BitVM



BitVM - Smarter Bitcoin Contracts - BitVM

BitVM/BitVM

- bitvm/acc
  - BitVM/accレポート 2025年6月
    - Gemini Research
    - ChatGPT

BitVM awesome

~~BitVMの解説｜株式会社Crypto Garage公式~~ BitVM1の内容だと思う

~~Bitcoin上で任意の計算の検証を可能にするBitVM - Develop with pleasure!~~ BitVM1

BitVM 2 - Develop with pleasure!

BitVM2ブリッジの制約(NotebookLMより)

BitVM - ChatGPT

概要

プログラミング言語でプログラムを書くと、コンパイラが CPU の理解できる機械語に翻訳し、機械語は CPU がそれに従って内蔵する電子回路 / 論理演算を動かしていくことで実行される。つまり、最終的に論理演算で実行しているのだ。

BitVMはそういう CPUが基本的な半導体の論理演算(NAND)で構成されていることから発想を得ている。すなわちスクリプトを論理演算レベルまで分解して電子回路のようなものを作る。

ただそうすると数十億くらいの回路になってBitcoinスクリプトにできないためオフチェーンでやりとりをする。Bitcoinスクリプトとしてはその回路を知っていないとできないようなデータを使ってスクリプトを解くか、1週間過ぎたら取り戻せるかになっている。

実装よりも理論がメインな感じがする。そしてBitVM/accに参加しているプロジェクトが実験したり実装に使ったりしているように思う。参加していると先行して情報が入手できていそうだ。参加していなくても公開されたレポートPDFは読むことができる。

2025年10月

BitVMリポジトリは Commit d2a4bb4 が最新
コミット履歴からすると 10月9日から活動が再開しているようだ。これが BitVM3 関連なのかどうか不明。Pull Request がマージされただけのように見えるし、内容も新規ではなく修正がほとんどのような気がする。

2025年7月

BitVMリポジトリは Commit b822f19 が最新

> 現時点（2025年5月）では、初代BitVM（2023年発表、Robin Linusの論文によるもの）は事実上、開発停止状態（＝de facto deprecated）です。現在はBitVM2が主流であり、開発・議論・実装はすべてBitVM2に移行しています。

- BitVM1 はおそらく2024年3月末が最後(README)
  - BitVM2 もその commit から続いているので明確な区切りはないのかもしれない
  - 半導体回路をベースにした考え方というものは引き継いでいるのだろう

できること

- チューリング完全なビットコイン契約を実現することができる
  - ループ文でぐるぐる同じ処理を繰り返したりできる
  - 普通のプログラミング言語みたいに書ける、くらいに思っておけば良い
- 今のBitcoinの仕様で実現できる(`OP_CAT` だったりのソフトフォーク無しで使える)

マイナスな点

- ~~(BitVM1)契約は2-party間に限られる~~
- ~~(BitVM1) 2 partyには多くの計算量およびインタラクションを要する（1つの契約を実行するにあたりお互いが交換および保持する情報は数百MB〜数GBになる）~~
- まだ高級言語で書くことはできない
- 制限事項はいろいろある
  - 信頼性
    - 少なくとも1人の正直なオペレーターが必要
    - 署名者委員会によるカベナンツのエミュレーション
      - Bitcoinスクリプトではまだできないのでエミュレーションしているのだが、オフチェーンでかなりの通信ややりとりが発生する
  - オンチェーン
    - 大規模な非標準トランザクション
      - 不正ChallengeにタイするAssertトランザクションやDisproveトランザクションはサイズが大きい
      - その分、手数料もかかる

- 運用
  - 悪意ある Prover による手数料の窃盗の可能性
  - オペレーターによるペグアウト時の先行資金提供
  - デポジット額の固定("Fixed deposit amounts")
  - 経済的なライトクライアントのセキュリティ
  - fraud-proof期間が長い
    - 2〜4週間(BitVM3では短縮できるらしい)
- 技術・設計
  - ライトクライアントの複雑性
  - サイドシステムライトクライアントの複雑性
  - 回路規模の制約
  - 「チャレンジ」DoS攻撃
  - ネイティブなOP_BLOCKHASHの欠如
    - 今は `OP_BLOCKHASH` みたいな命令がないので複雑になっているところがある
  - 将来のビットコインオペコードへの依存
    - 制限というか、 `OP_CAT` などが採用されると設計がよくなるという話

BitVM2

https://bitvm.org/bitvm2

bitVMとbitVM2 - ChatGPT

最初の BitVM(便宜的に BitVM1とする)は2025年5月の段階ですでに deprecated で、現在は BitVM2の開発になっている。

BitVM2は汎用なプラットフォームで、デモになっているのは派生した BitVM bridge である。

質問するときは BitVM2 なのか BitVM bridge なのかを明示しないと混乱することになるだろう。

BitVM2 は Layer2技術や ZKP技術とは直接関係ない。

現在の仕様の Bitcoin スクリプトでチューリング完全なプログラムを実行しようとしているだけである。

その動かそうとしているプログラムが ZKP の場合が多いというだけである。

BitVM1 は 2-party 間に限定されていたが、BitVM2 は誰でも検証できることを目指している。

BitVM2はオプティミスティックモデルで、後から異議申し立てがあればチャレンジ期間中にチャレンジするしくみ。

- 1. オフチェーン実行(重たいのでオンチェーンでは無理)

- 2. オンチェーンコミットメント
- 3. オンチェーン検証/紛争解決
  - 異議申し立ての「チャレンジ」

bitvm/acc の2025年2月レポートによると、BitVM2 で zkSNARK verifier(Groth16) を Bitcoinスクリプトにすると 1GB くらいになり、1tx = 400KB とすると 2500 txs くらいになるという。
1GB というのはオフチェーンで実行する部分のサイズで、オンチェーンでは最短 3トランザクションの展開とのこと。

bitvm-bridge.pdf は BitVM2も含んでいる。

> BitVM2は、ビットコイン上で任意のプログラム実行を可能にする新しいパラダイムであり、チューリング完全な表現力とビットコインコンセンサスのセキュリティを兼ね備えています。その核心において、BitVM2は、オペレーターが正直であると仮定し、チャレンジャーが不正証明を通じて不正を証明しない限りはその仮定が維持されるという楽観的計算を利用しています。また、SNARK証明検証スクリプトは、ビットコインのトランザクション内で実行されるサブプログラムに分割されます。その結果、BitVM2はわずか3つのオンチェーン・トランザクションでプログラムの正確性を保証します。BitVM2は、これまでのBitVM設計を大幅に改善し、初めてパーミッションレスな異議申し立てを可能にし、紛争解決に必要なオンチェーン・トランザクションの複雑さと数を削減しました。私たちの構成は、ビットコインのコンセンサス変更を一切必要としません。BitVM2は、ビットコインにおける全く新しい種類のアプリケーション設計を可能にします。私たちは、BitVMBridgeを提示することでこれを実証します。BitVMBridgeは、預金の安全性のための信頼仮定を、セットアップ時の正直な多数派（t-of-n）から存在論的正直さ（1-of-n）へと減らすことにより、従来のビットコインブリッジを強化するプロトコルです。活性を保証するためには、アクティブで合理的なオペレーターが1人いれば十分です（他のオペレーターが悪意を持っていたとしても）。どのユーザーでもチャレンジャーとして機能でき、プロトコルのパーミッションレスな検証を促進します。

bitvm-bridge.pdf には Light Client の概念が出てくるが、おそらく現状のリポジトリには含まれていない。そのため PDF にある ClaimLC transaction が存在せず、いろいろわかりづらくなっている・・・と思われる。
(Bitlayer や Citrea は実装しているとのこと。)

BitVM2 についていわれる「3トランザクション」というのは

solution

> Naive Solution https://bitvm.org/bitvm2#naive-solution

SNARK verifier は 20MB くらいの Bitcoinスクリプトになるが、もちろんそのサイズではBitcoin

のトランザクションとして展開できないので Lamport 署名というものを使って、1つのプログラムを複数に分割することを考える(Naive Solution)。
そうした場合、当然最後まで終わらないと1つのプログラムが完了しない。それまでトランザクションを展開する手数料などがかかり続けるなどの理由で、まだオンチェーンで実行するには無理がある。

Balanced Solution https://bitvm.org/bitvm2#balanced-solution

そこで SNARK verifier をそのままオンチェーンで実行するのではなく、まとめてやることにする。

Google Docs

元々は $f(x) = y$ で、これを Naive Solution で $f1_(x) = z_1$, $f_2(z_1) = z_2$,..., $f_{n-1}(z_{n-2}) = z_{n-1}$, $f_n(z_{n-1}) = y$ に分割し、それぞれの中間状態 z_k に不正があると気付いたらチャレンジするという考え方だった。
Balanced Solution は中間状態の z_1 , z_2 ,..., z_{n-1} を一度にコミットする。

- 一度にコミット：Operator などが特定のデータについて、そのデータの正確性を後から検証可能にする形で、オンチェーン上にそのデータの「証明」を公開することを意味する。つまり、verify 自体はオンチェーンで行わないことでサイズを削減しようとしている。

セットアップ時、 f_1 , f_2 ,..., f_n の計算不正を証明するためのスクリプトを含む 1つの TapTree を定義する。verifier は不正の申告をするのにどれか 1つの leaf についてだけ行えば良い。
不正な場合(assertionが成り立たない場合)は2ラウンドで完了する。

- 全体で 20MB あったりするが、チャレンジして TapScript を解くときはツリーのどれか 1つしか通らないのでトランザクションに収まるということでよいのか。BitVM2では 400KB くらいになるようにしているそうだが、Bridge のデモで一番大きいトランザクション(assert_commit_1)は 2.7MB くらいだった。
- $f_k(z_{k-1}) == k_k$ のことを assertion と呼ぶみたいだ
  - bridge の assert_commit_1 / 2 もそこから来ている名前？
- 2ラウンドというのは 2トランザクションか？
- 名前の "balanced" は何に対してかよくわからなかった。Tap Leaf のサイズ？

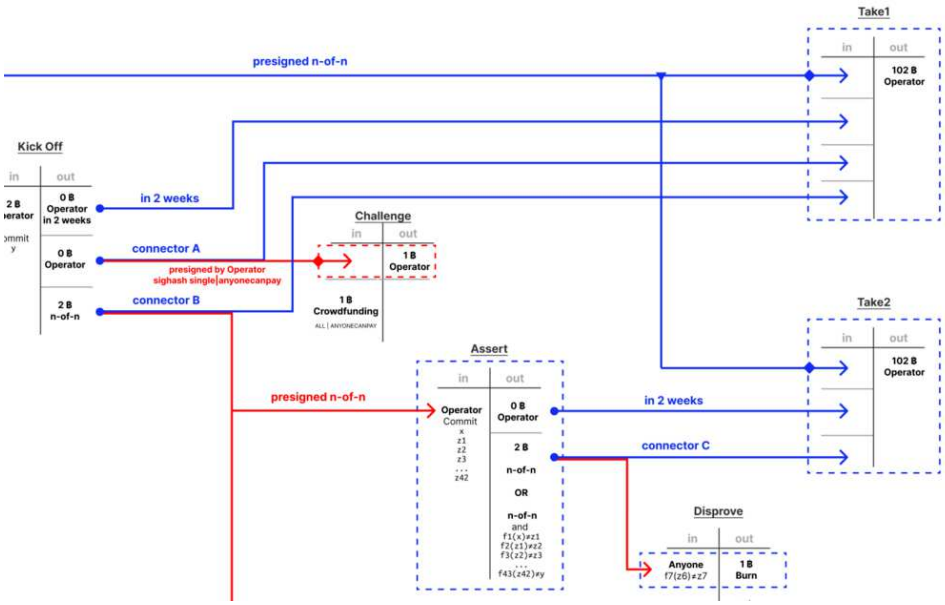
Optimistic Solution https://bitvm.org/bitvm2#optimistic-solution

さらに、これで不正がないケース("happy path")の改善のための解決策が導入された。それが Optimistic Solution。
これによって最悪ケースでは＋2ラウンドになるが通常時の効率は上がるとのこと。
まず、最終結果である `y` をコミットする。それが不正であれば誰でも良いのでチャレンジする。チャレンジされたら Operator はBalanced Solution と同じように中間結果 z_1 ,..., z_{n-1} を含んだ Assert Transaction をコミットする。 f_k に不正があれば気付いた人が証明(disprove)す

る。

challenge

この図からすると、challenge しないなら kickoff transaction の connector A と B は take_1 transaction への INPUT になるが、challenge するなら connector A を INPUT にし、connector B は assert transaction への INPUT になる。

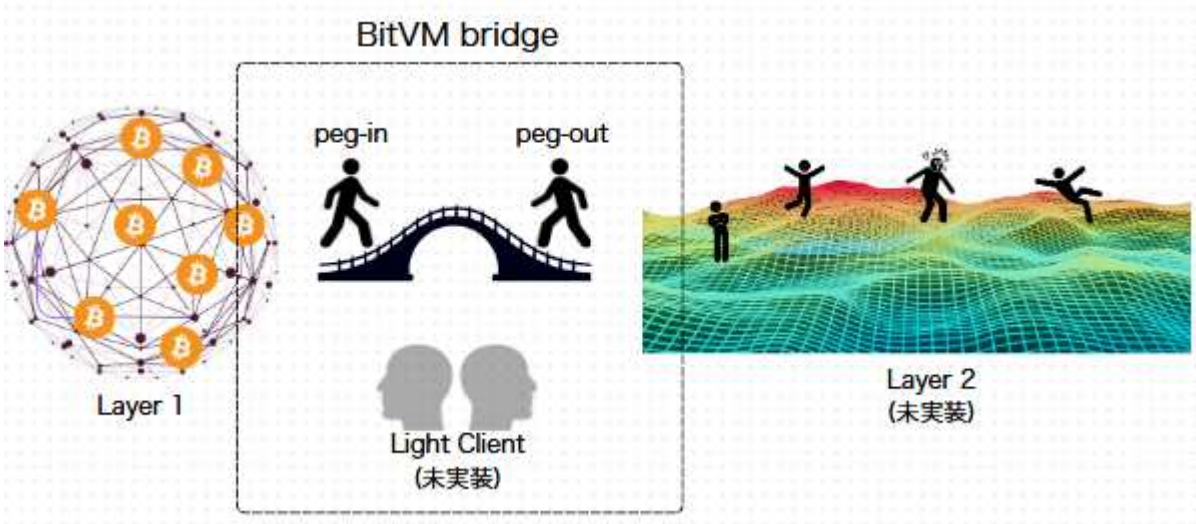


bridge

2025年6月現在、BitVM2はBitcoinと他チェーンの間のtrust-minimizedなブリッジ向けに実装(BitVM bridge)をしている。

理由は、その用途が多いことと、BitVMの構造としてやりやすいから。

L2 プロジェクトで BitVM2 を使っていると言った場合、bridge をそのまま使うパターンはあるのだろうか？



BitVMリポジトリは BitVM2 のしくみと、それを使った BitVMブリッジのコードがある。
bitvm-bridge.pdf にはブリッジとして必要な要素が書かれているが Light Client は載っていないのでこのしくみを使いたい人が自分で実装する。

BitVM bridgeのデモ

デモが動いたと公表した頃のものと最新のcommitの2つを試したが、どちらもPDFにあるトランザクションフローと若干異なる。

- BitVM bridge のデモ v0.1.0-alpha
  - ChallengeなしでAssertできる
- BitVM bridgeのデモ v0.1.1-b822f19
  - 実装に Light Client がない

現在のリポジトリには以下を含んでいる。

- Groth16 verifier のGraph生成・分割（＝ZKP検証用）
- チャレンジ型実行モデル（Fraud-proof型Bridge用）
- Playground / Graphビジュアライザー
- Kickoff / Assert / TakeトランザクションCLI群
- Light Client は実装していない
- Peg-In についてはある程度トラストが必要(デモではDEPOSITORという単独アドレス)

Light Client

いわゆる Bitcoin のブロックチェーンを参照する、フルノードではないという意味の Light Client のことでよいようだ。
peg-in 方向と peg-out 方向がある。peg-in は L2/bridge が L1状態を検証するし、peg-out は L2/bridge で Burn できているかなどの検証もやることになる。

- Gemini Deep Research

- How to integrate - Bitlayer
  - ターゲットチェーン(追加するL2)用のLight Client(Rust + zkVM)
    - L2の状態を正しく読み取ることをZKPで担保している
  - ターゲットチェーン上のブリッジコントラクト(Contract B)内の Bitcoin Light Client
    - SPV証明を検証できる Bitcoin Light Client を対象のスマコンに組み込む
    - イメージとしては `bitcoin-cli gettxoutproof` で取得したデータ(inclution proof(merkle branch))を検証する `bitcoin-cli verifytxoutproof` のようなもの
      - 1. トランザクションの inclusion proof を作成
      - 2. inclusion proof と TXID を BitVM2 verifier に渡す

- 3.

BitVM3

2025年6月にPDFが公開された (2025/10/16：←は BitVM3s に差し替えられている)

Abstract を翻訳してもらう。

> BitVM3は、ビットコイン上でSNARK証明を検証するためのプロトコルであり、前身であるBitVM2のオンチェーンフットプリントを大幅に削減します。ガーブル回路を用いた楽観的計算を活用することで、BitVM3は検証の負担をオフチェーンに移行します。この設計により、評価者は紛争発生時にコンパクトな不正証明を生成できます。結果として得られるオンチェーントランザクションは非常に効率的です。アサーショントランザクションは約56KB、反証トランザクションはわずか200バイトで、紛争発生時のオンチェーンコストを以前の設計と比較して1,000倍以上削減します。

Conclusionも翻訳してもらう。

> BitVM3は、RSAベースのガーブル回路を用いてSNARK検証をオフチェーンに移行することで、ビットコインの契約機能を大幅に向上させます。このアプローチにより、オンチェーンフットプリントはアサートトランザクションで56KB、反証トランザクションで200バイトに削減されますが、オフチェーンに数テラバイトのデータ設定が必要になります。このトレードオフにより、ロールアップやサイドチェーンといった第2層において、より資本効率が高く、信頼度を最小化したブリッジが可能になりますが、今後の取り組みではオフチェーンのデータ負荷軽減に重点を置く必要があります。最終的に、BitVM3は、ビットコインを任意の複雑な計算のための安全な決済層として利用するための現実的な道筋を示しています。

2025/10/16追記

最初の段階では各ノードに 5TB 程度のダウンロードが必要になる(「数テラバイトのデータ設定が必要」の部分)ということだったが、その後 BitVM3s に差し替えられた。

5TB が 280GB に削減されている。

garbled circuit / ガーブル回路 / ガーブリング回路

ChatGPT

論文では "garbled circuit" (garble は「歪曲する」で、これが暗号化すると意味でも使われる)だった。

日本語ではよくわからないが「ガーブリング回路」が伝わりやすい。

英語では "garbled circuit" でよいらしい。

ガーブリング回路（garbled circuit）とは、安全な二者間計算（2PC, two-party computation）を実現する暗号技術の一つ。

信頼できない二者が互いに秘密を保持したまま、ある関数 $f(x, y)$ の結果だけを正しく得るための手法である。
ブール回路C を知っている側が C を暗号化してガーブリング回路を作る。
評価側は回路C ではなくガーブリング回路を使って暗号化された入力を受け取って結果を計算する。この結果がそのまま回路C の結果ではなく、対応表を使ってようやく 0 or 1 がわかる。