Introduction to Blockchain and its possible applications to EDA

Naresh K. Sehgal, Ph.D.
Datacenter Security Initiatives Director,
Intel Corporation, Santa Clara CA

Acknowledgement: Prof. P. C. P. Bhatt, for his valuable suggestions.

Disclaimer: This work is solely based on publicly available information. Any newly proposed solutions have not been implemented.

Content

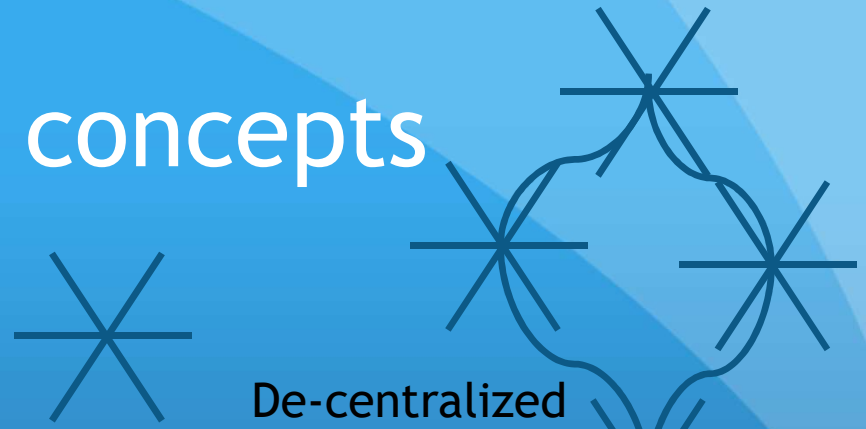
- What is Blockchain?
- Problems it solves
- A few problems from EDA
- Some Possible Solutions
- Next Steps

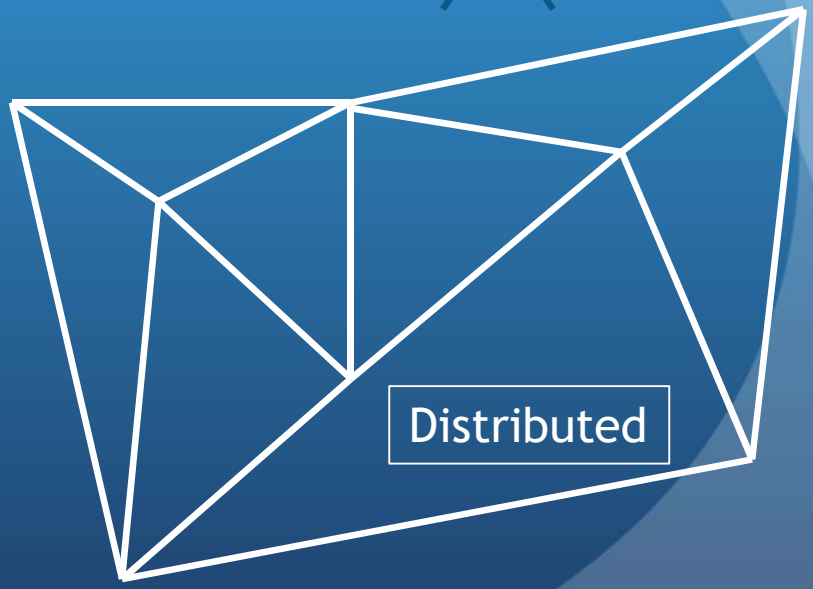
Blockchain = Block + Chain

- Block: A list of transactions from a certain time-period. Contains all the information processed on the network within past few minutes.
- Chain: Each block is time stamped, placed in a chronological order, and linked to the block before it using cryptographic algorithms.
- Hashing: takes an input of any length and creates output of fixed length. It is used to protect content of a block.
- Distributed Ledger: Every node on the network keeps its own copy of the ledger, and updates it when someone submits a new transaction.

A few more BC concepts

- Centralized vs. Decentralized consensus
- You store data in a linear blockchain container with your signatures.
- It is added to a distributed ledger.
- Anyone can verify it.
- Only you can unlock what's inside the container with your security keys.





Public visibility with Private inspection

Some applications of Blockchain

- **Bitcoins:** stores a token of value, i.e., money balance.
  - A value transfer system with no central authority.
  - Hashing prevents any malicious 3rd party tempering.
  - Distributed ledger prevents any duplicate transactions.
- **Smart Contracts:** self-managed w/o central rule maker
  - Two (or more) parties agree on the rules/terms.
  - Conditional money release when services fulfilled.
  - Incur penalties if not fulfilled.
- **Trusted Computing:** resource and transactions sharing
  - Combines BC, decentralized consensus and smart contracts
  - Supports spread of resources in a flat, peer-to-peer manner
  - Enables computers to trust one another w/o a central authority

A financial transaction example

- Every Block in ledger has an input, amount and output.
  - Input: if X wants to send a bitcoin, needs to show the source from where it came, this forms a link to a previous entry.
  - Amount: how much X wants to spend or send to someone?
  - Output: The address of new recipient, e.g., Y.
- A new transaction block is created and sent to the network
- Every node on the network updates its local ledger.
- Consensus among the nodes validates a new transaction.
- New block is thus added to the chain, forming a blockchain.
- Transactions are hard to undo.

Few possible applications for EDA

- Cell based vs. Custom designs
  - Depending on the number of cells, limits design choices
  - Cells may have bugs and incoming changes
  - Need root of trust for updating cell databases.
- Incremental requirements changes
  - New design requirements while design is underway
  - Need to maintain design consistency
  - Need to maintain history to roll-back changes
- EDA Tools versioning
  - Mixing and matching tools from multiple vendors
  - Multiple versions of the same tool

More areas of interest..

- Internet of Things: devices need to store and share information
  - Use a distributed ledger
  - Encrypt so that only the sender and receiver can decipher
  - Prevent any 3rd party malicious attacks
  - Maintain a verifiable history for debugging issues
- EDA in Cloud:
  - Use BC to securely transfer control and instructions from private to public cloud servers.
  - EDA tool vendors can use distributed ledgers to track their license usage and maintain compliance.
  - Cell library providers and 3rd party designers can contribute using smart contracts.

Proposed Solution for Cloud based hierarchical design Verification

- Lets say a design uses other 3rd party cells and IP libraries.
- Each cell provider maintains own ledger, showing who is using its cell, and in turn which sub-cells are in its own hierarchy.
- Only the top-level design knows the interconnects between all of its cells, thus owns its confidential design IP.
- Running a hierarchical verification puts out the inputs given to each cell on the BC network, and in turn that cell owner responds to the node that sent the inputs with its output.
  - Sender doesn't need to know where its cells are residing
  - A cell doesn't know how it is being used.
  - Only the top-level design owner knows all inputs and outputs.
  - No need to have a single or flattened database.
  - Any incremental Cell changes are easy to roll-up.

Next Steps

- Look for other areas of a design flow.
- An opportunity to mix in-house and cloud designs.
- Need to balance efficiency and confidentiality.