2.3. ГЕНЕРАТОРИ, БАЗИРАНИ НА ЕНТРОПИЕН ИЗТОЧНИК ОТ ИЗПЪЛНИТЕЛНАТА СРЕДА

Решава се четвъртата от поставените в т. 1.4 задачи: да се реализира физически генератор на действително-случайни последователности чрез наличния в паралелната изпълнителна среда източник на ентропия.

Формулировка на задачата:

Да се предложи реализация на генератор на действително-случайна последователност (*RRNG*, *TRNG*), използващ кръговите осцилатори на паралелната среда като ентропиен източник. Реализацията да бъде оформена като отделен проект, който включва и разработените до момента генератори.

Паралелната среда *XS1* съдържа няколко източника на ентропия, които са достъпни за изграждането на *RRNG* по представената на фиг. 1.4 обща блокова схема на физически генератор. Това са системните таймери и кръговите осцилатори [45]. Може да се използват и външни източници на случайни събития – *USB PHY* и *Ethernet PHY*[1].

RRNG по схемата на кръговите осцилатори са от особен интерес, защото са интегрирани в архитектурата *XS1* [38, 45]. Тя съдържа четири такива осцилатора, всеки от които тактува собствен брояч. Съдържанието на тези броячи не се изчиства при изключване или системен рестарт. Така се избягва връщането в предварително известна начална стойност и съответно се гарантира висока стойност на ентропията още от самото начало.

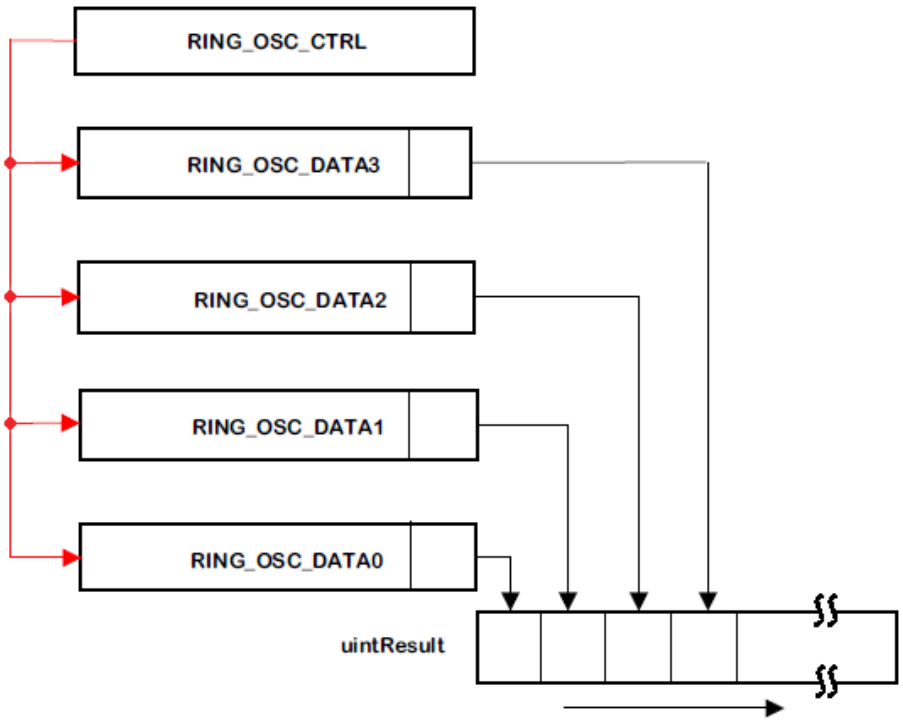
Четирите кръгови осцилатора са независими от системната тактова честота. Периодът на генерирания от тях сигнал е около 2.5 ns и се определя от сумата на времето за превключване на трите инвертора, образуващи осцилатора. Във фирмената документация [38, 45] се посочва, че при пасивен чип *XS1* за интервал от *50* μs броячите на тези осцилатори ще отброят *20000* – *25000* такта. Това се дължи на сумарното влияние на сложен комплекс от

1 *PHY* или *Physical Layer* е част *OSI* модела. Дадено *PHY* устройство служи за връзка между физическия и каналния слой.

физически фактори (температура, захранващо напрежение, влажност, атмосферно налягане) върху времето за превключване на инверторите на кръговите осцилатори. С натоварването на чипа нарастват флуктуациите на температурата и захранващото напрежение, оттам периодът за натрупване на достатъчно ентропия ще бъде по-къс от *50* μs.

В проекта се използва вече изградената и разгледана в предните точки паралелна система със *CSP* уравнение {*P* || *Q* || *L*}. Процесите *P*, *Q* и *L* имат идентично предназначение.

Процесът *P* вика генераторната функция *RNG_OSC()*, разпакетира получената *32* bit стойност и предава бит по бит по изходния си канал към *Q* крайната случайна последователност. Процесът *Q* пакетира обратно в *32* bit думи получените на входния си канал побитови последователности и ги записва в масива на извадката. Процесът *L* изпълнява помощната функция да индицира работата на *P* на светодиодната индикация.



Фиг. 2.8. Схема на генератора, използващ кръговите осцилатори на средата

Предложената генераторна функция `RNG_OSC()` използва представения принцип за формиране на действително-случайна последователност. Съдържа следните променливи

```
timer timerT;
UINT uintT, uintResult;
UINT r0a, r1a, r2a, r3a;
UINT r0, r1, r2, r3;
```

Променливите `timerT` и `uintT` служат за изчакване на работния период `ROSC_PERIOD` от *50* μs. В променливата `uintResult` на изхода се получава пакетираната *32* bit случайна величина. Променливите `r0a`, `r1a`, `r2a` и `r3a` са предназначени за фиксиране на началните, а `r0`, `r1`, `r2` и `r3` – на крайните стойности на четирите брояча.

Последователността на работа е следната:

1. При спрени осцилатори се четат началните им стойности;
2. Осцилаторите се стартират;
3. Изчаква се работния интервал `ROSC_PERIOD` от *50* μs;
4. Осцилаторите се спират;
5. При спрени осцилатори се четат крайните им стойности;
6. Намира се разликата между крайната и началната стойност за всеки отделен брояч и се отделя младшия бит на тази разлика. Така получените *4* bit се групират в тетрада.
7. Стъпки 1-6 се извършват общо *8* пъти за да се формира *32* bit случайна величина.

Посочената последователност на работа на генераторната функция `RNG_OSC()` е представена схематично на фиг. 2.8.

За четенето на броячите се използва функцията `getps()` с прототип

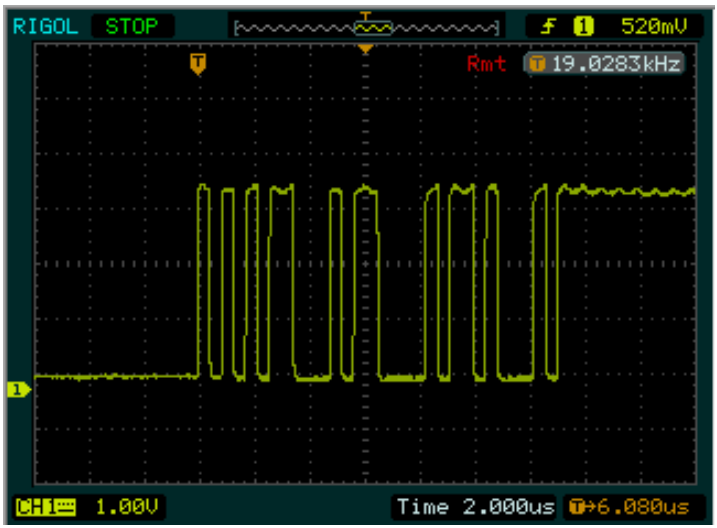
```
unsigned getps(unsigned reg);
```

където параметърът `reg` е идентификатора на регистъра. Тази функция отговаря на машинната инструкция `GETPS` [38, 52]. Двоичният код на идентификаторите на броячите на кръговите осцилатори е съответно *0x070B* (`XS1_L_PS_RING_OSC_DATA0`), *0x080B* (`XS1_L_PS_RING_OSC_DATA1`), *0x090B* (`XS1_L_PS_RING_OSC_DATA2`) и *0x0A0B* (`XS1_L_PS_RING_OSC_DATA3`).

Осцилаторите се стартират и спират чрез запис съответно на 0xF и 0x0 в управляващия им регистър. За този запис се използва функцията `setps()` с прототип

```
void setps(unsigned reg, unsigned value);
```

където параметърът `reg` е идентификатора на регистъра, а `value` – записваната в регистъра стойност. Функцията отговаря на машинната инструкция `SETPS` [38, 52]. Двоичният код на идентификатора на управляващия регистър на броячите на кръговите осцилатори е *0x060B* (`XS1_L_PS_RING_OSC_CTRL`).



Фиг.2.9. Осцилограмата на изходния сигнал на порт *oportRngBit* при *RRNG* в момента на разпакетиране

Времето за генериране на *32* bit пакет от разгледания алгоритъм е около *400* μs. Затова на показания на фиг. 2.9 участък от осцилограмата на изходния сигнал при използваната хоризонтална развивка от *2* μs се вижда само процеса на разпакетиране, извършван от *Q*. Самото разпакетиране продължава около *13* μs.

Осцилограмите от фиг. 2.3 и 2.4 показват побитовото генериране и извеждане, поради побитовия режим на работа на *LFSR* генераторите. При генератора, използващ примитивната функция *crc32()* се изработва цялостен *32* bit пакет, подобно на *RRNG*. Но времето за генерацията на пакета е много по-малко от времето за разпакетиране, поради което фиг. 2.7 има идентична с 2.3 и 2.4 интерпретация.