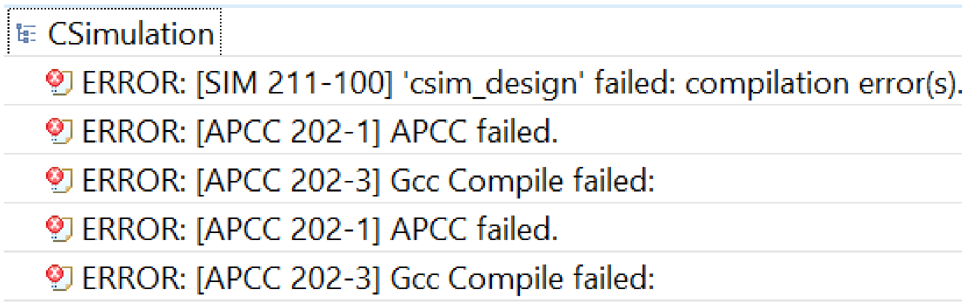
VLSI
MIDSEM PROJECT

Kasukabe Defence Group

Team Members :

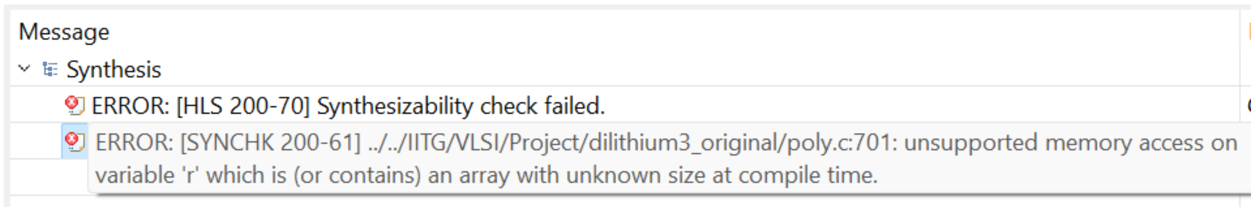
Name	Roll No.
Arpan Khandare	200101018
Naman Anand	200101070
Bhoomiraj Patel	200101075
Souradeep Kar	200101096
Sujeet Kamble	200101097

- The errors we got while executing the original file:
  Errors in simulation->



```
CSimulation
ERROR: [SIM 211-100] 'csim_design' failed: compilation error(s).
ERROR: [APCC 202-1] APCC failed.
ERROR: [APCC 202-3] Gcc Compile failed:
ERROR: [APCC 202-1] APCC failed.
ERROR: [APCC 202-3] Gcc Compile failed:
```

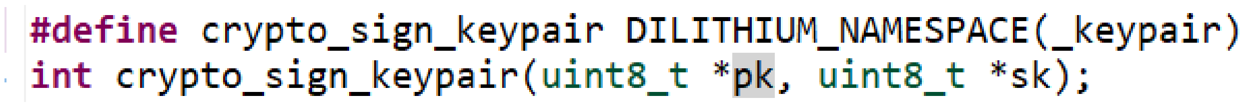
Errors in synthesis->



```
Message
Synthesis
ERROR: [HLS 200-70] Synthesizability check failed.
ERROR: [SYNCHK 200-61] ../../IITG/VLSI/Project/dilithium3_original/poly.c:701: unsupported memory access on variable 'r' which is (or contains) an array with unknown size at compile time.
```

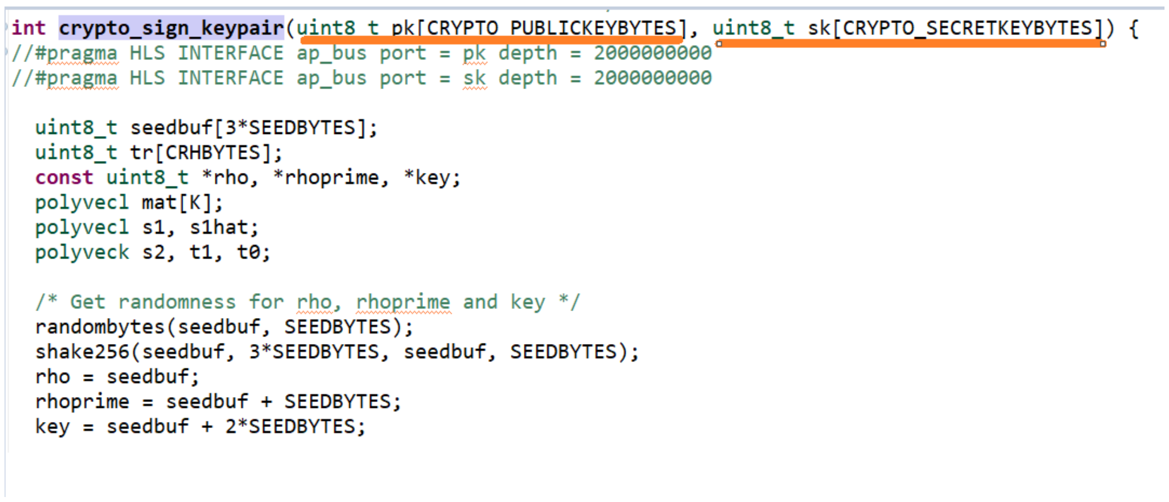
- Changes that we did in the project along with their reasoning:-

1. The Source will contain all files except PQCgenKAT_sign.c.
   Reasoning: Since PQCgenKAT_sign.c contains error checks and no new function is implemented, which gives us the clue that it should be in testbench.
2. Testbench contains only PQCgenKAT_sign.c.
   Reasoning: As we can see only PQCgenKAT_sign.c contains the top function- crypto_sign_keypair, it is the only file that we are keeping in testbench.
3. All the “#define macros” are commented out in api.h and sign.h.
   Reasoning: Since only these files contain our top function, crypto_sign_keypair. The reason why we did this is because using #define is renaming our function and hence its making it hard for our software to discover it. For example, we can see that here



```
#define crypto_sign_keypair DILITHIUM_NAMESPACE(_keypair)
int crypto_sign_keypair(uint8_t *pk, uint8_t *sk);
```

   The crypto_sign_keypair is getting renamed as DILITHIUM_NAMESPACE.
4. randombytes() function exists in two different files randombytes.c and rng.c. The function name is changed into randombytes2() in randombytes.c so as to distinguish those two functions.

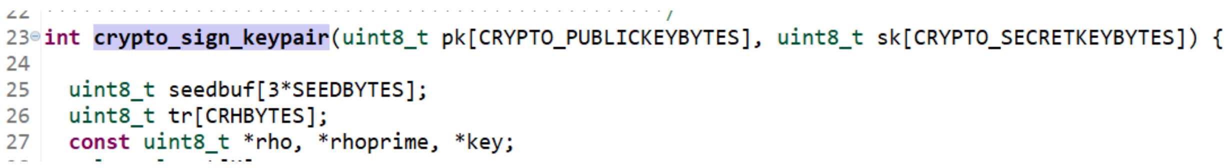


```
int crypto_sign_keypair(uint8_t pk[CRYPTO_PUBLICKEYBYTES], uint8_t sk[CRYPTO_SECRETKEYBYTES]) {
//#pragma HLS INTERFACE ap_bus port = pk depth = 2000000000
//#pragma HLS INTERFACE ap_bus port = sk depth = 2000000000

  uint8_t seedbuf[3*SEEDBYTES];
  uint8_t tr[CRHBYTES];
  const uint8_t *rho, *rhoprime, *key;
  polyvecl mat[K];
  polyvecl s1, s1hat;
  polyveck s2, t1, t0;

  /* Get randomness for rho, rhoprime and key */
  randombytes(seedbuf, SEEDBYTES);
  shake256(seedbuf, 3*SEEDBYTES, seedbuf, SEEDBYTES);
  rho = seedbuf;
  rhoprime = seedbuf + SEEDBYTES;
  key = seedbuf + 2*SEEDBYTES;
```

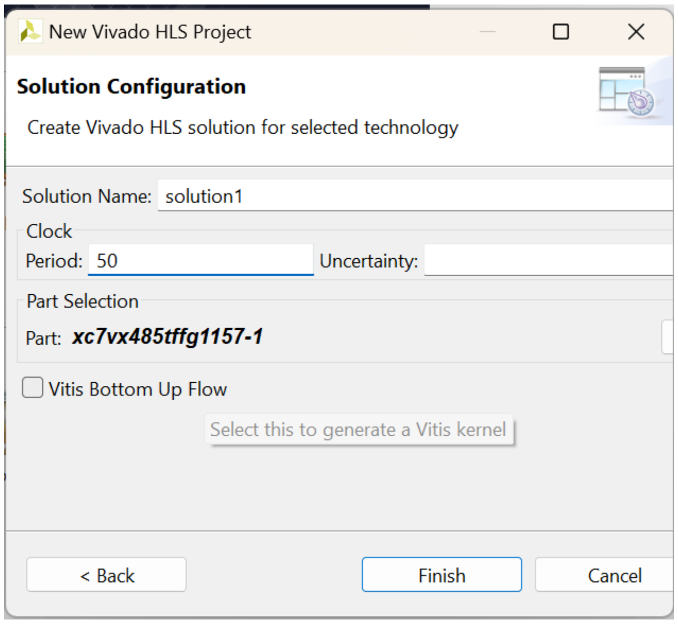
5. Unsupported memory access error was handled by replacing pointers with fixed sized array.



```
int crypto_sign_keypair(uint8_t pk[CRYPTO_PUBLICKEYBYTES], uint8_t sk[CRYPTO_SECRETKEYBYTES]) {

  uint8_t seedbuf[3*SEEDBYTES];
  uint8_t tr[CRHBYTES];
  const uint8_t *rho, *rhoprime, *key;
```

6. We tried it at 10 clock period but it wasn't working, so we increased it gradually and settled at 50 since it worked smoothly at that.



New Vivado HLS Project

Solution Configuration

Create Vivado HLS solution for selected technology

Solution Name: solution1

Clock

Period: 50 Uncertainty:

Part Selection

Part: *xc7vx485tffg1157-1*

☐ Vitis Bottom Up Flow

- Various other attempts to resolve the issues.
  1. As the array was referenced by the pointer, the interface can be synthesized only when it has been declared as ap_bus.
  So we used the pragma HLS INTERFACE ap_bus port = sk depth = d,
  Where we gradually incremented d but insufficient depth issue still arose even if depth was given max int value of 2^32 - 1.
  2. We tried to change the test bench by keeping only call to crypto_sign_keypair and commenting the rest but this attempt proved to be a failure.
  3.