща 7-сегментния код на входното число, от тип *std_logic_vector*.
3) Да се провери създадения VHDL модел.
4) Да се симулира модела с ***ModelSim***. Да се провери функционалността чрез задаване на всички цифри.
5) Да се снемат времедиаграмите.

- **Блокова схема на дешифратора.**

Фиг. 1. Блокова схема

- **Таблица на истинност.**

Таблица 1. Таблица на истинност

Входове		**Изходи**
Десетично число	Двоично число	
0	0000	0111111
1	0001	0000110
2	0010	1011011
3	0011	1001111
4	0100	1100110
5	0101	1101101
6	0110	1111101
7	0111	0000111
8	1000	1111111
9	1001	1101111
За всяка друга стойност		0111111

Упражнение № 21

Проектиране на делител на честота

1. Въведение.

В настоящото упражнение трябва да се проектира цифров делител на честота. В практиката като делител се използва брояч, един от разредите, на който е изведен като изход. В зависимост от поредния номер на разряда (n), който се използва, получения сигнал ще бъде с честота 2^n пъти по-ниска от тактовата честота. Проектирайки този делител, студентите ще се запознаят с основни конструкции, използвани в описанията на VHDL структури.

Последователност на работа:

1) Съставяне на VHDL модел на делителя на честота;
2) Проверка на синтаксиса и корекции на грешки;
3) Съставяне на тестова постановка *(test bench)*;
4) Симулация на проекта;
5) Снемане на симулационни резултати.

2. Цели на упражнението.

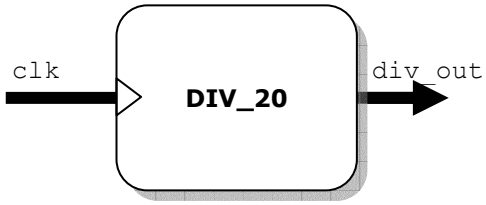
След изпълнението на поставените задачи, студентите ще могат:

- Да описват последователностна логика с VHDL;
- Да използват оператор **process**, условен преход, конструкции за работа по фронт на сигнал, оператори за паралелно присвояване *(concurrent signal assignment statements)*;
- Да използват компоненти във VHDL модул;
- Да описват тестови постановки *(test bench)*.

3. Задачи за изпълнение.

1) Да се проектира делител на честота с коефициент на делене 2^{20}.
2) Модулът има следните входно - изходни сигнали:
   - **Вход:** Тактов сигнал от тип **std_logic**;
   - **Изход:** еднобитов изход от тип **std_logic**.
3) Да се провери синтактично създадения VHDL модел;
4) Да се състави *test bench* за функционална проверка на делителя.
5) Да се симулира модела с ModelSim. Да се провери функционалността му като се изследват времевите параметри на получения в изхода импулс;
6) Да се снемат времедиаграмите.

- **Блокова схема на модула.**



Фиг. 1. Блокова схема

Упражнение № 22

Проектиране на BCD брояч

1. Въведение.

В настоящото упражнение трябва да се проектира двоично - десетичен брояч, като се използва метода на крайните автомати.

Последователност на работа:

1) Съставяне на VHDL модел на брояча;
2) Проверка на синтаксиса и корекции на грешки;
3) Съставяне на *test bench*;
4) Симулация на проекта;
5) Снемане на симулационни резултати;

2. Цели на упражнението.

След изпълнението на поставените задачи, студентите ще могат:

- Да синтезират крайни автомати (Finite State Machines);
- Да описват FSM чрез VHDL;
- Да ползват оператор **case**;
- Да дефинират асинхронна инициализация във VHDL.

3. Задачи за изпълнение.

1) Да се състави графичен модел на краен автомат с 10 състояния (едно за всяка цифра) и асинхронно задаване на начално състояние, който работи като BCD брояч.
2) Да се състави VHDL код, описващ крайния автомат, със следните изисквания:

a. Входно – изходни сигнали:

- **Вход:** тактов, нулиращ, count_enable от тип **std_logic**;
- **Изход:** четирибитов изход от тип **std_logic_vector**.

b. Преминаването през последователни състояния трябва да се извършва при нарастващ фронт на тактовия сигнал, и ‘1’ на входа count_enable.

c. Нулиращият вход е с активно състояние ‘0’ и най-висок приоритет.

3) Да се провери създадения VHDL модел.
4) Да се състави *test bench* за функционална проверка на крайния автомат.
5) Да се симулира модела с ModelSim. Да се провери функционалността му.
6) Да се снемат времедиаграмите.

Упражнение № 23

Проектиране на управляваща схема за LCD индикация

1. Въведение.

В настоящото упражнение трябва да се проектира модул за управление на седемсегментен LCD индикатор. Целта на този модул е да предпази от поляризация течните кристали в индикаторния елемент. В противен случай, след време цифрата би избледняла и изчезнала. Ще се синтезира логическа схема въз основа на описанието на език от високо ниво, използвайки вградения в ISE WebPack синтезатор XST.

Последователност на работа:

1) Съставяне на VHDL модел на управляващата схема;
2) Проверка на синтаксиса и корекции на грешки;
3) Съставяне на *test bench*;
4) Симулация на проекта;
5) Снемане на симулационни резултати;
6) Синтез на логическа схема;
7) Имплементация в FPGA;
8) Постсимулация на имплементирания модел *(post-place&route simulation)*.

2. Цели на упражнението.

След изпълнението на поставените задачи студентите ще могат да синтезират логически схеми и да имплементират проект в програмируеми устройства на *Xilinx*.

3. Задачи за изпълнение.

1) Да се проектира управляващо устройство за LCD индикация.
2) Да се състави VHDL код, описващ управляващия модул, със следните изисквания:

a. Входно – изходни сигнали:

- **Вход:** тактов и управляващ сигнал от тип **std_logic,** 7-битов за данни от тип **std_logic_vector**;
- **Изход:** седембитов за данни (сегменти) от тип **std_logic_vector,** за десетична точка и общ за индикатора от тип **std_logic**.

b. Изходите за сегменти по нарастващ фронт на тактовия сигнал трябва да получат стойността на входовете за данни, сумирани по модул 2 с управляващия сигнал.

c. Общият извод на индикатора и извода за десетична точка присвояват стойността на управляващия сигнал по нарастващ фронт на тактовия сигнал.

3) Да се симулира модела с ModelSim. Да се провери функционалността му.
4) Да се снемат времедиаграмите.
5) Използвайки XST, да се синтезира логическа схема въз основа на описанието на VHDL.
6) Да се имплементира проекта в устройство *Xilinx Spartan2 x2s100pq208-5.*
7) Да се анализират резултатите от процесите на синтез и имплементация.

Упражнение № 24

Структурно проектиране. Проектиране на десетичен брояч.

1. Въведение.

В настоящото упражнение трябва да се проектира двуразреден десетичен брояч, чиято стойност да бъде индицирана на LCD индикатор. Задачата изисква използването на готови VHDL модули, които да бъдат събрани в система и отразява методите на структурно проектиране на интегрални схеми и системи.

Последователност на работа:

1) Запознаване с блоковата схема на устройството;
2) Подготовка на структурните блокове;
3) Свързване на отделните компоненти в цяла система чрез сртруктурен VHDL;
4) Задаване условия *(constraints)* на синтезатора;
5) Синтез на проекта;
6) Имплементация на устройството;
7) Снемане на резултати за заетия ресурс на програмируемото устройство;
8) Зареждане на проекта в логическа матрица *Xilinx Spartan2.*

2. Цели на упражнението.

След изпълнението на поставените задачи, студентите ще могат:

- Да описват системи на високо ниво на абстракция;
- Да имплементират цифрови системи в програмируеми логически устройства.

3. Задачи за изпълнение.

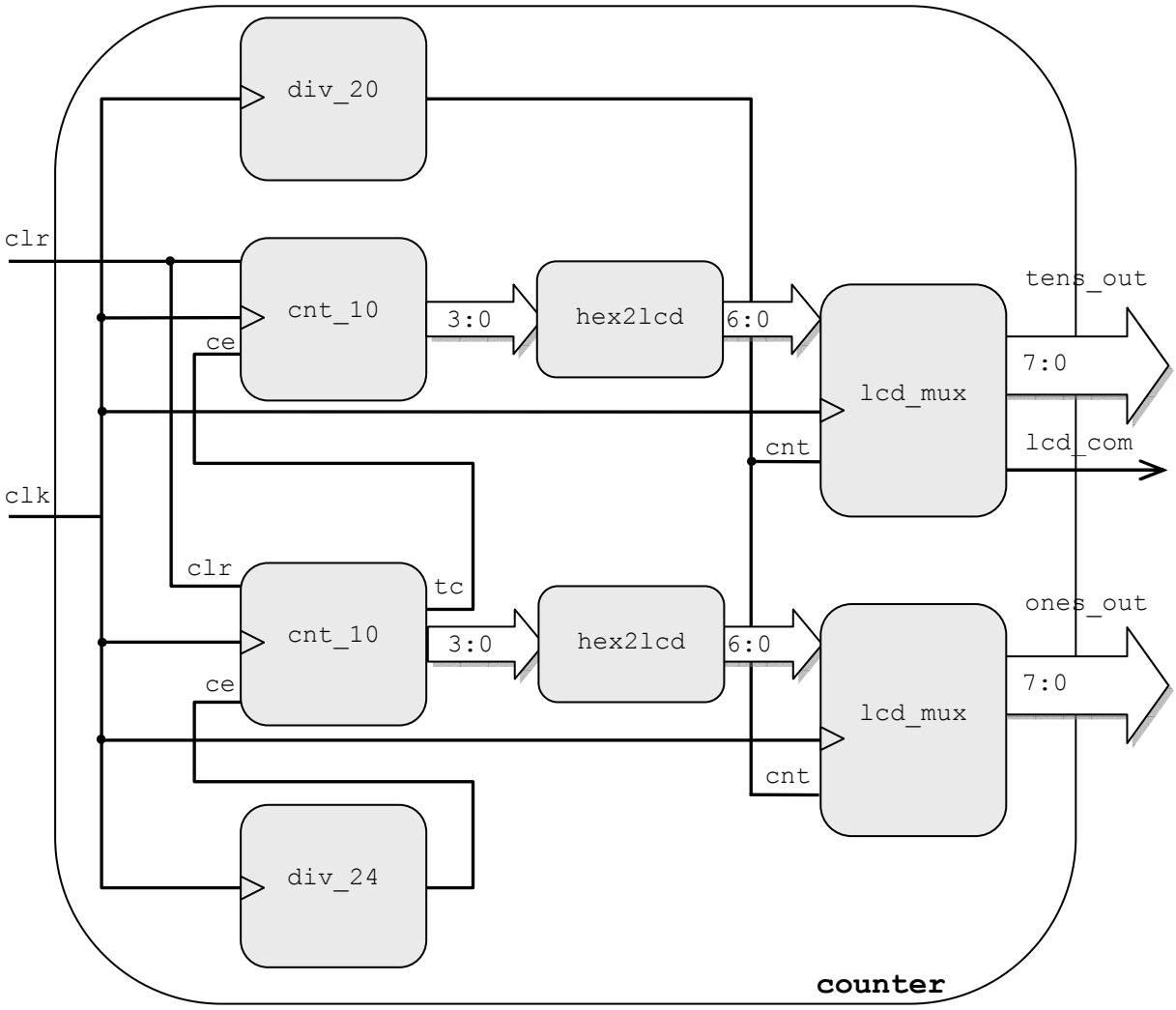
1) Да се състави блокова схема на десетичен брояч със следните входно-изходни сигнали:

- **Вход:** тактов и нулиращ сигнал от тип **std_logic**;
- **Изход:** двуразрядна 7-сегментна LCD индикация с десетична точка (сигнали от тип **std_logic_vector**).

2) Да се опише създадената блоковата схема, като се използва структурен VHDL. Като съставни компоненти да се използват проектираните в предните упражнения модули. VHDL кода на модула div_24 се дава от ръководителя на упражнението.
3) Да се състави *test bench* за функционална проверка на системата.
4) Да се симулира модела с ModelSim. Да се провери функционалността му като се наблюдава изходната поредица спрямо входния тактов сигнал.

5) Да се снемат времедиаграмите.
6) Да се зададат указания за работа на синтезатора *(constraints).* Файлът се дава от ръководителя на упражнението.
7) Да се синтезира логическа схема въз основа на VHDL – кода.
8) Да се имплементира проекта в устройство *Xilinx Spartan2 x2s100pq208-5*.
9) Да се зареди FPGA - устройството със създадения програмен файл.

- **Блокова схема на брояча.**



Фиг. 1. Блокова схема

Литература

1. Манолов Е., *Аналогови интегрални схеми. Схемотехника и проектиране*, Технически университет – София, София, 2002
2. Василева Т., В. Чумаченко, *Машинно проектиране на интегрални схеми и електронни възли*, Техника, София, 1999
3. Михов Г., *Цифрова схемотехника*, Технически университет – София, София, 1998
4. Hastings A., *The Art of Analog Layout*, Prentice Hall, NJ 07458, 2001, ISBN 0-13-087061-7
5. Saint C., J. Saint, *IC Layout Basics – A Practical Guide*, McGraw-Hill, 2002, ISBN 0-07-138625-4
6. Geiger R., P. Allen, N. Strader, *VLSI Design Techniques for Analog and Digital Circuits*, McGraw-Hill, 1990, ISBN 0-07-023253-9
7. Allen P., D. Holberg, *CMOS Analog Circuit Design*, Oxford University Press, 1987, ISBN 0-19-510720-9
8. Gray P., P. Hurst, S. Lewis, R. Meyer, *Analysis and Design of Analog Integrated Circuits*, 4th ed., John Wiley & Sons, 2001, ISBN 0-471-32168-0
9. Weste N., K. Eshraghian, *Principles of CMOSVLSI Design – A System Perspective*, 2nd ed., Addison-Wesley, 1992, ISBN 0-201-53376-6
10. Johns D., K. Martin, *Analog Integrated Circuit Design*, John Wiley & Sons,1997, ISBN: 0-471-14448-7
11. Baker J., H. Li, D. Boyce, *CMOS Circuit Design, Layout, and Simulation*, IEEE Press, USA, IEEE Order Number: PC5689, 1998, ISBN 0-07-002469-3
12. Clein D., *CMOS IC Layout – Concepts, Methodologies, and Tools*, Newnes, 2000, ISBN 0-7506-7194-7
13. Grebene A., *Bipolar and MOS Analog Integrated Circuit Design*, John Wiley & Sons, 1984, ISBN 0-471-08529-4
14. Razavi B., *Design of Analog CMOS Integrated Circuits*, McGraw-Hill, 2000, ISBN 0-07-237371-0
15. Daly J., D. Galipeau, *Analog BiCMOS Design: practices and pitfalls*, CRC Press, 2000, ISBN 0-8493-0247-1
16. Perry D., *VHDL*, 3rd ed., McGraw – Hill Education, 1998, ISBN 0-0704943-3-9
17. Smith D., *HDL Chip Design*, Doone Publications, 1997, ISBN 0-9651934-3-8
18. Breeding K., *Digital Design Fundamentals*, Prentice Hall, 1997, ISBN 0-1321127-7-9
19. Mazor S., P. Langstraat, *A guide to VHDL*, Kluwer Academic Publishers, 1992
20. Holmes C., M. Willoughby, *VHDL Language Course* , Rutherford Appleton Laboratory, 1997

21. Automated Custom Physical Design Flow Guide, Cadence Design Systems Inc., Product Version 5.0, July 2002
22. Cadence Library Manager User Guide, Cadence Design Systems Inc., Product Version 5.0, March 2003
23. Virtuoso Schematic Composer User Guide, Cadence Design Systems, Product Version 5.0, June 2003
24. Affirma Spectre Circuit Simulator Reference, Cadence Design Systems, Product Version 5.0, June 2003
25. Affirma Spectre Circuit Simulator User Guide, Cadence Design Systems, Product Version 5.0, June 2003
26. Affirma RF Simulator User Guide, Cadence Design Systems, Product Version 5.0, June 2003
27. Virtuoso Layout Editor User Guide, Cadence Design Systems, Product Version 5.0, June 2003
28. Virtuoso Layout Accelerator User Guide, Cadence Design Systems, Product Version 5.0, June 2003
29. Virtuoso Parameterized Cell Reference, Cadence Design Systems, Product Version 5.0, June 2003
30. Virtuoso Relative Object Design User Guide, Cadence Design Systems, Product Version 5.0, June 2003
31. Diva Reference, Cadence Design Systems, Product Version 5.0, April 2003
32. IC Shape Based Technology Chip Assembly User Guide, Cadence Design Systems, Product Version 11.0.4, March 2002
33. IEEE Standard VHDL Language Reference Manual, IEEE, ISBN 0-7381-1949-0 SS94817
34. Design Analyzer Reference Manual, Synopsys, Product Version 2002.05, May 2002
35. Design Compiler User Guide, Synopsys, Product Version U-2003.06, June 2003
36. Design Compiler Reference Manual, Synopsys, Product Version U-2003.06, June 2003
37. VHDL Simulation User Guide, Synopsys, Product Version 2002.12, December 2002
38. Scirocco-i User Guide, Synopsys, Product Version 2002.12, December 2002
39. VCS/VCSi User Guide, Synopsys, Product Version 7.0.1, April 2003
40. Programmable Logic Design Quick Start Book, Xilinx, Version 2, June 2003
41. Synthesis and Verification Design Guide, Xilinx, Version 4.0, June 2003
42. XST User Guide, Xilinx, Version 4.0, June 2003
43. Xilinx Constraints Guide, Xilinx, Product Version 6.1i, June 2003

44. Development System Reference Guide, Xilinx, Version 4.0, June 2003
45. ModelSim User Guide, Mentor Graphics, Product Version 5.7c, March 2003
46. ModelSim Command-line Reference Guide, Mentor Graphics, Product Version 5.7c, March 2003

WEB Sites:

- Cadence Design Systems Inc.: http://www.cadence.com
- Mentor Graphics Corporation: http://www.mentor.com
- Документация на Synopsys: http://www.synopsys.com/support/support.html
- Xilinx Data Sheets: http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp
- Xilinx Application notes: http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=Application+Notes
- Xilinx software tutorials: http://www.xilinx.com/support/techsup/tutorials/index.htm
- ModelSim, User manuals, Tech notes: http://www.model.com/support/technotes_all.asp
- Литература, форум и примерни проекти: http://ecad.tu-sofia.bg/soc
- Модули за дистанционно обучение: http://ecad.tu-sofia.bg/education/courses.html

Съдържание

СИСТЕМИ ЗА ПРОЕКТИРАНЕ В МИКРОЕЛЕКТРОНИКАТА

Автори:
© доц. д-р Марин Христов и колектив

Рецензент:
© проф. дтн Тихомир Таков

Даден за печат: м. юли 2004 г.

Излязъл от печат: м. юли 2004 г.

Печатни коли: 9.75

Поръчка № 84

Тираж 100 броя Формат 60/84/16

Цена 6.50 лв. **ISBN 954-438-411-1**

МП Издателство на Технически университет - София