## 2.4. ГЕНЕРАТОРИ, БАЗИРАНИ НА ВГРАДЕНИЯ В АЛТЕРНАТИВНАТА КОМАНДА НЕДЕТЕРМИНИЗЪМ

Решава се петата от поставените в т. 1.4 задачи: да се изследва възможността за генериране на случайни последователности чрез вградения в командата за алтернативен избор недетерминизъм.

Формулировка на задачата:

Да се провери адекватността на оператора *select* от езика *XC* на командата за алтернативен избор в частта, свързана с недетерминирания характер на избора между алтернативите.

Операторът *select* от езика *XC* е аналог на алтернативната команда на *CSP*

$P = \{G_1 \rightarrow P_1 \square G_2 \rightarrow P_2 \square \ldots \square G_n \rightarrow P_n\}$

Алтернативната команда е подобна на класическия оператор за избор *if/else*. Но за разлика от него, по дефиниция поддържа недетерминизъм. Недетерминизмът е вътрешно присъщ на всяка една паралелна система, което фактически е отчетено с въвеждането на оператора за алтернативен избор. Недетерминираността се проявява при едновременното сработване на повече от една защита. Тогава по случаен закон се избира една от тях и се преминава към изпълнение на подчинения ѝ процес. На тази важна особеност се набляга, когато алтернативната команда се нарича *команда за недетерминиран избор*.

Ако операторът *select*, като реализация на недетерминираната команда, е напълно адекватен на *CSP* дефиницията на командата, би било възможно това да се използва за изработването на случайни последователности.

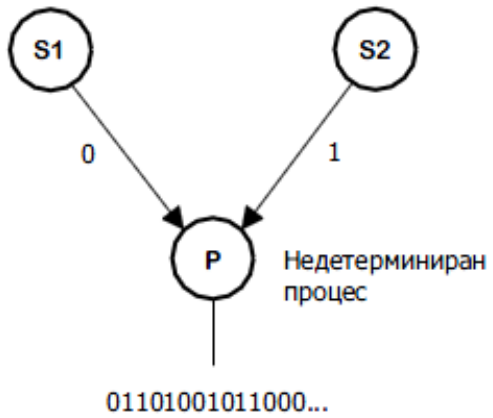
Отделна генераторна функция, за разлика от предните проекти не е необходима. За да се изследва посочената възможност един от паралелните процеси трябва да бъде недетерминиран, т.е. да има повече от един входен канал, които се проверяват за налични данни в рамките на един оператор *select*.

Следователно паралелната система има два допълнителни паралелни процеса, което е отразено в нейното *CSP* уравнение

{*S1* || *S2* || *P* || *Q* || *L*}.

Логическият граф на участъка от системата с вграден недетерминизъм е представен на фиг. 2.10. Процесите *Q* и *L* имат същото предназначение и алгоритъм на работа, както досега.

Процесът *P* обаче е изпълнен като недетерминиран – има два входни канала `chanIn1` и `chanIn2`, проверявани за налични данни в общ оператор *select*. Процесите *S1* и *S2* са свързани по изходния си канал `chanOut` съответно с `chanIn1` и `chanIn2`. Те формират детерминирана последователност от нули (*S1*) и единици (*S2*). Ако операторът *select* е напълно адекватен на *CSP* дефиницията на алтернативната команда, обединеният поток данни ще бъде случаен, което е необходимият ни критерий за адекватност.




**Фиг. 2.10. Граф на участъка от паралелната система с вграден недетерминизъм**

Процесите *S1* и *S2* са идентични и ползват една и съща главна функция `taskS()`

```
void taskS(UINT uintGen, chanend chanOut)
{
    timer timerT;
    int intT;

    while(TRUE)
    {
          chanOut <: (uintGen & 0x1);

          timerT :> intT;
          intT += 100;       // Период на изчакване 1 us
          timerT when timerafter(intT) :> void;
    }
}
```

Функцията има два параметъра. Първият параметър `uintGen` определя стойността на еднобитовия операнд, който трябва да се изпраща по изходния канал `chanOut`, задаван от втория параметър на функцията.

След изпращането на еднобитовото съобщение по канала `chanOut`, процесът *S*, т.е. копията му *S1* и *S2*, се задържа за *1000* ns. За целта, в променливата *intT* към текущата стойност на таймера се добавя константата *100*. Тя отговаря на броя *10* ns интервали на тактуване на таймера.

Участъкът в *P*, който изчаква получаването на данни по входните канали `chanIn1` и `chanIn2`, е изпълнен с оператор `select`

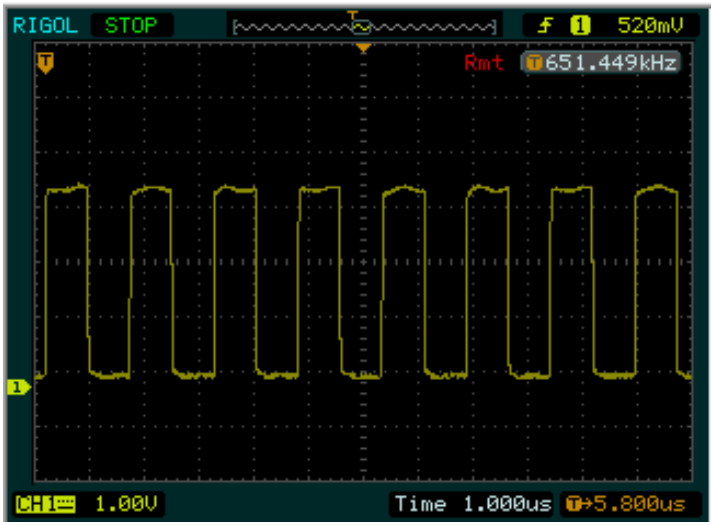
```
select
{
   case chanIn1 :> uintMsg:
   {
         break;
   }
   case chanIn2 :> uintMsg:
   {
         break;
   }
}

chanRight <: uintMsg;
```

Командата за въвеждане `:>` в двата оператора `case` изпълнява ролята на защита. Подобни случаи определят процеса като *недетерминиран*, тъй като двете защити не се изключват взаимно и могат да настъпят едновременно. Съгласно семантиката на алтернативната команда, за разлика от детерминирания избор *if/else*, тук не би трябвало да се дава приоритет на никоя от защитите.

Първоначално беше направен експеримент с нулева задръжка в тялото на *S*. Още при него се получи резултат, противоречащ на семантиката на алтернативната команда – получаваха се само съобщенията, отговарящи на канала от първия оператор `case`. В зависимост от това дали каналът в първия оператор беше `chanIn1` или `chanIn2` се извеждаха съответно нулеви или единични последователности[1]. Оттук и необходимостта от въвеждане на

1 Семантиката на алтернативната команда изключва даването на приоритет на защитите - проверката им би трябвало да се извършва едновременно, а не в реда на разположението им в програмната конструкция.




**Фиг. 2.11. Част от осцилограмата на изходния сигнал на порт *oportRngBit* за изследването на вградения недетерминизъм**

споменатата задръжка между съобщенията[2].

При варианта със задръжка в тялото на *S* съобщенията започват да се редуват, както би трябвало и да бъде. Но това редуване се оказва напълно периодично, т.е. детерминирано (фиг. 2.11).

Полученият резултат показва отсъствието в тази реализация на характерния за алтернативната команда недетерминизъм. Възниква задачата за вграждане в оператора *select* на необходимия недетерминизъм. Предложената техника е предмет на следващата точка - т. 2.5.

2 Въпросът, относно тази задръжка, е засегнат по-подробно в т. 2.5.