Written by Matthew Hannon (101754972)

# LSFR HLS- Lab 2

## Introduction

Lab #2 consisted of augmenting the previous lab (pairwise differences feed into histogram module) by adding a linear shift feedback register (LFSR) VHDL module that would provide the index value into the BRAM for the lower PN address (namely the value at that address) which would then be used to compute the pairwise difference as was done in the previous lab. The LFSR module was to be generated using high level synthesis (HLS). HLS is essentially a way to convert C code into HDL automatically. This new LFSR module will then be added to the project and setup via the *Top.vhd* file. Further the results of the now LFSR augmented previous lab project was to be compared with the slightly edited C code to verify functionality of the FPGA design. The software and the hardware-based histogram values were then to be compared as well as their mean and range values. If the lab was successfully completed the values for the histogram and the mean/range should be consistent.

## Understanding and Solving

Well, I started off this lab by trying to generate the HLS version of the provided C code for the LFSR. I first tried to utilize Vivado 2017.4 HLS application for this task. I ran into many errors, and finally conceded to a different approach of using Vivado 2019 HLS to do the task of generating the HLS LFSR module instead. This approach worked, and I was left with a new VHDL file that accurately passed the testbench cases provided by the teacher. I then moved back to Vivado 2017.4 for the remaining development.

I made a complete copy of the previous lab and renamed it to *HLS_LFSR_11Bit*. I then utilized the *rename* command to change the file names of the project and other corresponding files such that I was left with a new Vivado project that I could now start fresh from (at least starting from the finish point of the previous lab – which I did indeed finally get working earlier). The first step was to add our HLS LFSR module to the project and get it incorporated into our *Top.vhd* module. I had to instantiate a new entity for this LSFR module as well as create new signals such as *LFSR_start*, *LFSR_seed*, *LFSR_load_seed*, *LFSR_ap_return*, and the rest of them.

The next major events that needed to occur was how I was going to incorporate this new module into the control and data flow of our existing pairwise difference implementation (from the previous lab). I thought about it and considered a few options. One of the options I considered was adding the control logic to the *Controller.vhd* module. A second option I considered was isolating the management of the control and data flow for the LSFR module to the *Pairwise_Diffs.vhd* module alone (of course utilizing the *Top.vhd* to pass the signals between the modules). A third option I considered was essentially a mixture of the first 2 options where some of the logic (namely the seed initialization of the LFSR module) would be in *Controller.vhd* and the rest of the control and data logic would be resident inside of *Pairwise_Diffs.vhd* module. I choose the second option due to simplicity although I still feel the third option would have been more aligned with a proper architecture/design ethos.

Now that I made the choose to have my pairwise difference module handle the LFSR module I needed to actually do this in a way that made sense and actually worked in implementation/testing. I choose to add two additional cases to my *state_type* enumeration (for lack of a better word) listing called *initialize_lfsr* and *initialize_addrs_and_counters*. The reason I added two more cases rather than one case was because I originally thought that I needed to wait an entire clock cycle to obtain a valid LFSR output value. If that was indeed true, then my plan was to seed the LFSR module in the first added case (*initialize_lfsr*) and then compute the BRAM lower PN address in the second case (*initialize_addrs_and_counters)* because by then in the second case I would have had a valid LFSR initial value after the seeding of the module. I later discovered that I did not need to wait an entire clock cycle to obtain a valid LFSR output value from the HLS LFSR module. Therefore, I could have easily removed one of those extra case statements in my enumeration listing. However, I choose to continue with the original design.

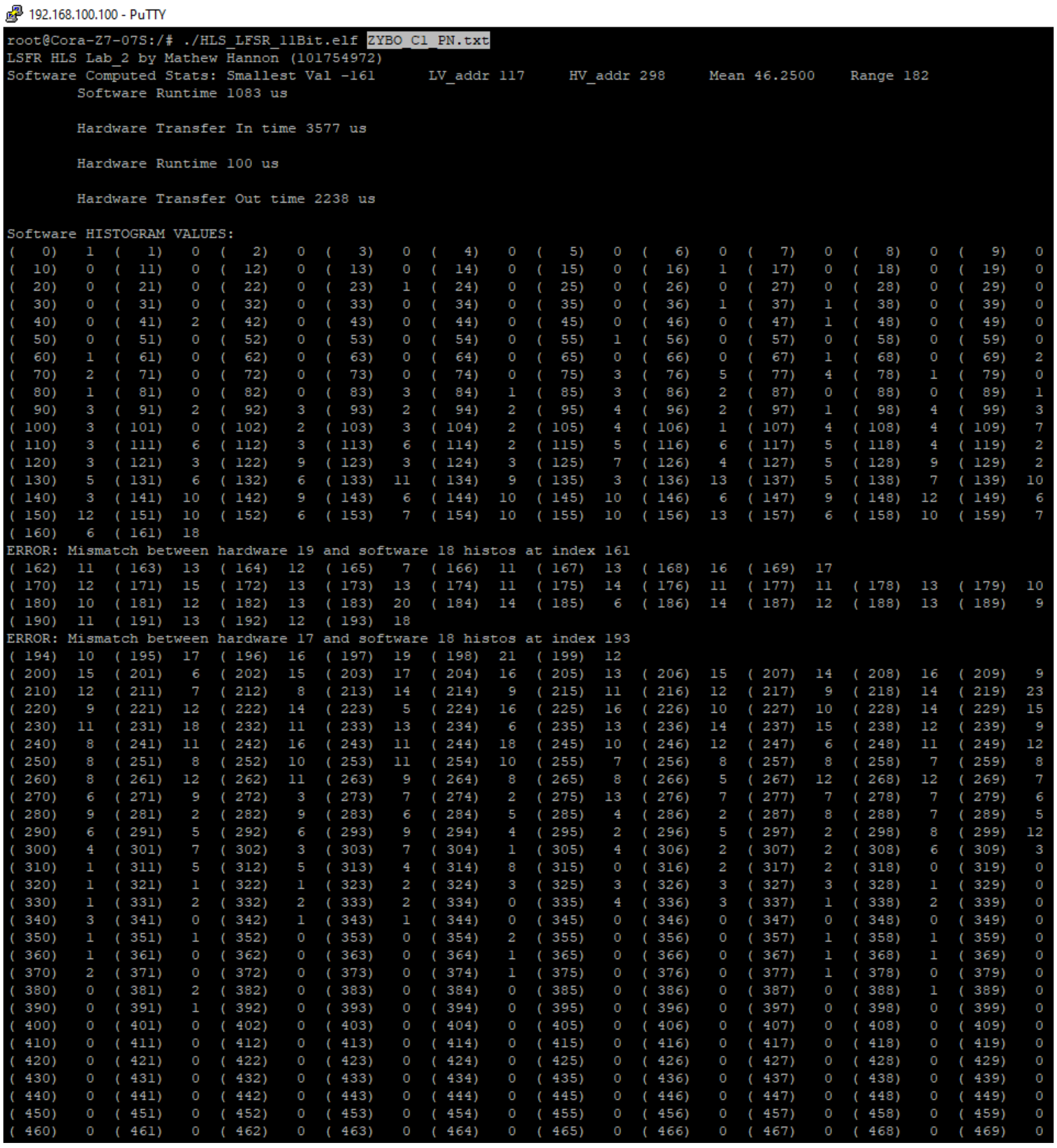
One thing I should note that was extremely important was that I made sure to always initialize the *LFSR_start* output *std_logic* signal line to a value of ‘0’ in my process’s default initialization listing. This is important because I did not want the LFSR module just computing the next valid LFSR values without me specifically telling it when to do that operation. Furthermore, I added implied flipflops to hold the most recent value of the *LSFR_start* signal as well as the most recent *LFSR_ap_return* value. I probably did not need to do this for the start signal because the seed initialization case was only going to be executed (I do not think this is the proper word) once. However, the flipflop/register for the *LFSR_ap_return* most recent value proved useful in how I computed each of the pairwise difference values which were to ultimately be stored in the lowest portion of the BRAM.

I added everything just described to my pairwise difference VHDL module and then generated the bitstream (making sure that there were no critical warnings occurring during the synthesis stage). I then moved onto the exporting of the hardware design (along with the bitstream) and into the SDK portion of this lab. The SDK portion of this lab was rather trivial compared to the modifications done to the HDL side. I simply utilized the teacher’s provided implementation (C code) for the LFSR and set the seed to 0 on the first iteration of the loop only. I then used the values generated by the C implementation of the LFSR to index the lower BRAM address which was to be used for the difference computation. I thought that there may be a problem with overflows (namely LFSR values greater than 2047), but the teacher was nice enough to ensure that this case never occurred. However, if indeed, that did occur I was going to use modulus operations to just simply round robin the index values (loop-around for lack of better wording).

I finally generated my new ELF file using SDK. I programmed the bitstream on the Cora Z7 board and SCP’ed over the newly generated ELF file to the running *PetaLinux* build (which the teacher was nice enough to provide to us earlier). I ran the ELF binary with the *ZYBO_C1_PN.txt* file which contained 4096 short (16-bit) values. I was mostly happy with the results as will be discussed in the following section.

## Results and Interpretation of the Results

The results of the program were surprisingly accurate and consistent (mostly) with the output of the C implementation. Figure 1 shows me running the ELF binary with the provided data file input. Figure 1 also shows the software computed mean and range values as well as the run time execution for the software implementation, the initial BRAM loading (from PS to PL), the hardware runtime to compute the LFSR values and the pairwise differences, and finally the histogram computation. Figure 1 also shows the hardware transfer time outputs back to the Linux side (PL to PS). Figure 2 shows the values generated by the hardware histogram computations. Finally Figure 3 shows the mean and range values computed by the hardware implementation.

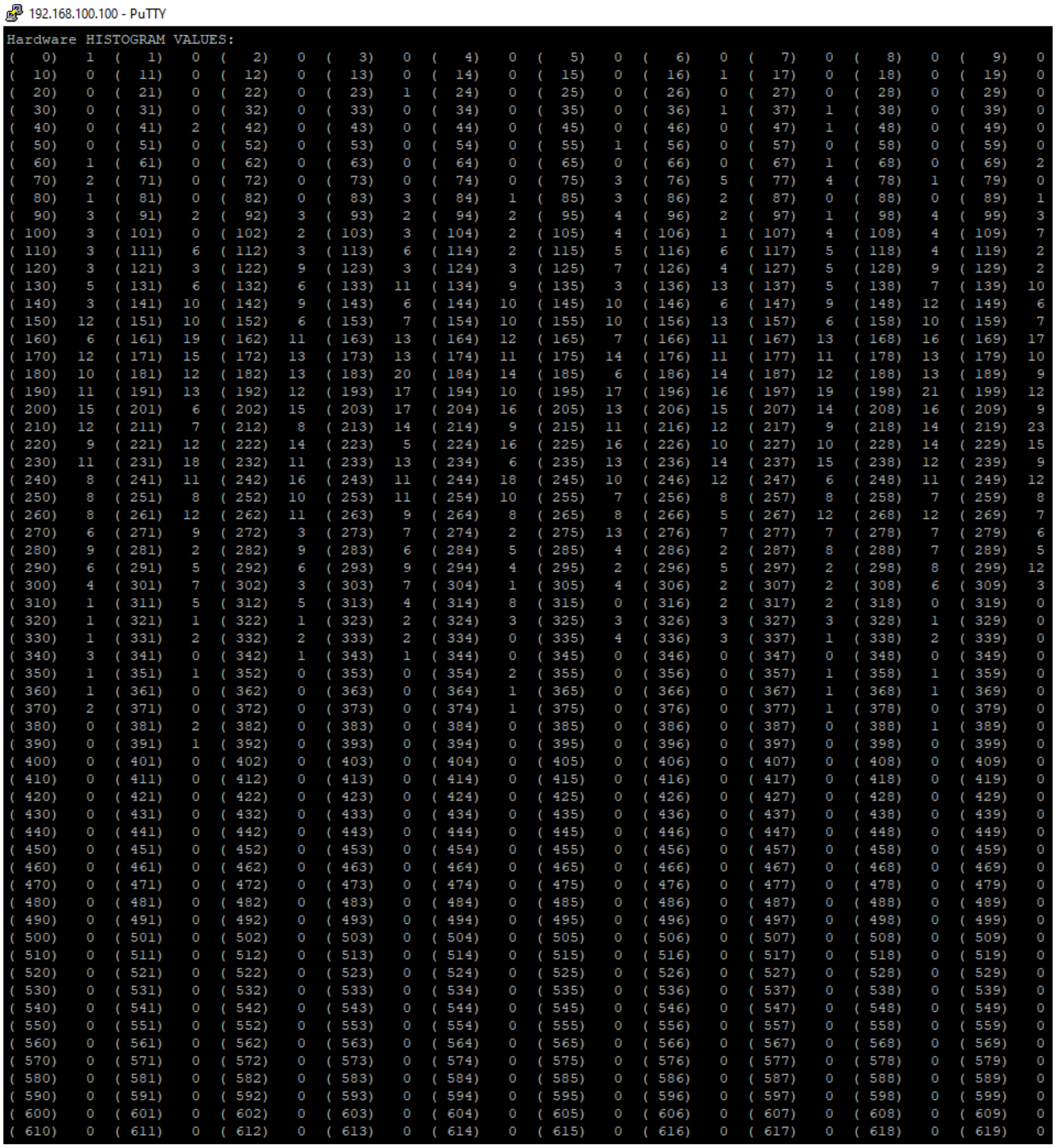


192.168.100.100 - PuTTY

```
root@Cora-Z7-07S:/# ./HLS_LFSR_11Bit.elf ZYBO_C1_PN.txt
LSFR HLS Lab_2 by Mathew Hannon (101754972)
Software Computed Stats: Smallest Val -161      LV_addr 117     HV_addr 298     Mean 46.2500    Range 182
        Software Runtime 1083 us

        Hardware Transfer In time 3577 us

        Hardware Runtime 100 us

        Hardware Transfer Out time 2238 us

Software HISTOGRAM VALUES:
(   0)   1  (   1)   0  (   2)   0  (   3)   0  (   4)   0  (   5)   0  (   6)   0  (   7)   0  (   8)   0  (   9)   0
(  10)   0  (  11)   0  (  12)   0  (  13)   0  (  14)   0  (  15)   0  (  16)   1  (  17)   0  (  18)   0  (  19)   0
(  20)   0  (  21)   0  (  22)   0  (  23)   1  (  24)   0  (  25)   0  (  26)   0  (  27)   0  (  28)   0  (  29)   0
(  30)   0  (  31)   0  (  32)   0  (  33)   0  (  34)   0  (  35)   0  (  36)   1  (  37)   1  (  38)   0  (  39)   0
(  40)   0  (  41)   2  (  42)   0  (  43)   0  (  44)   0  (  45)   0  (  46)   0  (  47)   1  (  48)   0  (  49)   0
(  50)   0  (  51)   0  (  52)   0  (  53)   0  (  54)   0  (  55)   1  (  56)   0  (  57)   0  (  58)   0  (  59)   0
(  60)   1  (  61)   0  (  62)   0  (  63)   0  (  64)   0  (  65)   0  (  66)   0  (  67)   1  (  68)   0  (  69)   2
(  70)   2  (  71)   0  (  72)   0  (  73)   0  (  74)   0  (  75)   3  (  76)   5  (  77)   4  (  78)   1  (  79)   0
(  80)   1  (  81)   0  (  82)   0  (  83)   3  (  84)   1  (  85)   3  (  86)   2  (  87)   0  (  88)   0  (  89)   1
(  90)   3  (  91)   2  (  92)   3  (  93)   2  (  94)   2  (  95)   4  (  96)   2  (  97)   1  (  98)   4  (  99)   3
( 100)   3  ( 101)   0  ( 102)   2  ( 103)   3  ( 104)   2  ( 105)   4  ( 106)   1  ( 107)   4  ( 108)   4  ( 109)   7
( 110)   3  ( 111)   6  ( 112)   3  ( 113)   6  ( 114)   2  ( 115)   5  ( 116)   6  ( 117)   5  ( 118)   4  ( 119)   2
( 120)   3  ( 121)   3  ( 122)   9  ( 123)   3  ( 124)   3  ( 125)   7  ( 126)   4  ( 127)   5  ( 128)   9  ( 129)   2
( 130)   5  ( 131)   6  ( 132)   6  ( 133)  11  ( 134)   9  ( 135)   3  ( 136)  13  ( 137)   5  ( 138)   7  ( 139)  10
( 140)   3  ( 141)  10  ( 142)   9  ( 143)   6  ( 144)  10  ( 145)  10  ( 146)   6  ( 147)   9  ( 148)  12  ( 149)   6
( 150)  12  ( 151)  10  ( 152)   6  ( 153)   7  ( 154)  10  ( 155)  10  ( 156)  13  ( 157)   6  ( 158)  10  ( 159)   7
( 160)   6  ( 161)  18
ERROR: Mismatch between hardware 19 and software 18 histos at index 161
( 162)  11  ( 163)  13  ( 164)  12  ( 165)   7  ( 166)  11  ( 167)  13  ( 168)  16  ( 169)  17
( 170)  12  ( 171)  15  ( 172)  13  ( 173)  13  ( 174)  11  ( 175)  14  ( 176)  11  ( 177)  11  ( 178)  13  ( 179)  10
( 180)  10  ( 181)  12  ( 182)  13  ( 183)  20  ( 184)  14  ( 185)   6  ( 186)  14  ( 187)  12  ( 188)  13  ( 189)   9
( 190)  11  ( 191)  13  ( 192)  12  ( 193)  18
ERROR: Mismatch between hardware 17 and software 18 histos at index 193
( 194)  10  ( 195)  17  ( 196)  16  ( 197)  19  ( 198)  21  ( 199)  12
( 200)  15  ( 201)   6  ( 202)  15  ( 203)  17  ( 204)  16  ( 205)  13  ( 206)  15  ( 207)  14  ( 208)  16  ( 209)   9
( 210)  12  ( 211)   7  ( 212)   8  ( 213)  14  ( 214)   9  ( 215)  11  ( 216)  12  ( 217)   9  ( 218)  14  ( 219)  23
( 220)   9  ( 221)  12  ( 222)  14  ( 223)   5  ( 224)  16  ( 225)  16  ( 226)  10  ( 227)  10  ( 228)  14  ( 229)  15
( 230)  11  ( 231)  18  ( 232)  11  ( 233)  13  ( 234)   6  ( 235)  13  ( 236)  14  ( 237)  15  ( 238)  12  ( 239)   9
( 240)   8  ( 241)  11  ( 242)  16  ( 243)  11  ( 244)  18  ( 245)  10  ( 246)  12  ( 247)   6  ( 248)  11  ( 249)  12
( 250)   8  ( 251)   8  ( 252)  10  ( 253)  11  ( 254)  10  ( 255)   7  ( 256)   8  ( 257)   8  ( 258)   7  ( 259)   8
( 260)   8  ( 261)  12  ( 262)  11  ( 263)   9  ( 264)   8  ( 265)   8  ( 266)   5  ( 267)  12  ( 268)  12  ( 269)   7
( 270)   6  ( 271)   9  ( 272)   3  ( 273)   7  ( 274)   2  ( 275)  13  ( 276)   7  ( 277)   7  ( 278)   7  ( 279)   6
( 280)   9  ( 281)   2  ( 282)   9  ( 283)   6  ( 284)   5  ( 285)   4  ( 286)   2  ( 287)   8  ( 288)   7  ( 289)   5
( 290)   6  ( 291)   5  ( 292)   6  ( 293)   9  ( 294)   4  ( 295)   2  ( 296)   5  ( 297)   2  ( 298)   8  ( 299)  12
( 300)   4  ( 301)   7  ( 302)   3  ( 303)   7  ( 304)   1  ( 305)   4  ( 306)   2  ( 307)   2  ( 308)   6  ( 309)   3
( 310)   1  ( 311)   5  ( 312)   5  ( 313)   4  ( 314)   8  ( 315)   0  ( 316)   2  ( 317)   2  ( 318)   0  ( 319)   0
( 320)   1  ( 321)   1  ( 322)   1  ( 323)   2  ( 324)   3  ( 325)   3  ( 326)   3  ( 327)   3  ( 328)   1  ( 329)   0
( 330)   1  ( 331)   2  ( 332)   2  ( 333)   2  ( 334)   0  ( 335)   4  ( 336)   3  ( 337)   1  ( 338)   2  ( 339)   0
( 340)   3  ( 341)   0  ( 342)   1  ( 343)   1  ( 344)   0  ( 345)   0  ( 346)   0  ( 347)   0  ( 348)   0  ( 349)   0
( 350)   1  ( 351)   1  ( 352)   0  ( 353)   0  ( 354)   2  ( 355)   0  ( 356)   0  ( 357)   1  ( 358)   1  ( 359)   0
( 360)   1  ( 361)   0  ( 362)   0  ( 363)   0  ( 364)   1  ( 365)   0  ( 366)   0  ( 367)   1  ( 368)   1  ( 369)   0
( 370)   2  ( 371)   0  ( 372)   0  ( 373)   0  ( 374)   1  ( 375)   0  ( 376)   0  ( 377)   1  ( 378)   0  ( 379)   0
( 380)   0  ( 381)   2  ( 382)   0  ( 383)   0  ( 384)   0  ( 385)   0  ( 386)   0  ( 387)   0  ( 388)   1  ( 389)   0
( 390)   0  ( 391)   1  ( 392)   0  ( 393)   0  ( 394)   0  ( 395)   0  ( 396)   0  ( 397)   0  ( 398)   0  ( 399)   0
( 400)   0  ( 401)   0  ( 402)   0  ( 403)   0  ( 404)   0  ( 405)   0  ( 406)   0  ( 407)   0  ( 408)   0  ( 409)   0
( 410)   0  ( 411)   0  ( 412)   0  ( 413)   0  ( 414)   0  ( 415)   0  ( 416)   0  ( 417)   0  ( 418)   0  ( 419)   0
( 420)   0  ( 421)   0  ( 422)   0  ( 423)   0  ( 424)   0  ( 425)   0  ( 426)   0  ( 427)   0  ( 428)   0  ( 429)   0
( 430)   0  ( 431)   0  ( 432)   0  ( 433)   0  ( 434)   0  ( 435)   0  ( 436)   0  ( 437)   0  ( 438)   0  ( 439)   0
( 440)   0  ( 441)   0  ( 442)   0  ( 443)   0  ( 444)   0  ( 445)   0  ( 446)   0  ( 447)   0  ( 448)   0  ( 449)   0
( 450)   0  ( 451)   0  ( 452)   0  ( 453)   0  ( 454)   0  ( 455)   0  ( 456)   0  ( 457)   0  ( 458)   0  ( 459)   0
( 460)   0  ( 461)   0  ( 462)   0  ( 463)   0  ( 464)   0  ( 465)   0  ( 466)   0  ( 467)   0  ( 468)   0  ( 469)   0
```

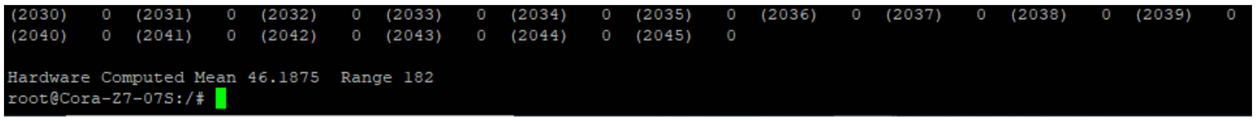
*Figure 1: this figure shows the results of running the modified pairwise difference based project augmented with the LFSR HLS generated module which computed the index values for the lower BRAM addresses. Namely this figure shows the software histogram values.*

Written by Matthew Hannon (101754972)



```
Hardware HISTOGRAM VALUES:
(   0)    1  (   1)    0  (   2)    0  (   3)    0  (   4)    0  (   5)    0  (   6)    0  (   7)    0  (   8)    0  (   9)    0
(  10)    0  (  11)    0  (  12)    0  (  13)    0  (  14)    0  (  15)    0  (  16)    1  (  17)    0  (  18)    0  (  19)    0
(  20)    0  (  21)    0  (  22)    0  (  23)    1  (  24)    0  (  25)    0  (  26)    0  (  27)    0  (  28)    0  (  29)    0
(  30)    0  (  31)    0  (  32)    0  (  33)    0  (  34)    0  (  35)    0  (  36)    1  (  37)    1  (  38)    0  (  39)    0
(  40)    0  (  41)    2  (  42)    0  (  43)    0  (  44)    0  (  45)    0  (  46)    0  (  47)    1  (  48)    0  (  49)    0
(  50)    0  (  51)    0  (  52)    0  (  53)    0  (  54)    0  (  55)    1  (  56)    0  (  57)    0  (  58)    0  (  59)    0
(  60)    1  (  61)    0  (  62)    0  (  63)    0  (  64)    0  (  65)    0  (  66)    0  (  67)    1  (  68)    0  (  69)    2
(  70)    2  (  71)    0  (  72)    0  (  73)    0  (  74)    0  (  75)    3  (  76)    5  (  77)    4  (  78)    1  (  79)    0
(  80)    1  (  81)    0  (  82)    0  (  83)    3  (  84)    1  (  85)    3  (  86)    2  (  87)    0  (  88)    0  (  89)    1
(  90)    3  (  91)    2  (  92)    3  (  93)    2  (  94)    2  (  95)    4  (  96)    2  (  97)    1  (  98)    4  (  99)    3
( 100)    3  ( 101)    0  ( 102)    2  ( 103)    3  ( 104)    2  ( 105)    4  ( 106)    1  ( 107)    4  ( 108)    4  ( 109)    7
( 110)    3  ( 111)    6  ( 112)    3  ( 113)    6  ( 114)    2  ( 115)    5  ( 116)    6  ( 117)    5  ( 118)    4  ( 119)    2
( 120)    3  ( 121)    3  ( 122)    9  ( 123)    3  ( 124)    3  ( 125)    7  ( 126)    4  ( 127)    5  ( 128)    9  ( 129)    2
( 130)    5  ( 131)    6  ( 132)    6  ( 133)   11  ( 134)    9  ( 135)    3  ( 136)   13  ( 137)    5  ( 138)    7  ( 139)   10
( 140)    3  ( 141)   10  ( 142)    9  ( 143)    6  ( 144)   10  ( 145)   10  ( 146)    6  ( 147)    9  ( 148)   12  ( 149)    6
( 150)   12  ( 151)   10  ( 152)    6  ( 153)    7  ( 154)   10  ( 155)   10  ( 156)   13  ( 157)    6  ( 158)   10  ( 159)    7
( 160)    6  ( 161)   19  ( 162)   11  ( 163)   13  ( 164)   12  ( 165)    7  ( 166)   11  ( 167)   13  ( 168)   16  ( 169)   17
( 170)   12  ( 171)   15  ( 172)   13  ( 173)   13  ( 174)   11  ( 175)   14  ( 176)   11  ( 177)   11  ( 178)   13  ( 179)   10
( 180)   10  ( 181)   12  ( 182)   13  ( 183)   20  ( 184)   14  ( 185)    6  ( 186)   14  ( 187)   12  ( 188)   13  ( 189)    9
( 190)   11  ( 191)   13  ( 192)   12  ( 193)   17  ( 194)   10  ( 195)   17  ( 196)   16  ( 197)   19  ( 198)   21  ( 199)   12
( 200)   15  ( 201)    6  ( 202)   15  ( 203)   17  ( 204)   16  ( 205)   13  ( 206)   15  ( 207)   14  ( 208)   16  ( 209)    9
( 210)   12  ( 211)    7  ( 212)    8  ( 213)   14  ( 214)    9  ( 215)   11  ( 216)   12  ( 217)    9  ( 218)   14  ( 219)   23
( 220)    9  ( 221)   12  ( 222)   14  ( 223)    5  ( 224)   16  ( 225)   16  ( 226)   10  ( 227)   10  ( 228)   14  ( 229)   15
( 230)   11  ( 231)   18  ( 232)   11  ( 233)   13  ( 234)    6  ( 235)   13  ( 236)   14  ( 237)   15  ( 238)   12  ( 239)    9
( 240)    8  ( 241)   11  ( 242)   16  ( 243)   11  ( 244)   18  ( 245)   10  ( 246)   12  ( 247)    6  ( 248)   11  ( 249)   12
( 250)    8  ( 251)    8  ( 252)   10  ( 253)   11  ( 254)   10  ( 255)    7  ( 256)    8  ( 257)    8  ( 258)    7  ( 259)    8
( 260)    8  ( 261)   12  ( 262)   11  ( 263)    9  ( 264)    8  ( 265)    8  ( 266)    5  ( 267)   12  ( 268)   12  ( 269)    7
( 270)    6  ( 271)    9  ( 272)    3  ( 273)    7  ( 274)    2  ( 275)   13  ( 276)    7  ( 277)    7  ( 278)    7  ( 279)    6
( 280)    9  ( 281)    2  ( 282)    9  ( 283)    6  ( 284)    5  ( 285)    4  ( 286)    2  ( 287)    8  ( 288)    7  ( 289)    5
( 290)    6  ( 291)    5  ( 292)    6  ( 293)    9  ( 294)    4  ( 295)    2  ( 296)    5  ( 297)    2  ( 298)    8  ( 299)   12
( 300)    4  ( 301)    7  ( 302)    3  ( 303)    7  ( 304)    1  ( 305)    4  ( 306)    2  ( 307)    2  ( 308)    6  ( 309)    3
( 310)    1  ( 311)    5  ( 312)    5  ( 313)    4  ( 314)    8  ( 315)    0  ( 316)    2  ( 317)    2  ( 318)    0  ( 319)    0
( 320)    1  ( 321)    1  ( 322)    1  ( 323)    2  ( 324)    3  ( 325)    3  ( 326)    3  ( 327)    3  ( 328)    1  ( 329)    0
( 330)    1  ( 331)    2  ( 332)    2  ( 333)    2  ( 334)    0  ( 335)    4  ( 336)    3  ( 337)    1  ( 338)    2  ( 339)    0
( 340)    3  ( 341)    0  ( 342)    1  ( 343)    1  ( 344)    0  ( 345)    0  ( 346)    0  ( 347)    0  ( 348)    0  ( 349)    0
( 350)    1  ( 351)    1  ( 352)    0  ( 353)    0  ( 354)    2  ( 355)    0  ( 356)    0  ( 357)    1  ( 358)    1  ( 359)    0
( 360)    1  ( 361)    0  ( 362)    0  ( 363)    0  ( 364)    1  ( 365)    0  ( 366)    0  ( 367)    1  ( 368)    1  ( 369)    0
( 370)    2  ( 371)    0  ( 372)    0  ( 373)    0  ( 374)    1  ( 375)    0  ( 376)    0  ( 377)    1  ( 378)    0  ( 379)    0
( 380)    0  ( 381)    2  ( 382)    0  ( 383)    0  ( 384)    0  ( 385)    0  ( 386)    0  ( 387)    0  ( 388)    1  ( 389)    0
( 390)    0  ( 391)    1  ( 392)    0  ( 393)    0  ( 394)    0  ( 395)    0  ( 396)    0  ( 397)    0  ( 398)    0  ( 399)    0
( 400)    0  ( 401)    0  ( 402)    0  ( 403)    0  ( 404)    0  ( 405)    0  ( 406)    0  ( 407)    0  ( 408)    0  ( 409)    0
( 410)    0  ( 411)    0  ( 412)    0  ( 413)    0  ( 414)    0  ( 415)    0  ( 416)    0  ( 417)    0  ( 418)    0  ( 419)    0
( 420)    0  ( 421)    0  ( 422)    0  ( 423)    0  ( 424)    0  ( 425)    0  ( 426)    0  ( 427)    0  ( 428)    0  ( 429)    0
( 430)    0  ( 431)    0  ( 432)    0  ( 433)    0  ( 434)    0  ( 435)    0  ( 436)    0  ( 437)    0  ( 438)    0  ( 439)    0
( 440)    0  ( 441)    0  ( 442)    0  ( 443)    0  ( 444)    0  ( 445)    0  ( 446)    0  ( 447)    0  ( 448)    0  ( 449)    0
( 450)    0  ( 451)    0  ( 452)    0  ( 453)    0  ( 454)    0  ( 455)    0  ( 456)    0  ( 457)    0  ( 458)    0  ( 459)    0
( 460)    0  ( 461)    0  ( 462)    0  ( 463)    0  ( 464)    0  ( 465)    0  ( 466)    0  ( 467)    0  ( 468)    0  ( 469)    0
( 470)    0  ( 471)    0  ( 472)    0  ( 473)    0  ( 474)    0  ( 475)    0  ( 476)    0  ( 477)    0  ( 478)    0  ( 479)    0
( 480)    0  ( 481)    0  ( 482)    0  ( 483)    0  ( 484)    0  ( 485)    0  ( 486)    0  ( 487)    0  ( 488)    0  ( 489)    0
( 490)    0  ( 491)    0  ( 492)    0  ( 493)    0  ( 494)    0  ( 495)    0  ( 496)    0  ( 497)    0  ( 498)    0  ( 499)    0
( 500)    0  ( 501)    0  ( 502)    0  ( 503)    0  ( 504)    0  ( 505)    0  ( 506)    0  ( 507)    0  ( 508)    0  ( 509)    0
( 510)    0  ( 511)    0  ( 512)    0  ( 513)    0  ( 514)    0  ( 515)    0  ( 516)    0  ( 517)    0  ( 518)    0  ( 519)    0
( 520)    0  ( 521)    0  ( 522)    0  ( 523)    0  ( 524)    0  ( 525)    0  ( 526)    0  ( 527)    0  ( 528)    0  ( 529)    0
( 530)    0  ( 531)    0  ( 532)    0  ( 533)    0  ( 534)    0  ( 535)    0  ( 536)    0  ( 537)    0  ( 538)    0  ( 539)    0
( 540)    0  ( 541)    0  ( 542)    0  ( 543)    0  ( 544)    0  ( 545)    0  ( 546)    0  ( 547)    0  ( 548)    0  ( 549)    0
( 550)    0  ( 551)    0  ( 552)    0  ( 553)    0  ( 554)    0  ( 555)    0  ( 556)    0  ( 557)    0  ( 558)    0  ( 559)    0
( 560)    0  ( 561)    0  ( 562)    0  ( 563)    0  ( 564)    0  ( 565)    0  ( 566)    0  ( 567)    0  ( 568)    0  ( 569)    0
( 570)    0  ( 571)    0  ( 572)    0  ( 573)    0  ( 574)    0  ( 575)    0  ( 576)    0  ( 577)    0  ( 578)    0  ( 579)    0
( 580)    0  ( 581)    0  ( 582)    0  ( 583)    0  ( 584)    0  ( 585)    0  ( 586)    0  ( 587)    0  ( 588)    0  ( 589)    0
( 590)    0  ( 591)    0  ( 592)    0  ( 593)    0  ( 594)    0  ( 595)    0  ( 596)    0  ( 597)    0  ( 598)    0  ( 599)    0
( 600)    0  ( 601)    0  ( 602)    0  ( 603)    0  ( 604)    0  ( 605)    0  ( 606)    0  ( 607)    0  ( 608)    0  ( 609)    0
( 610)    0  ( 611)    0  ( 612)    0  ( 613)    0  ( 614)    0  ( 615)    0  ( 616)    0  ( 617)    0  ( 618)    0  ( 619)    0
```

*Figure 2: this shows the hardware computed histogram values.*



```
(2030)    0  (2031)    0  (2032)    0  (2033)    0  (2034)    0  (2035)    0  (2036)    0  (2037)    0  (2038)    0  (2039)    0
(2040)    0  (2041)    0  (2042)    0  (2043)    0  (2044)    0  (2045)    0

Hardware Computed Mean 46.1875  Range 182
root@Cora-Z7-07S:/#
```

*Figure 3: this shows the hardware computed mean and range values.*

As can be seen from all three figures the computed histogram values as well as the mean and range values were mostly consistent with each other. However, there were two histogram values out of the total computed 2048 values that did not match up. Namely at indexes 161 and 193. The software and hardware implementations computed a histogram values that contained a difference of at most 1 value (for example 19 vs 18 for index 161; 17 vs 18 for index 193). I find it interesting that there was a inconsistency at index 161 when the value 161 was actually the lowest computed value in the computed input data set. I do not think this is a coincidence however I did not spend the time exploring to understand the two histogram inconsistences. I venture that they have to do with the floating-point calculations being done in the software side implementation versus the fixed-point computations being done on the PL side. Furthermore, examining the values for the computed mean and range values showed that the range values for both the C code and the hardware computed the same value of 182. The mean values were slightly off where the software computed a mean of 46.2500 and the hardware computed a mean value of 46.1875. This difference in the mean value I think can be directly tied to the differences in the two histogram inconsistencies just mentioned. Overall, both implementations of the algorithm in software and in hardware computed 2046 out of 2048 histogram values correctly. I feel that this is not perfect (obviously), but also feel the results are relatively not horrible. Horrible would have been no matching values between the software and hardware implementations.

A final thing I wanted to note here is that the mean value is strikingly close to the mean value computed in the previous lab (pairwise difference lab). I would have suspected that choosing the lower PN value to be used for subtraction from the output of the LFSR would have made a noticeable alteration to the final computed histogram bin values. I suspect two things: either I am doing something fundamentally wrong in both my software and hardware implementations (which is leading to this seemingly mostly consistent output) or that the teacher purposely selected the 4096 short (16-bit) values in the input data set in such a way that the final histogram mean values were consistent with simply indexing incrementally the values for the lower BRAM addresses. I hope the latter case is the truth…

## Learning Experience and Conclusion

I learned a great deal from this lab. I learned about how easy it is to use HLS to simply write some C code and have it automatically converted into HDL (while maintaining the accuracy of the testbench data to verify validity of the synthesized HDL). I learned how to add in yet another VHDL module into our project design. I further learned that in the case of the LFSR HLS module that one does not need to wait an entire clock cycle to obtain the computed LFSR output value. Rather I learned I could simply assert a signal line and on the next sequential line of VHDL I can already have the output of the LFSR computation. I wonder since it is the next line in the sequential VHDL process block whether that actually corresponds to a clock cycle… That is an interesting question I wish I understood the answer to. It is sequential and not concurrent, but the LFSR module is concurrently (always) computing a LFSR output value I believe. But it must take some amount of time for the *LFSR_start* signal to be asserted before that behavioral signal change can be reflected in the output of the LFSR value. I wish I understood this better.

Written by Matthew Hannon (101754972)

In conclusion of lab #2 I overall am happy with the results which showed a software and hardware implementation which computed 2046/2048 histogram values correctly corresponding to a 99.9023% consistency between the two implementations. I understand that a very high consistency does not imply a high level of accuracy. However, I have gone back and analyzed my HDL changes as well as my C code changes, and they appear sensible and aligned with what I think the proper and correct computation should look like. Therefore, I am optimistic that my consistency levels between my hardware and software implementations also are aligned with a high accuracy level. If I am wrong, then I would be very interested and curious to discover where I made my mistake/mistakes and how I could rectify and learn from them.