

2022  
DESIGN AND VERIFICATION™  
**DVCON**  
CONFERENCE AND EXHIBITION  
**EUROPE**  
MUNICH, GERMANY  
DECEMBER 6 - 7, 2022

# Development of a Core-Monitor for HW/SW Co- Debugging targetting Emulation Platform

Shreya Morgansgate

Dr. techn. Johannes Grinschgl

Dr. Ing. Djones Lettnin



# Outline

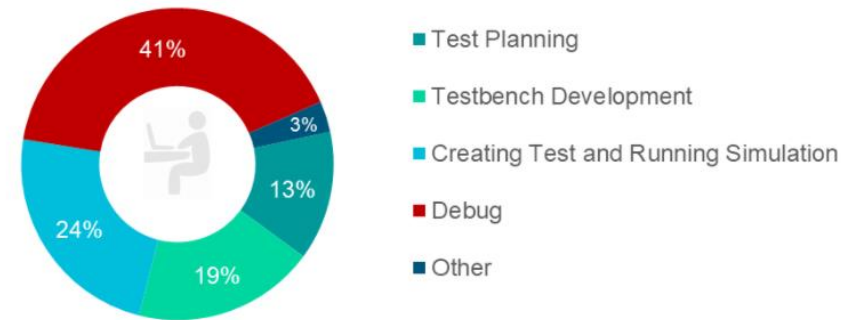
1. Motivation
2. Problem Statement
3. Technical Background
4. Methodology
5. Implementation
6. Results
7. Conclusion

# Outline

1. Motivation
2. Problem Statement
3. Technical Background
4. Methodology
5. Implementation
6. Results
7. Conclusion

# Motivation

- The SoC architecture that combines multiple heterogeneous cores and IPs are prevalent in a wide range of applications
- HW/SW co-debugging requires high effort:
  - Complex HW/SW system integration
  - More time spent in debugging
  - Shortens the time to market



Source: Wilson Research Group and Mentor, A Siemens Business, 2020 Functional Verification Study [1]

# Outline

1. Motivation
2. Problem Statement
3. Technical Background
4. Methodology
5. Implementation
6. Results
7. Conclusion

# What is a Core-Monitor?

- Core-Monitor is required to have a general HW/SW co-debugging environment for an embedded processor
- It generates a trace file that monitors the run-time behavior of the CPU
- The trace file is a human understandable log file with executed core instructions
- This trace file will be input to the debugging tools like Indago [2] for further processing

# Problem Statement(1)

- There is a need to trace interactions across multiple and heterogeneous processing elements
- There is no universal and reusable solution to trace the processor state of multiple cores in a processor
- There is no reusable platform to enable switching between simulation and emulation environments for HW/SW co-debugging

# Problem Statement(2)

- The Core Monitor aims to solve this problem:
  - It is standard for trace file generation in both the simulation and emulation environments
  - Reusable approach for processors with multiple cores
  - Reduces the HW/SW co-debugging time and shortens the time to market



# Outline

1. Motivation
2. Problem Statement
3. Technical Background
4. Methodology
5. Implementation
6. Results
7. Conclusion

# Why Emulation?

- Simulation-based techniques are unable to keep-up with today's growing size and complexity of designs with software and embedded processor core models
- Large design projects have moved on to emulation platform for SoC integration and verification

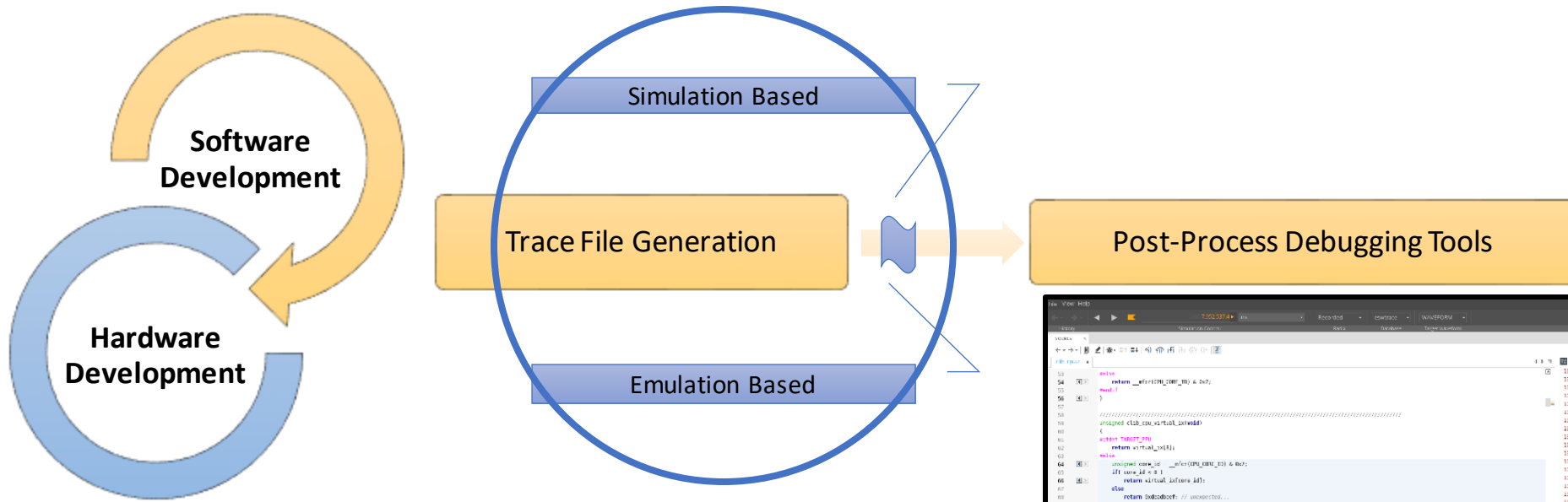
# Emulation Setup

- Zebu Server [3]
  - It is a Synopsys emulation system
  - In ZeBu emulation, design files and project files are compiled using VCS simulator
- Direct Programming Interface (DPI) [4]
  - System Verilog DPI is an interface for SV to interact with other languages like C, C++, SystemC
  - Zebu supports SV-DPI import calls to the host
  - Maximizes runtime efficiency

# Outline

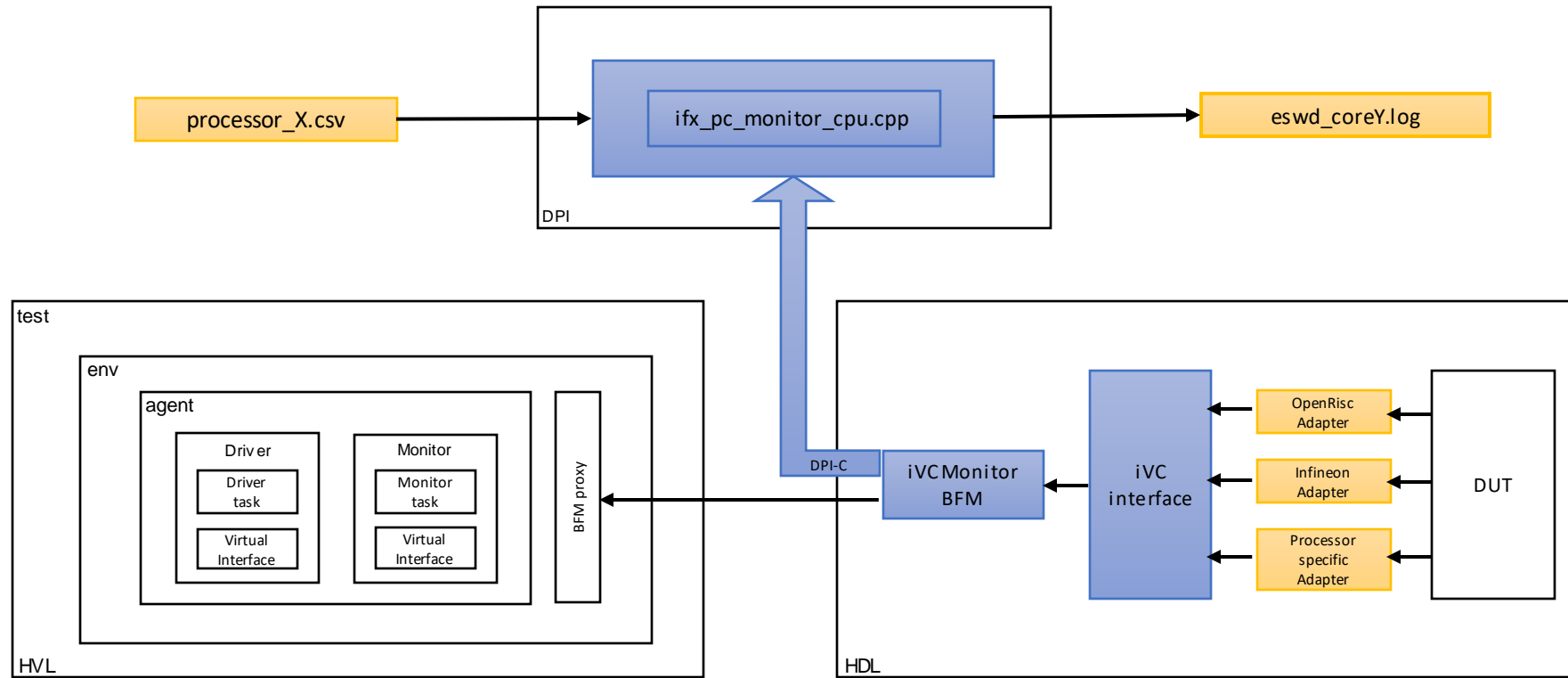
1. Motivation
2. Problem Statement
3. Technical Background
4. Methodology
5. Implementation
6. Results
7. Conclusion

# Typical System Flow



```
1170 0001360   
1171 01000000   
1172 01000000   
1173 01000000   
1174 01000000   
1175 01000000   
1176 01000000   
1177 01000000   
1178 01000000   
1179 01000000   
1180 01000000   
1181 01000000   
1182 01000000   
1183 01000000   
1184 01000000   
1185 01000000   
1186 01000000   
1187 01000000   
1188 01000000   
1189 01000000   
1190 01000000   
1191 01000000   
1192 01000000   
1193 01000000   
1194 01000000   
1195 01000000   
1196 01000000   
1197 01000000   
1198 01000000   
1199 01000000   
1200 01000000   
1201 01000000   
1202 01000000   
1203 01000000   
1204 01000000   
1205 01000000   
1206 01000000   
1207 01000000   
1208 01000000   
1209 01000000   
1210 01000000   
1211 01000000   
1212 01000000   
1213 01000000   
1214 01000000   
1215 01000000   
1216 01000000   
1217 01000000   
1218 01000000   
1219 01000000   
1220 01000000   
1221 01000000   
1222 01000000   
1223 01000000   
1224 01000000   
1225 01000000   
1226 01000000   
1227 01000000   
1228 01000000   
1229 01000000   
1230 01000000   
1231 01000000   
1232 01000000   
1233 01000000   
1234 01000000   
1235 01000000   
1236 01000000   
1237 01000000   
1238 01000000   
1239 01000000   
1240 01000000   
1241 01000000   
1242 01000000   
1243 01000000   
1244 01000000   
1245 01000000   
1246 01000000   
1247 01000000   
1248 01000000   
1249 01000000   
1250 01000000   
1251 01000000   
1252 01000000   
1253 01000000   
1254 01000000   
1255 01000000   
1256 01000000   
1257 01000000   
1258 01000000   
1259 01000000   
1260 01000000   
1261 01000000   
1262 01000000   
1263 01000000   
1264 01000000   
1265 01000000   
1266 01000000   
1267 01000000   
1268 01000000   
1269 01000000   
1270 01000000   
1271 01000000   
1272 01000000   
1273 01000000   
1274 01000000   
1275 01000000   
1276 01000000   
1277 01000000   
1278 01000000   
1279 01000000   
1280 01000000   
1281 01000000   
1282 01000000   
1283 01000000   
1284 01000000   
1285 01000000   
1286 01000000   
1287 01000000   
1288 01000000   
1289 01000000   
1290 01000000   
1291 01000000   
1292 01000000   
1293 01000000   
1294 01000000   
1295 01000000   
1296 01000000   
1297 01000000   
1298 01000000   
1299 01000000   
1300 01000000   
1301 01000000   
1302 01000000   
1303 01000000   
1304 01000000   
1305 01000000   
1306 01000000   
1307 01000000   
1308 01000000   
1309 01000000   
1310 01000000   
1311 01000000   
1312 01000000   
1313 01000000   
1314 01000000   
1315 01000000   
1316 01000000   
1317 01000000   
1318 01000000   
1319 01000000   
1320 01000000   
1321 01000000   
1322 01000000   
1323 01000000   
1324 01000000   
1325 01000000   
1326 01000000   
1327 01000000   
1328 01000000   
1329 01000000   
1330 01000000   
1331 01000000   
1332 01000000   
1333 01000000   
1334 01000000   
1335 01000000   
1336 01000000   
1337 01000000   
1338 01000000   
1339 01000000   
1340 01000000   
1341 01000000   
1342 01000000   
1343 01000000   
1344 01000000   
1345 01000000   
1346 01000000   
1347 01000000   
1348 01000000   
1349 01000000   
1350 01000000   
1351 01000000   
1352 01000000   
1353 01000000   
1354 01000000   
1355 01000000   
1356 01000000   
1357 01000000   
1358 01000000   
1359 01000000   
1360 01000000   
1361 01000000   
1362 01000000   
1363 01000000   
1364 01000000   
1365 01000000   
1366 01000000   
1367 01000000   
1368 01000000   
1369 01000000   
1370 01000000   
1371 01000000   
1372 01000000   
1373 01000000   
1374 01000000   
1375 01000000   
1376 01000000   
1377 01000000   
1378 01000000   
1379 01000000   
1380 01000000   
1381 01000000   
1382 01000000   
1383 01000000   
1384 01000000   
1385 01000000   
1386 01000000   
1387 01000000   
1388 01000000   
1389 01000000   
1390 01000000   
1391 01000000   
1392 01000000   
1393 01000000   
1394 01000000   
1395 01000000   
1396 01000000   
1397 01000000   
1398 01000000   
1399 01000000   
1400 01000000   
1401 01000000   
1402 01000000   
1403 01000000   
1404 01000000   
1405 01000000   
1406 01000000   
1407 01000000   
1408 01000000   
1409 01000000   
1410 01000000   
1411 01000000   
1412 01000000   
1413 01000000   
1414 01000000   
1415 01000000   
1416 01000000   
1417 01000000   
1418 01000000   
1419 01000000   
1420 01000000   
1421 01000000   
1422 01000000   
1423 01000000   
1424 01000000   
1425 01000000   
1426 01000000   
1427 01000000   
1428 01000000   
1429 01000000   
1430 01000000   
1431 01000000   
1432 01000000   
1433 01000000   
1434 01000000   
1435 01000000   
1436 01000000   
1437 01000000   
1438 01000000   
1439 01000000   
1440 01000000   
1441 01000000   
1442 01000000   
1443 01000000   
1444 01000000   
1445 01000000   
1446 01000000   
1447 01000000   
1448 01000000   
1449 01000000   
1450 01000000   
1451 01000000   
1452 01000000   
1453 01000000   
1454 01000000   
1455 01000000   
1456 01000000   
1457 01000000   
1458 01000000   
1459 01000000   
1460 01000000   
1461 01000000   
1462 01000000   
1463 01000000   
1464 01000000   
1465 01000000   
1466 01000000   
1467 01000000   
1468 01000000   
1469 01000000   
1470 01000000   
1471 01000000   
1472 01000000   
1473 01000000   
1474 01000000   
1475 01000000   
1476 01000000   
1477 01000000   
1478 01000000   
1479 01000000   
1480 01000000   
1481 01000000   
1482 01000000   
1483 01000000   
1484 01000000   
1485 01000000   
1486 01000000   
1487 01000000   
1488 01000000   
1489 01000000   
1490 01000000   
1491 01000000   
1492 01000000   
1493 01000000   
1494 01000000   
1495 01000000   
1496 01000000   
1497 01000000   
1498 01000000   
1499 01000000   
1500 01000000   
1501 01000000   
1502 01000000   
1503 01000000   
1504 01000000   
1505 01000000   
1506 01000000   
1507 01000000   
1508 01000000   
1509 01000000   
1510 01000000   
1511 01000000   
1512 01000000   
1513 01000000   
1514 01000000   
1515 01000000   
1516 01000000   
1517 01000000   
1518 01000000   
1519 01000000   
1520 01000000   
1521 01000000   
1522 01000000   
1523 01000000   
1524 01000000   
1525 01000000   
1526 01000000   
1527 01000000   
1528 01000000   
1529 01000000   
1530 01000000   
1531 01000000   
1532 01000000   
1533 01000000   
1534 01000000   
1535 01000000   
1536 01000000   
1537 01000000   
1538 01000000   
1539 01000000   
1540 01000000   
1541 01000000   
1542 01000000   
1543 01000000   
1544 01000000   
1545 01000000   
1546 01000000   
1547 01000000   
1548 01000000   
1549 01000000   
1550 01000000   
1551 01000000   
1552 01000000   
1553 01000000   
1554 01000000   
1555 01000000   
1556 01000000   
1557 01000000   
1558 01000000   
1559 01000000   
1560 01000000   
1561 01000000   
1562 01000000   
1563 01000000   
1564 01000000   
1565 01000000   
1566 01000000   
1567 01000000   
1568 01000000   
1569 01000000   
1570 01000000   
1571 01000000   
1572 01000000   
1573 01000000   
1574 01000000   
1575 01000000   
1576 01000000   
1577 01000000   
1578 01000000   
1579 01000000   
1580 01000000   
1581 01000000   
1582 01000000   
1583 01000000   
1584 01000000   
1585 01000000   
1586 01000000   
1587 01000000   
1588 01000000   
1589 01000000   
1590 01000000   
1591 01000000   
1592 01000000   
1593 01000000   
1594 01000000   
1595 01000000   
1596 01000000   
1597 01000000   
1598 01000000   
1599 01000000   
1600 01000000
```

# Architecture: Core-Monitor

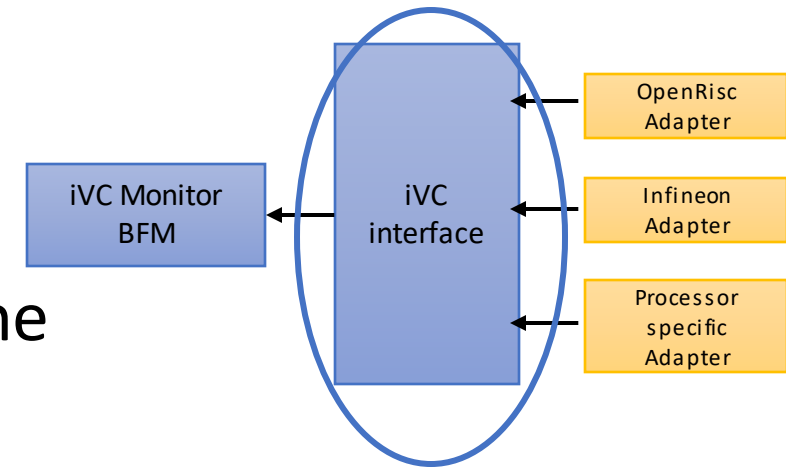


# Outline

1. Motivation
2. Problem Statement
3. Technical Background
4. Methodology
5. Implementation
6. Results
7. Conclusion

# Implementation - iVC Interface

- Connects the output from different adapters to the monitor BFM.
- The signals in the interface are CPU independent and can be used by all the adapters.



```
logic cpu_rst_n; // low active reset
logic cpu_clk; // clock
logic [ifx_pc_monitor_pkg::MAX_PC_LENGTH-1:0] cpu_pc[ifx_pc_monitor_pkg::PC_NUM]; // program counter
logic [ifx_pc_monitor_pkg::MAX_INSTRUCTION_LENGTH-1:0] cpu_instruction[ifx_pc_monitor_pkg::PC_NUM];
logic cpu_pc_valid[ifx_pc_monitor_pkg::PC_NUM]; // particular PC valid
logic [ifx_pc_monitor_pkg::MAX_MEM_ADDR_LENGTH-1:0] cpu_mem_addr; // memory store/load address
logic [ifx_pc_monitor_pkg::MAX_MEM_DATA_LENGTH-1:0] cpu_data; // memory load/store data
logic [ifx_pc_monitor_pkg::MAX_MEM_SIZE:0] cpu_mem_size;
logic cpu_mem_ack; // acknowledgement from the memory subsystem so that the memory write/read operation can proceed
logic cpu_mem_str_enable; // memory write enable
logic [ifx_pc_monitor_pkg::MAX_REG_DATA_LENGTH-1:0] reg_data[ifx_pc_monitor_pkg::MAX_REG_NUM]; // accessed register values
logic [ifx_pc_monitor_pkg::MAX_CPU_ID_NUM:0] cpu_id; // id for processor config file
```



# Implementation - Register Tracing(1)

```
always@(posedge cpu_clk or negedge cpu_reset_n)
  if(!cpu_reset_n) begin
    vif.reg_data[0:15] <='{default:0};
  end
  else begin
    if((cpu_rf_we != cpu_rf_we_reg) && cpu_rf_we) begin
      vif.reg_data[cpu_rf_wr_addr] <= cpu_rf_wr_data;
    end
  end
end
```

```
always@(posedge cpu_clk or negedge cpu_reset_n)
  ...
  if((cpu_except_started != cpu_except_started_reg) &&cpu_except_started) begin
    vif.reg_data[EPCRO] <= exception_reg[0]; //cpu_epr;
    vif.reg_data[ESR0] <= exception_reg[1]; //cpu_esr;
    vif.reg_data[EEAR0] <= exception_reg[2]; //cpu_eear;
  end
end
```

```
always@(posedge cpu_clk or negedge cpu_reset_n)
  ...
  if((cpu_supervision_reg_we != cpu_supervision_reg_we_reg) && cpu_supervision_reg_we) begin
    vif.reg_data[SR] <= cpu_supervision_reg_wr_data;
  end
end
```

Since the registers R0..R15 of the OpenRisc processor are driven in different clock cycles, they are put in their corresponding addresses of the interface array

The exception and supervision registers may be driven in the same cycle and hence they take in the corresponding addresses of the interface signal array.

# Implementation - Register Tracing(2)

- In the case of the Infineon's processor the register is accessed as a complete array in the respective adapter.

```
assign vif.reg_data = register_data;
```

- Here reg\_data is an array with all the register values of the processor.
- It can be updated to each array location where the address is parameterized in the adapter.

# Implementation - Memory Tracing

- The interface signals for memory tracing are as below:

```
logic [ifx_pc_monitor_pkg::MAX_MEM_ADDR_LENGTH-1:0] cpu_mem_addr; // memory store/load address  
logic [ifx_pc_monitor_pkg::MAX_MEM_DATA_LENGTH-1:0] cpu_mem_data; // memory load/store data  
logic [ifx_pc_monitor_pkg::MAX_MEM_SIZE-1:0] cpu_mem_size; // memory size  
logic cpu_mem_ack; // acknowledgement from the memory subsystem so that the memory write/read operation can proceed  
logic cpu_mem_str_enable; // memory write enable
```

- The adapter for Infineon's processor included a FIFO to initially store the memory signals.
- The OpenRisc processor required an additional if condition.
- The BFM then calls the C functions to write the relevant information into the trace file.

# Implementation - PC Tracing

- PC interface signals

```
logic [ifx_pc_monitor_pkg::MAX_PC_LENGTH-1:0] cpu_pc[CPU_PC_NUM];           // program counter array
logic [ifx_pc_monitor_pkg::MAX_INSTRUCTION_LENGTH-1:0] cpu_instruction[CPU_PC_NUM]; // executed instruction array
logic cpu_pc_valid[CPU_PC_NUM];                                           // undefined instruction (particular
```

- PC values from the DUT are assigned to the interface signals in the adapter using the simple assign statements.
- The signals from the interface are processed in the monitor BFM in every clock cycle and the PC value is driven in the generate block.
- Once the PC value is processed, it will be used by the C function call via DPI-C interface for further PC tracing

# Implementation – Parameterization

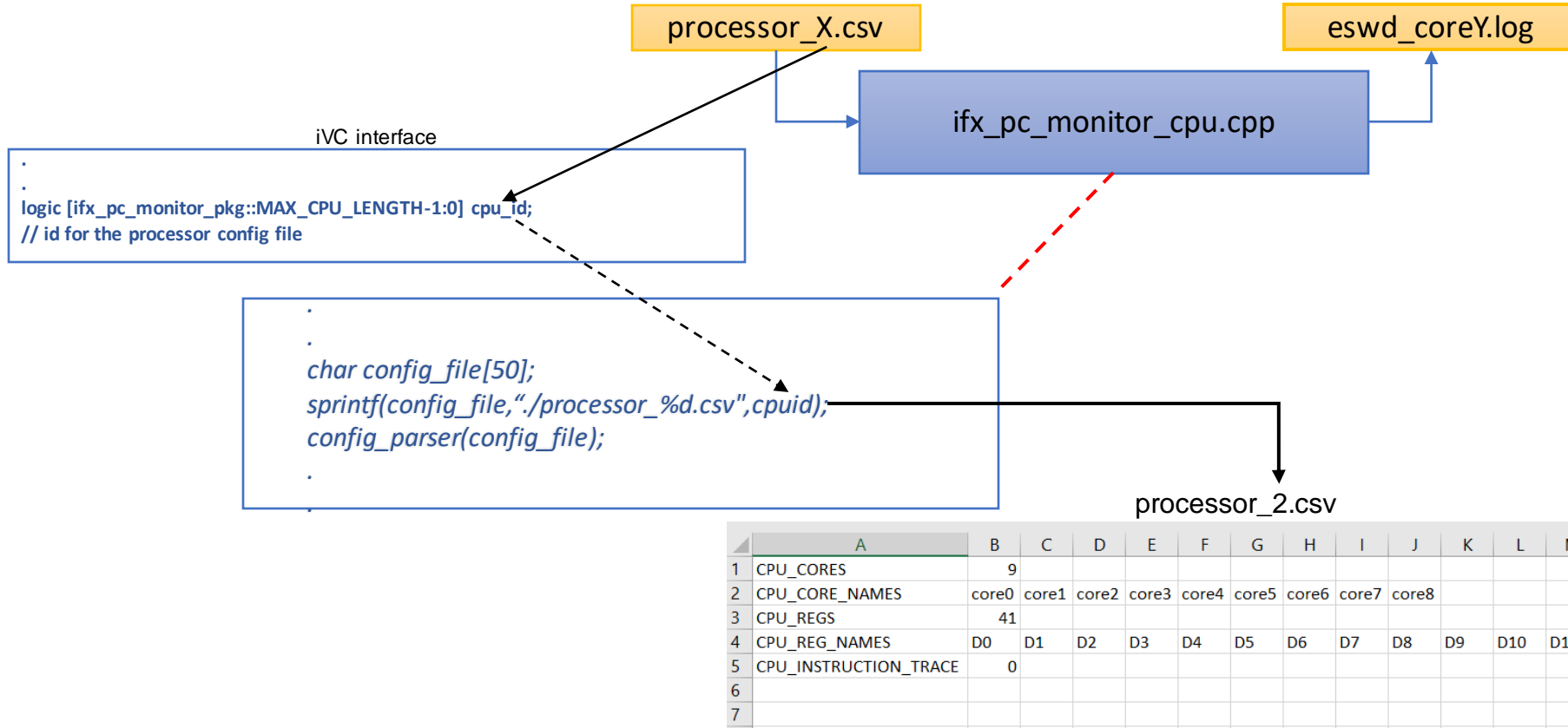
- Interface and BFM definitions are parameterized for PC count and register count signals

```
interface ifx_pc_monitor_if#(parameter CPU_PC_NUM = 16, parameter CPU_REG_NUM = 100)(input logic [3:0] cpu_num);
```

```
interface ifx_pc_monitor_monitor_bfm#(parameter CPU_PC_NUM = 16, parameter CPU_REG_NUM = 100)(ifx_pc_monitor_if ifc);
```

- This provides flexibility for heterogeneous cores with varying PC and register count

# Implementation – Processor specific config file



# Outline

1. Motivation
2. Problem Statement
3. Technical Background
4. Methodology
5. Implementation
6. Results
7. Conclusion

# Core-Monitor Setup for Emulation

- Core-Monitor setup for the Emulation environment with ZeBu emulator.
- Designed an adapter for the Infineon's processor.
- Successfully generated the trace files.
- The generated trace file is an input to post-processing HW/SW co-debugging tools such as Indago.

CPU trace of core0 of Infineon processor, eswd\_core0.log

1	5826200	ps	PC=0xafffff00
2	7126200	ps	PC=0xaafffc000
3	7176200	ps	PC=0xaafffc004
4	7176200	ps	REG=A10, 0xd0010000
5	7226200	ps	PC=0xaafffc008
6	7226200	ps	REG=A10, 0xd0009820
7	7276200	ps	PC=0xaafffc00c
8	7276200	ps	REG=A15, 0xf0060000
9	7326200	ps	REG=A15, 0xf0064404
10	8026200	ps	PC=0xaafffc010
11	8076200	ps	REG=D15, 0x00010001
12	8126200	ps	PC=0xaafffc012
13	8176200	ps	MR=0xf0064404, 0x00010001
14	8476200	ps	PC=0xaafffc016
15	8476200	ps	PC=0xaafffc01a
16	8526200	ps	REG=A15, 0xf0060000
17	8576200	ps	REG=D15, 0x44140000
18	9626200	ps	PC=0xaafffc01e
19	9676200	ps	PC=0xaafffc022
20	9676200	ps	PC=0xaafffc026
21	9676200	ps	REG=A15, 0xf0064410
22	9726200	ps	PC=0xaafffc028
23	9776200	ps	PC=0xaafffc02c
24	9776200	ps	REG=D15, 0x44141224
25	9776200	ps	REG=A15, 0xf0060000



# Performance Evaluation – Emulation

- The eswd option is necessary for trace file generation.
- Emulation Runtime comparison.
- Results are based on two testcases
- There is no deterioration in performance.
- The Core-Monitor provides a high degree of reusability.
- Substantially decreases the coding-effort.

Emulation Runtime	Core-Monitor	Existing-Method
Testcase 1	3197.3s (39053.3 Hz)	3635.9s (34344.6 Hz)
Testcase 2	889.2s (32186.3 Hz)	921.3s (31063.8 Hz)

# Outline

1. Motivation
2. Problem Statement
3. Technical Background
4. Methodology
5. Implementation
6. Results
7. Conclusion

# Conclusion

- A novel methodology that targets a reusable Core-Monitor
- Compatible with simulation and emulation platforms
- Designed processor specific adapters for Infineon processor and OpenRisc
- Verified the trace files generated
- No deterioration in performance
- Reduces the coding effort in large SoC designs

# References

[1] <https://blogs.sw.siemens.com/verificationhorizons/2021/01/06/part-8-the-2020-wilson-research-group-functional-verification-study/>

[2] Cadence Design Systems, Indago Embedded Software Debug App User Guide.

[3] Synopsis, Zebu Server User Guide <https://www.synopsys.com/verification/emulation/zebu-server.html>

[4] <https://www.doulos.com/knowhow/systemverilog/systemverilog-tutorials/systemverilog-dpi-tutorial/>

# Questions?

THANK YOU