BitVM bridgeのデモ v0.1.1-b822f19

BitVM のこれを書いている時点での最新 commit でデモを動かす。

https://github.com/hirokuma/BitVM/tree/demo/b822f19

実行ログ1(challenge無し)

実行ログ2(challenge無し + starttime)

`v0.1.0-alpha` と違い、デモ用ではなくなっている。

BitVM bridge のデモ v0.1.0-alpha にあったサブコマンド `assert_commit_1` と `assert_commit_2` がなくなっている。その代わり `assert_commits` というコマンドにまとめられたような感じがする。

デモの手順では、問題がない方は `kick_off_2` のあとに `assert_initial` ～ `assert_final` 、 `take_2` という手順になっているが、実際は challenge 無しで assert フェーズを実行することはない。最新 commit もそのようになっているので `kick_off_2` の次は `take_1` で Operator が手間賃を取り戻している。

この方が展開するトランザクション数やサイズが小さくて済む。

チャレンジする場合はトランザクション数が増えてしまうが、

ビルド

データストアをローカルファイルでテストするので `--release` ではなくデバッグ版(オプション無し)で `cargo build` する。

リリースビルドした `bridge` だとエラーになる(ローカルファイルに対応するのは `v0.1.1-dev` 以降)。

環境

ローカルファイルを使うようにしたので BitVM bridge のデモ v0.1.0-alpha よりは楽かもしれない。

https://github.com/hirokuma/BitVM/blob/demo/b822f19/DEMO_INSTRUCTIONS.md#environment-setup

想定するトランザクション

BitVM2の説明ページでは kickoff transaction の vout は 3つになっている。

https://github.com/BitVM/bitvm.github.io/blob/main/bitvm2.md#permissionless-bridge-design

これだと prover が手数料をかすめ取ることが出来てしまうらしいので、少し変更が入ってい

る。

Challenge と Assert が n-of-n Operator から支払うようになっている。

https://github.com/BitVM/bitvm.github.io/blob/main/bitvm2.md#limitation-fees

不正な peg-out があった場合でも対抗できるように、最低でも1人の真実を告げる OPERATOR が必要である。もし peg-out したのが不正であっても、正直な OPERATOR がいれば申告できるということである。

https://github.com/BitVM/bitvm.github.io/blob/main/bitvm2.md#limitation-honest-operator

全体のトランザクション図がわからないが、これか？ peg-in/out confirm や assert のトランザクションが不足しているがおおよそ正しい感じがする。

permissionless_bridge_spv_v4.png

図に出てくるトランザクションは bitvm-bridge.pdfがベースになっているか。Figure 7 などが結構近しい感じがする。

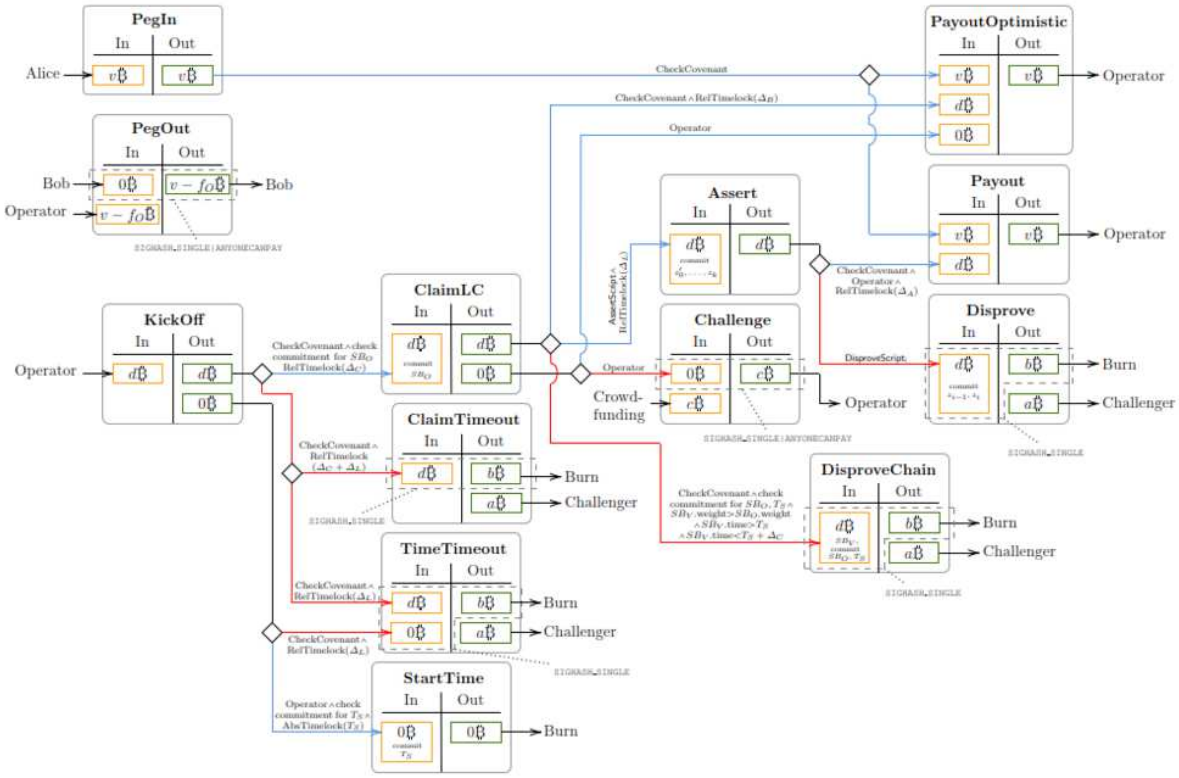


Fig. 7. Transactions of the BitVM Bridge protocol. The transactions **ClaimLC**, **PayoutOptimistic**, **Assert**,

Rejected Disprove Scenario (a.k.a. 'happy peg-out' execution path)

https://github.com/hirokuma/BitVM/blob/demo/b822f19/DEMO_INSTRUCTIONS.md#rejected-disprove-scenario-aka-happy-peg-out-execution-path

"happy path" はチャレンジ不要な場合である。

いろいろ実装を調べたが、 `v0.1.0-alpha` とコマンドが違うのは最初に述べたが、そもそも challenge する関数はあるもののコマンドは用意されていなかった(関数が誰からも呼ばれていない)。

実際にやってみると、kick_off_1 transaction は vout が 3つあったが、kick_off_2 transaction は 2つであった。これは `v0.1.0-alpha` も同じである。
手順にない `starttime` コマンドを使うと Start Time transaction というものを発行するが、これは kick_off_1 の vout から延びた。
正常系に関しては全体のトランザクション図(青い線)と一致する。

