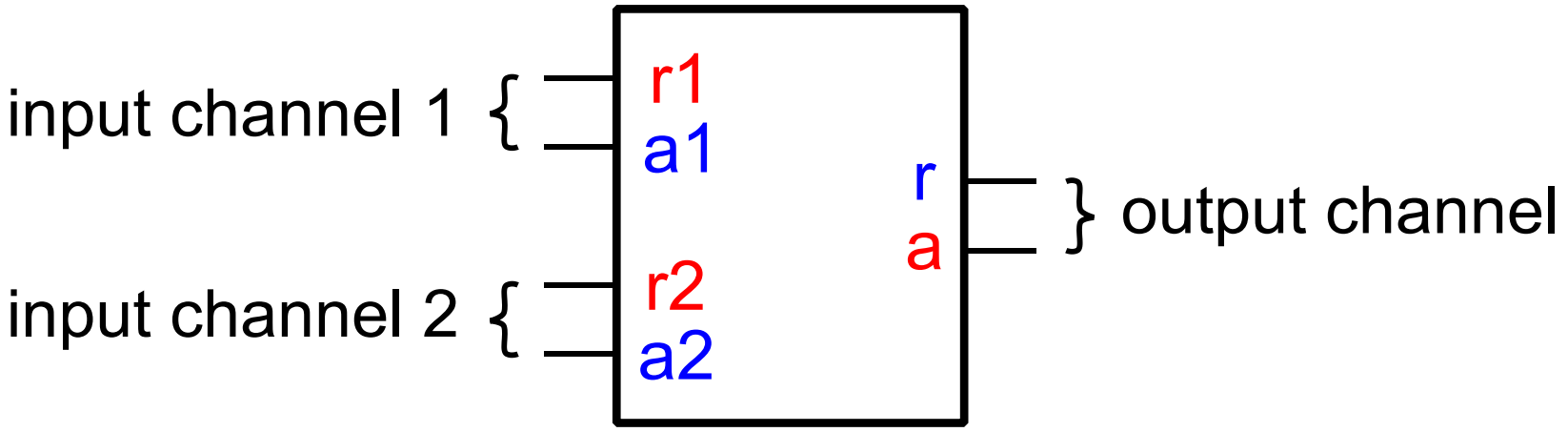
Opportunistic Merge Element

Andrey Mokhov, Victor Khomenko,
Danil Sokolov, Alex Yakovlev

Merge Element



Purpose: merge independent requests

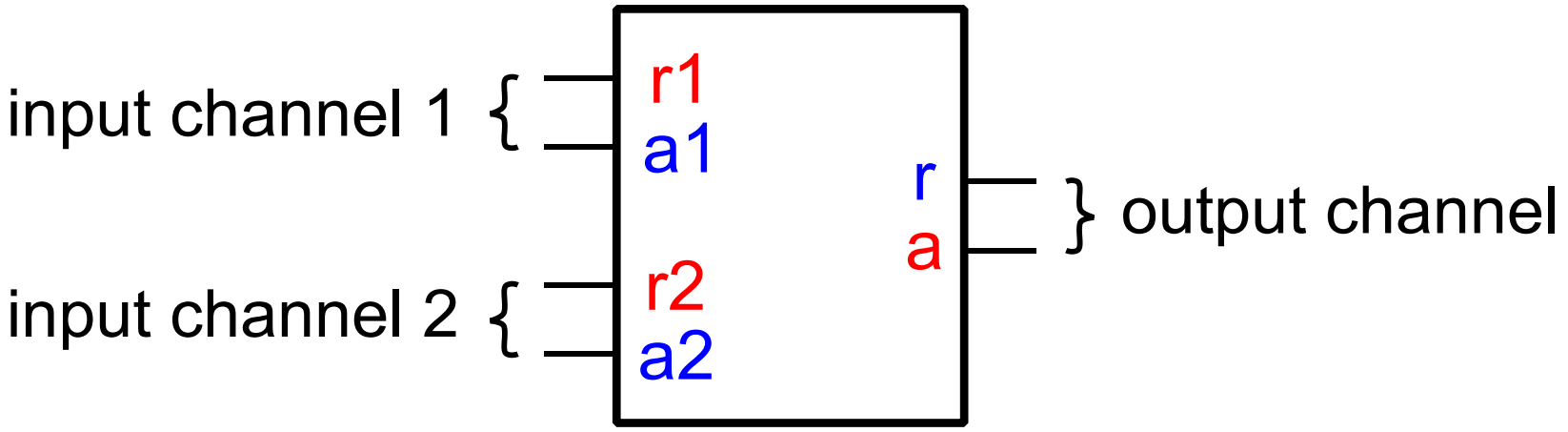
Example: count the total number of requests

Property: requests are never lost, $\mathbf{I_1 + I_2 = O}$

Requires arbitration

- between requests
- better outside the critical path

Opportunistic Merge Element



Purpose: merge independent requests, *bundling closely arriving requests together*

Example: respond to an alarm (two sensors)

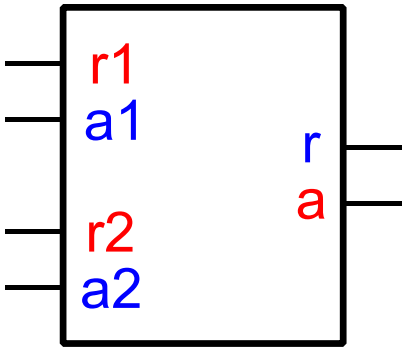
Property: $\mathbf{max(I_1, I_2) \leq O \leq I_1 + I_2}$

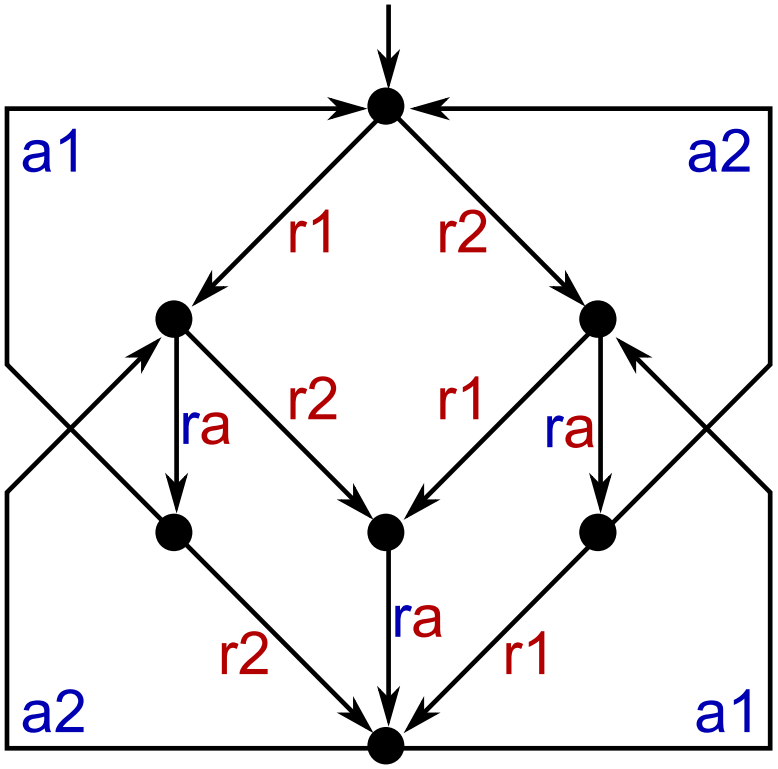
OMs in the real world

Our motivation:

on-chip power management

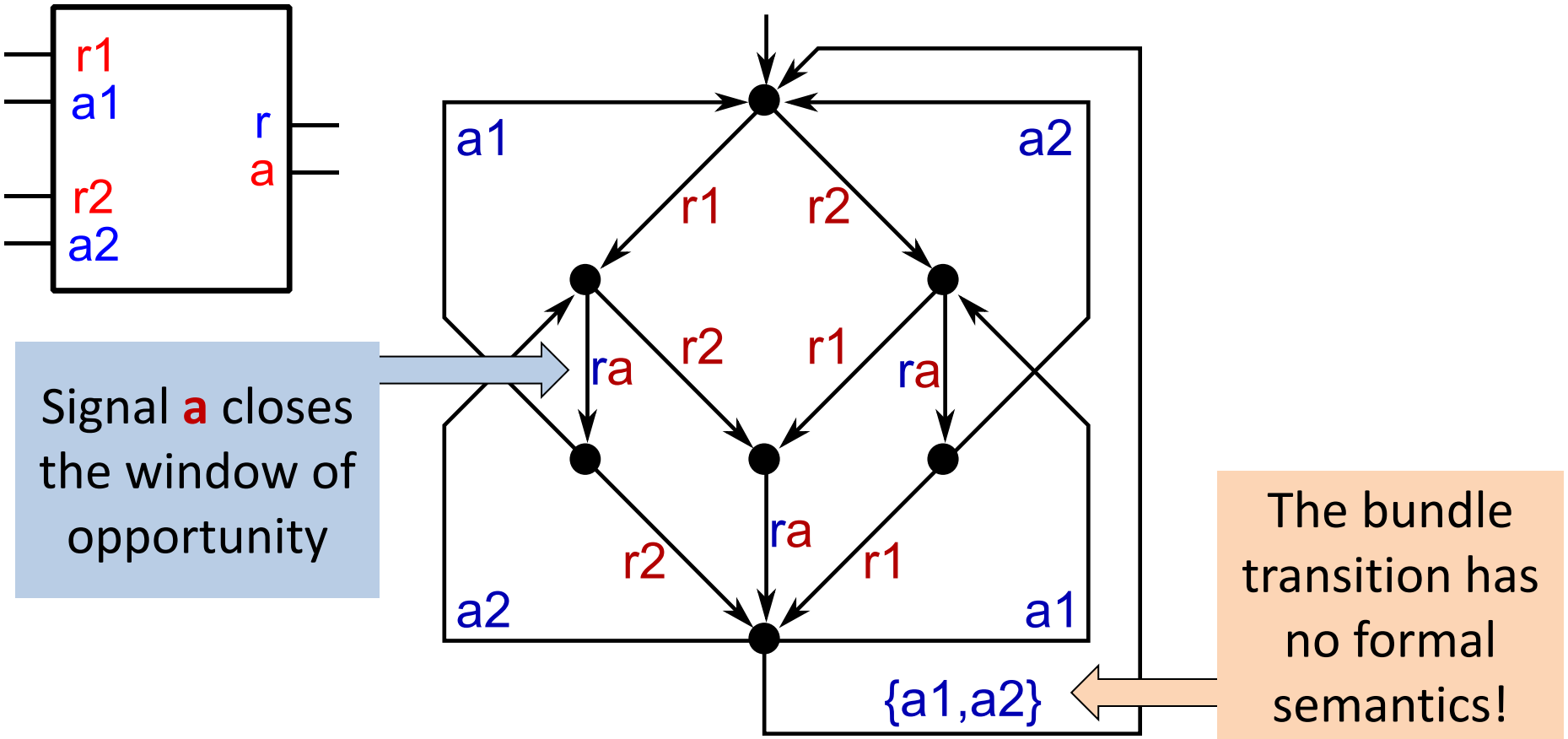
Conceptual specification





Merge

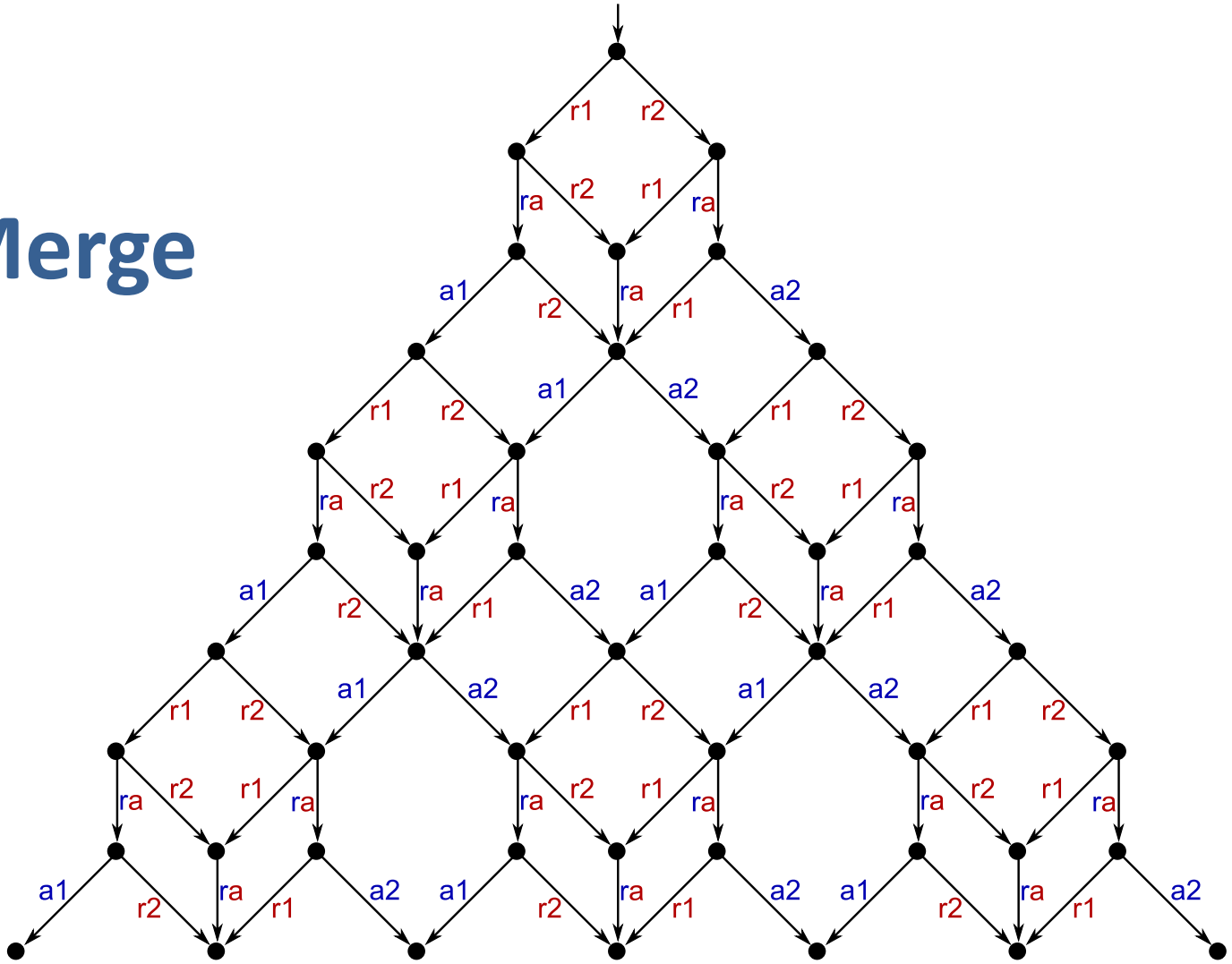
Conceptual specification



OM

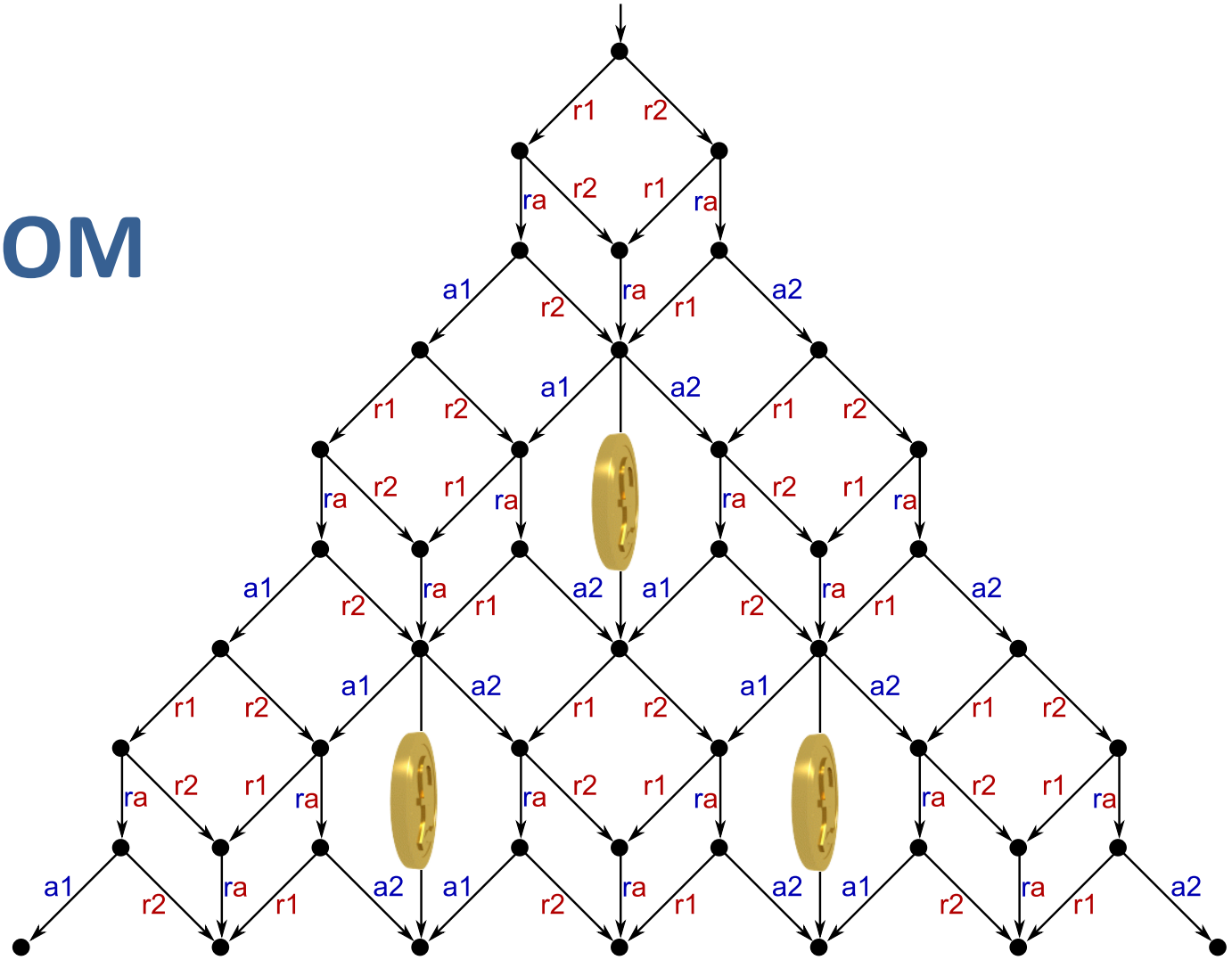
Conceptual specification (unrolled)

Merge

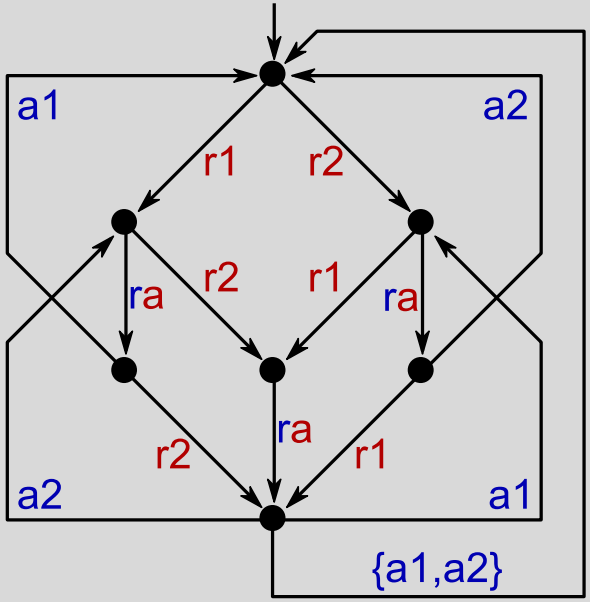


Conceptual specification (unrolled)

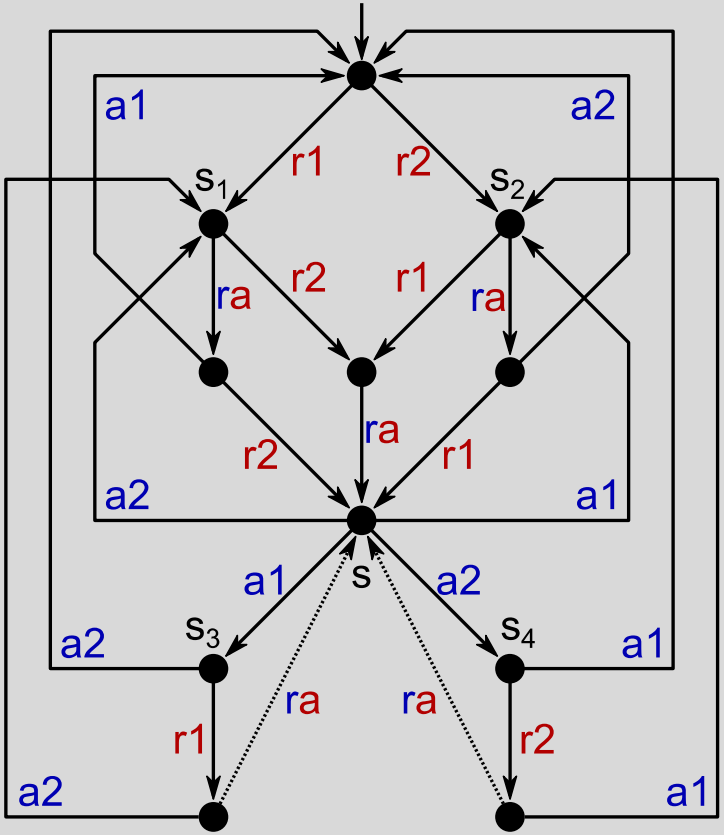
OM



Decomposing the bundle



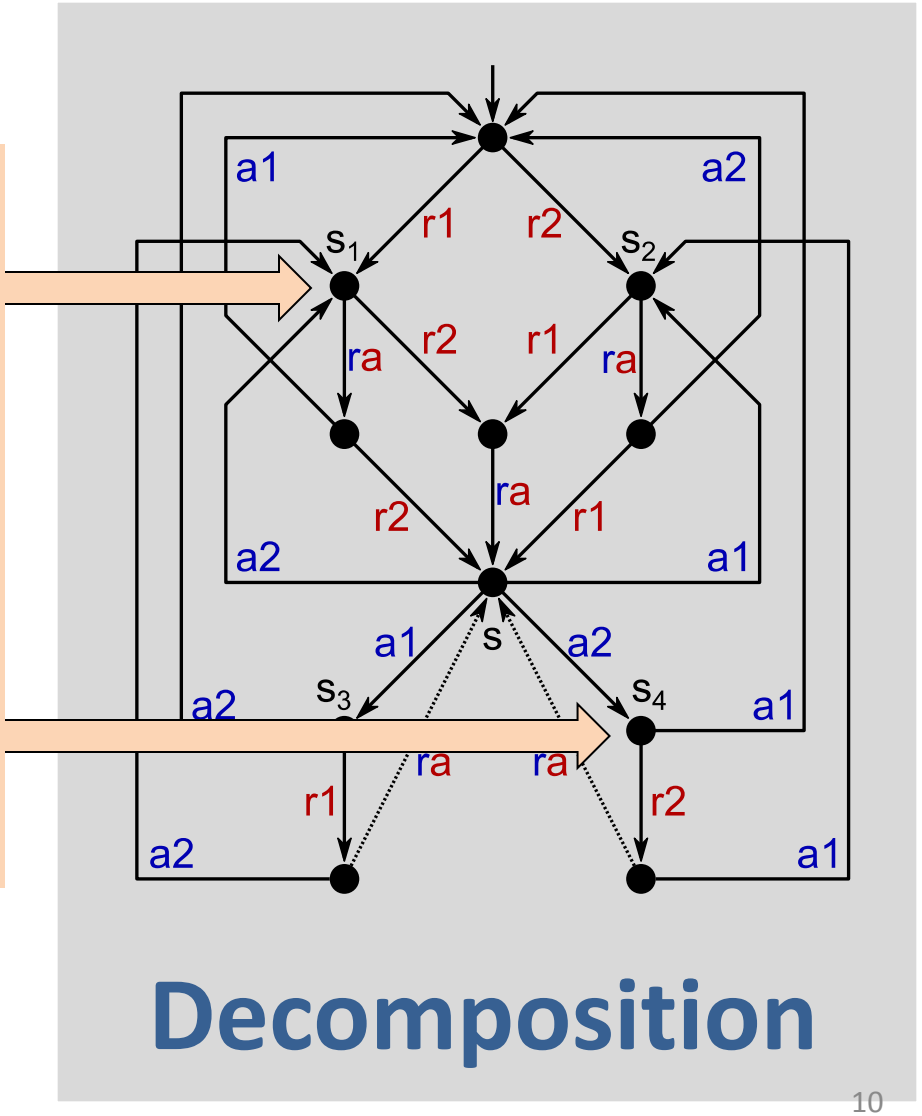
OM with bundle



Decomposition

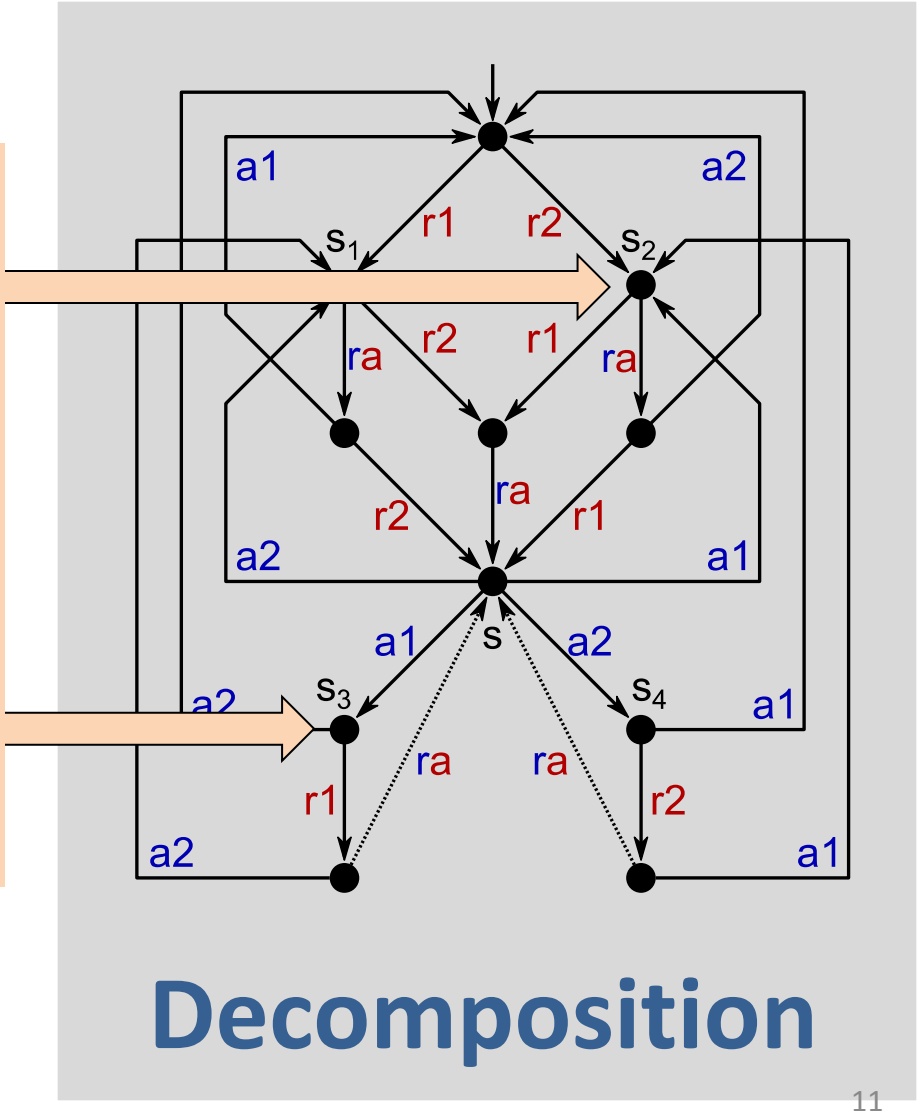
Decomposing the bundle

Problem: decomposed specification cannot be synthesised due to *irreducible state encoding (CSC) conflicts* between $\mathbf{s_1}$ and $\mathbf{s_4}$, and between $\mathbf{s_2}$ and $\mathbf{s_3}$



Decomposing the bundle

Problem: decomposed specification cannot be synthesised due to *irreducible state encoding (CSC) conflicts* between $\mathbf{s_1}$ and $\mathbf{s_4}$, and between $\mathbf{s_2}$ and $\mathbf{s_3}$



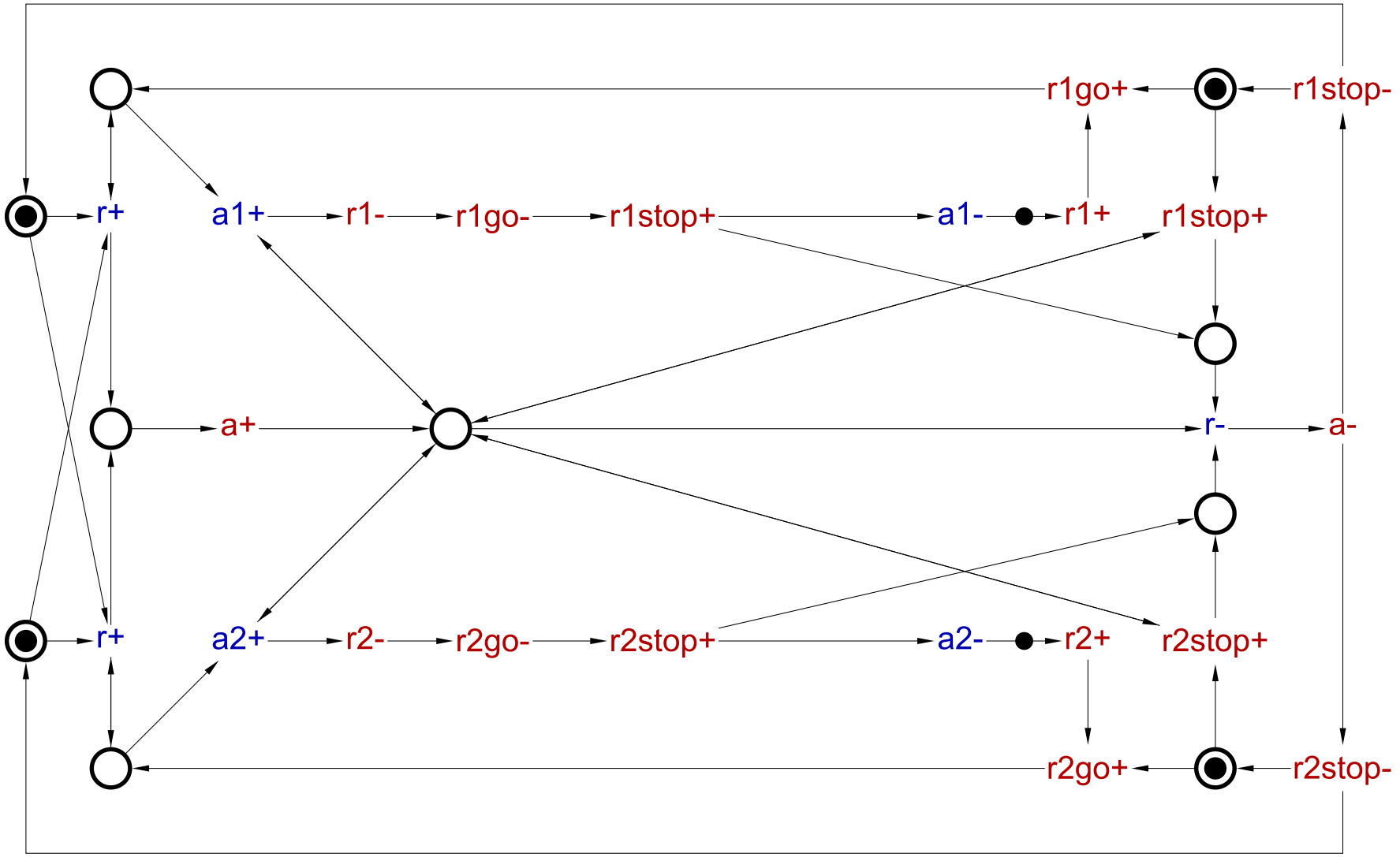
Is this a dead end?

Decomposing the bundle **{a1,a2}** is highly non-trivial:

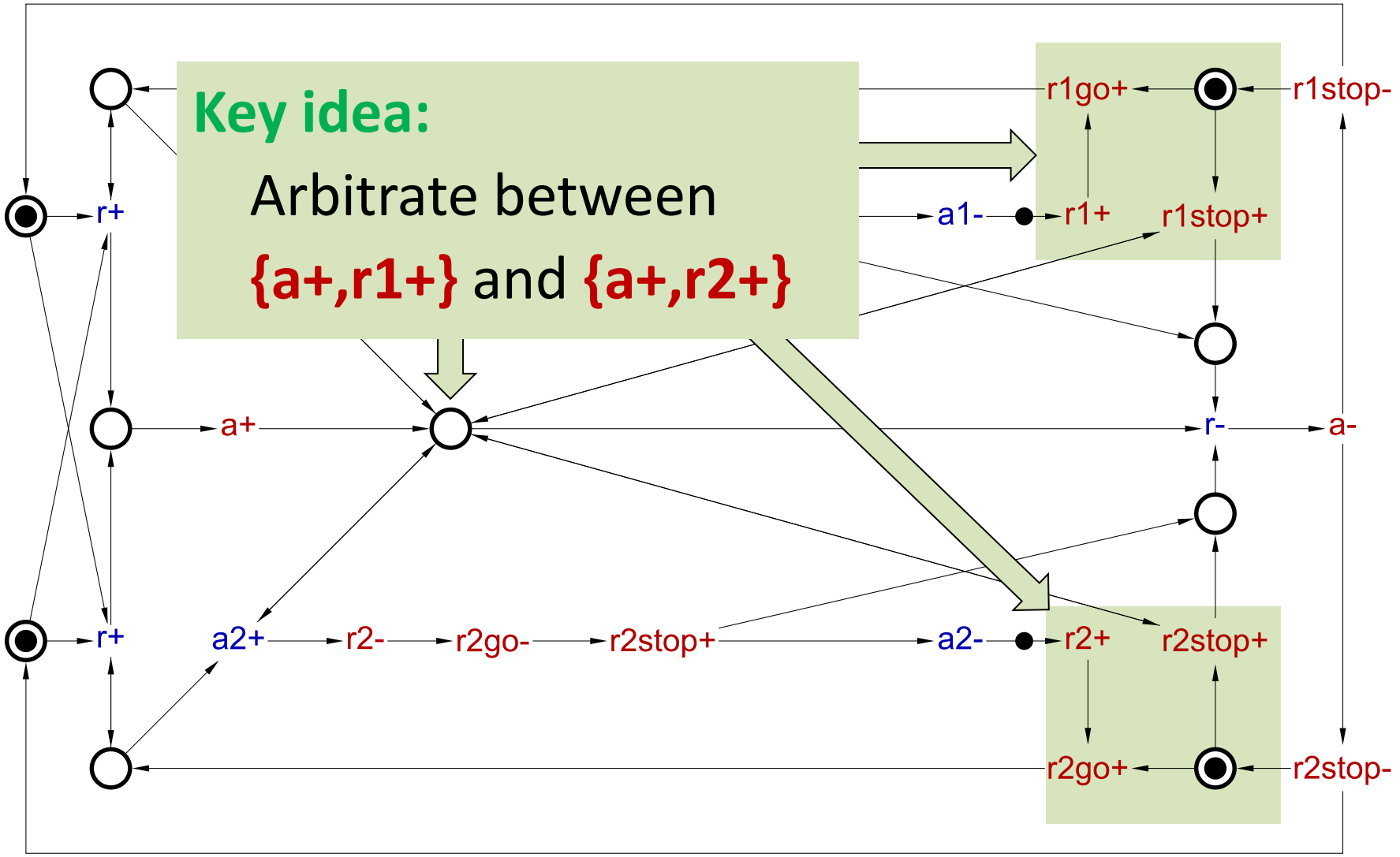
- Output-determinacy violations
- Non-commutativity of inputs
- Irreducible CSC conflicts
- …

…then a miracle occurs…

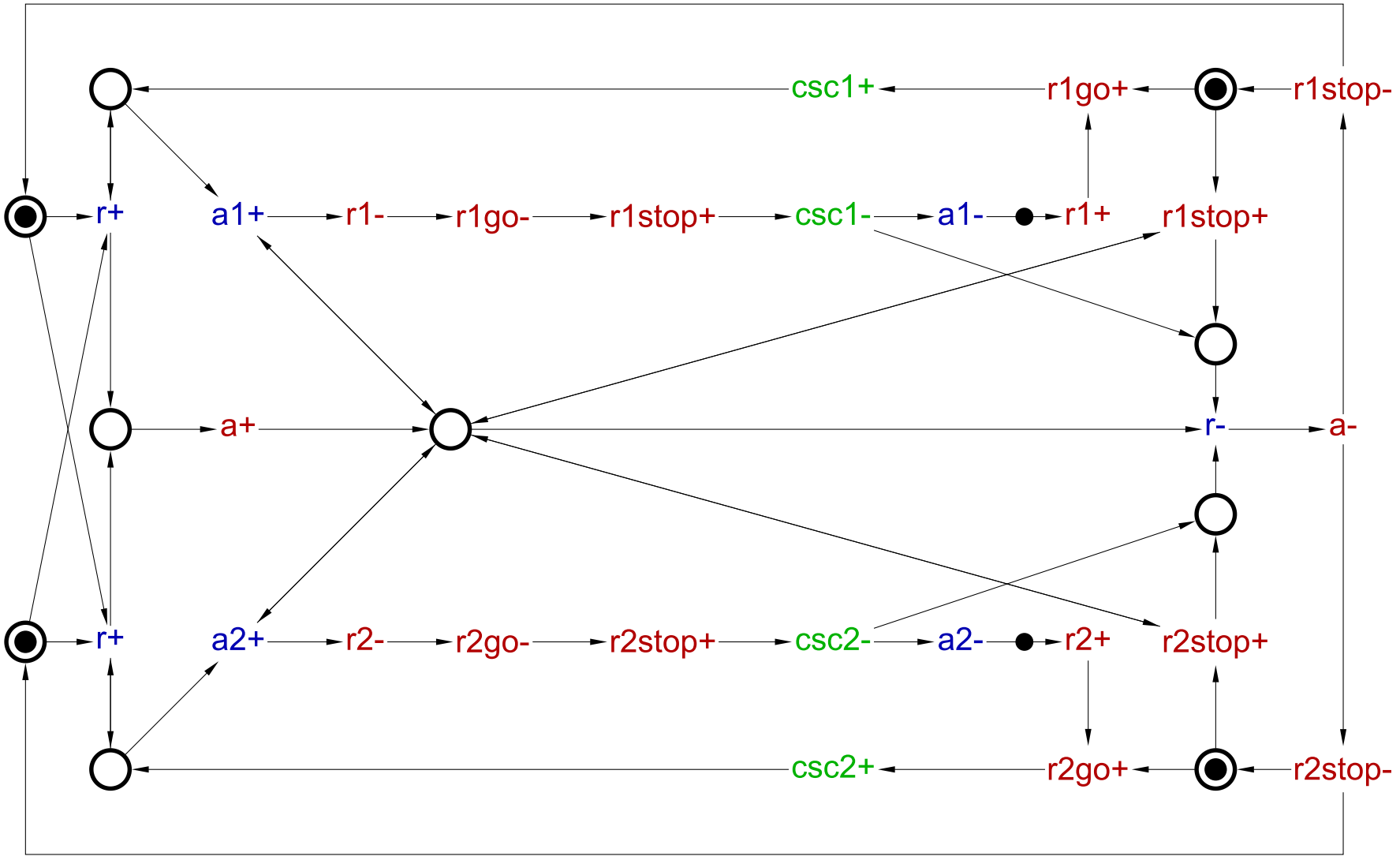
STG specification



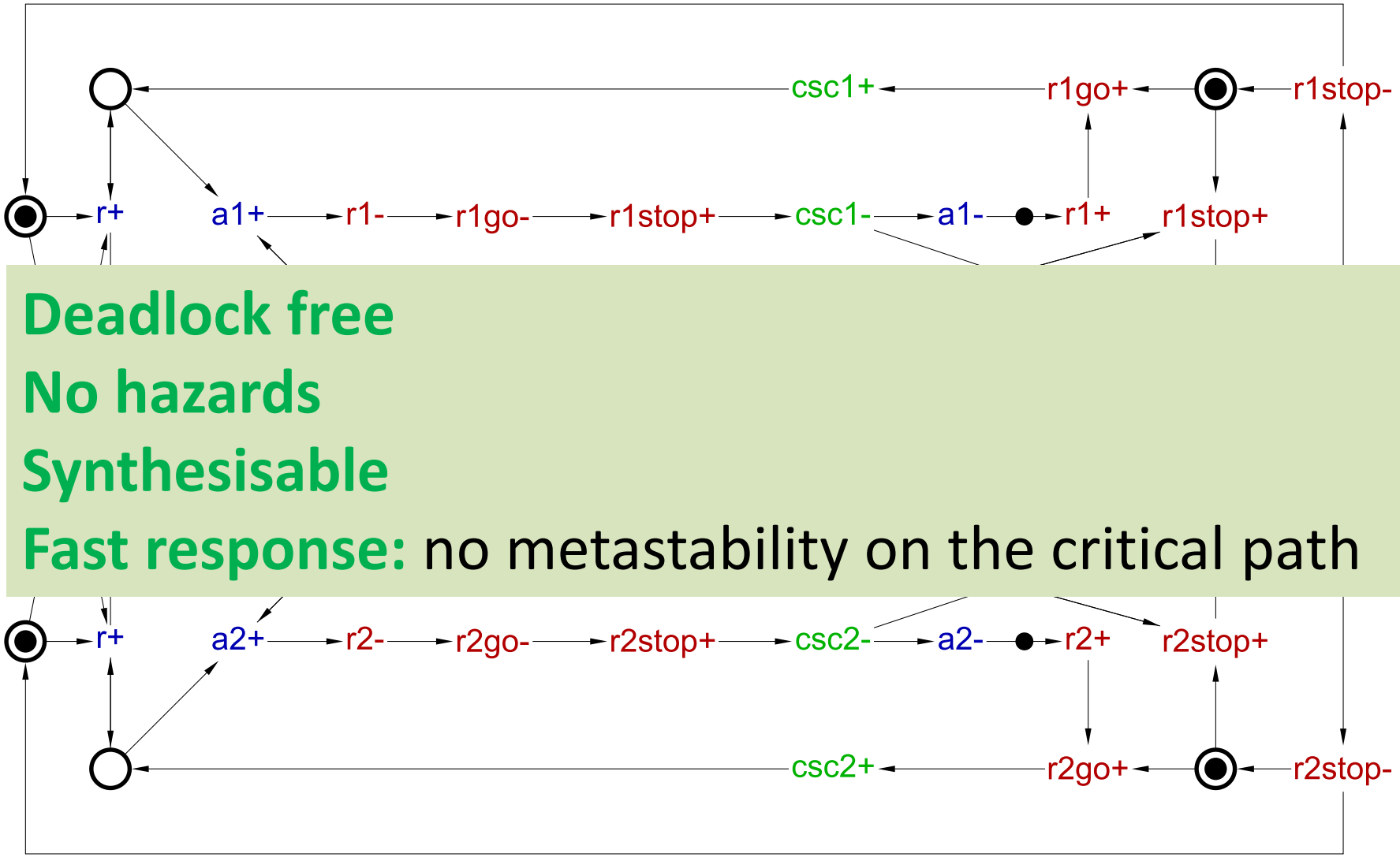
STG specification



CSC resolution (MPSAT)



CSC resolution (MPSAT)



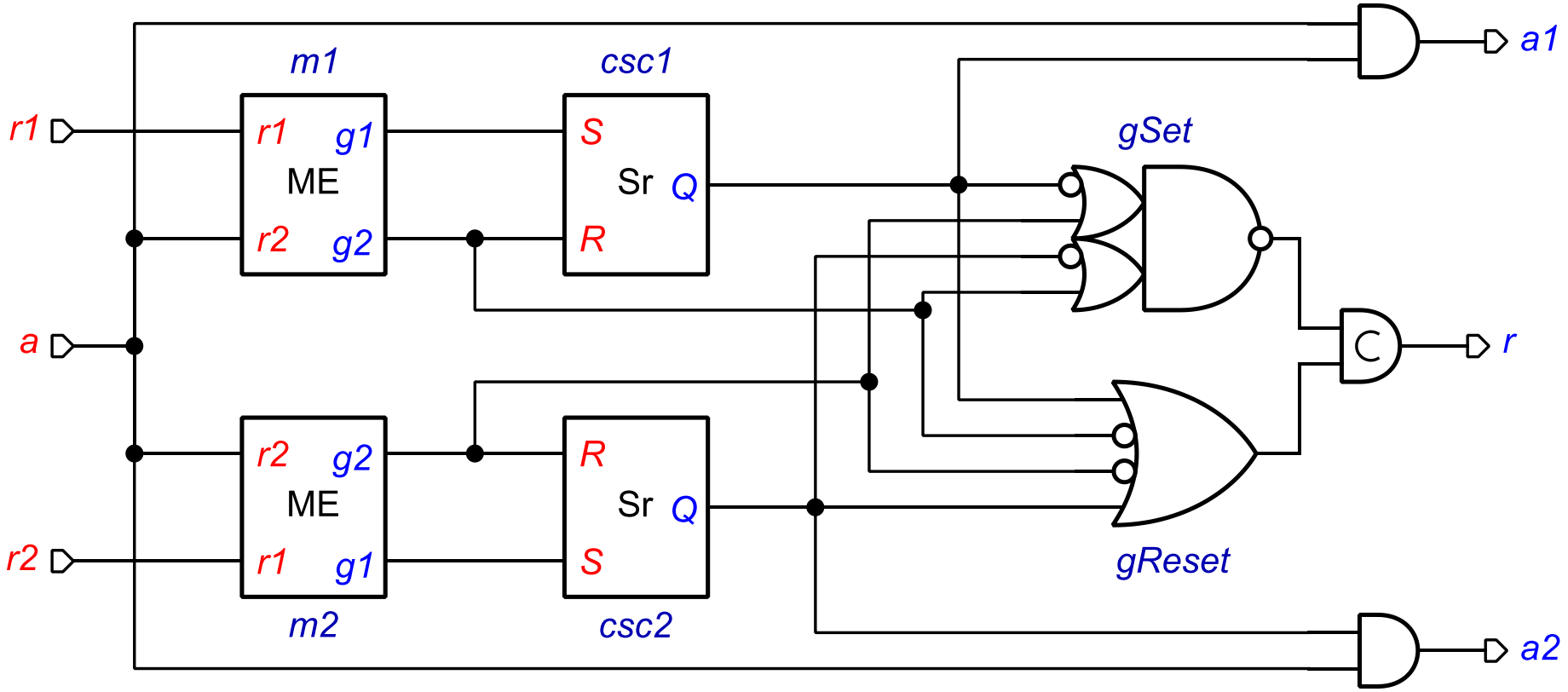
Deadlock free

No hazards

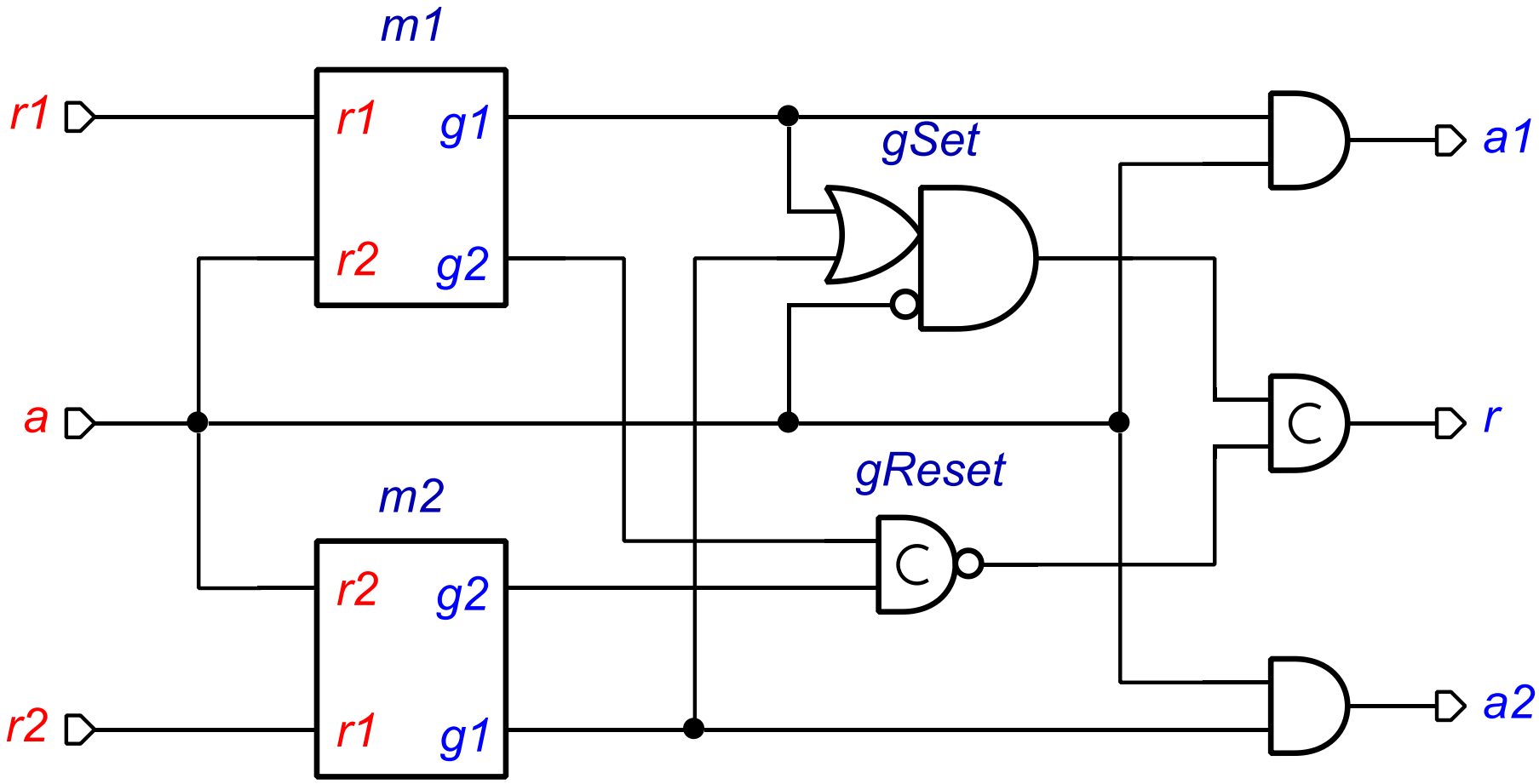
Synthesisable

Fast response: no metastability on the critical path

Synthesised circuit (MPSAT)

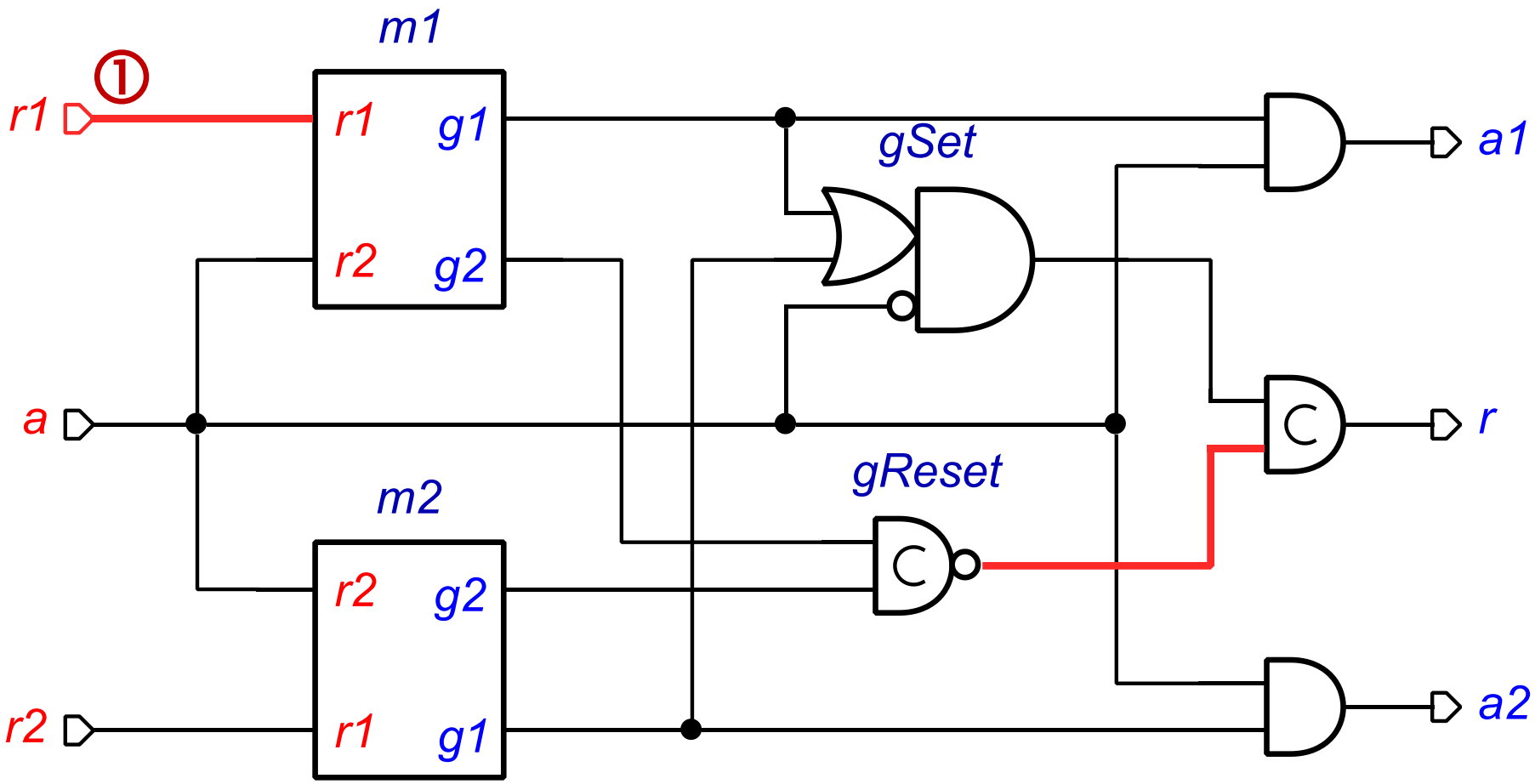


Simplified (hacked up) circuit

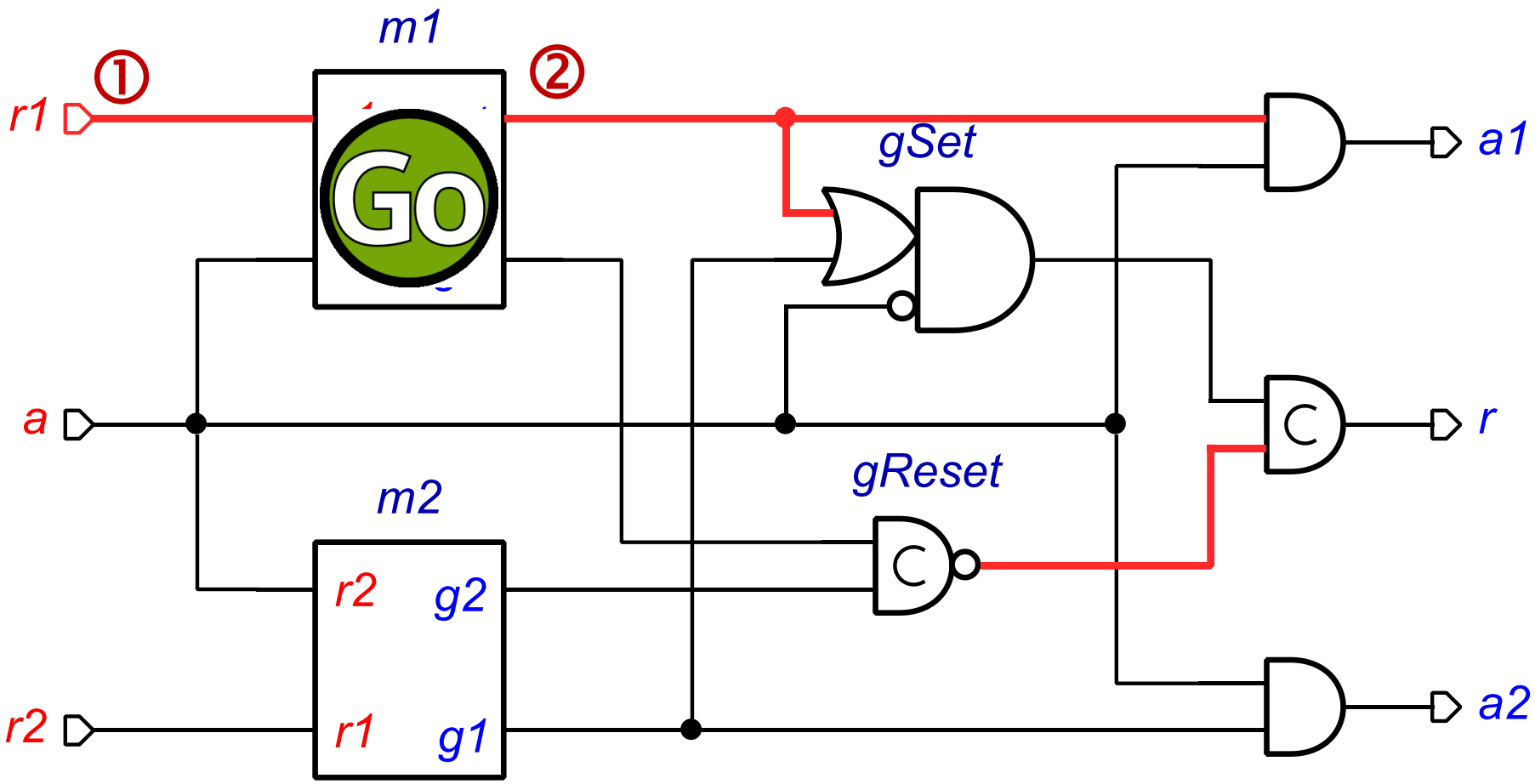


New optimisation technique: **fairness-based optimisation**

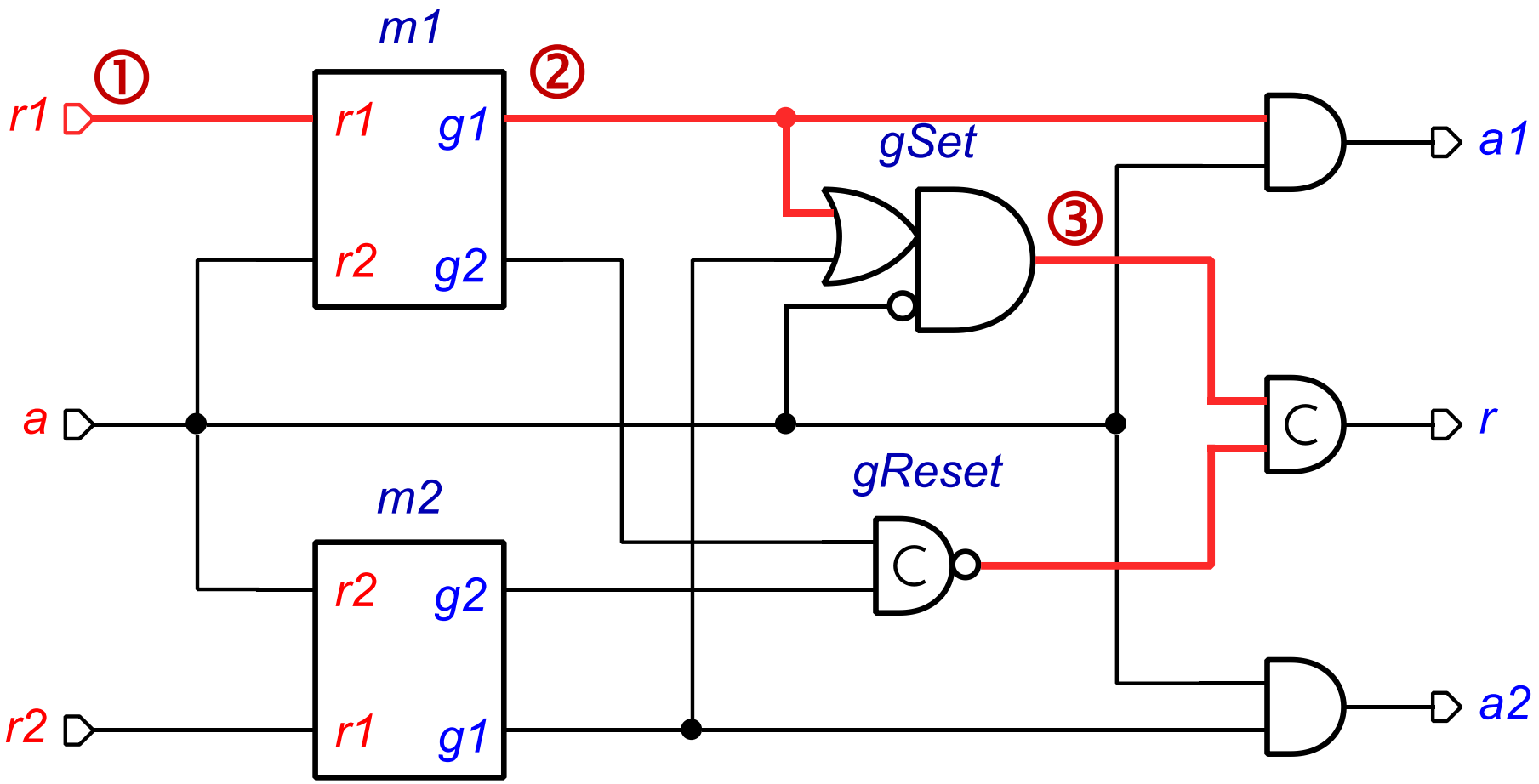
Simplified (hacked up) circuit



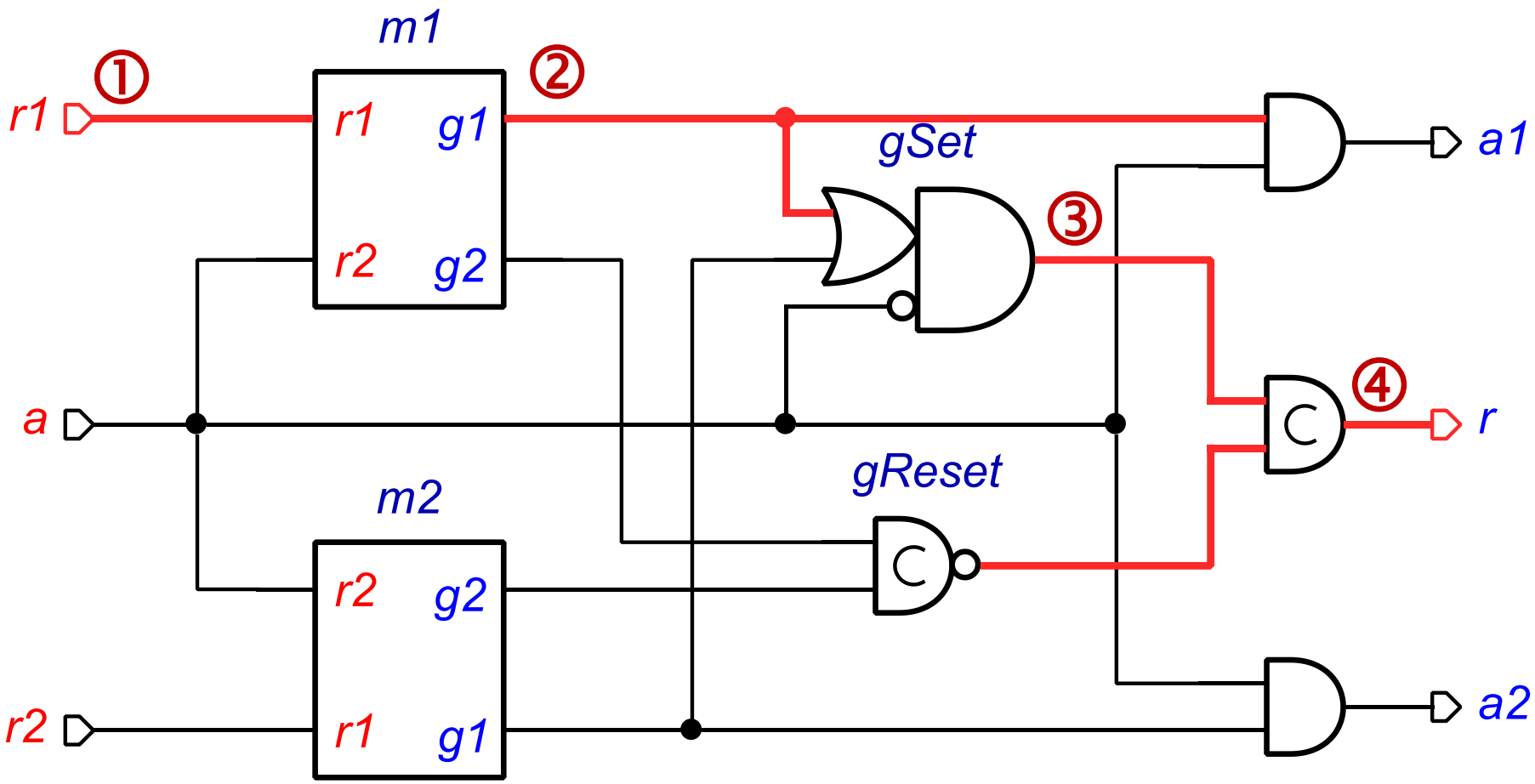
Simplified (hacked up) circuit



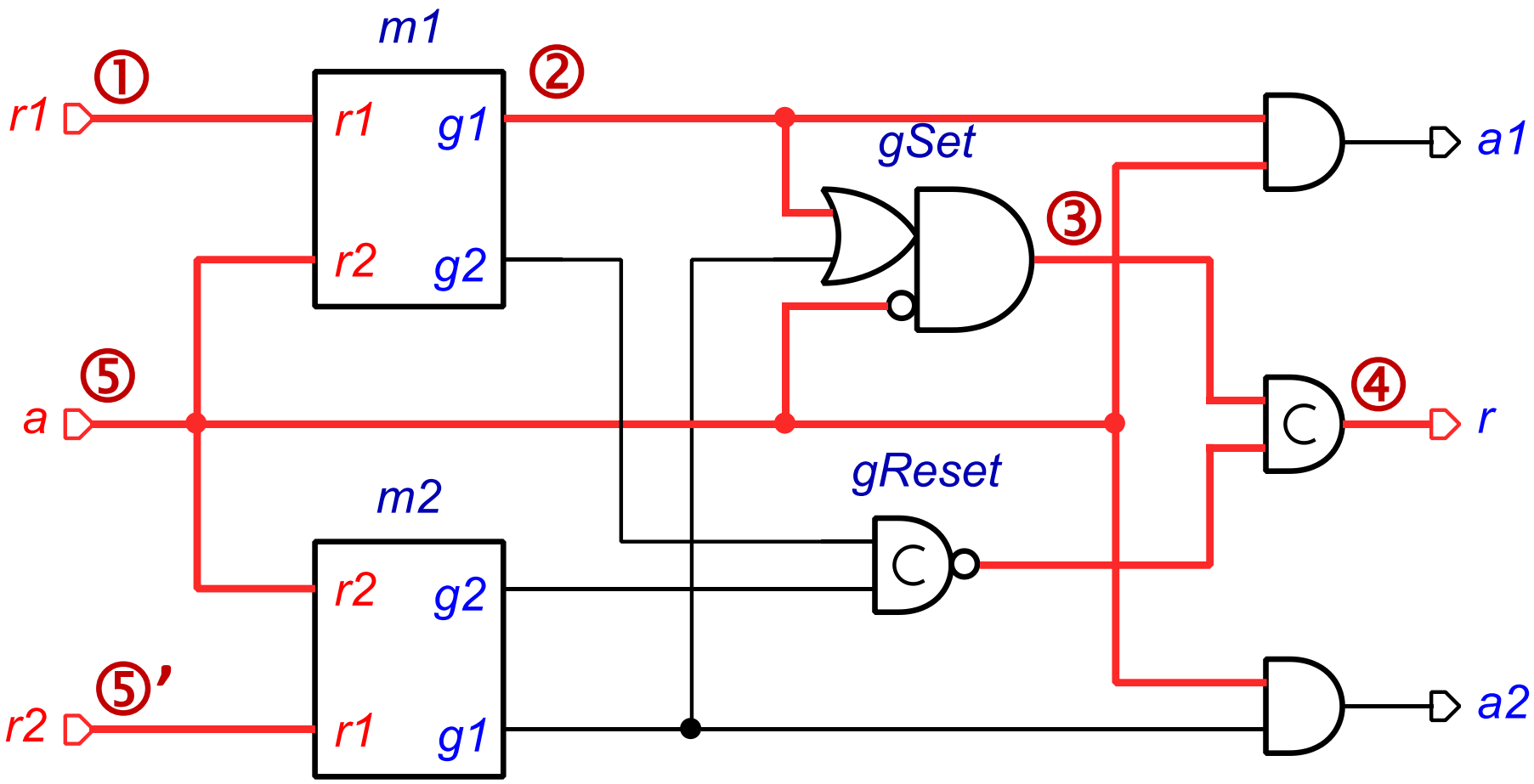
Simplified (hacked up) circuit



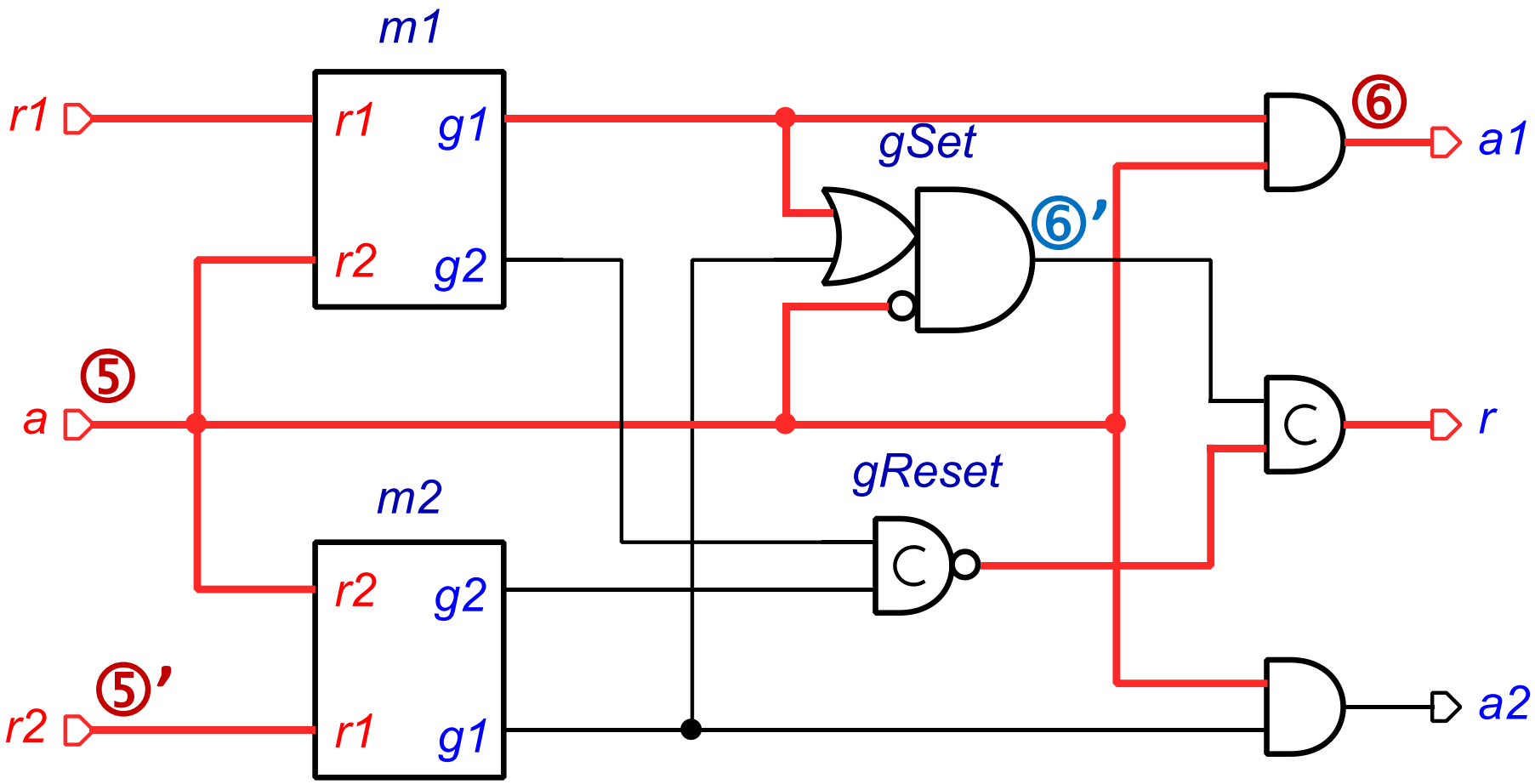
Simplified (hacked up) circuit



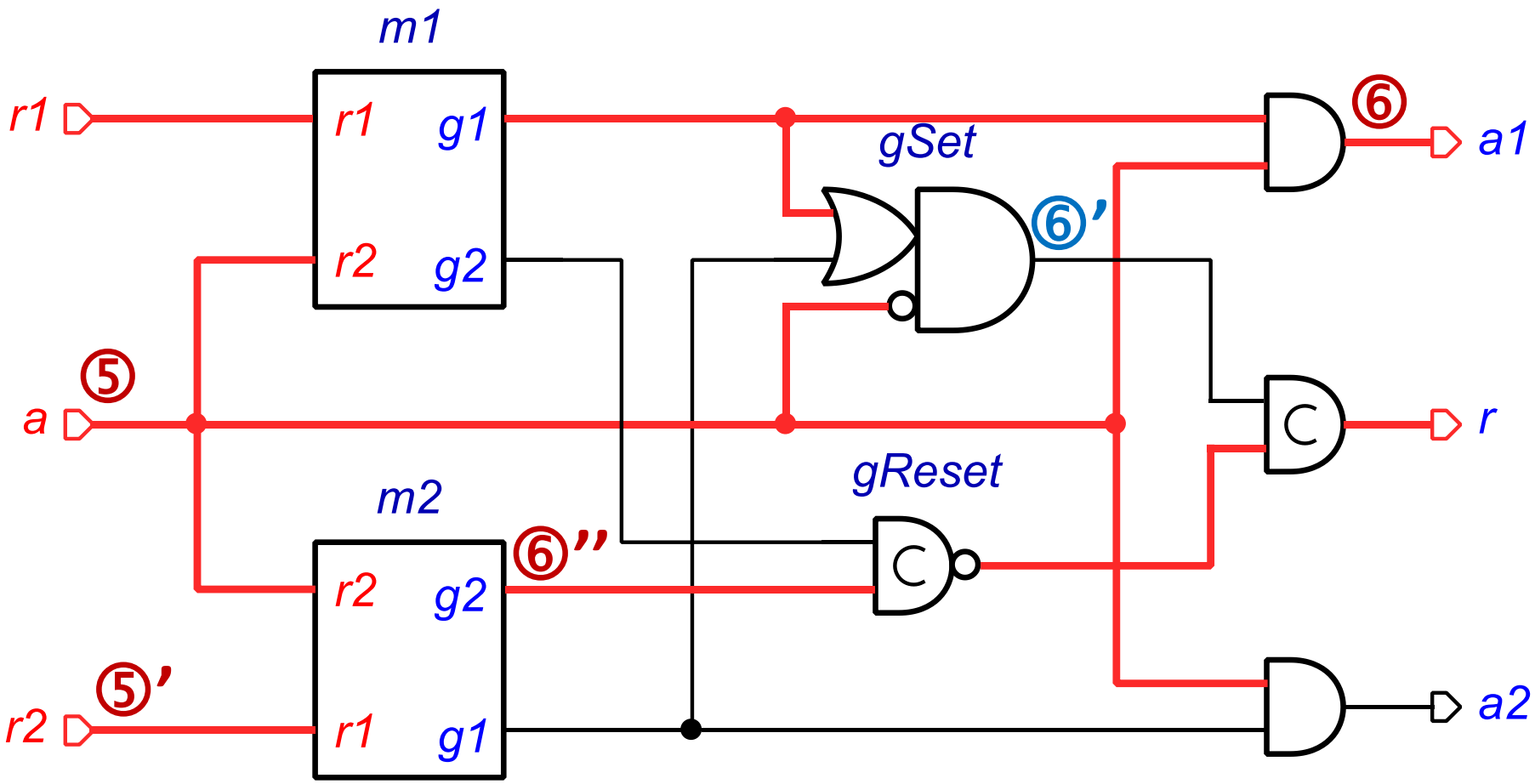
Simplified (hacked up) circuit



Simplified (hacked up) circuit

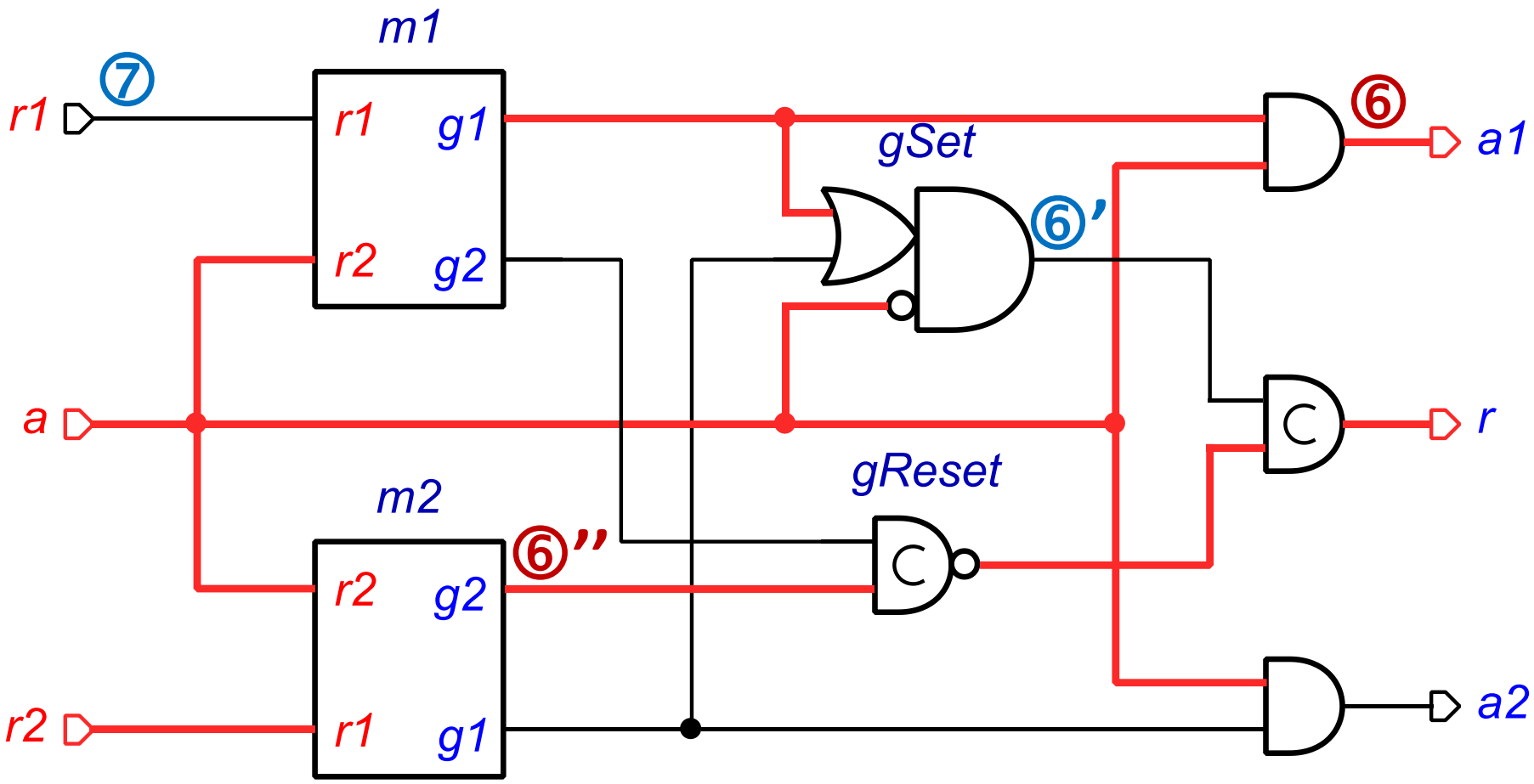


Simplified (hacked up) circuit



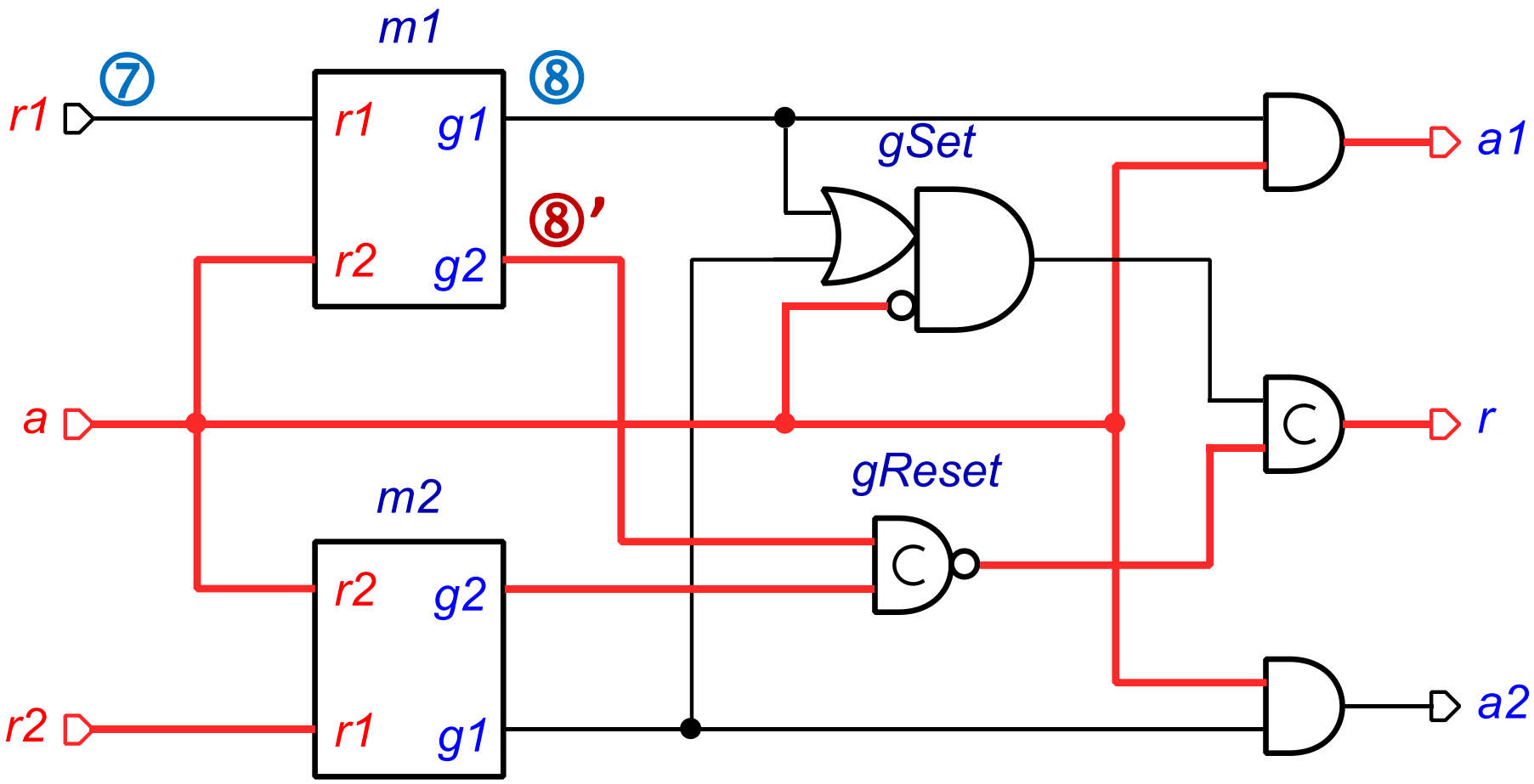
Scenario 1: acknowledgement **a** wins the arbitration

Simplified (hacked up) circuit



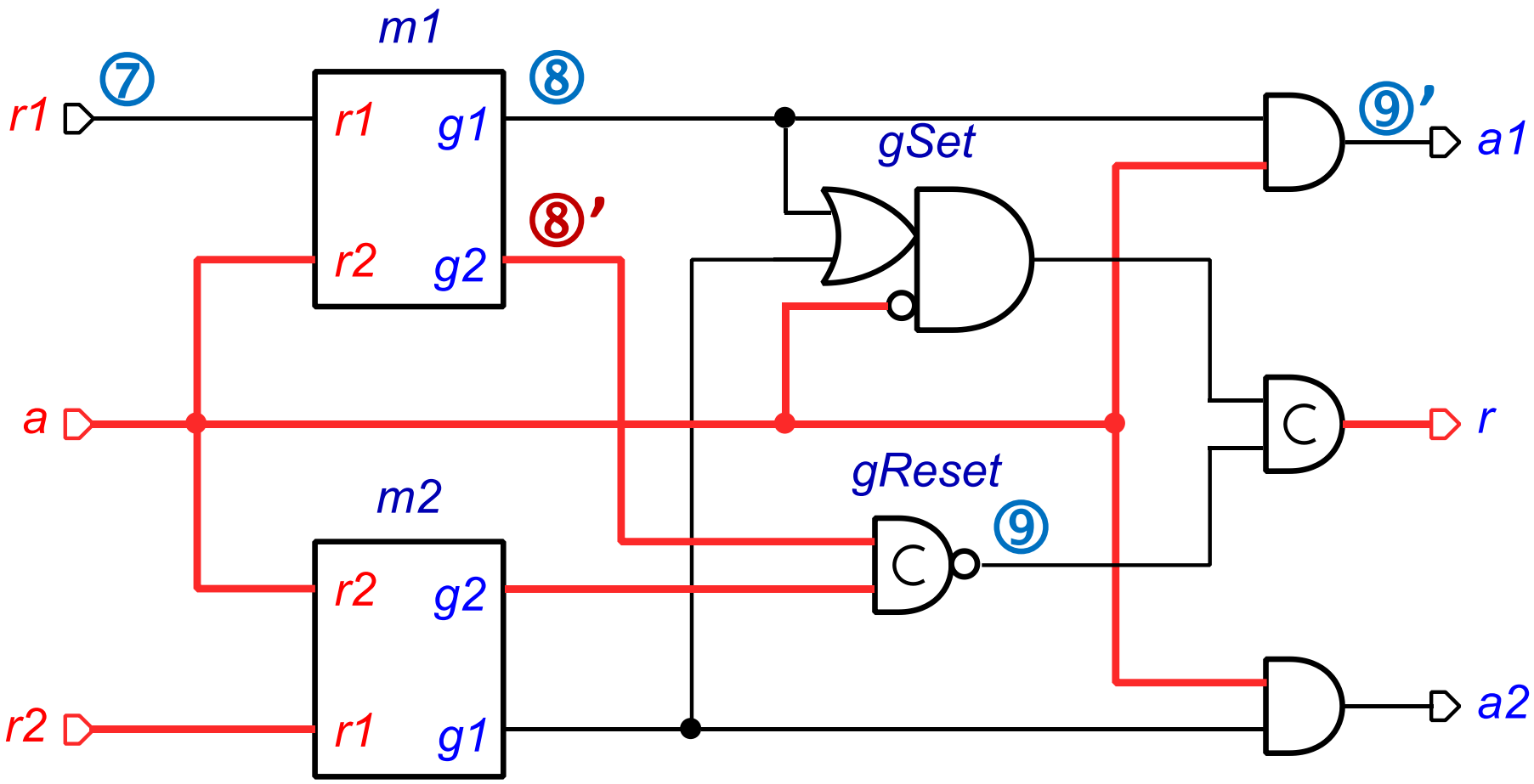
Scenario 1: acknowledgement **a** wins the arbitration

Simplified (hacked up) circuit



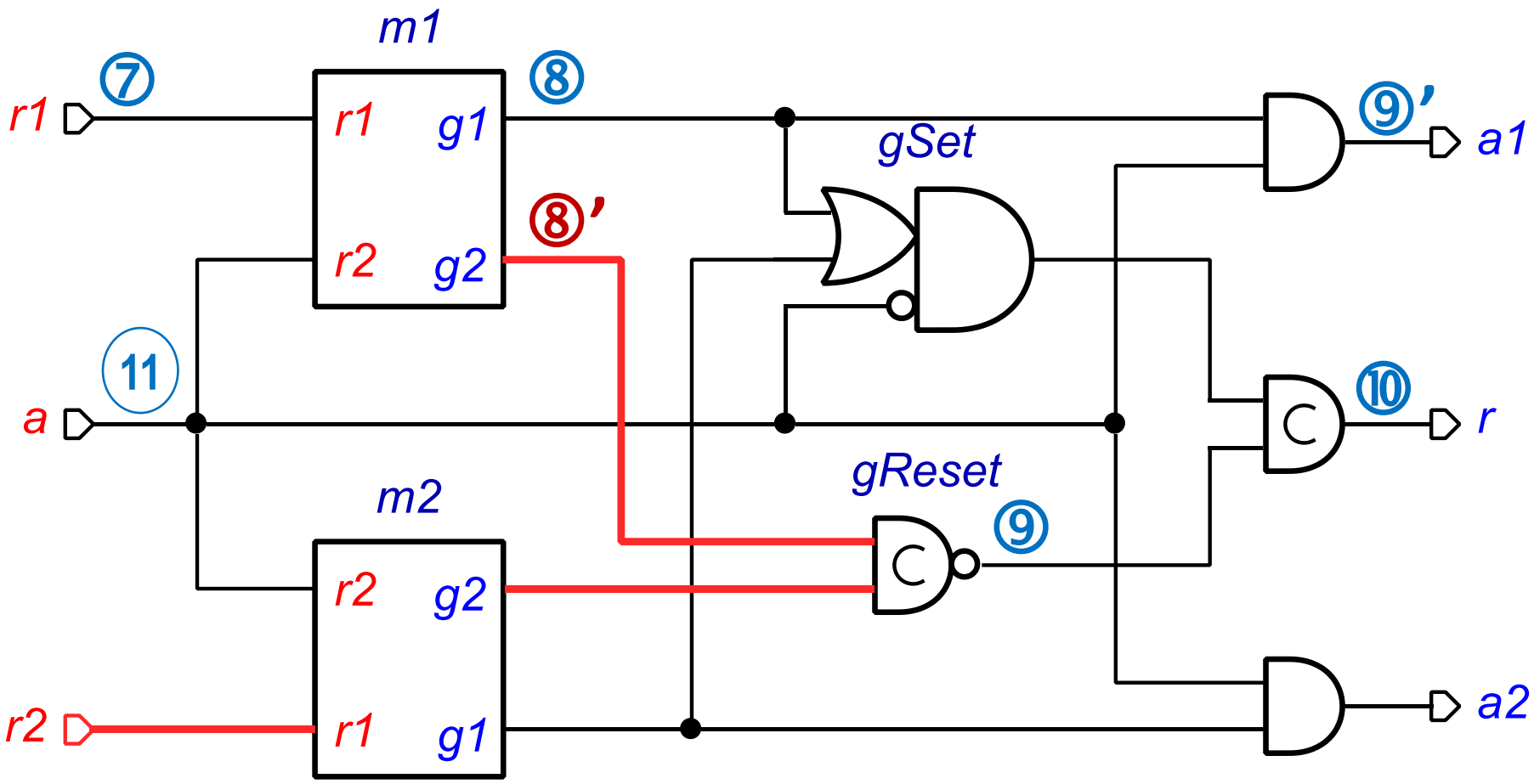
Scenario 1: acknowledgement **a** wins the arbitration

Simplified (hacked up) circuit



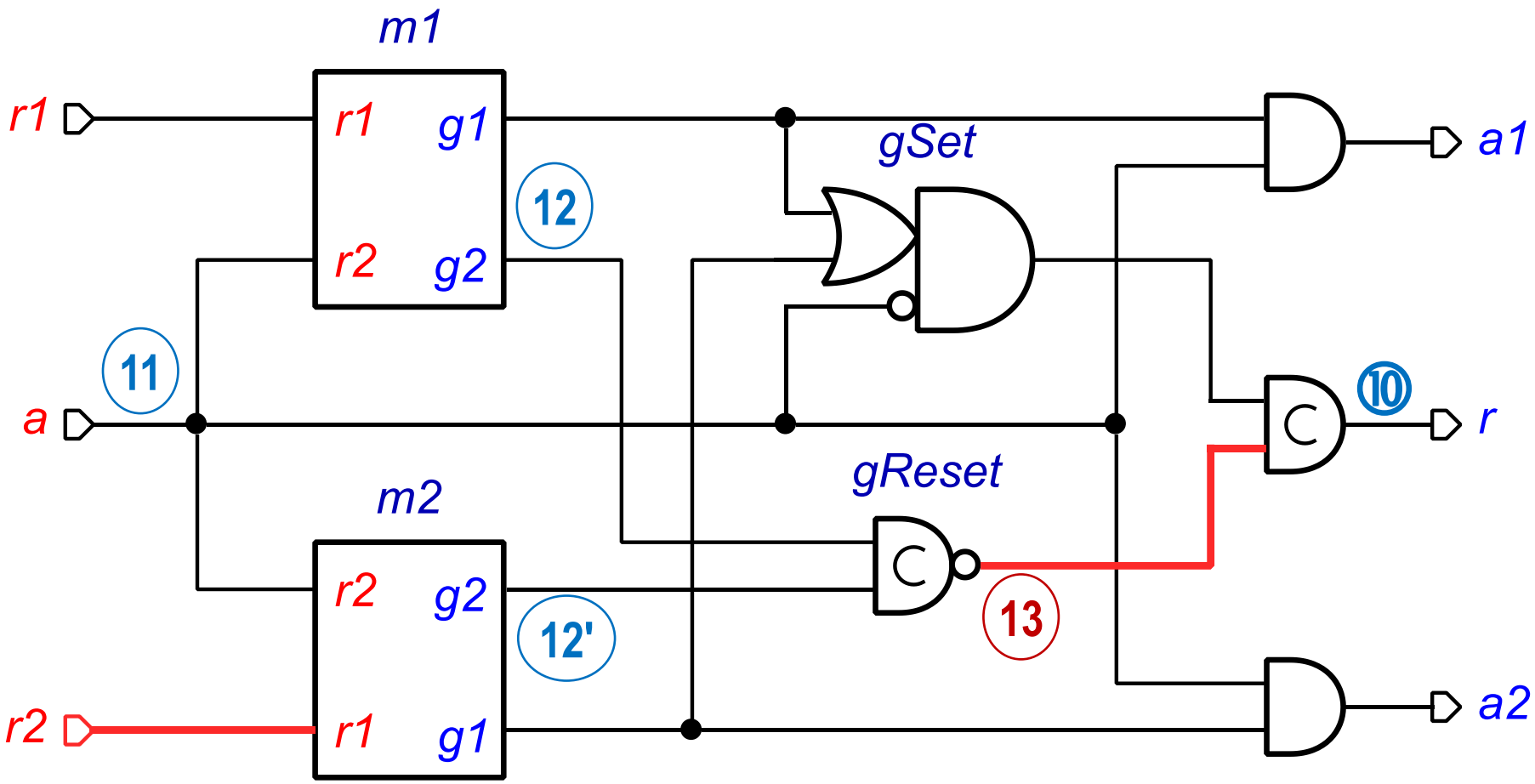
Scenario 1: acknowledgement **a** wins the arbitration

Simplified (hacked up) circuit



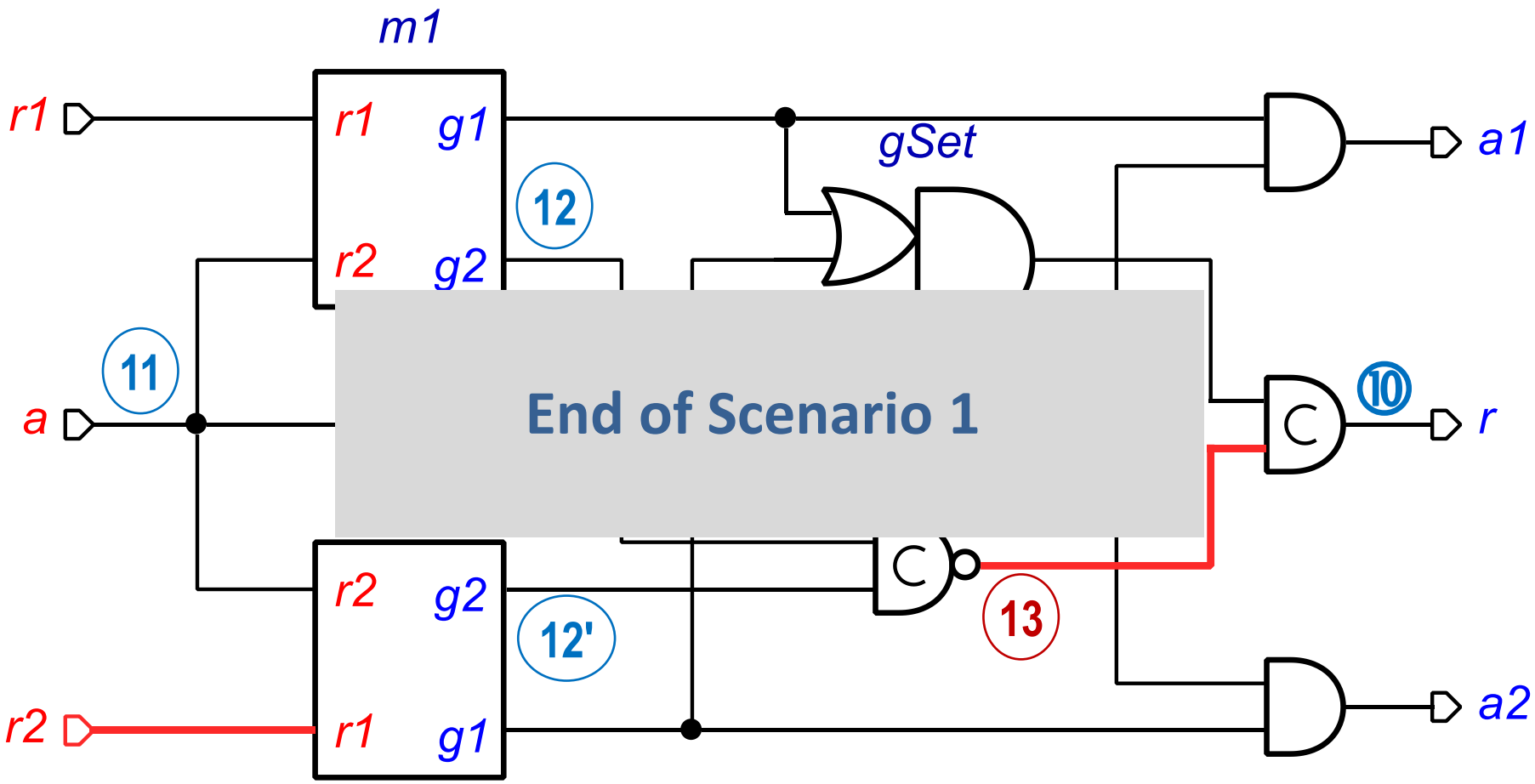
Scenario 1: acknowledgement **a** wins the arbitration

Simplified (hacked up) circuit



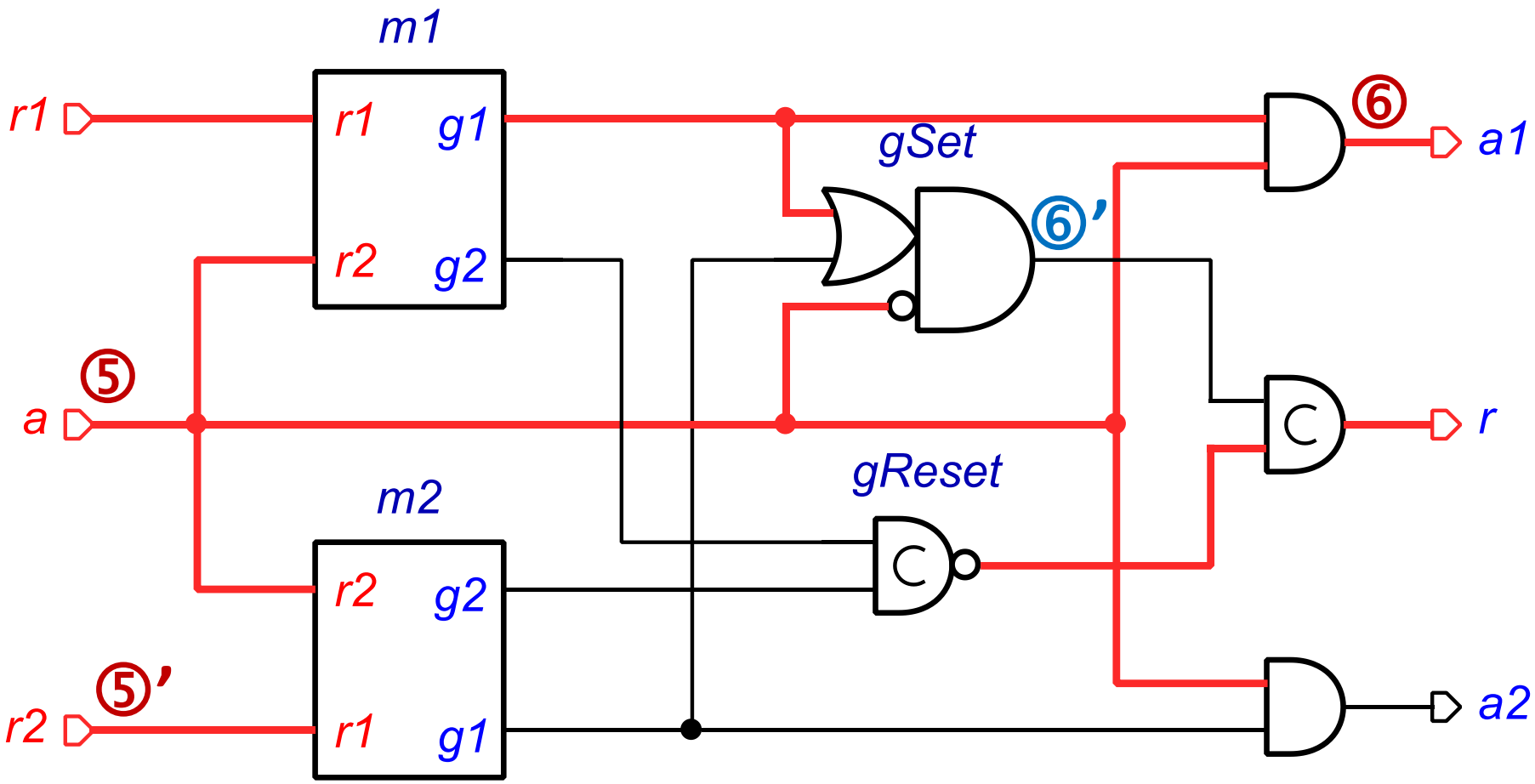
Scenario 1: acknowledgement **a** wins the arbitration

Simplified (hacked up) circuit



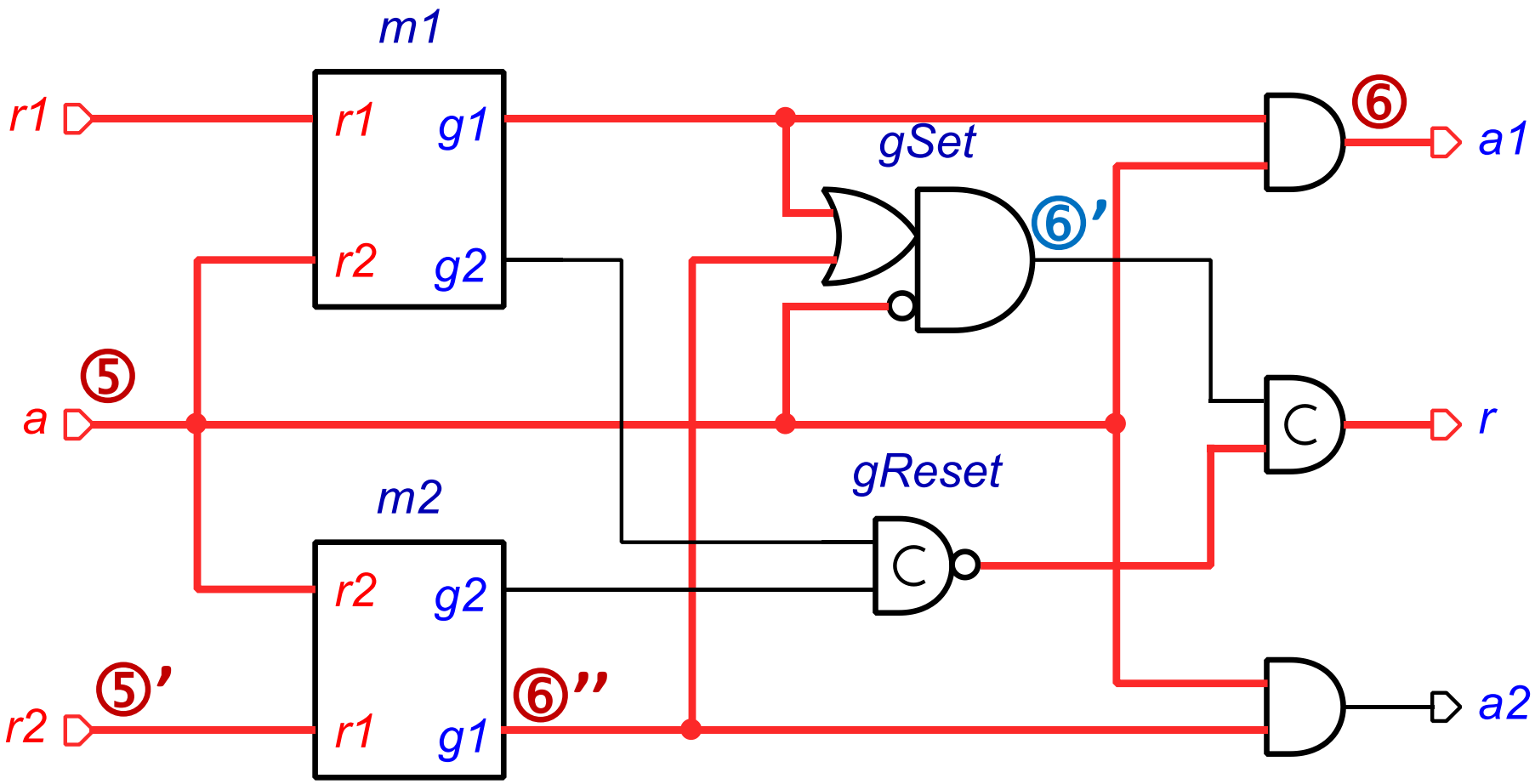
Scenario 1: acknowledgement **a** wins the arbitration

Simplified (hacked up) circuit



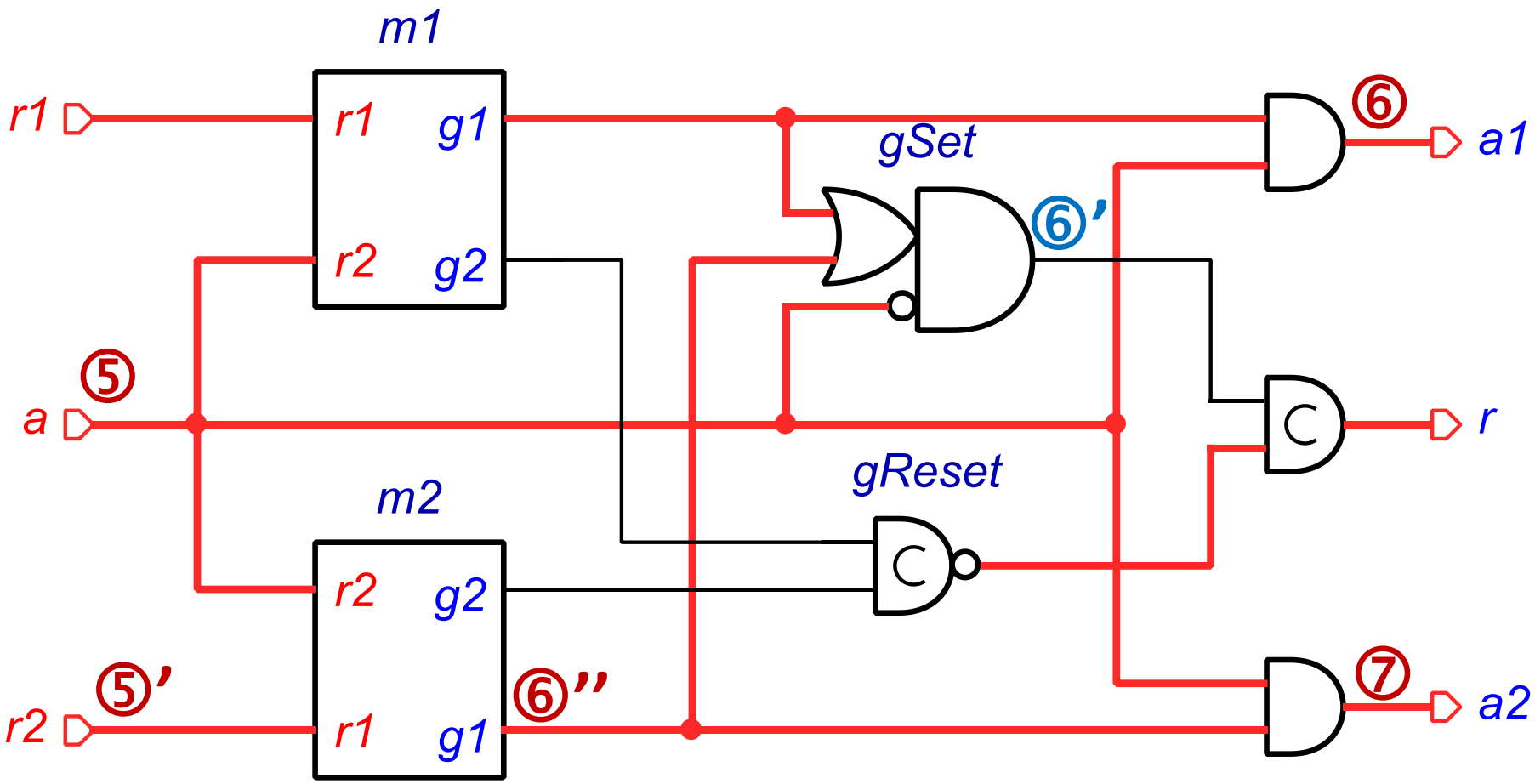
Scenario 2: request **r2** wins the arbitration

Simplified (hacked up) circuit



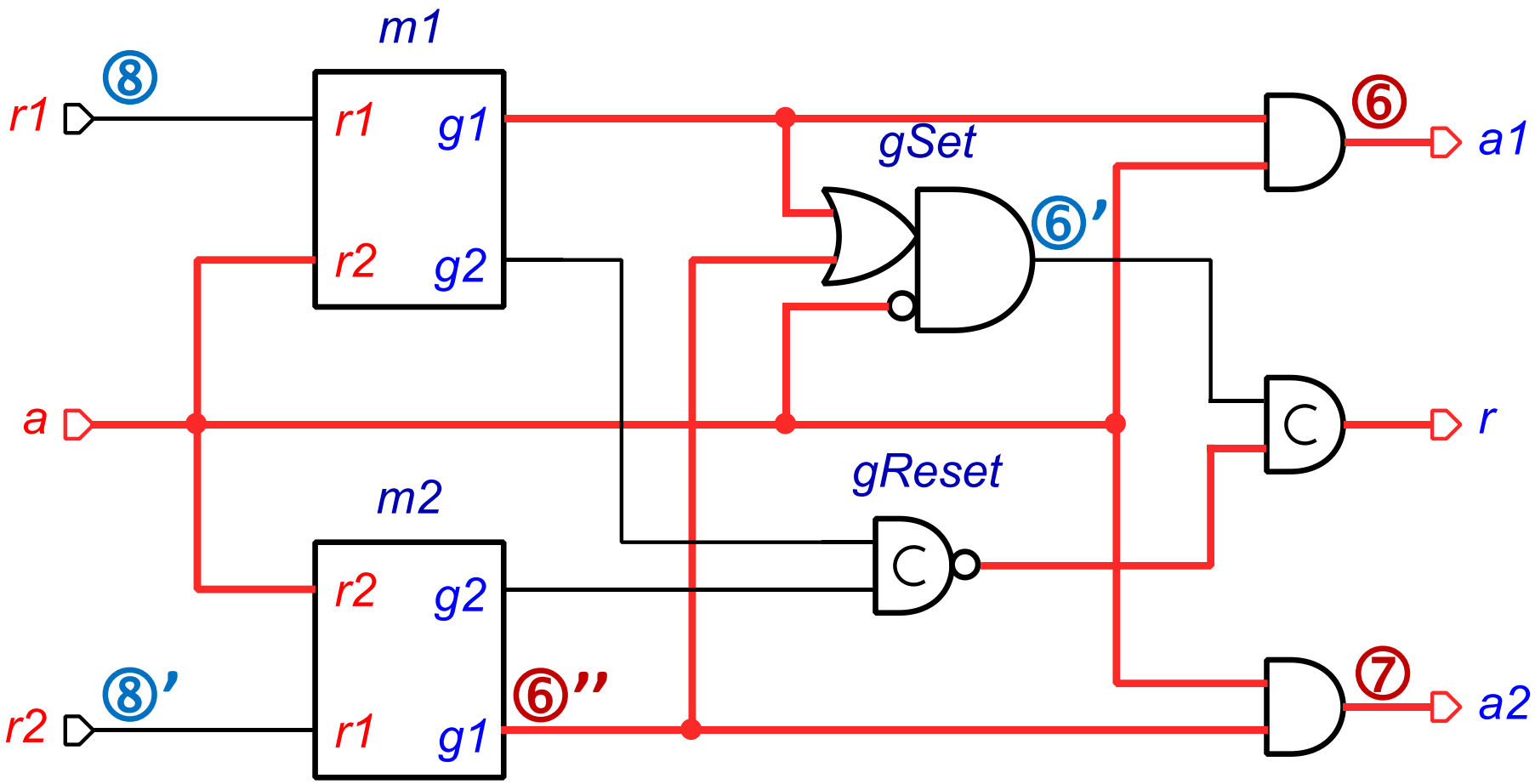
Scenario 2: request **r2** wins the arbitration

Simplified (hacked up) circuit



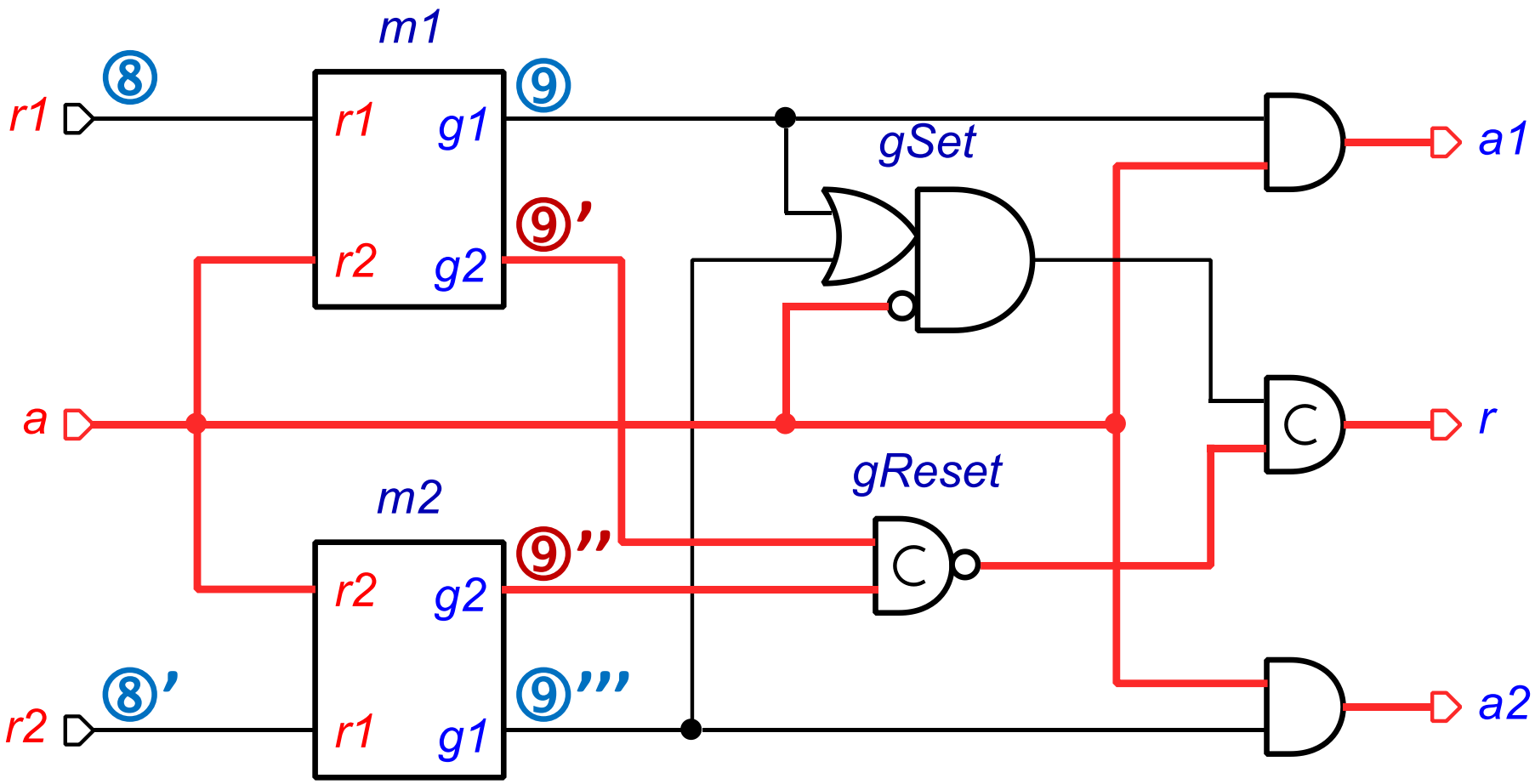
Scenario 2: request **r2** wins the arbitration

Simplified (hacked up) circuit



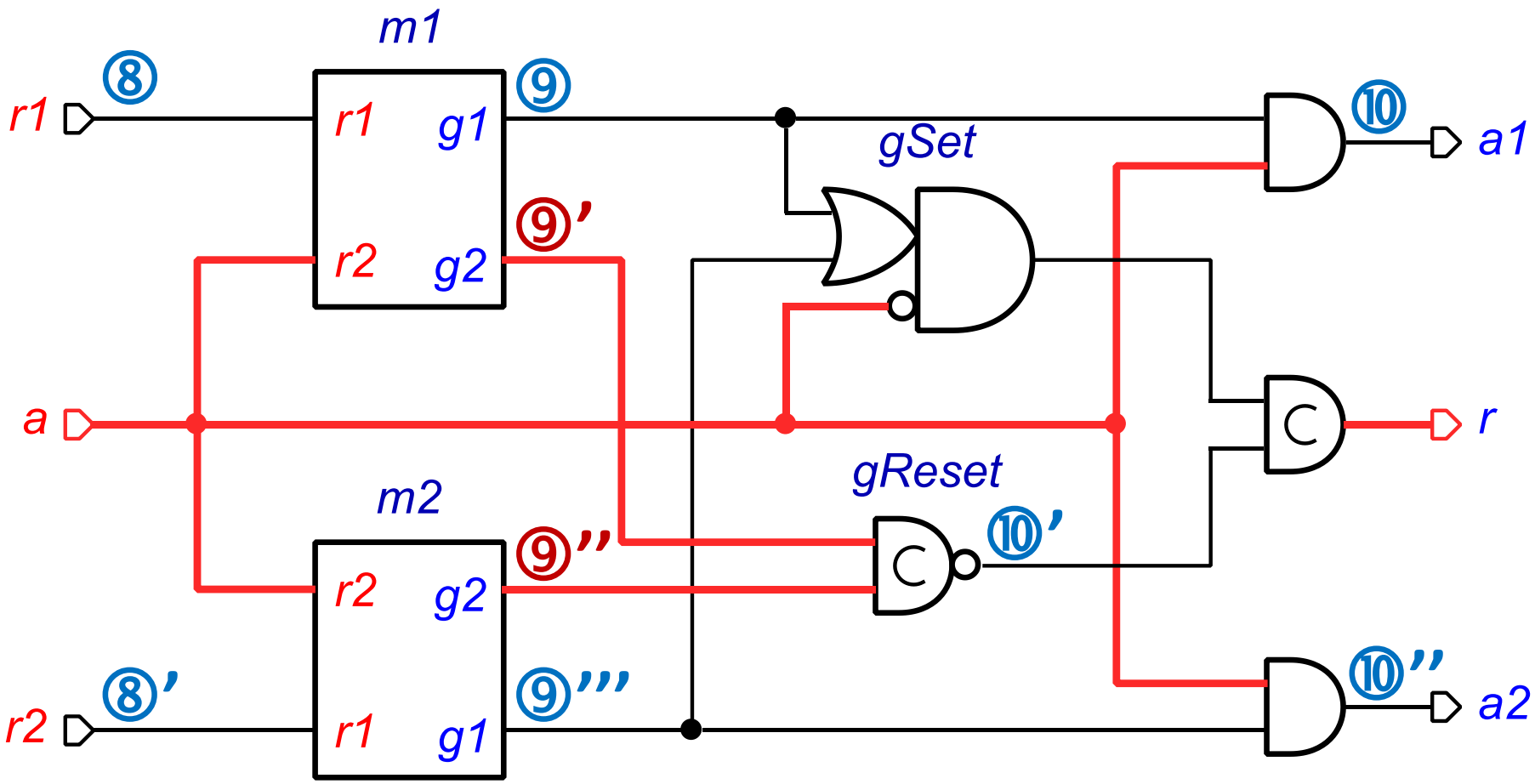
Scenario 2: request **r2** wins the arbitration

Simplified (hacked up) circuit



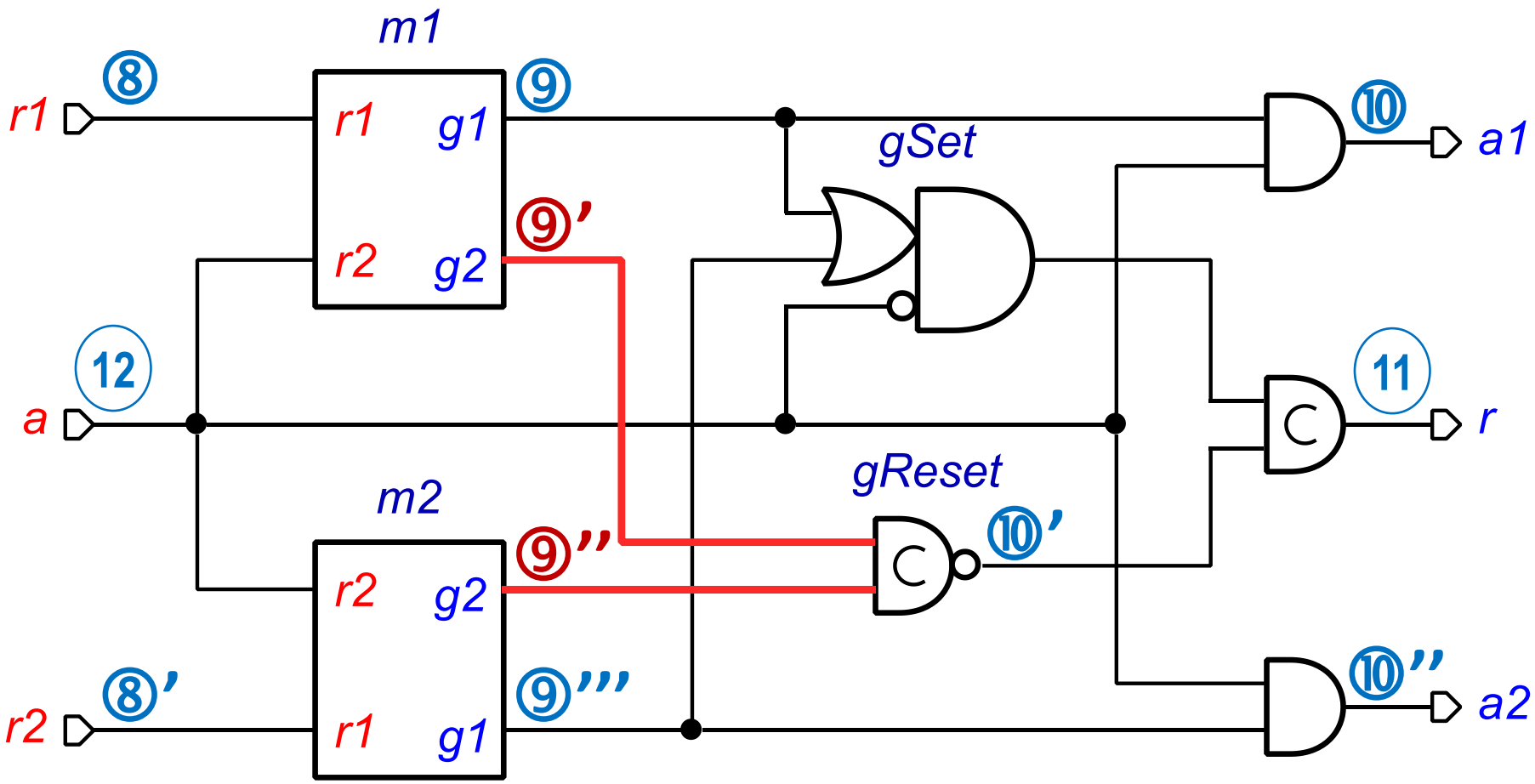
Scenario 2: request **r2** wins the arbitration

Simplified (hacked up) circuit



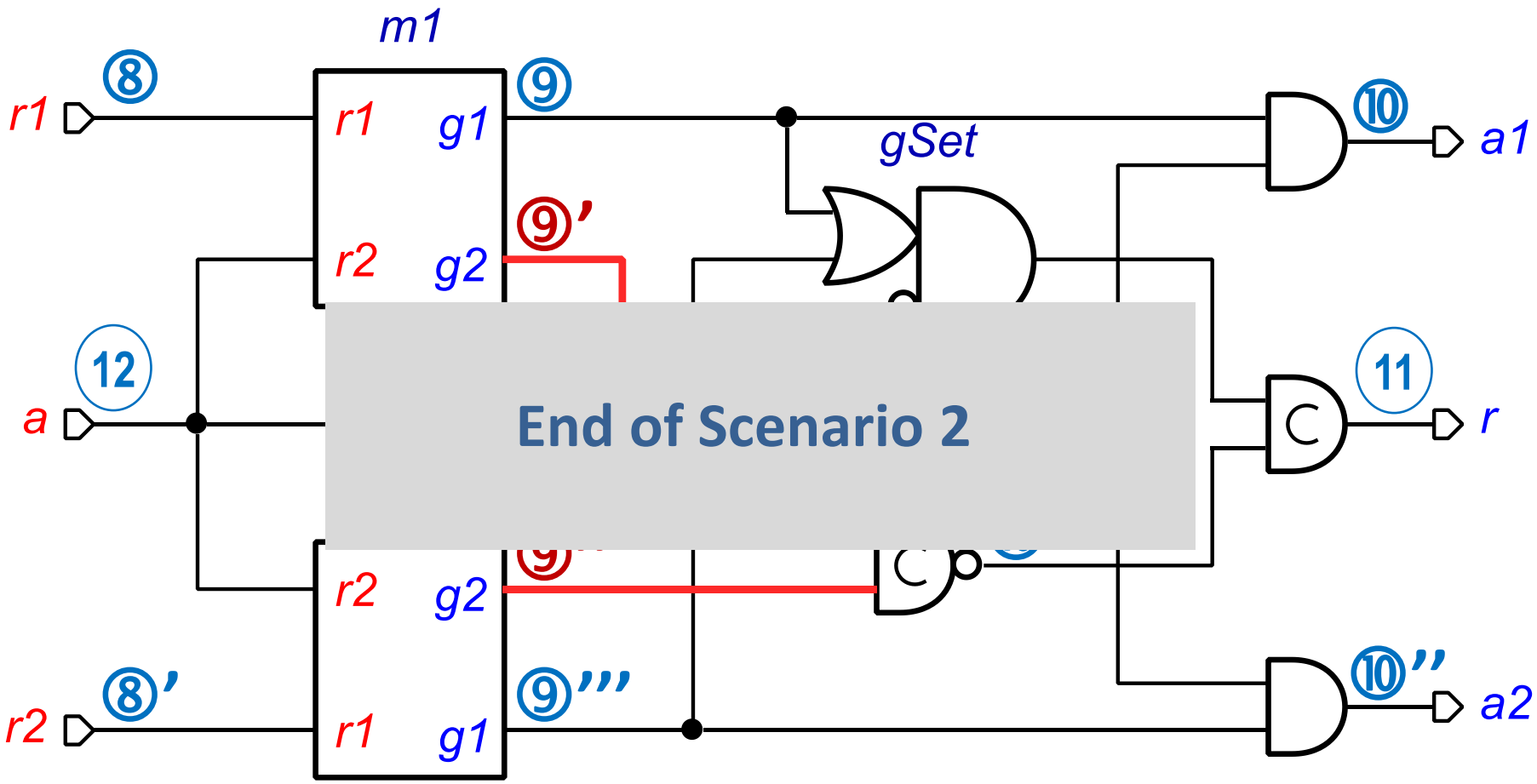
Scenario 2: request **r2** wins the arbitration

Simplified (hacked up) circuit



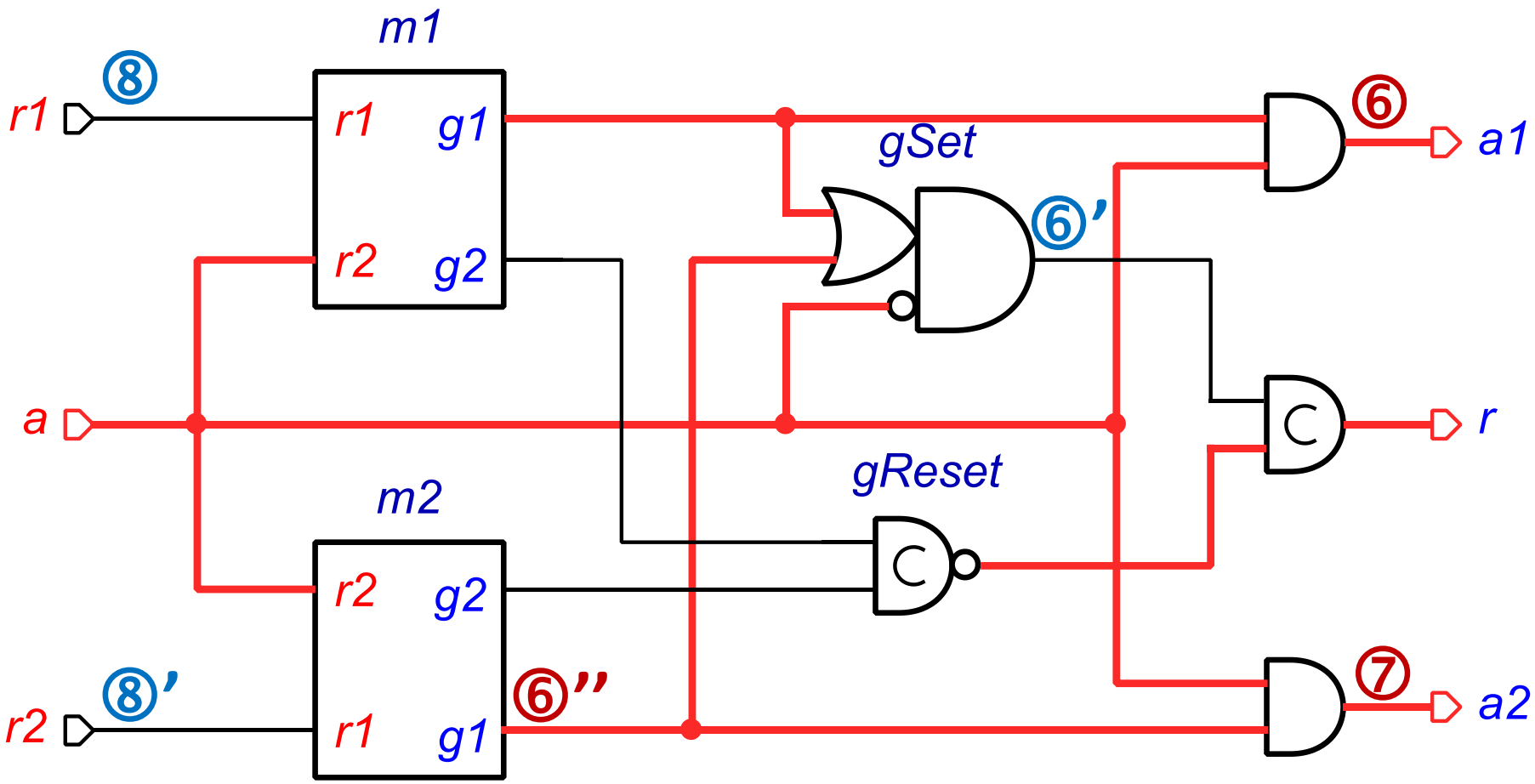
Scenario 2: request **r2** wins the arbitration

Simplified (hacked up) circuit



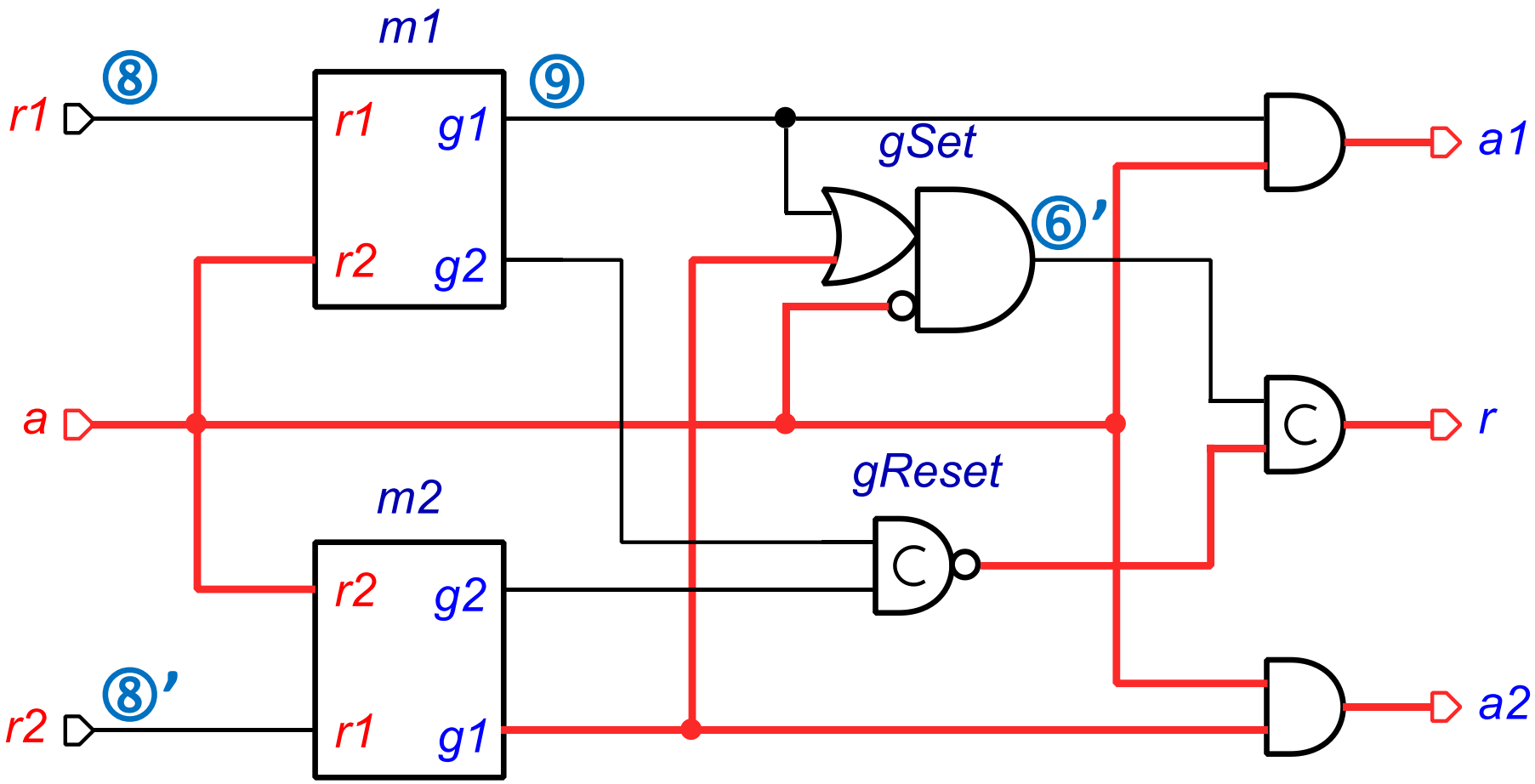
Scenario 2: request **r2** wins the arbitration

Simplified (hacked up) circuit



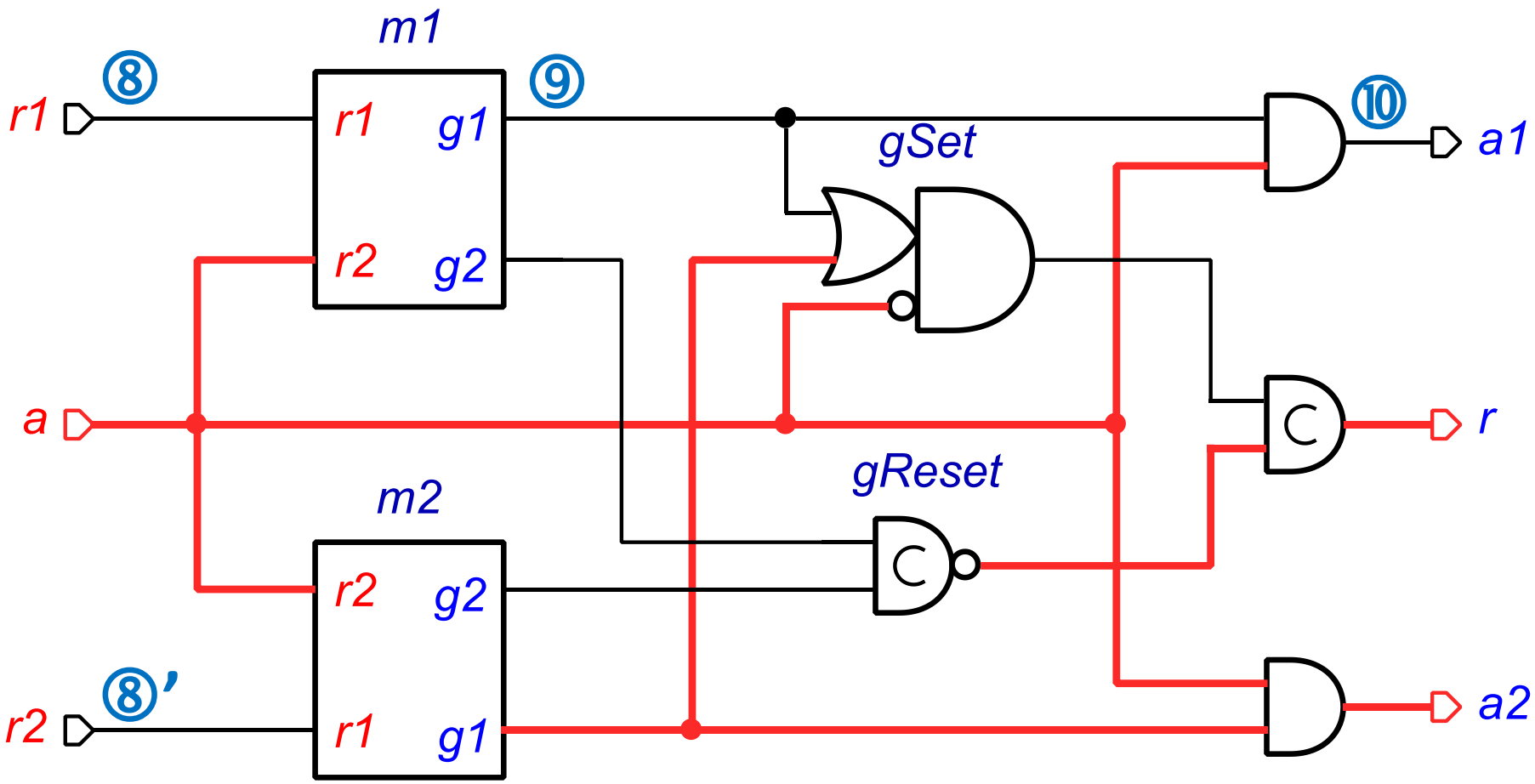
Scenario 3: sequential bundling of requests

Simplified (hacked up) circuit



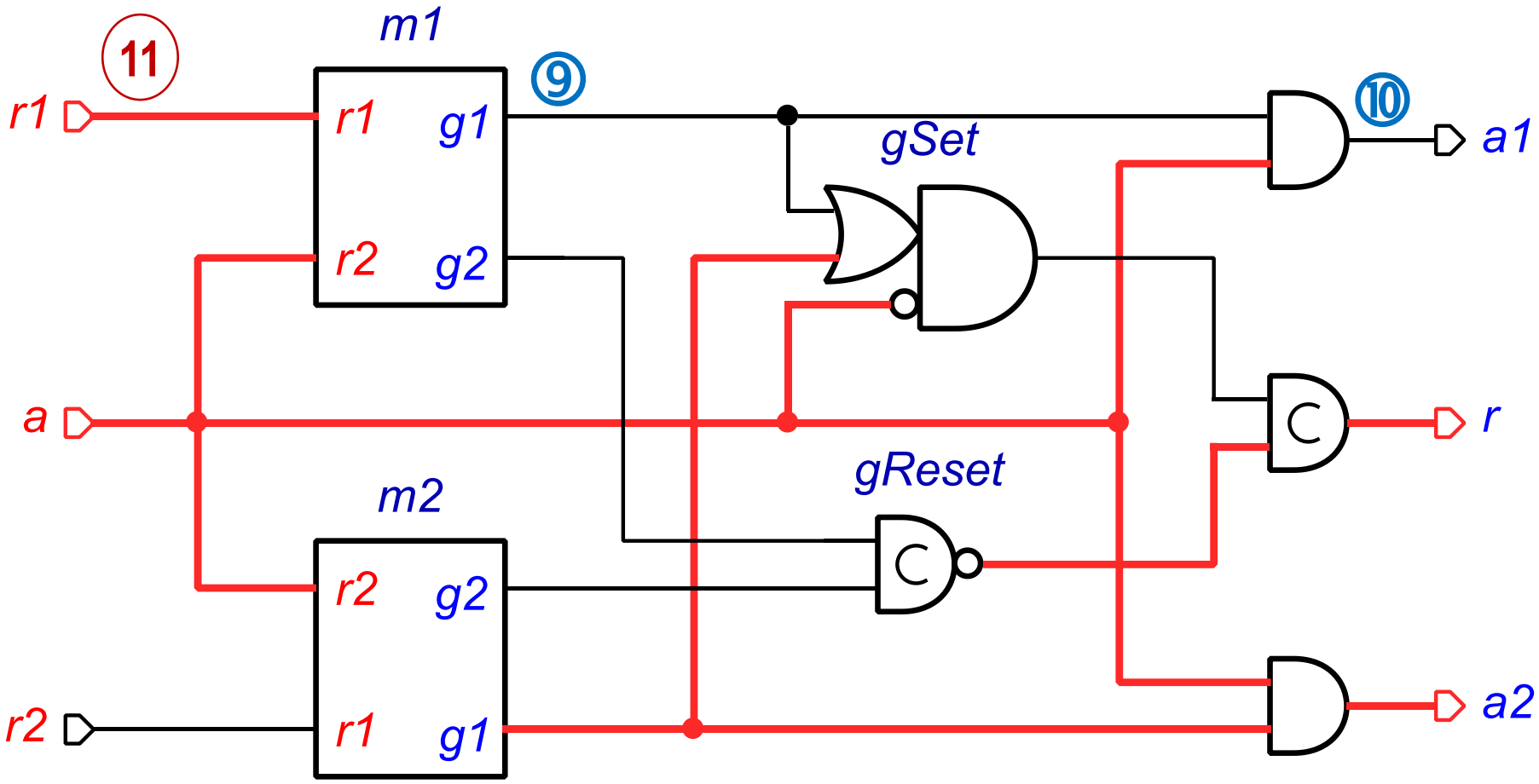
Scenario 3: sequential bundling of requests

Simplified (hacked up) circuit



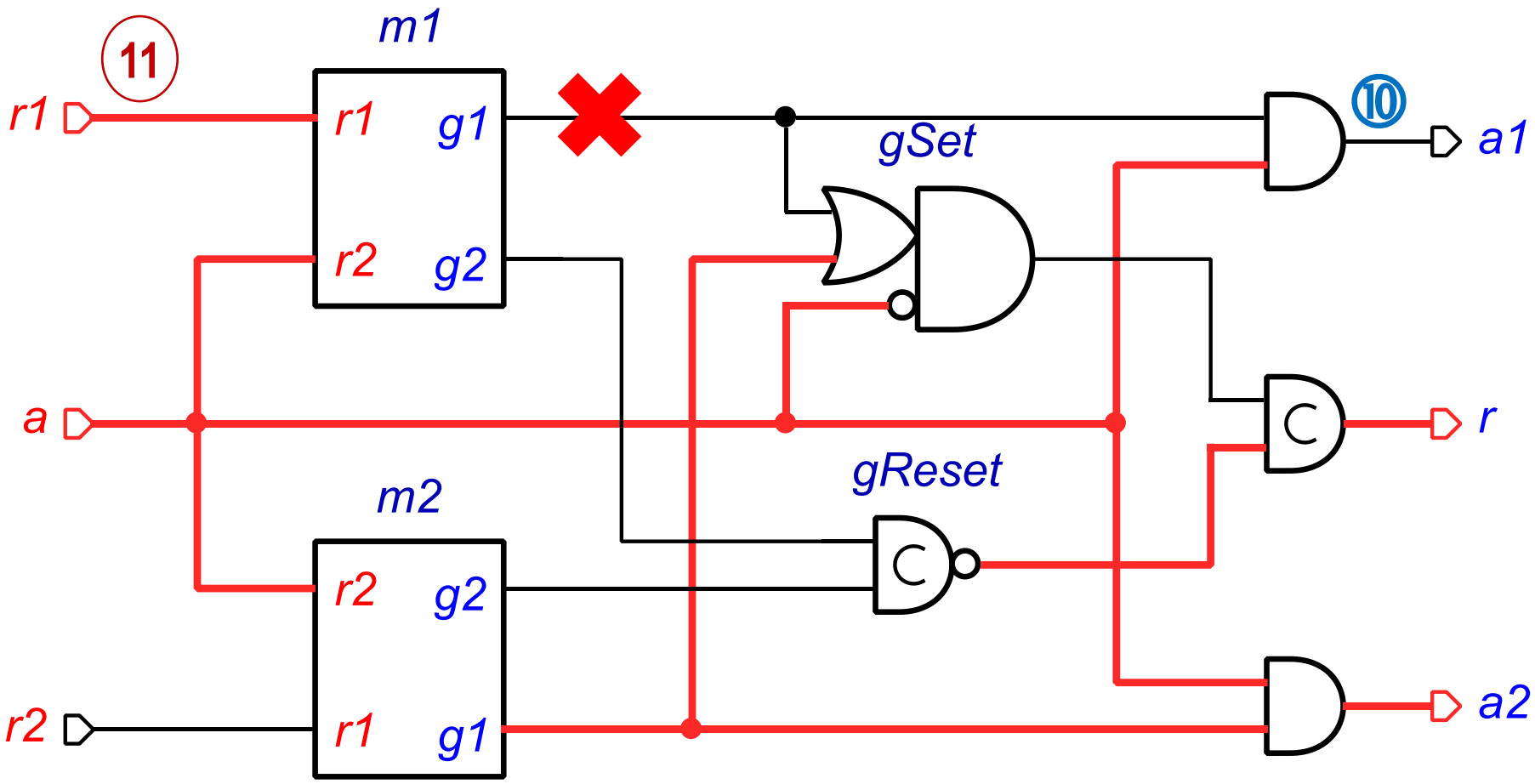
Scenario 3: sequential bundling of requests

Simplified (hacked up) circuit



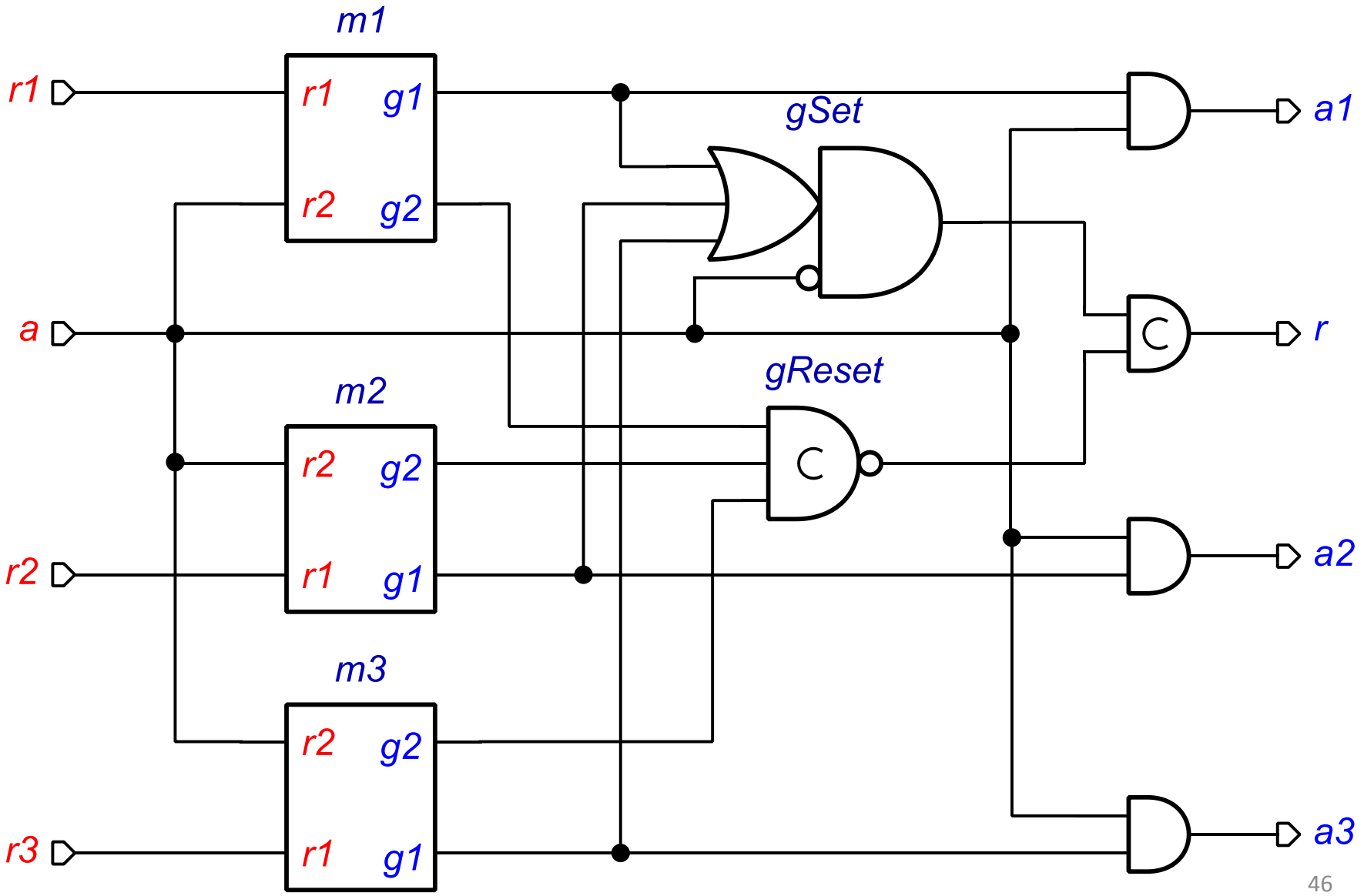
Scenario 3: sequential bundling of requests

Simplified (hacked up) circuit

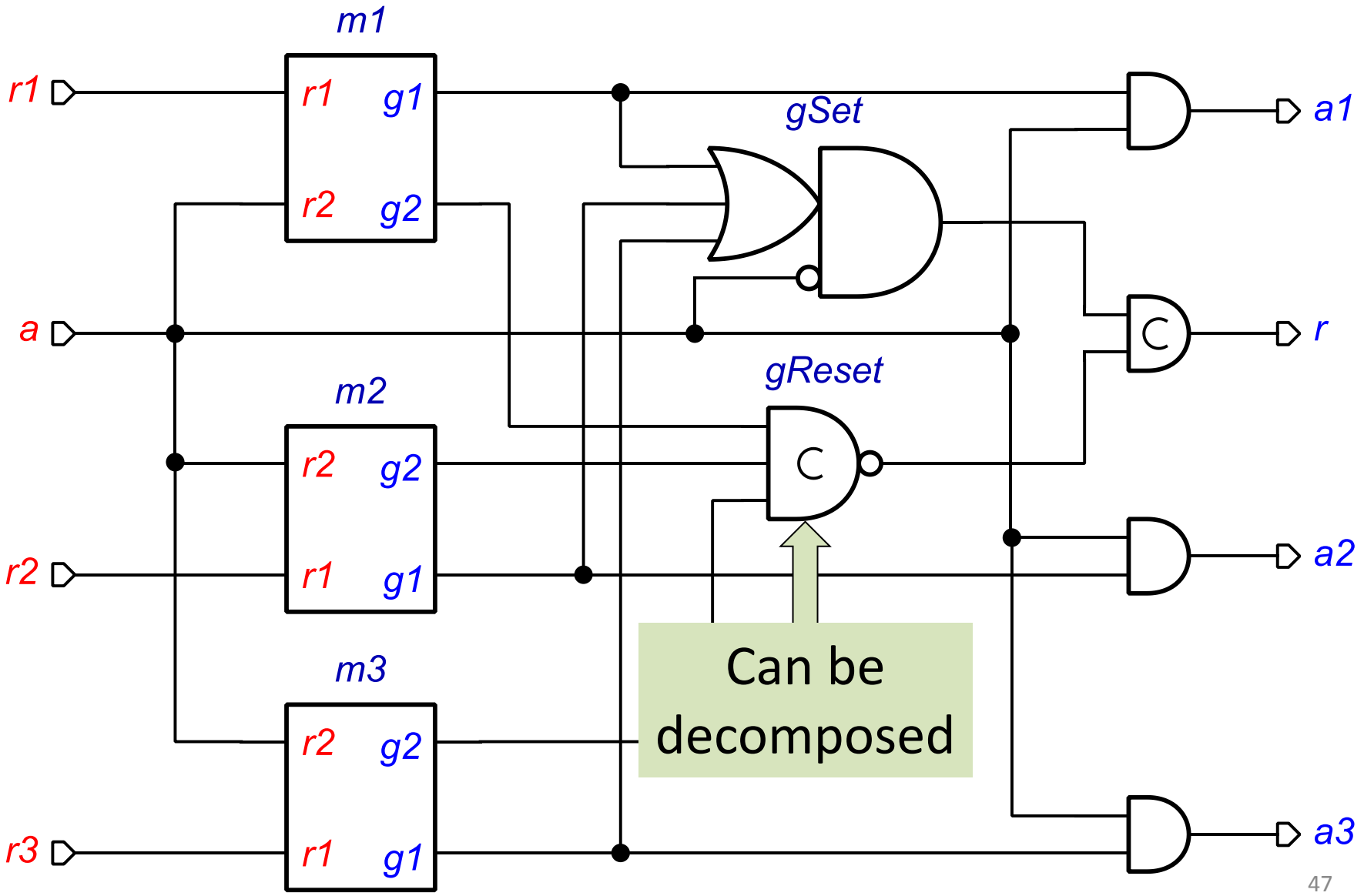


Fair mutexes do not permit sequential bundling

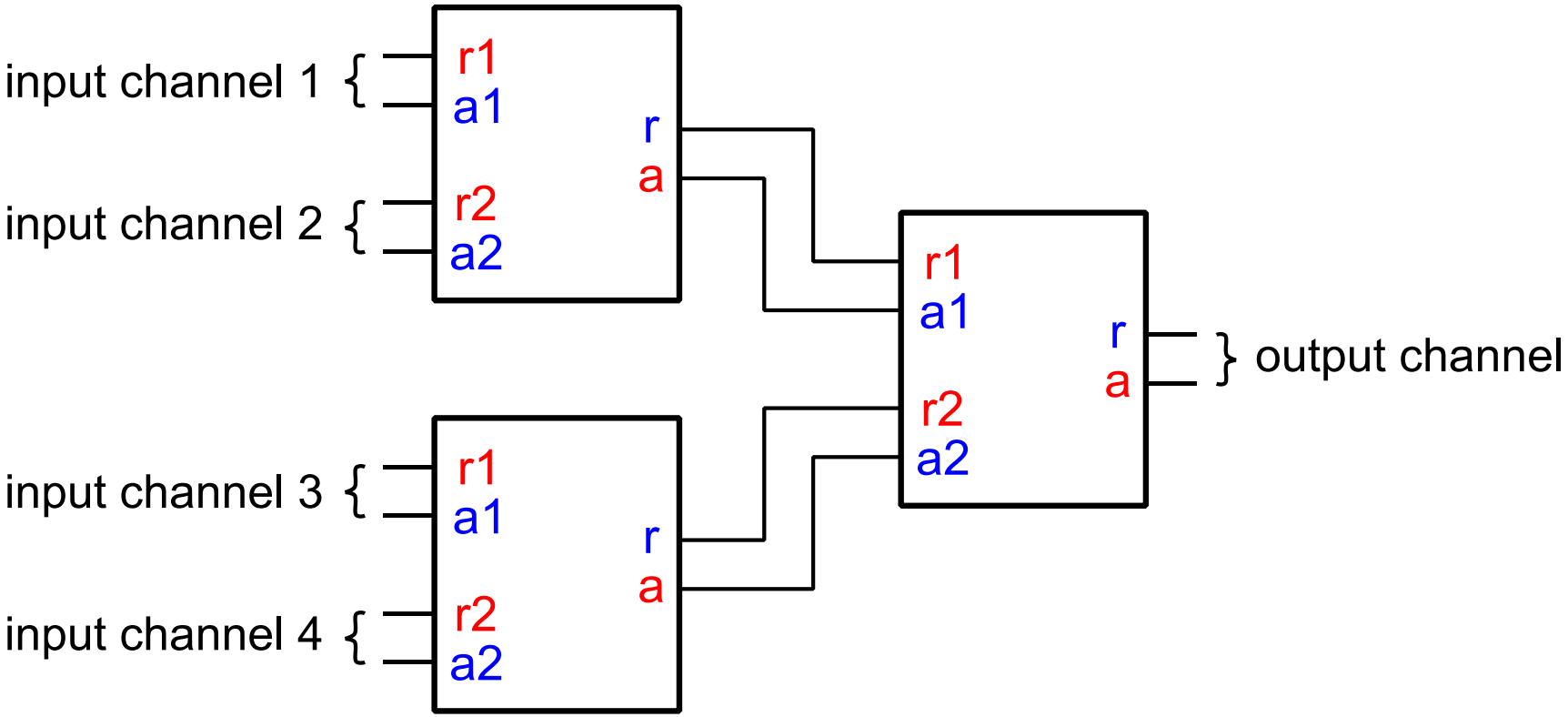
Scaling to more inputs



Scaling to more inputs



Scaling to more inputs



Conclusion

- New reusable asynchronous component – surprisingly difficult for just 3 handshakes!
- Fast implementation – no metastability on critical path
- Discovered *fairness-based optimisation*
- Scalable
- Formally verified using Workcraft and Versify
- To be integrated into a real multiphase buck

- Challenge for asynchronous community:

Design OM in a non-monolithic way

(how to design it without a miracle?)

Thank you!

Opportunistic bundling of questions is encouraged (fairness assumption on the session chair to prevent sequential bundling) ☺