**Лабораториялық жұмыс №2**

**Комбинациялық және тізбекті схемаларды ПЛИС негізінде жобалау**

**Жұмыстың мақсаты**

ПЛИС құрылғыларында комбинациялық және тізбекті схемаларды жобалау үшін Verilog аппараттық құралды сипаттау тілімен жұмыс істеуде негізгі білім мен дағдыларды алу.

**Теориялық бөлім**

**Комбинациялық және тізбекті схемалар**

Цифрлық схемалар комбинациялық (combinational) және тізбекті (sequential) болып бөлінеді. *Комбинациялық* тізбектердің шығыстары тек кірістердегі ток мәндеріне байланысты; басқаша айтқанда, мұндай схемалар шығыс мәнін есептеу үшін кіріс сигналдарының ағымдағы мәнін біріктіреді. Комбинациялық логикалық схеманың әрекетін шығыс функциялар жиынын пайдаланып анықтауға болады. Мысалы, логикалық элемент комбинациялық схема болып табылады.

Комбинациялық логикалық элементтер көбінесе күрделі жүйелерді жасау үшін пайдаланылатын «құрылыс блоктарына» топтастырылған. Бұл логикалық элементтер деңгейін шамадан тыс егжей-тегжейленуін абстракциялауға және «құрылыс блогының» функциясын атап өтуге мүмкіндік береді. Мультиплексорлар мен дешифраторлар жиі қолданылатын блоктар болып табылады.

Схема комбинациялық болып табылады, егер ол өзара байланысты элементтерден тұрса және келесі шарттар орындалса:

* Схеманың әрбір элементінің өзі комбинациялық болып табылады;
* Схеманың әрбір қосылымы кіріс болып немесе басқа тізбек элементінің бір шығысына қосылған болып табылады;
* Схемада циклдік жолдар жоқ: схемадағы әрбір жол кез келген қосылым арқылы бір реттен артық емес өтеді.

Шығыс мәні кіріс мәндерінен, сондай-ақ сақталған ақпаратқа тәуелді болатын логикалық схема *тізбекті* логикалық схема деп аталады. Сондықтан тізбектелген логикалық схемалар жадыға ие. Тізбекті схемаларды құрудың негізгі шарты кері байланыстың болуы болып табылады.

Аппараттық құралдардың бөлігі ретінде қолданылатын ең көп таралған тізбекті схемалардың арасында триггерлер, сондай-ақ олардың негізінде жүзеге асырылатын регистрлер мен есептегіштер жатады.

**Комбинациялық және тізбекті схемаларға арналған Verilog**

Verilog жүйесіндегі абстракцияның барлық үш деңгейі комбинациялық логикалық схемаларды жобалау үшін пайдаланылуы мүмкін. Ал тізбекті схемаларды жобалау үшін мінез-құлық (Behavioral) абстракция деңгейі оңтайлы болып табылады. Өйткені, мәндерді сақтауға арналған *reg* деректер түрі *always* блокпен бірге, мінез-құлық деңгейінде модельдеуде қолданылады.

Қақпа деңгейіндегі (gate-level) жобалау кезінде комбинациялық схеманың ішкі құрылымы белгілі болуы керек. Бұл оның қандай логикалық элементтердің негізінде жасалғанын білу керек дегенді білдіреді. Бұл абстракция деңгейі шағын комбинациялық схемаларды жобалау үшін қолайлы.

Деректер ағынының деңгейі (data flow) комбинациялық логикалық элементтерді байланыстыратын сигналдарды пайдалануға бағытталған. Жобалау кезінде разрядтық, логикалық және үштік (ternary) операторлар пайдаланылады. Операторлардың алғашқы екі түрін пайдалану түсінікті. Бірақ үштік операторды қолдануда кейбір нюанстар бар. Үштік оператор – үш операндты пайдаланатын оператор. Үштік оператор синтаксисі:



𝑐𝑜𝑛𝑑𝑖𝑡𝑖𝑜𝑛 — логикалық өрнекке арналған орын. Егер жауап дұрыс болса, онда бірінші өрнек орындалады (𝑒𝑥𝑝𝑟𝑒𝑠𝑠𝑖𝑜𝑛1), ал егер ол жалған болса, екінші өрнек сәйкес орындалады (𝑒𝑥𝑝𝑟𝑒𝑠𝑠𝑖𝑜𝑛2). Үштік оператор мысалы:



Егер х 10-нан үлкен болса, онда у 1 болады, ал егер х 10-нан кіші болса, онда у 0-ге орнатылады.

Мінез-құлық деңгейі комбинациялық және тізбекті логикалық схемаларды жобалауға мүмкіндік береді. Мінез-құлық дизайнындағы операциялар *always* блокта шоғырланған. *Always* блоктардың әрқайсысы жеке деректерді өңдеу ағынын қамтамасыз етеді және олардың барлығы параллель жұмыс істейді.

Қарапайым логикалық элементтермен жұмыс істегенде кіріс/шығыс сигналының мәндері бір разрядты болады. Бірақ комбинациялық схемалармен жұмыс істегенде бір разрядты (скаляр) сигналдарды қолдану жеткіліксіз. Мұны шешу үшін вектор (vector) деп аталатын көпразрядты сигналдар (шиналар) қолданылады. Вектор – скалярлық сигналдар жиыны. Мысалы, 5-биттік сигнал 𝑤 берсеңіз, былай жариялауға болады:



Бұл w [4], w[3], w[2], w[1] және w[0] бес сигналдың жиынтығы ретінде анықтайды, олардың әрқайсысы бір бит. Векторды немесе портты жариялағанда, жақшадағы бөлік (жоғарыдағы мысалда 4:0) вектордың элементтері үшін индекс ауқымын көрсетеді. Бірінші мән - сол жақтағы алғашқы элементтің индексі, ал екінші мән - оң жақтағы соңғы элементтің индексі.

Мінез-құлық деңгейінде схемаларды жобалауды құрылғының жұмыс істеу алгоритмін құрастыру ретінде қарастыруға болады. Ал алгоритм құру белгілі бір шарттар негізінде жиі шешім қабылдауды білдіреді. Ал комбинациялық және тізбекті схемаларды құрастырған кезде оның ақиқат кестесіне сүйене отырып, қандай мән шығатыны анықталады. Verilog бағдарламасында шарттарды орындау үшін шартты операторлар if және case пайдаланылады. If операторының синтаксисі:

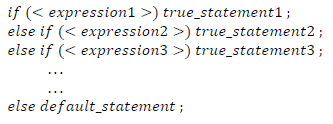
* Бірінші түрі else блогы жоқ шартты оператор. Оператор орындалады немесе орындалмайды



* Екінші түрі else блогы бар шартты оператор. Мәлімдеме ақиқат немесе жалған деп бағаланады.

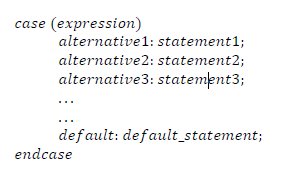


* Шартты оператордың үшінші түрі, кірістірілген if-else-if. Таңдау бірнеше мәлімдемелерден тұрады, бірақ олардың біреуі ғана орындалады



< 𝑒𝑥𝑝𝑟𝑒𝑠𝑠𝑖𝑜𝑛> өрнегі бағаланады, егер ол шын болса (1 немесе нөл емес мән), онда 𝑡𝑟𝑢𝑒\_ 𝑠𝑡𝑎𝑡𝑒𝑚𝑒𝑛𝑡 орындалады. Алайда, егер ол жалған болса (нөл), онда 𝑓𝑎𝑙𝑠𝑒\_𝑠𝑡𝑎𝑡𝑒𝑚𝑒𝑛𝑡 орындалады. 𝑡𝑟𝑢𝑒\_𝑠𝑡𝑎𝑡𝑒𝑚𝑒𝑛𝑡 немесе 𝑓𝑎𝑙𝑠𝑒\_ 𝑠𝑡𝑎𝑡𝑒𝑚𝑒𝑛𝑡 бірнеше оператордан тұратын өрнек болуы мүмкін. Бірнеше мәлімдемелер блогын әдетте *begin* және *end* кілт сөздерін пайдаланып топтастыру керек.

Егер көптеген нұсқалар болса, кірістірілген if-else-ifs қиын болуы мүмкін. Бірдей нәтижеге жетудің ең қысқа жолы - case операторын пайдалану. Case операторының синтаксисі:



𝑠𝑡𝑎𝑡𝑒𝑚𝑒𝑛𝑡 әрқайсысы бір өрнек немесе бірнеше өрнектер блогы болуы мүмкін. Бірнеше мәлімдемелер блогын әдетте *begin* және *end* кілт сөздерін пайдаланып топтастыру керек. *E*𝑥𝑝𝑟𝑒𝑠𝑠𝑖𝑜𝑛 өрнегі олардың жазылу тәртібінде баламалармен салыстырылады. Егер баламалардың ешқайсысы сәйкес келмесе, 𝑑𝑒𝑓𝑎𝑢𝑙𝑡\_𝑠𝑡𝑎𝑡𝑒𝑚𝑒𝑛𝑡 орындалады. *D𝑒*𝑓𝑎𝑢𝑙𝑡\_𝑠𝑡𝑎𝑡𝑒𝑚𝑒𝑛𝑡 міндетті емес.

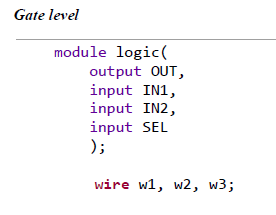
**Verilog-те схеманы жобалау**

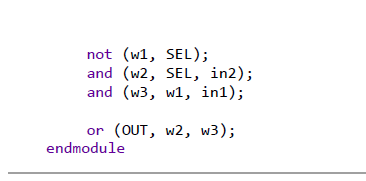
Комбинациялық схеманың мысалы үшін 2х1 мультиплексор сұлбасын және оның ақиқат кестесін алайық (1-сурет).

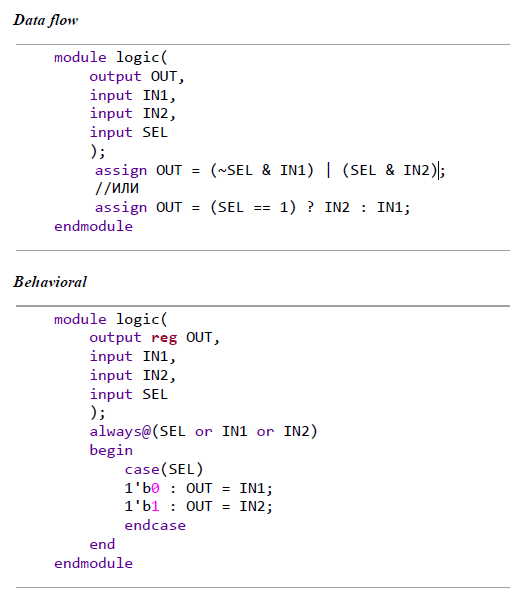


Сурет 1. Мультиплексордың схемасы және оның ақиқат кестесі

Деңгейдегі схемалық дизайн:







**Практикалық бөлім**

Тапсырма №1

1x4 демультиплексор сұлбасының программасын құрыңыз.