Alex Allahar
928009686
Dec 1st, 2024

Hardware Fuzzing Report

This assignment is about using a hardware fuzzer that's not fully finished, to find weaknesses in hardware designs. TheHuzz fuzzer is focused on testing only the Cva6 processor. For this assignment, we must act like a verification engineer, and complete the unfinished parts of TheHuzz, and use it to test the Cva6 processor. By running the completed code, we can see how well the fuzzer works based on coverage after each of the four function correction.

```
------------------------------------------------------------
  Benchmark              : cva6
  Run time               : 84.47 sec
  No. of testcases       : 20
  No. of coverage points : {'line': 6183, 'branch': 11147, 'cond': 44943, 'fsm': 199, 'tgl': 335978, 'Total': 398450}
  No. of points covered  : 0
  % coverage achieved    : 0%
------------------------------------------------------------

[84.48 sec] Running TheHuzz on given benchmark, cva6 done
[alex.allahar@n01-zeus thehuzz]$
```

Figure 1: Cva6 Benchmark (Setup)

Once the setup of TheHuzz was completed an initial test over 20 test cases was run, to verify that current code has no points covered and 0% coverage achieved. To fix this the four function we need to fix are:

F1: The coverage calculation function in the feedback engine.
F2: The function to get test cases from the input database.
F3: The test case selector function in the feedback engine.
F4: The random function in the mutation engine (bonus points)

Before modifying any code the following benchmark is shown, this will be the reference when modifying the function below to determine if we have correctly fixed the function at hand.

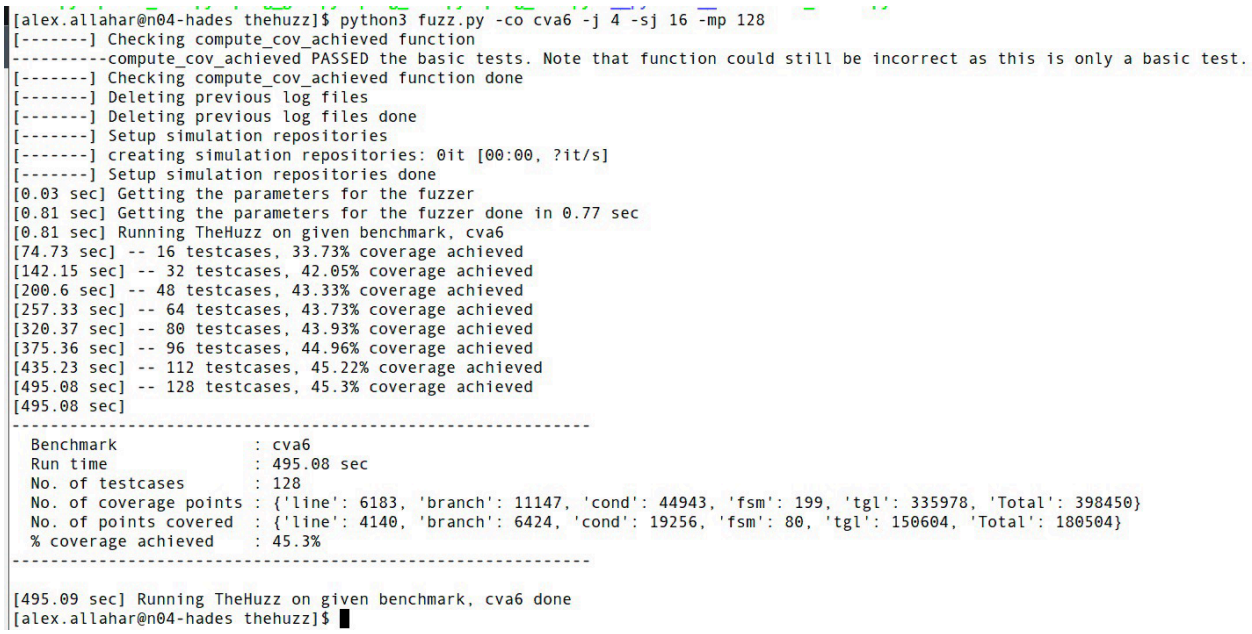

Figure 2: Cva6 Benchmark (Initial)

F1: compute_cov_achieved function

This function takes two inputs, *merged_cov_dict,* this dictionary contains each key that represents a different coverage type and is represented by a binary string to determine if a coverage point was achieved (1) or not achieved (0). The input tot_cov_points is the total number of coverage points. The function returns the number of coverage points achieved and the percentage of coverage points achieved. After making the two edits outlined in the code, the fuzzer was run again for 128 testcase to determine if the total number of coverage points and % of coverage points achieved was displayed. The first edit counts the number of 1s in the dictionary for each coverage type. It is then summed up to determine the total achieved coverage points. The updated function worked correctly, and the following cva6 benchmark was printed:

```
[alex.allahar@n04-hades thehuzz]$ python3 fuzz.py -co cva6 -j 4 -sj 16 -mp 128
[-------] Checking compute_cov_achieved function
----------compute_cov_achieved PASSED the basic tests. Note that function could still be incorrect as this is only a basic test.
[-------] Checking compute_cov_achieved function done
[-------] Deleting previous log files
[-------] Deleting previous log files done
[-------] Setup simulation repositories
[-------] creating simulation repositories: 0it [00:00, ?it/s]
[-------] Setup simulation repositories done
[0.05 sec] Getting the parameters for the fuzzer
[0.27 sec] Getting the parameters for the fuzzer done in 0.22 sec
[0.27 sec] Running TheHuzz on given benchmark, cva6
[75.14 sec] -- 16 testcases, 33.59% coverage achieved
[137.52 sec] -- 32 testcases, 34.9% coverage achieved
[195.05 sec] -- 48 testcases, 41.42% coverage achieved
[253.75 sec] -- 64 testcases, 42.02% coverage achieved
[320.69 sec] -- 80 testcases, 44.41% coverage achieved
[375.31 sec] -- 96 testcases, 44.63% coverage achieved
[422.72 sec] -- 112 testcases, 44.75% coverage achieved
[484.09 sec] -- 128 testcases, 45.31% coverage achieved
[484.09 sec]
------------------------------------------------------------
  Benchmark              : cva6
  Run time               : 484.09 sec
  No. of testcases       : 128
  No. of coverage points : {'line': 6183, 'branch': 11147, 'cond': 44943, 'fsm': 199, 'tgl': 335978, 'Total': 398450}
  No. of points covered  : {'line': 4152, 'branch': 6481, 'cond': 19289, 'fsm': 80, 'tgl': 150555, 'Total': 180557}
  % coverage achieved    : 45.31%
------------------------------------------------------------

[484.14 sec] Running TheHuzz on given benchmark, cva6 done
[alex.allahar@n04-hades thehuzz]$
```

Figure 3: Cva6 Benchmark (After F1 Edits)

F2: get_testcases_to_sim function

This function is a part of the Database class, and gets the total number of test cases available in the database. If the total number of test cases requested is greater than the number of test cases available the function should exit and provide an error message. The second edit to complete his function is to ensure that the first test cases from the self.new_testcases array are assigned to the self-testcases_to_sim. After applying these edits to the function the fuzzer was run again and worked correctly. The following benchmark was given:

```
[alex.allahar@n04-hades thehuzz]$ python3 fuzz.py -co cva6 -j 4 -sj 16 -mp 128
[-------] Checking compute_cov_achieved function
----------compute_cov_achieved PASSED the basic tests. Note that function could still be incorrect as this is only a basic test.
[-------] Checking compute_cov_achieved function done
[-------] Deleting previous log files
[-------] Deleting previous log files done
[-------] Setup simulation repositories
[-------] creating simulation repositories: 0it [00:00, ?it/s]
[-------] Setup simulation repositories done
[0.01 sec] Getting the parameters for the fuzzer
[0.15 sec] Getting the parameters for the fuzzer done in 0.14 sec
[0.15 sec] Running TheHuzz on given benchmark, cva6
[71.02 sec] -- 16 testcases, 35.13% coverage achieved
[126.93 sec] -- 32 testcases, 42.59% coverage achieved
[189.7 sec] -- 48 testcases, 43.55% coverage achieved
[247.79 sec] -- 64 testcases, 44.3% coverage achieved
[310.39 sec] -- 80 testcases, 44.53% coverage achieved
[372.93 sec] -- 96 testcases, 45.01% coverage achieved
[428.58 sec] -- 112 testcases, 45.17% coverage achieved
[483.09 sec] -- 128 testcases, 45.33% coverage achieved
[483.09 sec]
------------------------------------------------------------
  Benchmark              : cva6
  Run time               : 483.09 sec
  No. of testcases       : 128
  No. of coverage points : {'line': 6183, 'branch': 11147, 'cond': 44943, 'fsm': 199, 'tgl': 335978, 'Total': 398450}
  No. of points covered  : {'line': 4169, 'branch': 6451, 'cond': 19264, 'fsm': 81, 'tgl': 150665, 'Total': 180630}
  % coverage achieved    : 45.33%
------------------------------------------------------------

[483.1 sec] Running TheHuzz on given benchmark, cva6 done
[alex.allahar@n04-hades thehuzz]$
```

Figure 4: Cva6 Benchmark (After F2 Edits)

F3: calc_no_times_to_mut function

This function is a part of the fuzz.py file. The function aims to determine how many times to mutate a given test case and returns two values: the mutation count and the type of mutation. This helper function is in the feedback engine. The first edit for this function is to the num_times_to_mut_local variable. It reduces its value based on the decr_factor and then ensures the value is an integer for mutation. The second edit ensures that the mutation variable is at least 1. The third edit for the function implements randomness to the mutation for new inputs. The local variable is changed into a new random int within a reasonable range to prevent excessive mutation. After applying this edit the code ran successfully.

```
[alex.allahar@n04-hades thehuzz]$ python3 fuzz.py -co cva6 -j 4 -sj 16 -mp 128
[-------] Checking compute_cov_achieved function
----------compute_cov_achieved PASSED the basic tests. Note that function could still be incorrect as this is only a basic test.
[-------] Checking compute_cov_achieved function done
[-------] Deleting previous log files
[-------] Deleting previous log files done
[-------] Setup simulation repositories
[-------] creating simulation repositories: 0it [00:00, ?it/s]
[-------] Setup simulation repositories done
[0.0 sec] Getting the parameters for the fuzzer
[0.15 sec] Getting the parameters for the fuzzer done in 0.14 sec
[0.15 sec] Running TheHuzz on given benchmark, cva6
[64.41 sec] -- 16 testcases, 33.89% coverage achieved
[123.01 sec] -- 32 testcases, 37.58% coverage achieved
[183.2 sec] -- 48 testcases, 38.51% coverage achieved
[238.65 sec] -- 64 testcases, 43.11% coverage achieved
[297.57 sec] -- 80 testcases, 44.44% coverage achieved
[357.39 sec] -- 96 testcases, 44.92% coverage achieved
[412.91 sec] -- 112 testcases, 45.52% coverage achieved
[466.09 sec] -- 128 testcases, 45.79% coverage achieved
[466.09 sec]
------------------------------------------------------------
  Benchmark              : cva6
  Run time               : 466.09 sec
  No. of testcases       : 128
  No. of coverage points : {'line': 6183, 'branch': 11147, 'cond': 44943, 'fsm': 199, 'tgl': 335978, 'Total': 398450}
  No. of points covered  : {'line': 4184, 'branch': 6527, 'cond': 19743, 'fsm': 86, 'tgl': 151926, 'Total': 182466}
  % coverage achieved    : 45.79%
------------------------------------------------------------

[466.1 sec] Running TheHuzz on given benchmark, cva6 done
[alex.allahar@n04-hades thehuzz]$
```

Figure 5: Cva6 Benchmark (After F3 edits)

F4: my_random function

This function located in prog_mut.py, is responsible for performing random mutations on the instructions. This random mutation is based on specified indices and bit-flipping logic. The edit for this function is to convert the binary string instruction mut into a list to easier iterate with the m_index variable. Looping through m_index and flipping a bit randomly of the mut instruction list. The mut list is then joined back together and returned. After implementing this edit the code ran successfully and the benchmark from this edit is shown below:

```
[alex.allahar@n04-hades thehuzz]$ python3 fuzz.py -co cva6 -j 4 -sj 16 -mp 128
[-------] Checking compute_cov_achieved function
----------compute_cov_achieved PASSED the basic tests. Note that function could still be incorrect as this is only a basic test.
[-------] Checking compute_cov_achieved function done
[-------] Deleting previous log files
[-------] Deleting previous log files done
[-------] Setup simulation repositories
[-------] creating simulation repositories: 0it [00:00, ?it/s]
[-------] Setup simulation repositories done
[0.03 sec] Getting the parameters for the fuzzer
[0.21 sec] Getting the parameters for the fuzzer done in 0.18 sec
[0.21 sec] Running TheHuzz on given benchmark, cva6
[74.63 sec] -- 16 testcases, 40.25% coverage achieved
[135.43 sec] -- 32 testcases, 41.73% coverage achieved
[195.38 sec] -- 48 testcases, 43.28% coverage achieved
[252.66 sec] -- 64 testcases, 43.77% coverage achieved
[314.16 sec] -- 80 testcases, 44.0% coverage achieved
[368.86 sec] -- 96 testcases, 44.09% coverage achieved
[422.88 sec] -- 112 testcases, 44.2% coverage achieved
[482.82 sec] -- 128 testcases, 44.33% coverage achieved
[482.82 sec]
------------------------------------------------------------
  Benchmark              : cva6
  Run time               : 482.82 sec
  No. of testcases       : 128
  No. of coverage points : {'line': 6183, 'branch': 11147, 'cond': 44943, 'fsm': 199, 'tgl': 335978, 'Total': 398450}
  No. of points covered  : {'line': 4149, 'branch': 6386, 'cond': 19132, 'fsm': 80, 'tgl': 146903, 'Total': 176650}
  % coverage achieved    : 44.33%
------------------------------------------------------------

[482.85 sec] Running TheHuzz on given benchmark, cva6 done
[alex.allahar@n04-hades thehuzz]$
```

Figure 6: Cva6 Benchmark (After F4 edit)

After implementation of all 4 function fixes, a final test over 512 test cases was run. This code ran successfully and the benchmark produced is shown below:

```
[alex.allahar@n04-hades thehuzz]$ python3 fuzz.py -co cva6 -j 4 -sj 16 -mp 512
[-------] Checking compute_cov_achieved function
----------compute_cov_achieved PASSED the basic tests. Note that function could still be incorrect as this is only a basic test.
[-------] Checking compute_cov_achieved function done
[-------] Deleting previous log files
[-------] Deleting previous log files done
[-------] Setup simulation repositories
[-------] creating simulation repositories: 0it [00:00, ?it/s]
[-------] Setup simulation repositories done
[0.01 sec] Getting the parameters for the fuzzer
[0.16 sec] Getting the parameters for the fuzzer done in 0.15 sec
[0.16 sec] Running TheHuzz on given benchmark, cva6
[70.14 sec] -- 16 testcases, 34.26% coverage achieved
[126.88 sec] -- 32 testcases, 40.51% coverage achieved
[189.21 sec] -- 48 testcases, 43.0% coverage achieved
[250.58 sec] -- 64 testcases, 43.82% coverage achieved
[319.56 sec] -- 80 testcases, 45.45% coverage achieved
[382.64 sec] -- 96 testcases, 45.53% coverage achieved
[447.11 sec] -- 112 testcases, 45.65% coverage achieved
[514.03 sec] -- 128 testcases, 47.05% coverage achieved
[578.84 sec] -- 144 testcases, 47.18% coverage achieved
[638.73 sec] -- 160 testcases, 47.41% coverage achieved
[704.14 sec] -- 176 testcases, 47.57% coverage achieved
[760.28 sec] -- 192 testcases, 47.58% coverage achieved
[819.28 sec] -- 208 testcases, 47.61% coverage achieved
[874.79 sec] -- 224 testcases, 47.68% coverage achieved
[931.42 sec] -- 240 testcases, 47.72% coverage achieved
[988.72 sec] -- 256 testcases, 47.73% coverage achieved
[1043.65 sec] -- 272 testcases, 47.85% coverage achieved
[1103.06 sec] -- 288 testcases, 47.95% coverage achieved
[1164.22 sec] -- 304 testcases, 48.09% coverage achieved
```

```
[1225.16 sec] -- 320 testcases, 48.11% coverage achieved
[1285.31 sec] -- 336 testcases, 48.14% coverage achieved
[1341.45 sec] -- 352 testcases, 48.17% coverage achieved
[1395.14 sec] -- 368 testcases, 48.22% coverage achieved
[1447.31 sec] -- 384 testcases, 48.24% coverage achieved
[1506.94 sec] -- 400 testcases, 48.32% coverage achieved
[1562.55 sec] -- 416 testcases, 48.34% coverage achieved
[1618.15 sec] -- 432 testcases, 48.37% coverage achieved
[1673.76 sec] -- 448 testcases, 48.4% coverage achieved
[1727.02 sec] -- 464 testcases, 48.42% coverage achieved
[1782.09 sec] -- 480 testcases, 48.43% coverage achieved
[1839.57 sec] -- 496 testcases, 48.45% coverage achieved
[1891.74 sec] -- 512 testcases, 48.45% coverage achieved
[1891.74 sec]
------------------------------------------------------------
  Benchmark              : cva6
  Run time               : 1891.74 sec
  No. of testcases       : 512
  No. of coverage points : {'line': 6183, 'branch': 11147, 'cond': 44943, 'fsm': 199, 'tgl': 335978, 'Total': 398450}
  No. of points covered  : {'line': 4253, 'branch': 6706, 'cond': 20686, 'fsm': 84, 'tgl': 161327, 'Total': 193056}
  % coverage achieved    : 48.45%
------------------------------------------------------------

[1891.76 sec] Running TheHuzz on given benchmark, cva6 done
[alex.allahar@n04-hades thehuzz]$
```

Figure 7: Cva6 Benchmark (All Edits, 512 test cases)

Feedback

This assignment was a great way to introduce theHuzz fuzzer. It provides an interactive demo-like walkthrough through the various functions and components of the hardware fuzzer. Having worked on the fuzzer this semester already due to my team selection project 6, this assignment was able to provide new insight into the fuzzer that I previously had not noticed. To improve for upcoming semesters, I would introduce this assignment soon in the year to help students taking theHuzz team project get more experience with theHuzz before diving into creating a new implementation.