A Secure Gateway for Enabling Application Specific Integrated Circuit Design Collaborations

Steve Bogol*, Paul Brenner*, Adam Brinckman*, Ewa Deelman‡, Rafael Ferreira da Silva‡
Sandeep Gupta§, Jarek Nabrzyski*, Soowang Park§, Damian Perez*, Sarah Rucker*, Mats Rynge‡, Ian J. Taylor*†
Karan Vahi‡, Matt Vander Werf*, Sebastian Wyngaard*
*Center for Research Computing, University of Notre Dame, Notre Dame, IN, USA
‡Information Sciences Institute, University of Southern California, Marina Del Rey, CA, USA
§Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA
†School of Computer Science & Informatics, Cardiff University, Cardiff, UK

***Abstract*—Leading CAD companies are currently developing virtual secure environments to help lower the barriers for adopting new ASIC design flows. However, such services are proprietary, lack key features, and present barriers for collaboration and sharing. This paper covers the transition of the CRAFT repository, originally designed as a repository for discovery and documentation of ASIC design flows, to a fully secure environment called the CRAFT Vault hosted in Amazon Gov Cloud that allows designers to collaborate and actually implement these flows. The Vault is a fully configured environment that is deployed with all the necessary electronic design automation tools and IP making it easier for end users to implement a CRAFT Design Flow, augmented with Blockchain functionalities to provide a non-repudiable audit trail of what happened and when.**

***Keywords*—Collaborative Environments, Design Flows, Chip Design**

I. Introduction

A new Application-Specific Integrated Circuit (ASIC) design flow often faces high barriers for adoption, even by experienced design teams. New design flows typically require new tools; Intellectual Property (IP)/licensing considerations of acquiring new tools; installation of new tools and associated license servers; and the need to ensure that a user's environment are properly configured and that the new settings do not interfere with other installed software. Additionally, new design flows typically require new models and libraries, especially Process Design Kits (PDKs) (which contain all information provided by the semiconductor foundry regarding fabricated devices), cell libraries, libraries of larger IP modules, and scripts for designs. IP/licensing considerations make all this acquisition time-consuming and expensive, strongly deterring even the evaluation of promising new design flows. Increasingly, members of design teams span different parts of a large organization or even multiple organizations. Hence, coordinating the execution of a new design flow on various computational resources available to sub-teams is likely to be time consuming and inefficient.

To tackle such challenges, leading CAD companies are beginning to develop commercial services, such as Cadence VCAD [1]. However, such services are proprietary, lack key features, and present barriers for collaboration and sharing. In particular, these provide a private environment for each design team. In contrast, US Department of Defense (DoD) programs such as DARPA CRAFT [2] require sharing of design flows, IP, and best practices across design teams while protecting all information about specific designs being carried out by individual design teams. Also, the commercial services are typically limited to the vendor's own tools and design services and hence not optimized for rapid evaluation and adoption of new design flows, especially those that use tools and IP from multiple vendors. In short, commercially available services are difficult to adapt to serve the needs of programs like CRAFT that are trying to develop new tools and flows and that are interested in building a new community of design teams.

In this paper, we present the CRAFT Secure Vault, an extension to the CRAFT Repository [3] gateway, that provides design teams ready access to new design flows, tools, and IP, and an environment that supports the unique combination of rapid learning and quick time to design productivity, security, access control, and community-wide sharing of best practices of DoD programs like CRAFT.

II. Requirements

The goal of the CRAFT Secure Vault is to provide distributed design teams a ready-to-use platform that streamlines the process which otherwise requires teams to acquire new tools and IP from multiple vendors, install and configure these tools on their computing platform, and learn a new design flow using voluminous user manuals. It also provides a collaborative environment for geographically and institutionally diverse design teams to rapidly learn and evaluate the new flow and low time-to-productivity for ASIC design using new flows, tools, and IP.

To achieve the above goal, the CRAFT Vault has been designed to support the following use cases:

1) From the view of the design team, for each design flow hosted by the Vault, all tools required are installed and ready to use, simply upon the completion of an off-the-shelf licensing agreement by the team (more ahead). Also, all required PDKs, models, IP libraries, are also pre-installed upon completion of the agreement.

2) Each hosted design flow is captured precisely and represented using interactive views of the design flow for rapid learning. Also, each design flow and associated tools, PDKs, models, and IP libraries are tested thoroughly by using the flow to design appropriate example Very Large Scale Integration (VLSI) modules (called *cores*) for integration into the design of ASICs. The instructions for each step of a design flow are accompanied by design files used/obtained during the design of the example VLSI cores, and these instructions are refined based on the example design experience to make the new design flow more accessible.
3) Each design team is provided an environment for collaborative design which supports multiple design projects, sharing of best practices with members of the team and selected best practices with the wider community of design teams. Further, each design team may bring additional tools and IP to the Vault and fork from a hosted design flow to modify it for its unique needs.
4) The tools, PDKs, models, and IP libraries hosted in the Vault are governed by off-the-shelf licensing agreements designed to fit the needs of the range of design teams likely to use the Vault. Each design team may customize the licensing agreement by selecting the tools, PDKs, models, IP libraries, and the level of resources for collaboration and computing they wish to option. A single licensing agreement will make a customized instance of Vault available to the user and reduce the barrier to entry by avoiding prolonged delays associated with lengthy licensing, installation, customization, and testing.
5) From the view of the vendors/developers of tools, PDKs, models, and IP libraries, the Vault guarantees that each design team can only access the items it has optioned. The Vault is designed to ensure that PDKs, models, and IP are protected, i.e., cannot be exported outside, unless designated by the vendor as open. At the same time, each design team is allowed to export reports of simulations, characterization, and verification of their design. Out-of-Vault design improvements, simulation/verification, and/or tape-outs (i.e., sending the design to a foundry for fabrication of chips) are only permitted if the design team has separately secured out-of-Vault licenses.

III. System Architecture

Our broad goal is to provide a system that is capable of provisioning a secure environment, called the CRAFT Secure Vault, for a group of CRAFT collaborators that span a number of different teams, each having particular authentication and authorization privileges to tools, IP, and design data. A CRAFT Secure Vault is a secure collaborative environment, synergistic to a collaborative project in the CRAFT repository [3], which gives users direct access to ASIC design tools and data engineering capabilities, regardless of geography, so that team members can work in a predictable environment that defines securely who has access to what. It also avoids the lengthy process of acquiring tools (and their dependencies), IP, system setup, and administration. The high level overview of the CRAFT Secure Vault architecture is shown in Fig. 1.

In order to lock down access to the CRAFT Secure Vault (Fig. 1-*left*), participants cannot access it directly, instead they connect via a client side gateway using a single secure communication channel [4]. The gateway enforces the traffic of information to and from the Vault. The architecture of the Vault therefore consists of three core components, namely, a (1) secure CRAFT gateway, a (2) secure CRAFT Vault, and a (3) secure virtual private channel that securely connects the two environments.

From a CRAFT user's perspective, the result is that they are provided seamless access to a remote desktop in a secure Vault environment from their own desktop. From a technical standpoint however, this remote desktop connection involves a secure virtual private connection to an AWS GovCloud [5] hosted client side gateway and to a second AWS GovCloud virtual private cloud (VPC) environment, and an authorization and connection proxy to one of multiple secure Linux (or Windows) systems to provide remote desktop access to a secure Vault that hosts IP, data, and tools for collaboration in ASIC designs.

IV. System Design

In addition to our goal of providing a standalone, functional and secure environment for designing chip flows, we also aim to capture and provide an audit trail for the whole design process. We decided to leverage Blockchain as a means of hooking the various components into a common distributed ledger. The CRAFT Secure Vault consists of the following 4 main components, with Fig. 2 illustrating the interaction between the various pieces.

1) *Blockchain:* The data backbone runs on a distributed ledger with a data store per node for storing sensitive data.
2) *Secure Vault Servers:* A full audit trail is collected for each server so we have access to who did what and when.
3) *ASIC Design Flows:* To support the existing CRAFT repository, visualization tools have been ported to the CRAFT Secure Vault environment.
4) *Collaborative Workflow Tracking:* To record transactions on the ledger when users run scripts on certain data for a design flow.

A. Blockchain

A Blockchain [6] is a distributed ledger that provides a shared write only file system. Data is stored through consensus, meaning that all parties sign off on each transaction. Once data is written it cannot be removed by any single entity and hence, it provides a non-repudiable audit trail of what happened and when. All Vault implementations sit on such an underlying Blockchain system, for interaction with the distributed ledger.

Currently, we are using three Ethereum [7] nodes per Vault virtual private network (VPN) deployed in the Vault environment. We plan to augment this deployment with additional

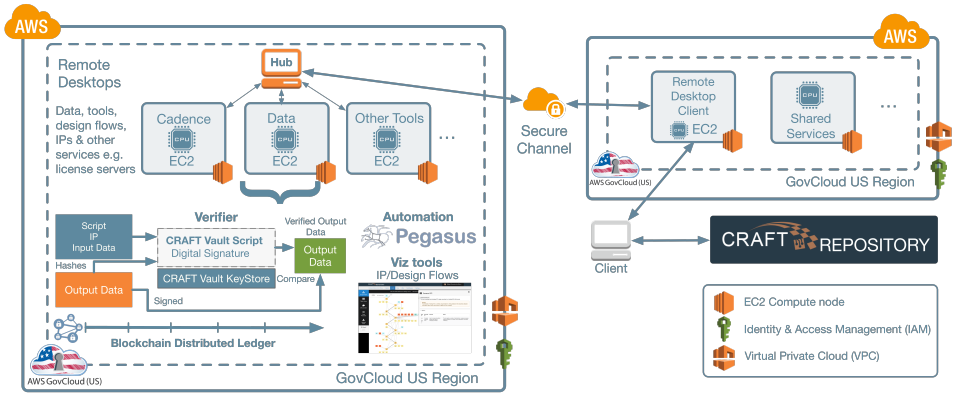

Fig. 1: Architectural overview of the CRAFT Secure Vault.

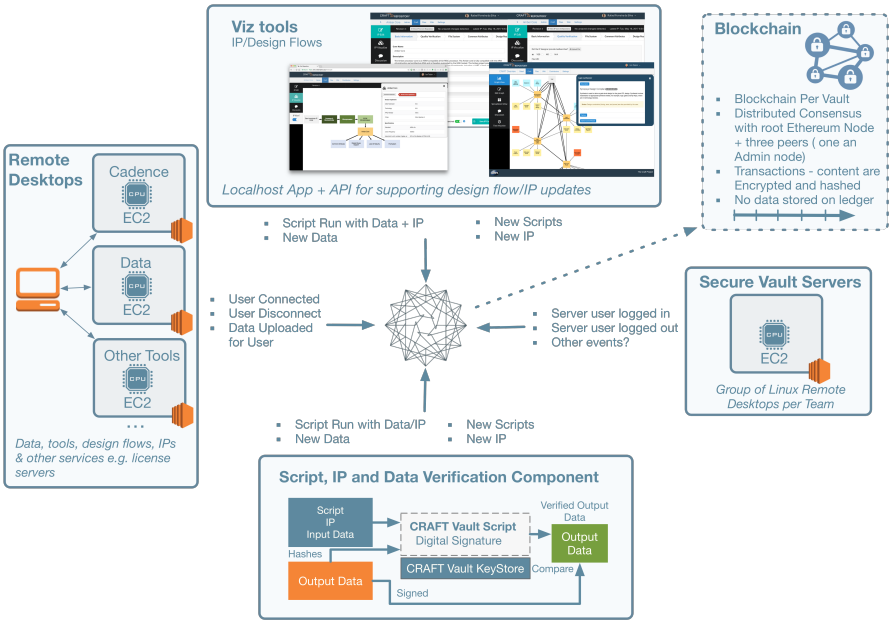

Fig. 2: Overview of the CRAFT Secure Vault Software Component Design.

Ethereum nodes one which will be hosted by the University of Notre Dame, and another controlled by DARPA in the near future. Addition of more nodes results in them being involved in the consensus, resulting in a stronger consensus backbone, insuring a tamper proof audit chain of events.

The Blockchain creates a cryptographic fingerprint (a *hash*) unique to each block and a consensus protocol, the process by which the nodes in the network agree on a shared history. The hash serves as proof that the miner who added the block to the Blockchain did the computational work. It also serves as a kind of seal, since altering the block would require generating a new hash. Verifying whether or not the hash matches its block is easy, and once the nodes have done this they update their respective copies of the Blockchain with the new block. This mechanism makes the Blockchain tamper proof, or "immutable". Since, in CRAFT, each node has similar computational capability, it makes it impossible for one party to change the audit trail or events.

The general approach for interacting with the Blockchain in the Vault is to store data locally with the Blockchain only storing the hash of the data (anything used by CRAFT designers e.g. a design flow, IP, PDK, etc). Hence, it is sensitive material and as such, it is never stored on the Blockchain, it is kept on the local filesystem in the Vault. But because data maps to hashes, it provides signed non-repudiable proof that an event happened e.g. design flow has been updated, someone logged in, a script was run with this data etc. Copies of actual data are stored privately on each Vault using a write only filesystem.

In the Vault, we use Blockchain in two core areas:

- ***Design Flow Visualization and Editing***, which consists of a Web GUI for visualizing and editing the design flows and a backend API that stores design flow version hashes on the Blockchain (distributed ledger) and design flows themselves on a local filesystem (local to the vault).
- ***Collaborator Workflow Tracking***, which consists of a script wrapping approach and a backend API that stores verification/integrity data on the Blockchain (distributed ledger) and a local filesystem (local to the vault).

B. ASIC Design Flows

Each CRAFT performer team is developing a user-oriented version of their new design flows that can be used by DoD ASIC designers. As such, their representations are specific to their individual design approaches. Since the CRAFT repository bridges between the various flows and the target DoD design community, we have developed a way for designers to capture, document, store, and visualize such flows in a systematic and common methodology. Initially, we conducted several requirements gathering meetings (in-person, teleconferences, email exchanges, etc.) to create a high-level view of the design flow (no flow specific information was required at this point). Then, we extracted expert knowledge to expand the high-level flow into a complete running example flow, which included information about options, flow controls, and data. These interactions allowed us to iterate towards a common schema for documenting the flows.

One of the major capabilities that we wanted to demonstrate in this representation was the ability for the users to visualize and edit these flows, and perform flow validation (syntactically and semantically). We decided to describe and formalize the template for the design flows in JSON format, where the schema has three major sections, tools, files, and stages. The tools section describes a list of tools that are used within the flow and specifies tool functionality and specific options or configuration parameters. The stage section describes the individual steps of the flow, and includes its input and output files (which may be provided by external vendor tools), a tool (referenced by the its unique identification), and flow controls, which capture the decision flow based on the tool output. We provide the users two options for describing their flows: (*i*) upload a JSON document conformant to the schema—the repository UI automatically validates and identifies any errors in the document; or (*ii*) via the use of an interactive visualization tool, as presented in [3].

C. Collaborative Workflow Tracking

Many hardware modules (i.e., IP blocks incorporated in a hardware design) must interoperate with other hardware and software modules. Across multiple vendors, such interoperability is often ensured by adherence to industry standards (e.g., USB 3.0 for serial ports and DDR4 for process-memory interfaces). Even when a complex ASIC is designed within a single organization, various sub-teams need to coordinate the design to ensure interoperability. Such design coordination typically takes the form of the sub-teams agreeing to adhere to specific released versions of key IP modules. Since the CRAFT design flow also includes IP modules created using generators and other scripts and tools, it is also necessary for the design framework to include adherence to specific releases of scripts and tools during design and use. Further, achievement of desired design robustness may also require various sub-teams to adhere to specific releases of scripts and tools for verification of design's correctness, including adherence to industry/internal standards.

Currently, the CRAFT repository does not provide a mechanism to ensure adherence between produced outcomes and the scripts, tools and input data used to generate them. To address this issue in the Vault, we developed a system to provide the necessary generation and verification steps using cryptographic encryption. The generation step encrypts outcomes, i.e., output data, based on the scripts, tools, and input data used for generation. The verification step applies then the reverse operation, so that a user that would like to make use of an output can verify that it adheres to the versions/metadata of the scripts, tools, and input data used for its generation.

V. Implementation

A. Blockchain Implementation

In order to prepare the CRAFT Secure Vault for operations, we must generate an identity for the vault server itself. We then need to generate users in a global way (one single Vault account) but allow those users to access different vaults in different ways. We can achieve a number of these core goals by interacting with the ALADDIN framework.

ALADDIN and the CRAFT Secure Vault: We have developed a Blockchain service architecture, called ALADDIN (Any Ledger, Any Distributed Data using Intelligent Networks), which incorporates the following services:

- A common interface to different distributed ledger Blockchain systems;
- A common interface to different data stores;
- A conceptual model, using assets and transactions, that define the relationships between things that need to be tracked (assets) and how that tracking occurs (transactions) to feed into smart contracts;
- A Smart Contract Design Tool to help generate smart contracts automatically from the conceptual model, defined using a UI; and
- An auto API generator tool that converts the ALADDIN conceptual model to a Node.js REST API to help software integration with the Smart Contract.

Defining an ALADDIN application: Each ALADDIN application is defined using assets and transactions that are written for a specific Blockchain system, along with data (files) to be saved in a third party data store. We use the ALADDIN private Ethereum [7] network adapter for Blockchain, and IPFS [8] or MongoDB [9] for the data store. ALADDIN Data Bundles wrap data for an asset into an asset Manifest, which specifies data relationships using links to hashes that represent the off-chain data. We use assets to represent entities, (design flows, datasets, or scripts) and transactions to represent updates, or usages of those assets. Once assets and transactions are defined, ALADDIN converts this specification into a format that can be used with the Blockchain system by generating: (1) a smart contract (in Solidity for Ethereum); and (2) a custom business REST API that is generated for the assets and transactions that the user defined. This allows client applications and SDKs to be written directly against this REST

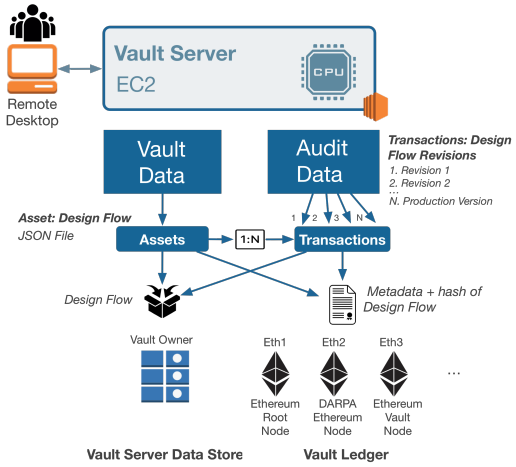

Fig. 3: A schematic showing how the design flow is stored onto the ledger and data store.

API, which in turn converts the calls into Smart contract calls, which in turn writes data to the distributed ledger network.

ALADDIN Tooling: Our approach has involved creating some flexible developer-based tooling to help generate backend Blockchain implementations and APIs, which includes a:

- **Smart Contract Design Tool** that can auto create Ethereum Solidity Smart Contracts from an asset (design flow) and transaction (change on the design flow) model.
- We use the same underlying model in our **API Generation tool**, which can parse a Smart Contract to create a Swagger OpenAPI specification [10] and then auto generate an ExpressJS [11] API in Javascript for interfacing with the Smart contract and Blockchain.

B. Design Flows

The CRAFT repository Design Flow mechanism is based on an EmberJS App [3], which talks to backend filesystem for storing the design flows. The repository also performs versioning, so the "time machine" uses this functionality to browse previous versions of the design flow. To enable this capability in the Vault we have:

- Redesigned the EmberJS App so it talks to a local backend;
- Created a new backend capable of storing the different versions of the Design Flows, which also connects to our common distributed ledger backbone, (save or update transactions are saved). These tools enable flow editing within the Vault system.

Here, an asset is a design flow and transactions are updates on this initial design flow. Then, when changes are made to that design flow, transactions on that original design flow are written along with the changes to it. This allows us to create a non-repudiable chain of transactions for such a flow representing versions of the design flow made by the users.

Fig. 3 shows this process in detail. The CRAFT Secure Vault server, shown on top, hosts the design flow application in the form of an EmberJS Web app. The App then talks to

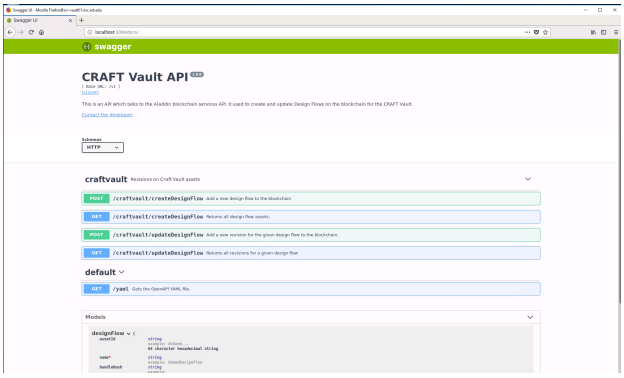

Fig. 4: The Design Flow API

the ALADDIN framework, which splits the request (a design flow) into the data component (the JSON design flow file) and a hash of the data. The Design Flow itself is stored in the data store which is local (private) to the vault itself and the hash is recorded on the blockchain to tie this event into the distributed ledger. If the JSON design flow changes then the blockchain is broken and the system will know that there was an attempt to tamper with the local data.

The API is used to interface with the blockchain for storing hashes of Design Flows. The data, i.e., design flows, are stored privately on each vault server, using a write only filesystem (IPFS). The purpose is to provide a non-repudiable audit of each version of a design flow, identifying the author at each stage. In order to port this to the vault environment, we have migrated the EmberJS App to an AngularJS/Blockchain API to work locally in the closed Vault secure environment – since the CRAFT repository backend has multiple outgoing connections to support its infrastructure. The local App communicates (locally) via a Blockchain ExpressJS [11] API, generated by our Smart Contract Design and API Generation Tools. The app provides an editor view and a screen that shows all versions of the design flow. A user can simply click the version they want to view and the app will update the JSON flow accordingly. Fig. 4 shows the backend API that interfaces with the blockchain. This shows two endpoints: one to create a design flow, which is used to initialize the app with the design flow that was imported from the CRAFT repository; and one to update Design Flow that allows revisions of this design flow by transacting Ethereum revisions on it.

C. Collaborator Workflow Tracking

For the design flow implementation, we developed the underlying blockchain implementation and implemented a prototype of a lightweight tool for supporting design flows within the Vault. In creating the API, we built tooling that allowed us to generically develop smart contracts and build APIs from them automatically, to support extensibility for the other Blockchain uses we have in CRAFT.

To track the designer's collaborative workflow, we use the same tooling to create a Blockchain API for addressing the Data security and integrity assurance task. Since the CRAFT design flow includes IP modules created using generators and

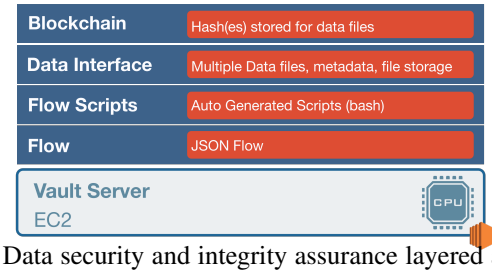

Fig. 5: Data security and integrity assurance layered approach.

other scripts and tools, it is also necessary for us to track specific releases of scripts and tools that were used during design and verification. In this task therefore, we set out to design a system that tracks a designer's use of the underlying IP modules, scripts, tools, and data so that we can record the specific versions or releases of scripts and tools that were used for verification purposes. Each tracking event is stored on a distributed ledger in AWS GovCloud for transparency, support and for audit purposes. Fig. 5 shows the general overview of the different layers of the approach.

Design Flows: A number of extensions were made to the design flow specification to enable realization of the execution of the design flow scripts. These modifications included the addition of file system locations specifying the physical paths of the IP, scripts, data, and so on. Essentially, we converted the passive design flow specification that existed previously and made them actionable, so that they could run within a command line environment. For each stage of the design flow, a Linux shell script is generated to automate (when possible) the steps involved in the flow generation.

Flow Scripts: The execution scripts mentioned above are organized into five steps: (1) creation of a blockchain session; (2) registration of the input data as transactions (a transaction is performed per input file); (3) execution of the flow's stage program (s); (4) registration of the output data produced by the previous step (a transaction is performed per output file); and (5) finalizing the session.

Data Interface: To connect the Flow Scripts to the blockchain, we designed a set of interfaces that could be used to track a particular design flow session:

- *Create Session:* starts a working session to be tracked. On the blockchain, we create a transaction that identifies a new thread of transactions. We call this an asset (running design flow) that we wish to track. Once the asset has been registered, transactions can be written against it.
- *Transaction:* records a transaction on the blockchain using the address to the asset, registering a file hash and metadata about the file being used. The file hash is a fingerprint (i.e. a SHA2 hash) of the contents of the file being recorded.
- *End Session:* ends the session being tracked. Once a session has ended, it cannot be reopened, but a new session can be started.

Blockchain: The data Interface interacts with:

- *The Blockchain.* To write to the Blockchain, the Node.js script interfaces with the API that was generated to the Ethereum Blockchain for this application. The API is basically a file oriented API that allows the application to specify the filename, its file hash, and other metadata.
- *A Data Store,* which is responsible for the storing of data files that are used by a designer.

The CRAFT Blockchain API is based around a Web API concept. It contains adapters to the Blockchain and it provides interfaces to connect to IPFS [8], which is a lightweight, distributed, versioned, Web-based file system, designed specifically for blockchain use. To cope with large design files (> 15 GBs in some cases), we redesigned the approach to interface through the file system rather than the Web based interface. To achieve this, we implemented an adapter that manages file usage on the local file system. The scheme, first generates a hash of a file that a designer wants to register. It then checks to see if this file is already being tracked. If so, then a copy is not stored again, because the hashes already matches a file that has been stored in the file system. If the file has not previously been tracked then it is copied to the file system and the hash code is used as its filename, for future checking. The resulting hash is then passed to the Blockchain adapter to be stored on the Blockchain. For the most part, only the hashes are stored, to track usage and so the approach is scalable with respect to the file space required for tracking. A shortcoming of this approach is the time it takes to generate the hash. To generate a SHA2 hash of a 20GB file can take more than two minutes. We are investigating solutions to address this issue currently.

VI. Conclusion

This paper covers the transition of the CRAFT repository, originally designed to be a repository for users to discover and document ASIC design flows to a fully secure environment called the CRAFT Vault hosted in Amazon Gov Cloud that allows designers to collaborate and actually implement these flows. The Vault is a fully configured environment that is deployed with all the necessary Electronic design automation tools and IP making it easier for end users to implement a CRAFT Design Flow. In addition to the technical challenges, we had to invest a fair amount of effort to get the legal agreements in-place for hosting the tools and providing necessary licenses for our users, which re-iterates one of the motivating reasons for developing the Vault in the first place.

Acknowledgments. This work was funded by DARPA under contract #HR0011-16-C-0043 Repository and Workflows for Accelerating Circuit Realization (RACE).

References

[1] "Cadence VCAD," https://www.cadence.com/content/cadence-www/global/en_US/home/services/vcad-services.html.
[2] "DARPA Circuit Realization at Faster Timescales (CRAFT) program," https://www.darpa.mil/program/circuit-realization-at-faster-timescales.
[3] A. Brinckman *et al.*, "Collaborative circuit designs using the craft repository," *Future Generation Computer Systems*, vol. 94, pp. 841–853, 2019.
[4] "Ericom," https://www.ericom.com.
[5] "AWS GovCloud," https://aws.amazon.com/govcloud-us.

[6] M. Swan, *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.
[7] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.
[8] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.
[9] "MongoDB," https://www.mongodb.com.
[10] "Swagger OpenAPI Specification," https://swagger.io/specification.
[11] "Express.js," https://expressjs.com.