Secure Processor
True Random Number Generator

Arthur Chung

Outline

- **Target**
- **TRNG Evaluation Platform**

  - Shared Folder function in Oracle VM VirtualBox

- **Q & A**

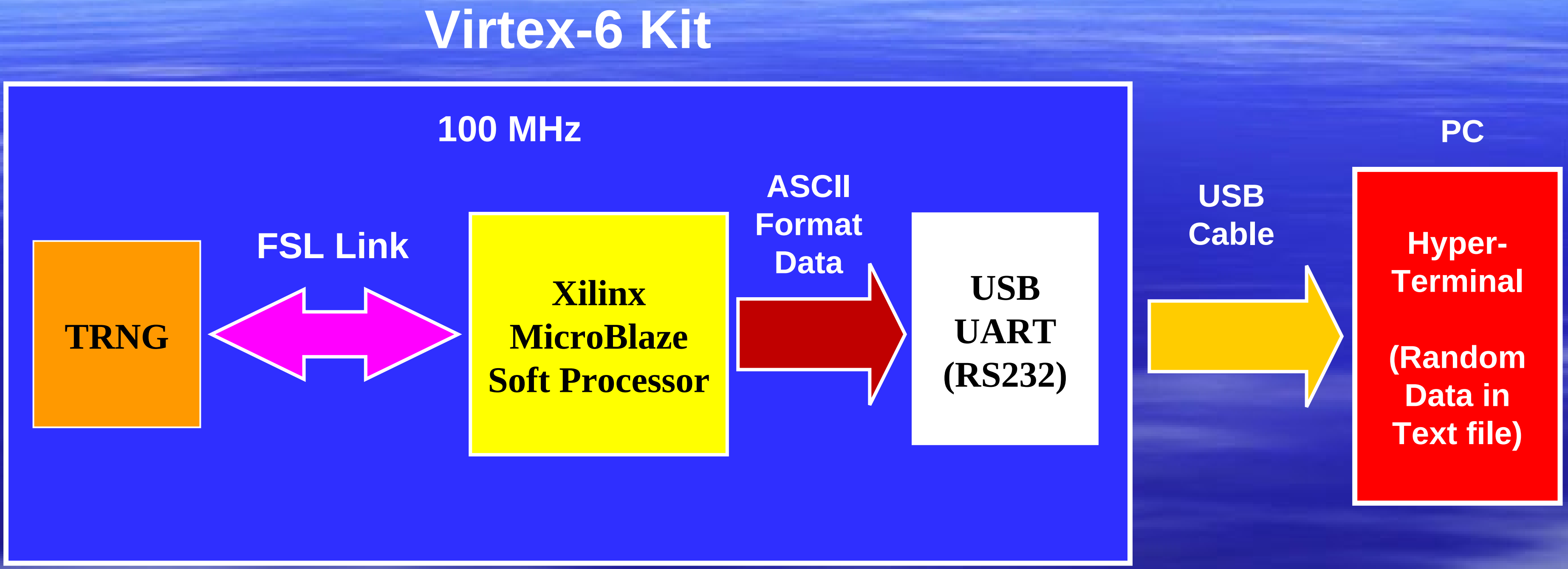
Target

- An FPGA-based TRNG without post-processing

  - it can pass both DIEHARD and NIST Tests

- Study the proposed TRNG
- Figure out its best performance
- Optimize the proposed TRNG to achieve the goal

TRNG Evaluation Platform



- ***Throughput: About 200 MB Random Data per 40 mins***

TRNG Evaluation Platform

- Pass Random Data (RN data) into Linux
  (Gust OS in Oracle VM VirtualBox)

- Perform DIEHARD and NIST Tests

- Reason of using Linux:

  i) DIEHARD test program runs with bugs in Windows but it runs perfectly in Linux

  ii) NIST test program is designed for running in Linux

Shared Folder Function in Oracle VM VirtualBox

- **How to efficiently pass RN data into Linux (Gust OS in VM) ?**

  Ans: Use Share Folder

- **Use the Shared Folder Function provided by VirtualBOX**

- **Drag and Drop the RN data into the shared folder and the Linux in VM can get it at once**

- **Then, do the tests directly !!**

- **Advantages:**

  i) everything can be done internally inside PC (convenient)
  ii) no need to go through share network drive (no network, much safer)
  iii) no need to use any portable device mapping (no portable device)

TRNG Evaluation Platform

- **DIEHARD Test Conditions:**

  - Input RN data: 11.5 MB (binary format)
  - Comply with DIEHARD standard

  (Use the ASCII to Binary converter provided by the test suite for data conversion)

- **NIST Test Conditions:**

  - Input RN data: 260 MB (ASCII format)
  (No need to do any conversion as NIST accepts ASCII format)

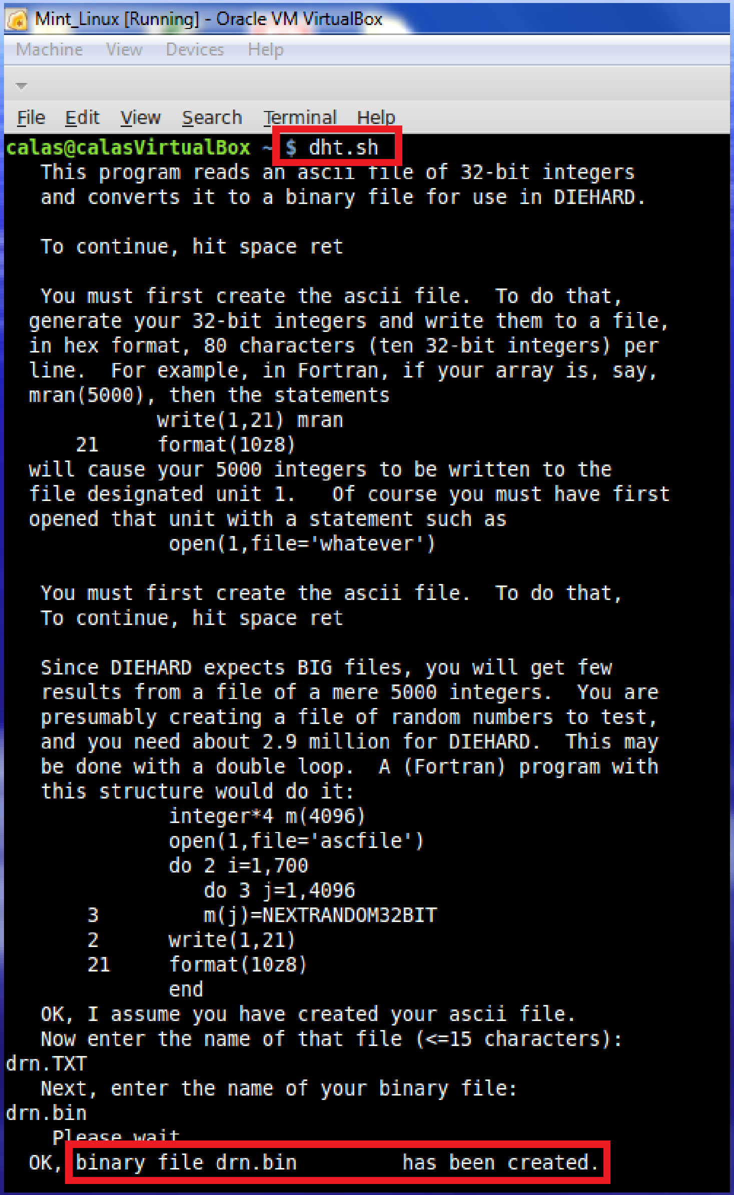
  - Bitstream Length: 2068480

  - No. of bitstreams: 128

  - Comply with NIST standard

TRNG Evaluation Platform

- **DIEHARD Test**


Mint_Linux [Running] - Oracle VM VirtualBox
Machine View Devices Help
File Edit View Search Terminal Help

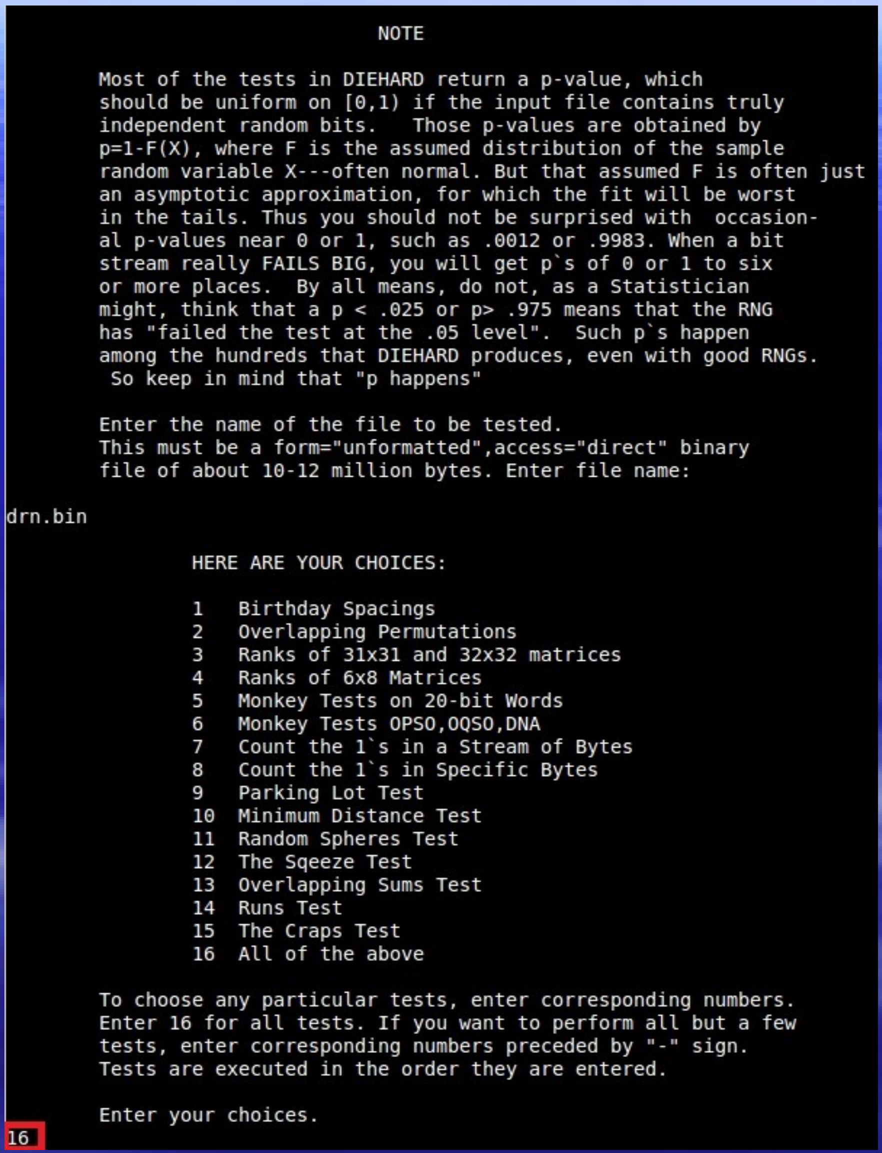
```
calas@calasVirtualBox ~ $ dht.sh
   This program reads an ascii file of 32-bit integers
   and converts it to a binary file for use in DIEHARD.

   To continue, hit space ret

   You must first create the ascii file.  To do that,
  generate your 32-bit integers and write them to a file,
  in hex format, 80 characters (ten 32-bit integers) per
  line.  For example, in Fortran, if your array is, say,
  mran(5000), then the statements
             write(1,21) mran
     21      format(10z8)
  will cause your 5000 integers to be written to the
  file designated unit 1.   Of course you must have first
  opened that unit with a statement such as
              open(1,file='whatever')

   You must first create the ascii file.  To do that,
   To continue, hit space ret

   Since DIEHARD expects BIG files, you will get few
   results from a file of a mere 5000 integers.  You are
   presumably creating a file of random numbers to test,
   and you need about 2.9 million for DIEHARD.  This may
   be done with a double loop.  A (Fortran) program with
   this structure would do it:
              integer*4 m(4096)
              open(1,file='ascfile')
              do 2 i=1,700
                 do 3 j=1,4096
       3         m(j)=NEXTRANDOM32BIT
       2      write(1,21)
       21     format(10z8)
              end
   OK, I assume you have created your ascii file.
   Now enter the name of that file (<=15 characters):
drn.TXT
   Next, enter the name of your binary file:
drn.bin
    Please wait
  OK, binary file drn.bin         has been created.
```



```
                              NOTE

        Most of the tests in DIEHARD return a p-value, which
        should be uniform on [0,1) if the input file contains truly
        independent random bits.   Those p-values are obtained by
        p=1-F(X), where F is the assumed distribution of the sample
        random variable X---often normal. But that assumed F is often just
        an asymptotic approximation, for which the fit will be worst
        in the tails. Thus you should not be surprised with  occasion-
        al p-values near 0 or 1, such as .0012 or .9983. When a bit
        stream really FAILS BIG, you will get p`s of 0 or 1 to six
        or more places.  By all means, do not, as a Statistician
        might, think that a p < .025 or p> .975 means that the RNG
        has "failed the test at the .05 level".  Such p`s happen
        among the hundreds that DIEHARD produces, even with good RNGs.
         So keep in mind that "p happens"

        Enter the name of the file to be tested.
        This must be a form="unformatted",access="direct" binary
        file of about 10-12 million bytes. Enter file name:

drn.bin

                HERE ARE YOUR CHOICES:

                1   Birthday Spacings
                2   Overlapping Permutations
                3   Ranks of 31x31 and 32x32 matrices
                4   Ranks of 6x8 Matrices
                5   Monkey Tests on 20-bit Words
                6   Monkey Tests OPSO,OQSO,DNA
                7   Count the 1`s in a Stream of Bytes
                8   Count the 1`s in Specific Bytes
                9   Parking Lot Test
                10  Minimum Distance Test
                11  Random Spheres Test
                12  The Sqeeze Test
                13  Overlapping Sums Test
                14  Runs Test
                15  The Craps Test
                16  All of the above

        To choose any particular tests, enter corresponding numbers.
        Enter 16 for all tests. If you want to perform all but a few
        tests, enter corresponding numbers preceded by "-" sign.
        Tests are executed in the order they are entered.

        Enter your choices.
16
```

TRNG Evaluation Platform

- **DIEHARD Test (con’t)**

  - after getting the results,
  execute bash script “ca.sh” (written by me)

  - delete the Input RN data and the generated binary file

  - prevent any confusion

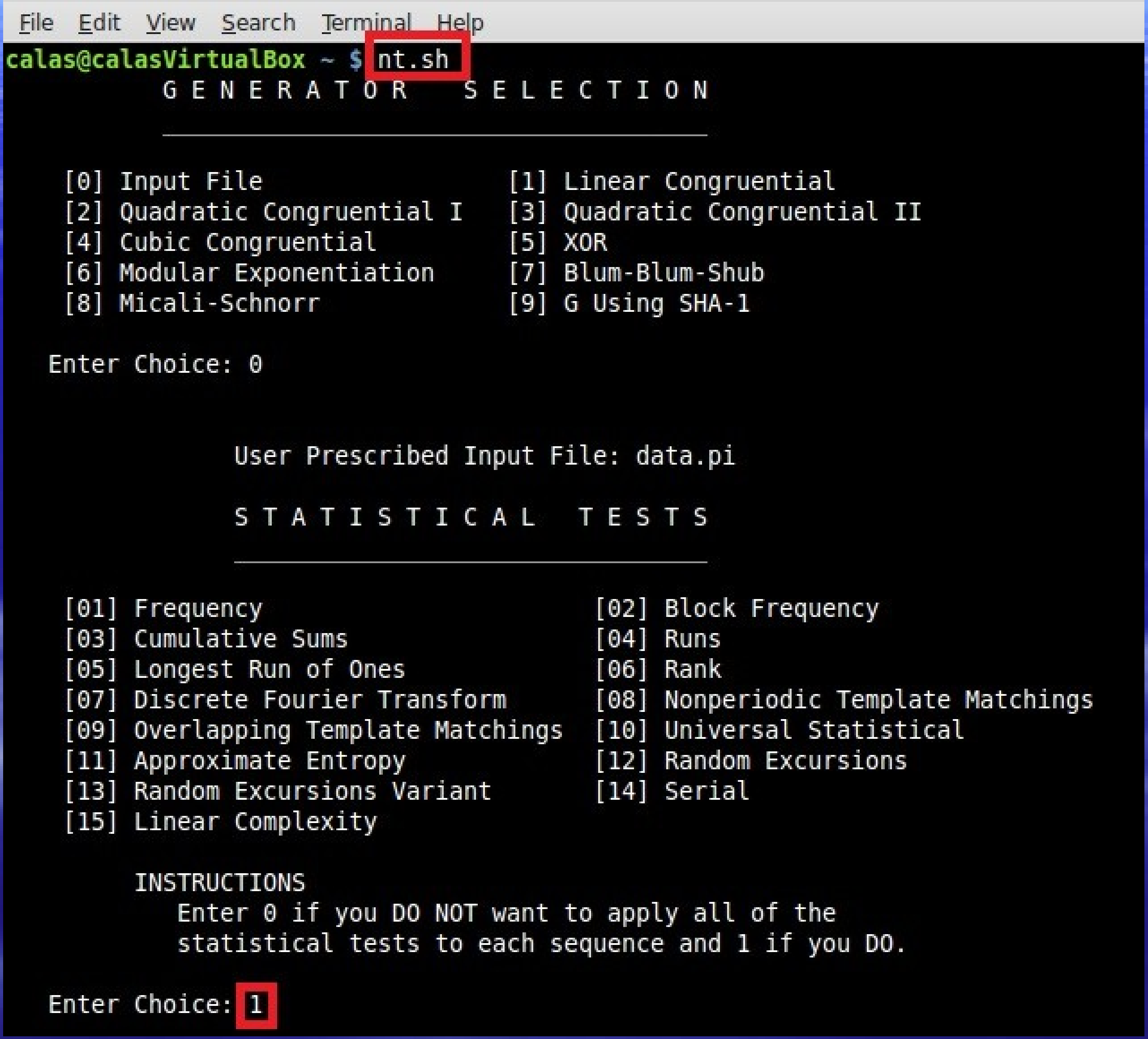
  - reset the Terminal

- **Clear Up everything !!**

- **Start flash in the next test !!**

TRNG Evaluation Platform

- NIST Test



```
calas@calasVirtualBox ~ $ nt.sh
           G E N E R A T O R    S E L E C T I O N
           ______________________________________

    [0] Input File                  [1] Linear Congruential
    [2] Quadratic Congruential I    [3] Quadratic Congruential II
    [4] Cubic Congruential          [5] XOR
    [6] Modular Exponentiation      [7] Blum-Blum-Shub
    [8] Micali-Schnorr              [9] G Using SHA-1

   Enter Choice: 0


                User Prescribed Input File: data.pi

                S T A T I S T I C A L   T E S T S
                _________________________________

    [01] Frequency                       [02] Block Frequency
    [03] Cumulative Sums                 [04] Runs
    [05] Longest Run of Ones             [06] Rank
    [07] Discrete Fourier Transform      [08] Nonperiodic Template Matchings
    [09] Overlapping Template Matchings  [10] Universal Statistical
    [11] Approximate Entropy             [12] Random Excursions
    [13] Random Excursions Variant       [14] Serial
    [15] Linear Complexity

          INSTRUCTIONS
             Enter 0 if you DO NOT want to apply all of the
             statistical tests to each sequence and 1 if you DO.

   Enter Choice: 1
```

Q & A