**Лабораториялық жұмыс №1**

**Қарапайым сандық логикалық схемаларды ПЛИС негізінде жобалау**

**Жұмыстың мақсаты**

Логикалық элементтерге негізделген қарапайым цифрлық схемаларды жобалау үшін Verilog аппараттық құралды сипаттау тілімен жұмыс істеуде негізгі білім мен дағдыларды алу.

**Теориялық бөлім**

**ПЛИС**

ПЛИС сандық схемаларды жобалауға мүмкіндік береді. Ал цифрлық схемалар жай ғана белгілі бір тапсырманы орындау үшін біріктірілген логикалық элементтердің (және, немесе, не және т.б.) жиынтығы болып табылады. Сіз жасайтын жобалар жарық диодты жыпылықтайтын есептегіш сияқты қарапайым нәрседен көп ядролы процессор сияқты күрделі нәрсеге дейін болуы мүмкін.ПЛИС-те цифрлық схемаларды жобалау аппараттық құралдарды сипаттау тілдеріндегі бағдарламалау құралдарын қолдану арқылы жүзеге асырылады. Аппараттық құралдарды сипаттау тілдері (HDL) дискретті электрондық жүйелердің архитектурасын және әрекетін сипаттайды. HDL пайдалану графикалық редакторларды қолданумен байланысты әртүрлі схемалық инженерия тәсілдерінен түбегейлі айырмашылығы бар цифрлық схемаларды жобалаудың жаңа технологиясы болып табылады.Қазіргі заманғы бағдарламалау тілдері тек құрылғының құрылымы мен әрекетін сипаттауға ғана емес, сонымен қатар талдау үшін сынақ векторларының тізбегін дайындауға, модельдеуге нәтижелерін тексеруді орындауға, жобаның сыртқы ортасын эмуляциялауға, нәтижелерді әртүрлі тәсілдермен көрсетуге, модельдеуге және т.б. мүміндік береді.

Басқаша айтқанда, бағдарламалау тілдері дизайнды ұсынудың көптеген кезеңдерінде, логикалық синтезден функционалдық және уақытша модельдеуге дейін, қолданылатын қуатты құрал. Ең көп қолданылатын бағдарламалау тілдері Verilog және VHDL болып табылады.

**Verilog аппараттық құралды сипаттау тілі**

Verilog, VHDL бәсекелес тілімен қатар ПЛИС-ті бағдарламалаудың ең кең тараған тәсілі болып табылады. Схеманы қолданатын ПЛИС-ті бағдарламалау таныс екенін байқаған шығарсыз, сондықтан дәл осылай істеу үшін неге күрделі бағдарламалау тілін үйрену керек? Жауап мынада: шын мәнінде, жобалар барған сайын күрделене түскеді, сол себепті оны сызудан гөрі бағдарламалау тілін пайдаланып дизайнды елестету оңайырақ. Verilog басқа бағдарламалау тілдеріне ұқсайды және сіз «if» мәлімдемелерін, код блоктарын және басқа бағдарламаға ұқсас құрылымдарды, соның ішінде сандарды қосу және азайту мүмкіндігін таба аласыз.

Жаңадан бастаушылар жіберетін ең үлкен қателіктердің бірі - HDL кодын цифрлық жабдықты сипаттау құралы ретінде емес, компьютерлік бағдарлама ретінде ойлау. Егер сізде HDL кодының синтезделуі керек екенін білмесеңіз, онда нәтиже сізге ұнамауы мүмкін. Сіздің цифрлық схемаңыз қажет болғаннан әлдеқайда үлкен болуы мүмкін немесе сіздің кодыңыз дұрыс модельденіп, бірақ аппараттық құралда іске асырылмауы мүмкін. Оның орнына дизайн туралы комбинациялық логика, регистрлер және күй машиналары тұрғысынан ойлану керек. Кодтауды бастамас бұрын осы блоктарды қағазға салып, олардың қалай қосылатынын көрсеткен жөн.

Verilog бағдарламасында модульді абстракцияның төрт түрлі деңгейі арқылы анықтауға болады.

* Мінез-құлық (Behavioral) немесе алгоритмдік деңгей: бұл абстракцияның ең жоғарғы деңгейі. Модульді жобалау алгоритмі тұрғысынан жүзеге асыруға болады.Дизайнерге аппараттық құралдарды іске асыру туралы ешқандай білімнің болуы қажет емес.
* Деректер ағынының деңгейі (data flow): Бұл деңгейде модуль деректер ағынын ескере отырып жасалған. Дизайнер дизайндағы әртүрлі регистрлер арасында деректердің қалай тасымалданатынын білуі керек.
* Қақпалар деңгейі (gate level): Модуль логикалық элементтер және осы элементтер арасындағы байланыстар тұрғысынан жүзеге асырылады. Дизайнер логикалық элемент деңгейінде дизайн макетін білуі керек.
* Ауыстырып-қосқыш деңгейі (switch level): бұл абстракцияның ең төменгі деңгейі. Дизайн коммутаторлар/транзисторлар арқылы жүзеге асырылады.

Қақпа деңгейіндегі модельдеу абстракцияның ең төменгі деңгейі болып табылады, өйткені коммутатор деңгейіндегі абстракция сирек қолданылады. Жалпы алғанда, қақпа деңгейіндегі модельдеу толық қосқыштар, мультиплексорлар және т.б. сияқты цифрлық тізбектерде ең төменгі деңгей модульдерін жүзеге асыру үшін қолданылады. Verilog HDL барлық негізгі қақпалар үшін қақпа примитивтері бар. Мысалы, and, nand, or, nor, xor, xnor, not, buf және т.б. Примитивтерді қолдануға арналған синтаксис



Мұнда and кілт сөзі ЖӘНЕ қақпасының көшірмесін жасайды, 𝑛𝑎𝑚𝑒\_𝑜𝑝𝑡𝑖𝑜𝑛𝑎𝑙 қақпаның жасалған көшірмесінің атауы, ол міндетті емес. Аргументтер 𝑜𝑢𝑡, 𝑖𝑛1, 𝑖𝑛2, … , 𝑖𝑛𝑁 - бұл құрылған элементтің кірістері мен шығыстары, біріншісі әрқашан жалғыз шығыс, ал кірістерден кейін екіден көп болады.

Деректер ағынының деңгейінің дизайны үздіксіз тағайындау assign операторын тағайындау арқылы жүзеге асырылады. Аssign операторының синтаксисі:



Әрбір үздіксіз тапсырманы схеманың бөлігі ретінде қарастыруға болады. Сол жақтағы сигнал шығыс болып табылады, ал оң жақтағы өрнекте қолданылатын сигналдар кірістер болып табылады. Өрнек осы тізбектің қызметін сипаттайды. Мысалы, мәлімдемені қарастырайық:



Бұл ЖӘНЕ операциясын орындайтын схема. in1 немесе in2 мәнін өзгерткенде, бұл оператор іске қосылады және өрнек есептеледі. Жаңа мән тарату кідірісінен кейін шығыс сигналына тағайындалады.

Мінез-құлық деңгейінде жұмыс істеу always блокты пайдалануды білдіреді. Always блокты мінез-құлқы ішкі процедуралық нұсқаулармен сипатталатын қара жәшік ретінде қарастыруға болады. Процедуралық нұсқаулар әртүрлі құрылымдарды қамтиды, бірақ олардың көпшілігінде нақты аппараттық аналогтары жоқ. Нашар кодталған блок жиі қажетсіз күрделі іске асыруға әкеледі немесе мүлде синтезделмейді. Always блокты пайдалану синтаксисі:

𝑎𝑙𝑤𝑎𝑦𝑠@([ сезімталдық тізімі])

𝑏𝑒𝑔𝑖𝑛

… процедуралық нұсқаулар …

𝑒𝑛𝑑

Сезімталдық тізіміндегі кез келген сигнал өзгергенде немесе оқиға орын алғанда, әрқашан блок іске қосылады және ішкі процедуралық нұсқаулар орындалады.

**Verilog-те схеманы жобалау**

Жоғарыда айтылғандай, Verilog аппараттық құралды сипаттау тілі сандық схемаларды жобалаудың бірнеше нұсқасын ұсынады. Бұл мүмкіндікті көрсету үшін логикалық схеманы қарастырайық және оның ақиқат кестесі 1-суретте көрсетілген. А қосымшасы Verilog тілінде қолданылатын операторлар кестесін береді.



Сурет 1. Логикалық схема және оның ақиқат кестесі

Деңгейдегі схемалық дизайн:









**Практикалық бөлім**

Тапсырма №1

Берілген схеманы сипаттайтын программа жазу.



EDA playground ортасында бағдарламалау үшін келесі сілтеме арқылы өті қажет: <https://www.edaplayground.com>

EDA playground ортасында бағдарламалау мысалы:



