Samsung Semiconductor India Research

2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SANTA CLARA, CA, USA
MARCH 2 - 5, 2026

Integrating RTL design and UVM Testbench with Hyperledger Blockchain and Machine Learning for better efficiency and optimisation

Sougata Bhattacharjee,
Samsung Semiconductor India Research (SSIR)

Problem Statement and Motivation

Problem:

- Complexity in the ASIC Verification process.
- Meeting time to market within the stipulated time.
- Reducing Disk Space Issues & Regression Management
- Bug Fixes

Solutions:

- Hyperledger Fabric Blockchain integrated with Testbench / DUT
- Smart Contracts
- Bug Fix with the help of Machine Learning

Agenda

- Basic Blockchain and Hyperledger Structure
- Hyperledger Framework Integration with UVM
- Integration of Fabric with Python and UVM
- Hyperledger Commands Compilation and Execution
- Ethereum to UVM Integration Flow
- Preference of Hyperledger over Ethereum and CI/CD Framework
- Usage of Machine Learning in an Integrated Blockchain Network
- Results
- Future Scope and Challenges

Basic Blockchain Framework

Blockchain is a growing list of records or transactions connected in a distributed peer-to-peer network. All blocks in the network are securely connected using the SHA-256 algorithm, making the network completely tamper-proof.

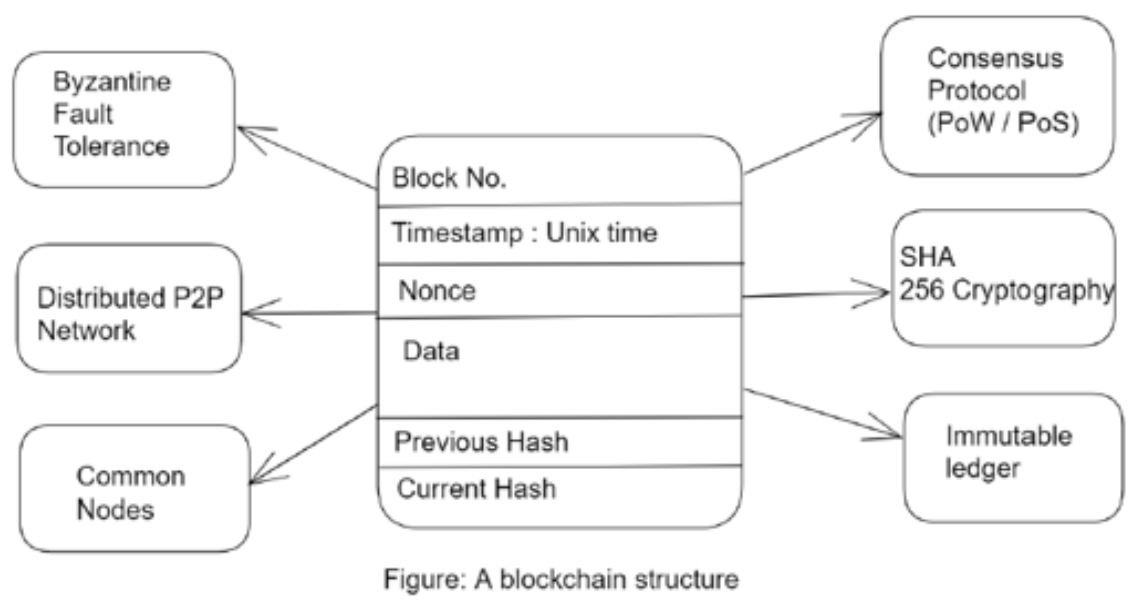
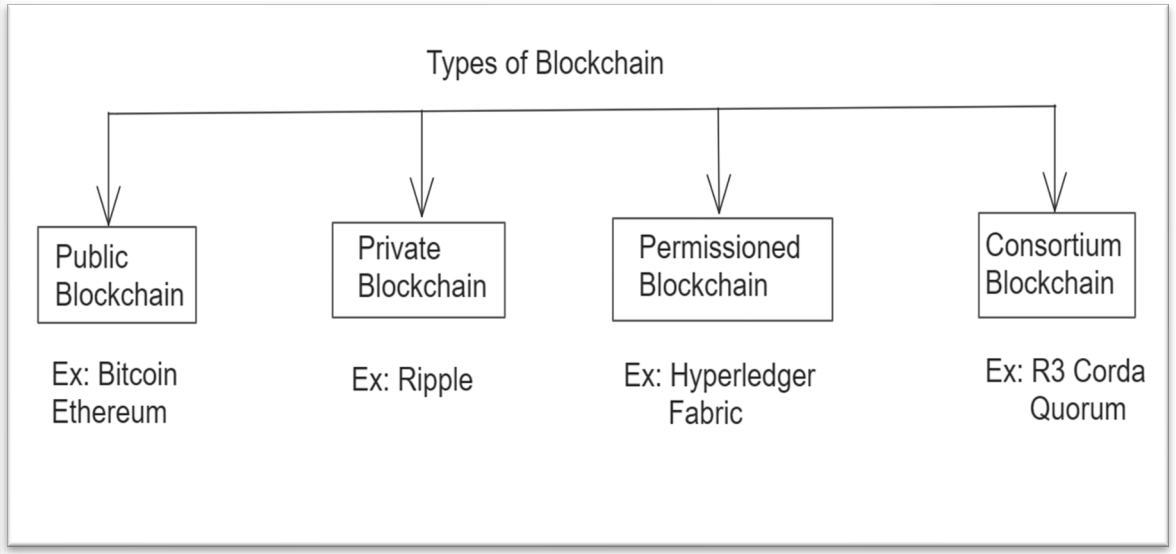


Figure: A blockchain structure



Hyperledger Structure

- ***Certificate Authority (CA):***

Hyperledger Fabric is a permissioned blockchain that differs from the public blockchain, like Ethereum, due to its confidential and private nature. To maintain this confidentiality, a Certificate Authority (CA) is provided to the selected peer to participate in the network.

- ***Endorsing Peer (EP):***

Endorsing Peers are the members of the network who have access to vote and validate the transaction.

- ***Gossip Protocol:***

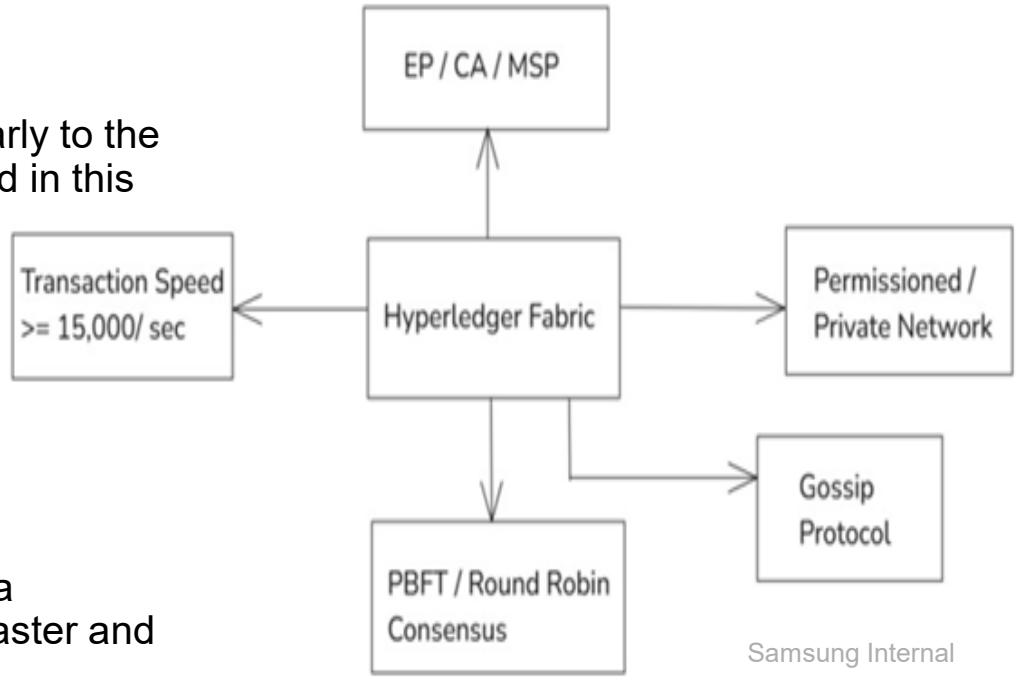
The method through which transactions flow in the network. It works similarly to the spreading of rumours and is lightning fast, and the same concept is applied in this framework for the reachability of transactions.

- ***Chaincode:***

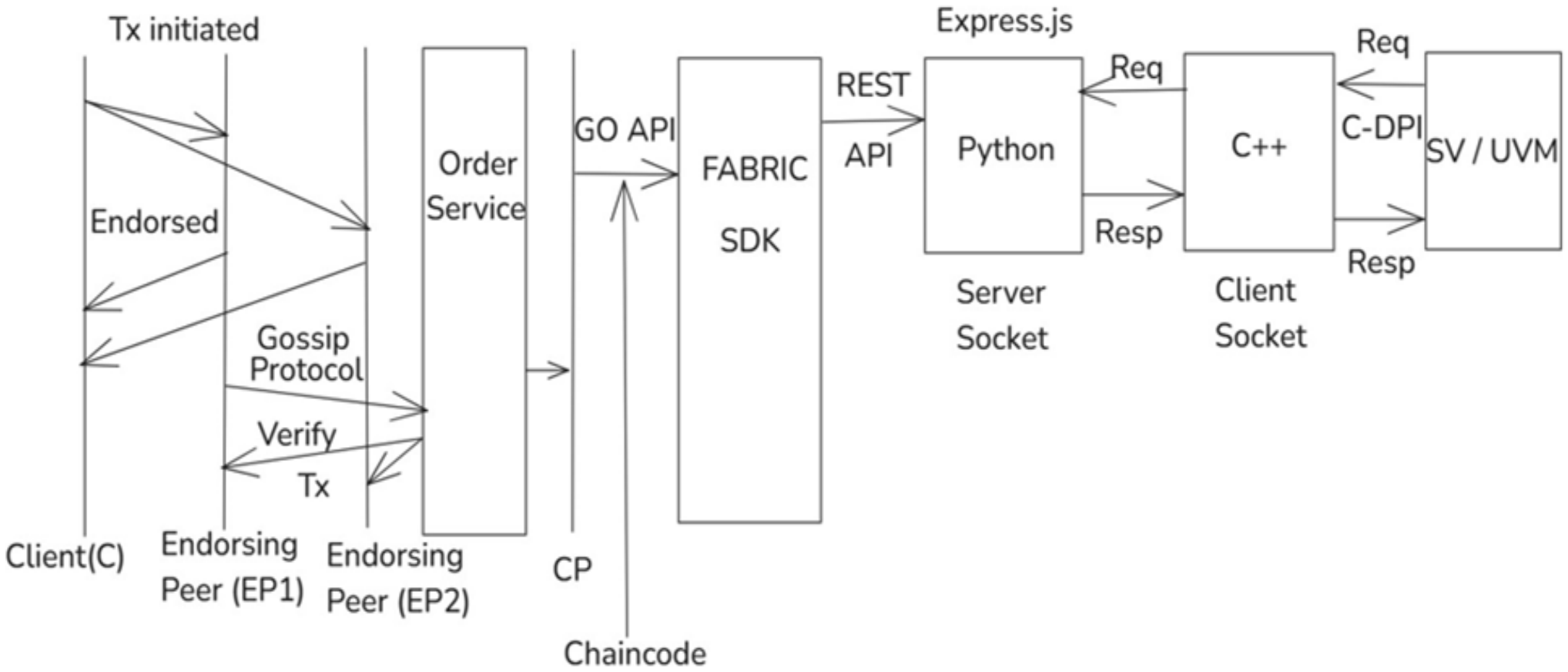
Self-executing code running on the Blockchain.

- ***Practical Byzantine Fault Tolerance (PBFT) consensus:***

The mechanism through which the majority of peers in the network reach a conclusion or arrive at a consensus that the transaction is valid. PBFT is faster and can validate a transaction at a speed of 15,000/sec.

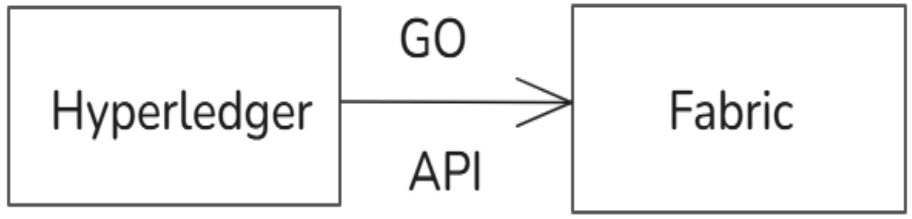


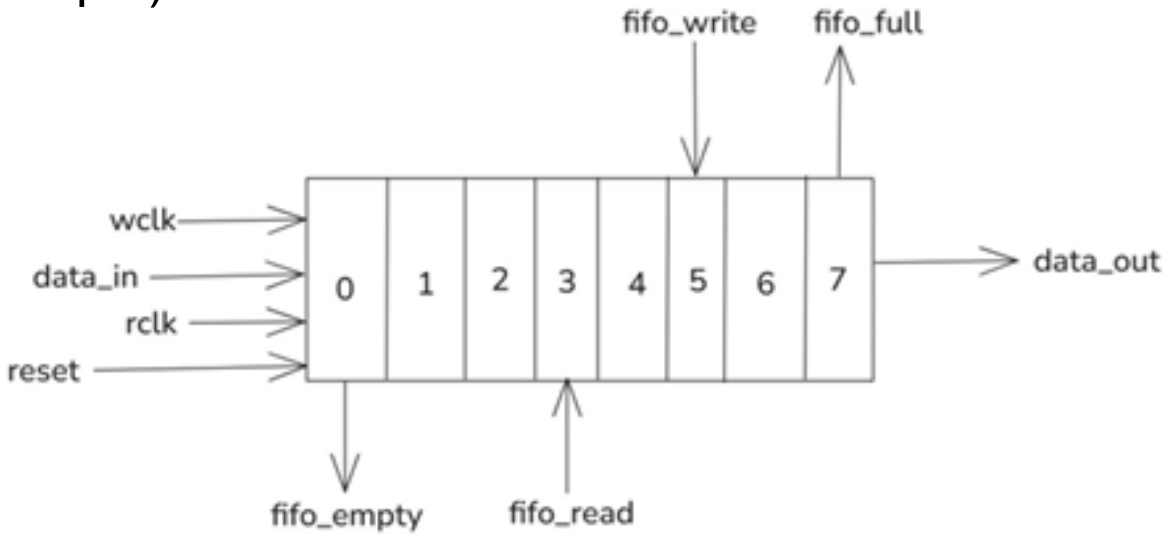
Hyperledger Execution & Integration with UVM



Integration of Hyperledger ←→ Fabric

GO API is the core foundation layer of Hyperledger Fabric, which is useful for Chaicode (self-executable code that runs on Blockchain and cannot be modified) execution, transaction validation, peer review, and as a key parameter for connecting Blockchain with Semiconductor/VLSI. In the context of the paper where FIFO is taken as a reference, the GO API acts as a Smart contract on top of FIFO RTL (Basically A wrapper of self-executable code written in the GO language, which defines the rules of FIFO or in short, the working principle)





Integration of Fabric ←→Python

- **fabric_client.js** → TCP-based gateway that receives verification requests and securely invokes Hyperledger Fabric Smart Contracts.
- **python_server.py** → Server that interacts with fabric through the REST API and handles responses.
- **organization.json** → Defines the Hyperledger network configuration, peer channel and connection parameters.
- **admin.js** → Fabric administrator identity used to manage blockchain access.
- **reg_user.js** → Enrolls the application users who are authorised to interact with the blockchain.

Integration of Fabric ←→Python ←→ C←→ UVM

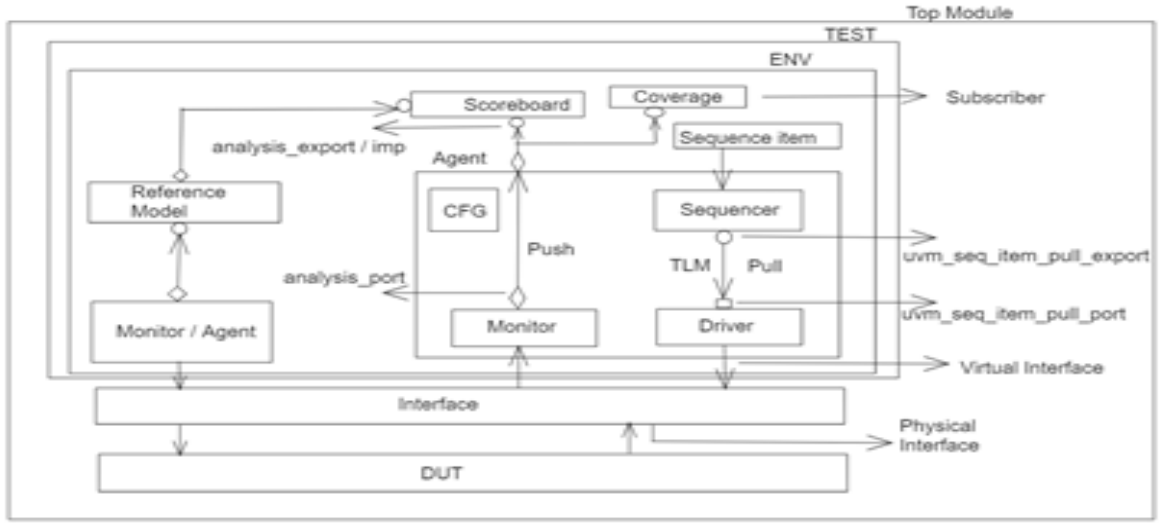
- **python3 python_server.py**

```
soug1@DESKTOP-TESGUBL:~/blockchain_council/src2/github.com/soug2/fabric-samples/test-network$ python3 python_server.py
[Python] Starting server...
[Python] Listening on 127.0.0.1:6000
```

- **node fabric_client.js**

```
soug1@DESKTOP-TESGUBL:~/blockchain_council/src2/github.com/soug2/fabric-samples/test-network$ node fabric_client.js
Fabric connecting to Python server...
Connected to Python server
Response from Python: {"status": "SUCCESS", "tx": {"command": "query", "args": ["ReadAsset", "asset1"]}}
Connection closed
```

- **g++ -fPIC -shared fabric_client.cpp -o libdpi_client.so**
  - libdpi_client.so is generated, which is used in the C-DPI interface to connect with SV/UVM Testbench by using the import DPI-C function.



✓ **Overall Flow:**

Hyperledger → Fabric Client → Python Server → C++ → C-DPI → UVM

Hyperledger Command and Execution

- ./network.sh up createChannel → Start the network
- go build fifo.go → Compile the GO API for FIFO
- peer lifecycle chaincode package fifo.tar.gz --path ../asset-transfer-basic/chaincode-go/fifo --lang golang --label fifo

```
soug1@DESKTOP-TESGUBL:~/blockchain_council/src2/github.com/soug2/fabric-samples/test-network$ peer lifecycle chaincode package fifo.tar.gz --path ../asset-transfer-basic/ch
aincode-go/fifo/ --lang golang --label fifo
soug1@DESKTOP-TESGUBL:~/blockchain_council/src2/github.com/soug2/fabric-samples/test-network$ ls
README.md  addOrg3  basic.tar.gz  channel-artifacts  configtx  docker  fifo.tar.gz  log.txt  memory.tar.gz  network.sh  organizations  scripts  system-genesis-block
```

- peer lifecycle chaincode install fifo.tar.gz →Basic chaincode command of Hyperledger

```
soug1@DESKTOP-TESGUBL:~/blockchain_council/src2/github.com/soug2/fabric-samples/test-network$ peer lifecycle chaincode install fifo.tar.gz
2024-10-21 19:54:48.952 UTC [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200 payload:"\nEfifo:3de25ce9d85a44eeacd22ad0e1
0131dd33163cc7d6b93604b70015fb3000c0ae\022\004fifo" >
2024-10-21 19:54:48.956 UTC [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: fifo:3de25ce9d85a44eeacd22ad0e10131dd33163cc7d6b9
3604b70015fb3000c0ae
```

- peer lifecycle chaincode queryinstalled

```
Installed chaincodes on peer:
Package ID: fifo:3de25ce9d85a44eeacd22ad0e10131dd33163cc7d6b93604b70015fb3000c0ae, Label: fifo
```

Hyperledger Command and Execution Cont'd

- peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1 --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" → Gossip Protocol

```
soug1@DESKTOP-TESGUBL:~/blockchain_council/src2/github.com/soug2/fabric-samples/test-network$ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnam
eOverride orderer.example.com --channelID mychannel --name basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1 --tls --cafile "${PWD}/organizations/ordererOrganiza
tions/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"
2024-10-19 20:30:59.681 UTC [chaincodeCmd] ClientWait -> INFO 001 txid [157715be23aeeca16b080aa28e2480fddc9fb050a152578194b0603ca52cce31] committed with status (VALID) at l
ocalhost:9051
```

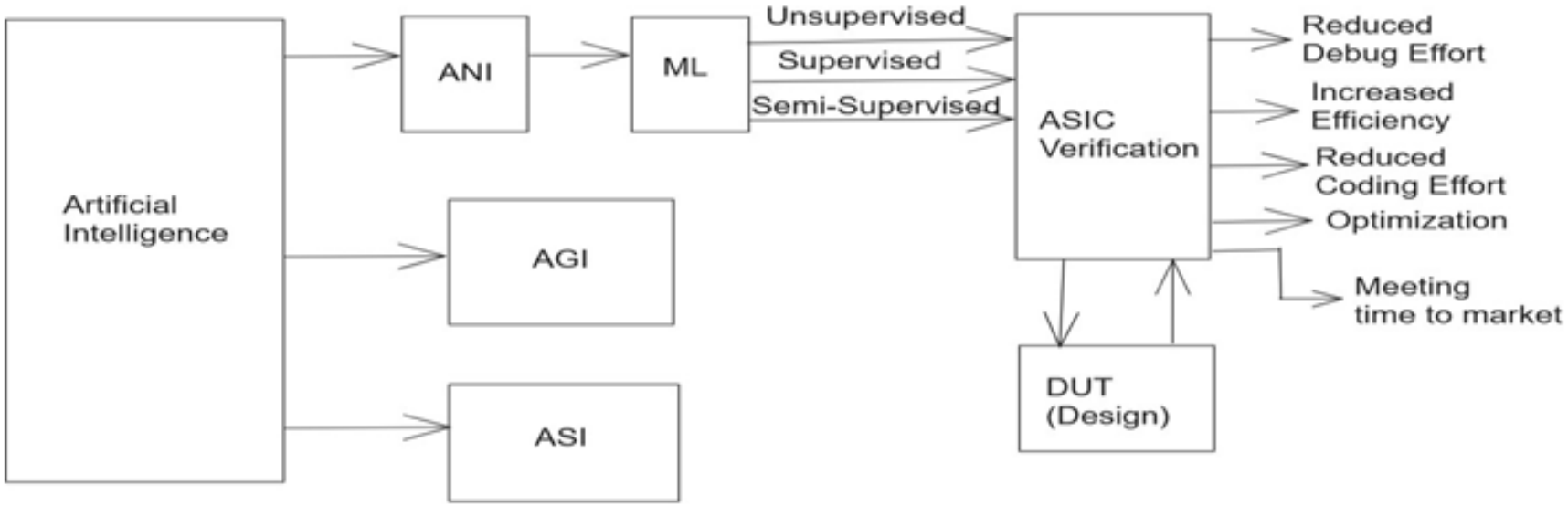
- peer lifecycle chaincode querycommitted --channelID mychannel --name basic --cafile "${*PWD*}/organizations/ordererOrganizations/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"

```
Committed chaincode definition for chaincode 'fifo' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]
```

```
soug1@DESKTOP-TESGUBL:~/blockchain_council/src2/github.com/soug2/fabric-samples/test-network$ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name fi
o --version 1.0 --sequence 1 --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"
-output json
{
        "approvals": {
                "Org1MSP": true,
                "Org2MSP": true
        }
}
```

Use of Machine Learning in Blockchain

- In UVM, the bugs can be categorized based on their failure signature, but there are outliers as well, which will not fit into this category, and for these features, like anomaly detection, it is useful. This Anomaly detection based ML learning mode helps to find out the protocol violations and corner case scenarios. Hence, for this paper Integrated Framework of Hyperledger Framework + Unsupervised learning ML model is implemented.
- This type of model is specifically useful for automatic capture of failure modes and error signatures without prior labelling.



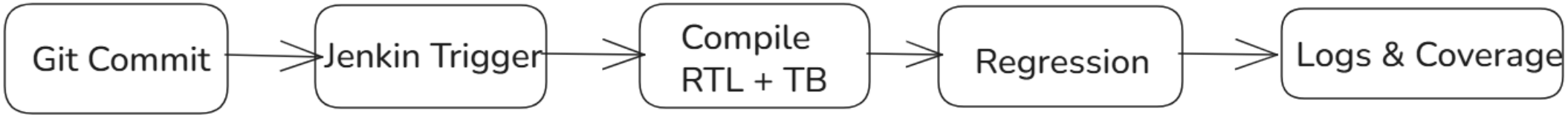
Use of CI/CD framework in a UVM Environment

- Jenkins is a CI/CD framework used to automate pipelines and build or test frameworks across multiple Developers within a centralised space and a single trust domain. In the context of UVM,

  CI = Build + Simulate + Check

  CD = Delivering Results (logs, coverage reports)

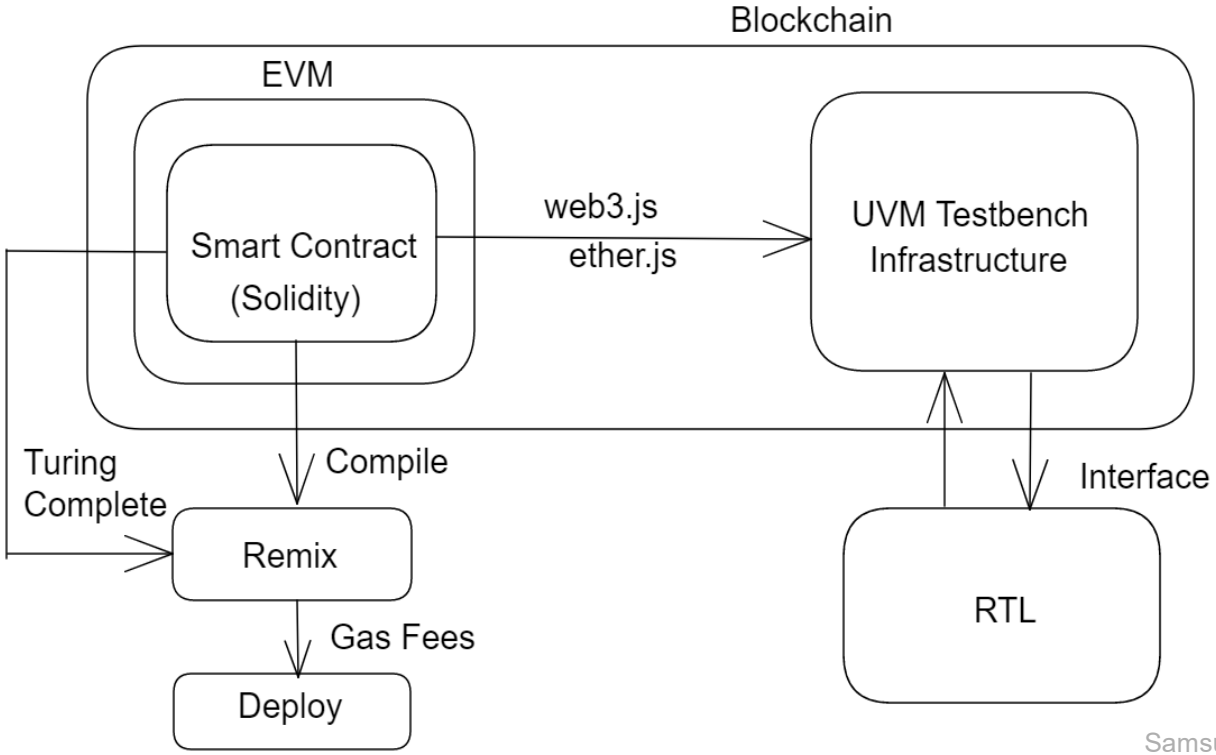
- A simple CI/CD flow in the context of UVM looks like this:



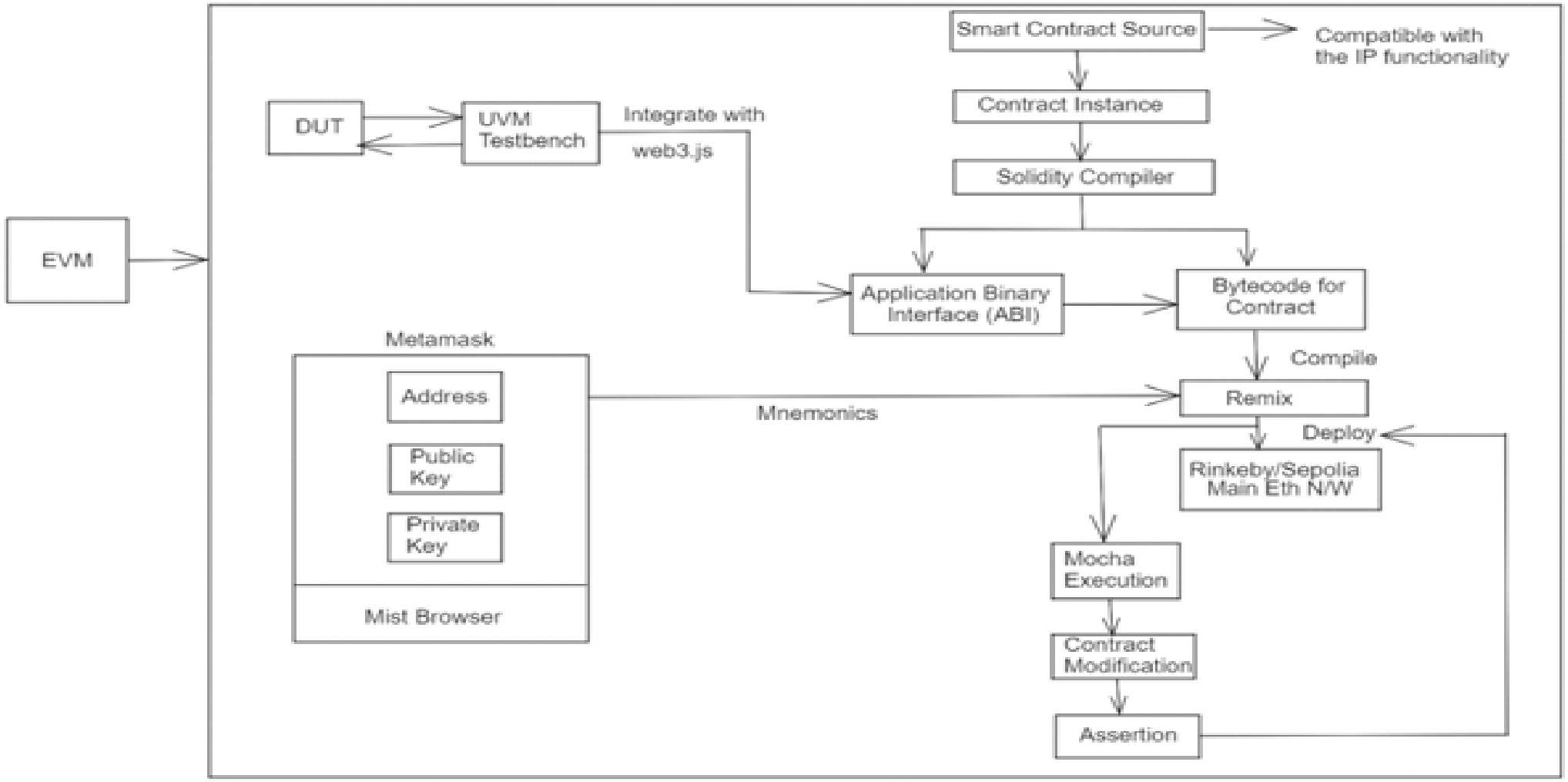
- Although Jenkins has been considered for lower latency, faster outcomes and storing CI artifacts, and integration with UVM, considering the decentralized working space, the use of Jenkins in the context of the Integrated Blockchain Environment remains a scope of future use case.

Preference of Hyperledger over Ethereum

- Provides Confidentiality and Privacy as compared to complete openness in Ethereum.
- No or limited Gas fees, and hence, there is no computation cost involved for coding the testbench.
- Transaction speed is higher (15,000/sec) as compared to Ethereum (20/sec)
- The PBFT consensus mechanism provides more flexibility to users as compared to the PoS (Proof of Stake) consensus in Ethereum.



Ethereum Integration Flow with UVM



Results

- The infrastructure cost of the server has been reduced by ~25% due to blockchain technology.
- The security issues with the hardware chips have been significantly reduced by 20% to 30%
- Meeting the time-to-market and executing complex projects will be faster as more engineers can collaborate on a private, decentralized server to work together towards their completion.
- Further implementing Anomaly detection using unsupervised learning in the UVM testbench helps categorize errors based on their signature and also detects any exceptions to them.
- With the implementation of Smart Contract / Chaincode, bug tracking will be easier as the bugs can be easily filtered and segregated depending on their signatures, as smart contracts are immutable.
- One of the key benefits of using Hyperledger is to use it in the Automotive Sector for Functional Safety ISO26262, where Audit and traceability are of utmost concern.

Future Scope & Challenges

- Hyperledger Fabric uses an ECDSA cryptography algorithm for signing transactions and an AES symmetric encryption mechanism for data privacy. Moreover, today's Machine learning algorithm, which is applied to perform a certain task on Blockchain, also offers security. However, the challenge is that, with the recent rise of Quantum computers and Shor's algorithm, it can solve complex logarithmic and discrete factorisation problems in polynomial time, making AES encryption and other security measures vulnerable.
- As a result, research is being conducted on post-quantum cryptography algorithms to make the existing Blockchain network and AI models resistant to Quantum attacks. Some of the algorithms are:

- Lattice Cryptography
- Code Cryptography
- Isogeny Cryptography

THANK YOU