Presentation

GRA PORJECT

GROUP 174

ÜBERSICHT

PROBLEM-STELLUNG

Praktische Nutzung des TLBs

1-Beschleunigung von Speicherzugriffen
(Caching)

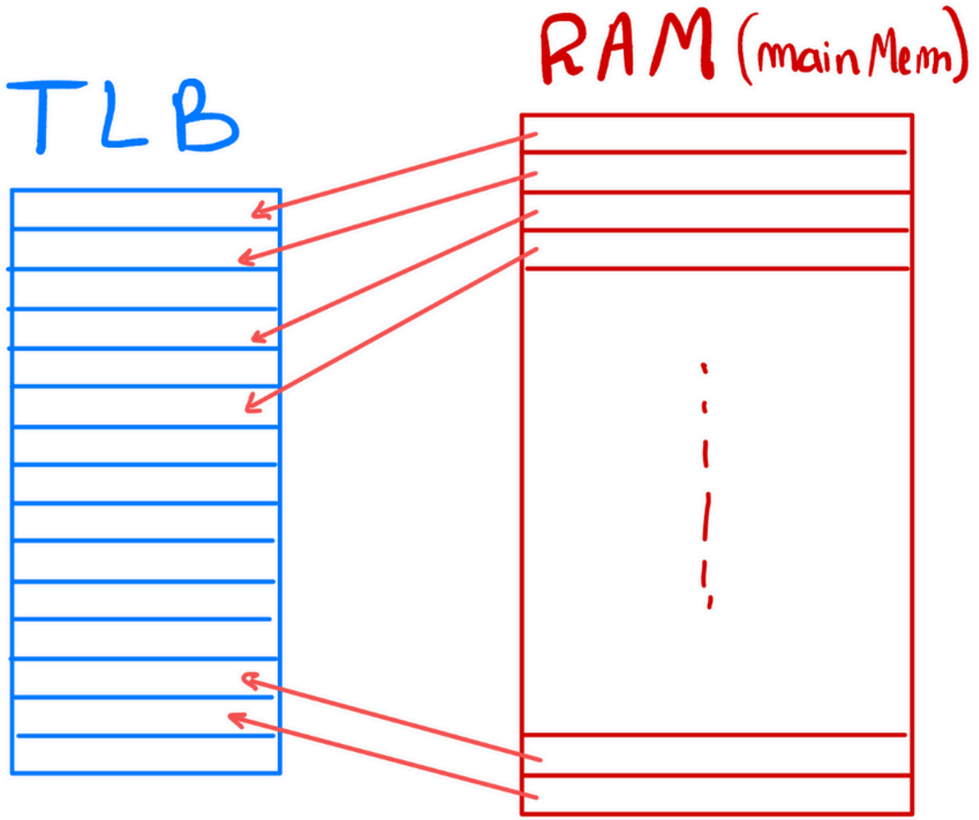
2-Effizienzsteigerung in Mehrbenutzersystemen
(Isolation und Sicherheit)

3-Verbesserte Systemleistung

PROBLEM-STELLUNG

TLB-Eigenschaften

Direct Mapped TLB



PROBLEM-STELLUNG

TLB-Eigenschaften

1-Beschleunigung von Speicherzugriffen

2-Effizienzsteigerung in Mehrbenutzersystemen

3-Verbesserte Systemleistung

SIMULATIONSANSATZ UND IMPLEMENTIERUNG

SIMULATIONS-ANSATZ

Wie wird das TLB-Verhalten nachgebildet?

- Nutzung maximaler Hardwarekomponenten

- Beginn mit einem Hardwaredesign (Logisim evolution) zur Minimierung der Softwareabstraktion

- Geringerer Einsatz von Datenstrukturen

SIMULATIONS-ANSATZ

Wie es auf reale Situationen anwendbar gemacht wird?

-Integration des virtuellen Speichers

-Virtuelle Lookup Table

-Kommunikation mit einem virtuellen Prozessor

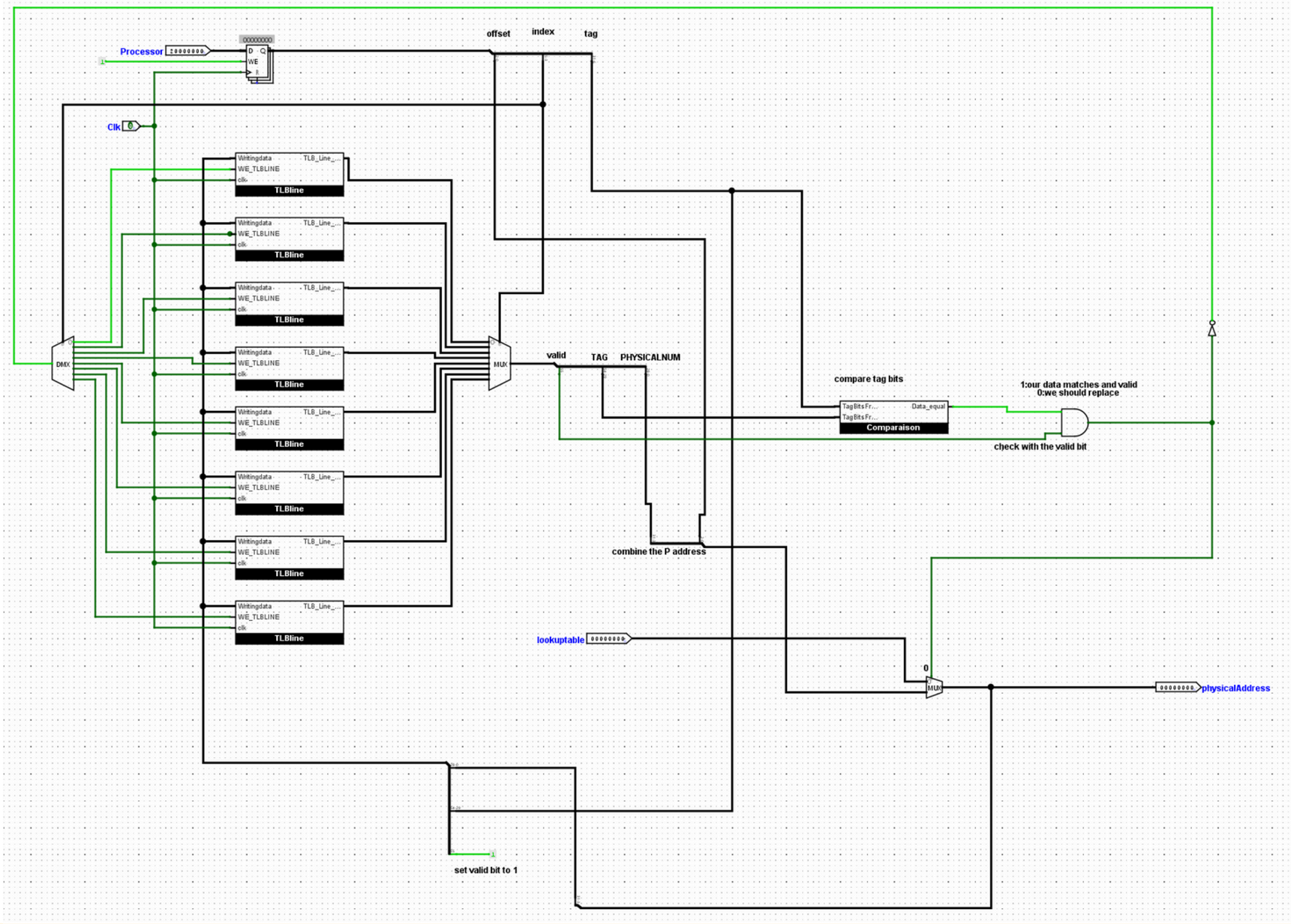
IMPLEMENTIERUNG

Rahmenprogramm

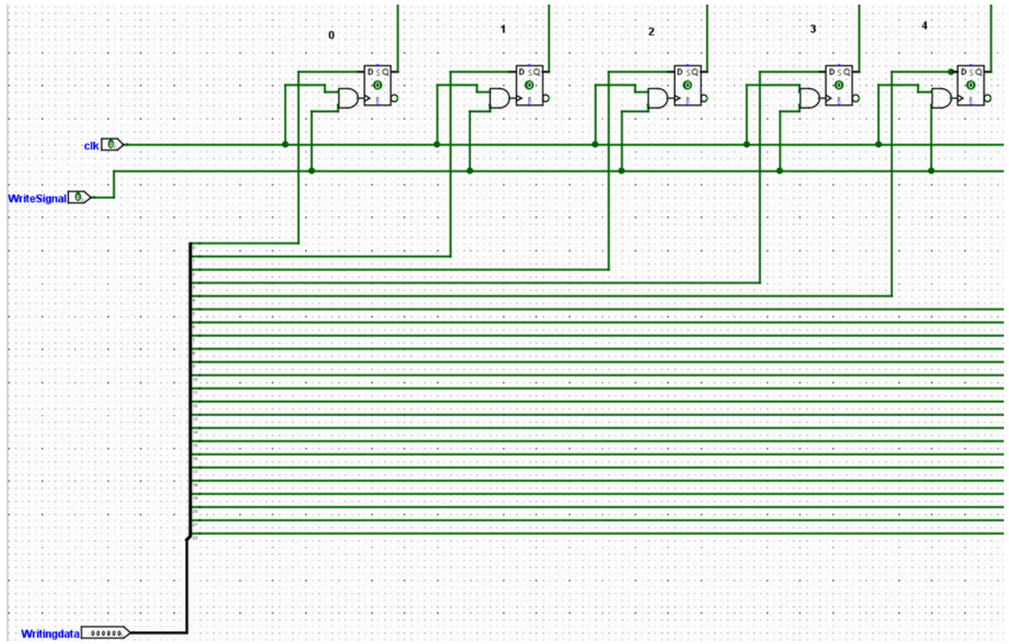
-Abdeckung von Randfällen

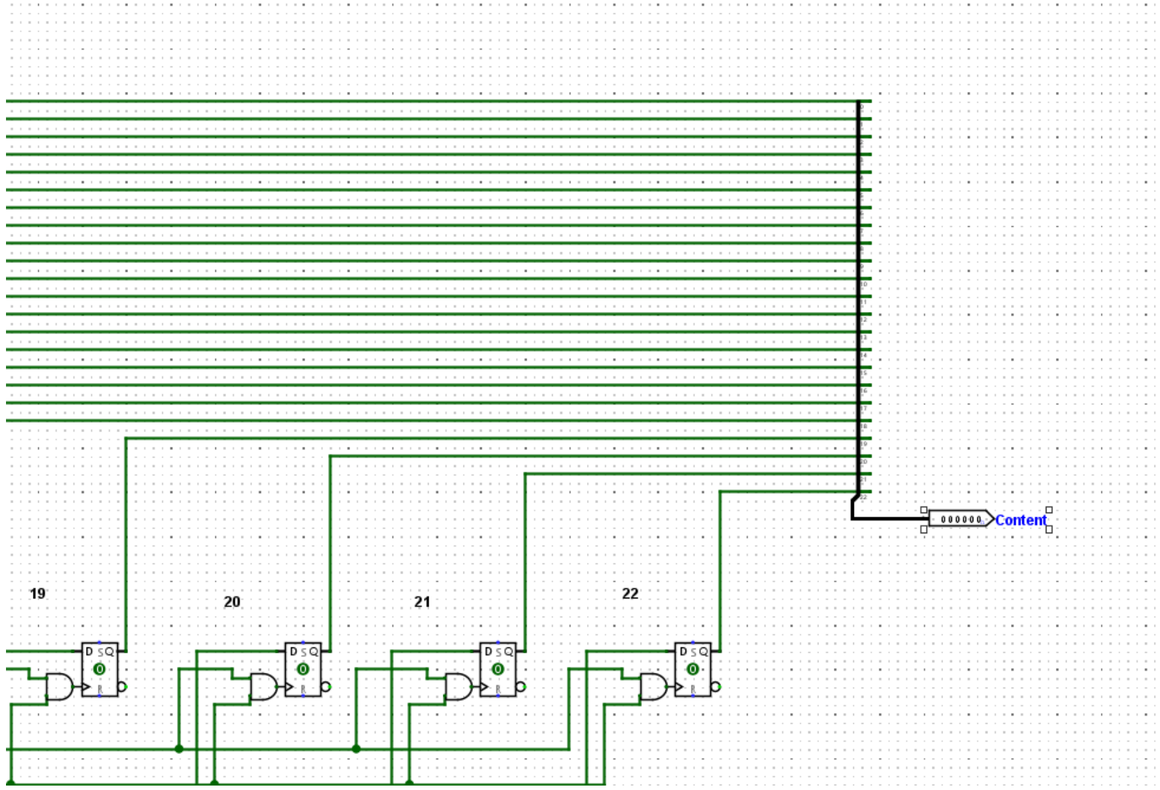
-Benutzerfreundlichkeit

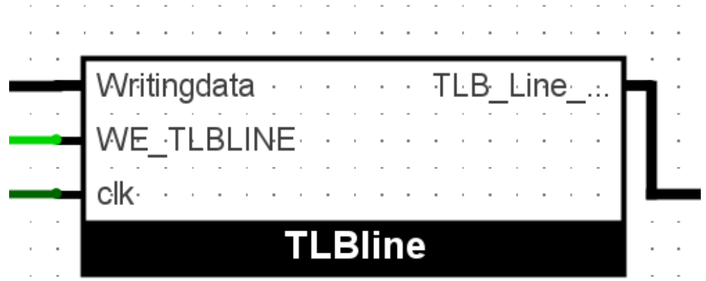
-Sinnvolle Rückmeldungen und Kommentare



VON DER HARDWARE ZU .HPP-MODULEN

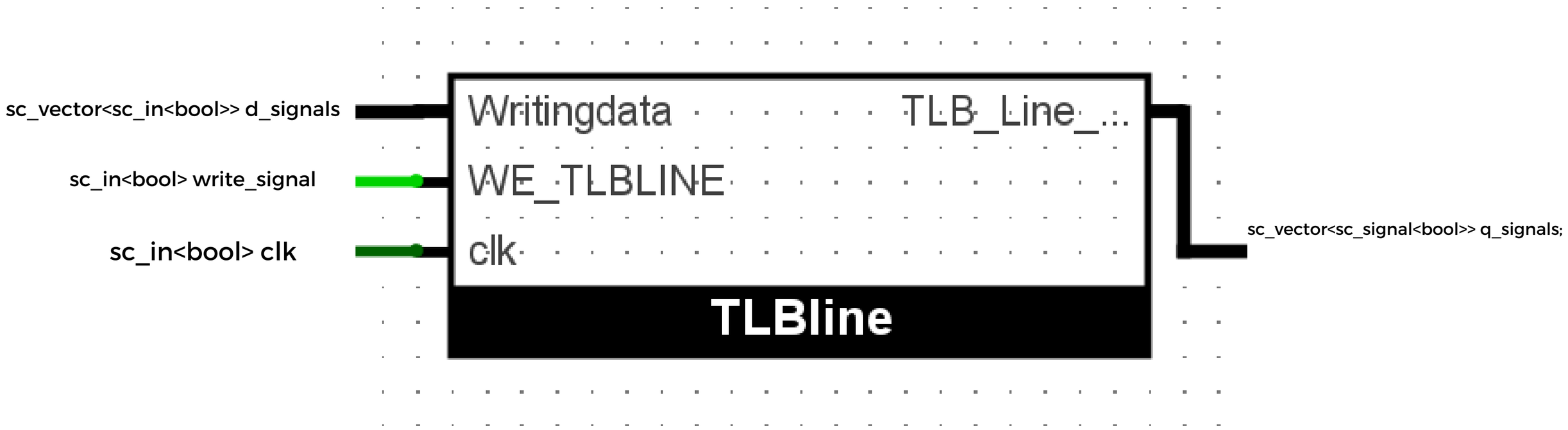

0
1
2
3
4
clk
WriteSignal
Writingdata


19
20
21
22
Content

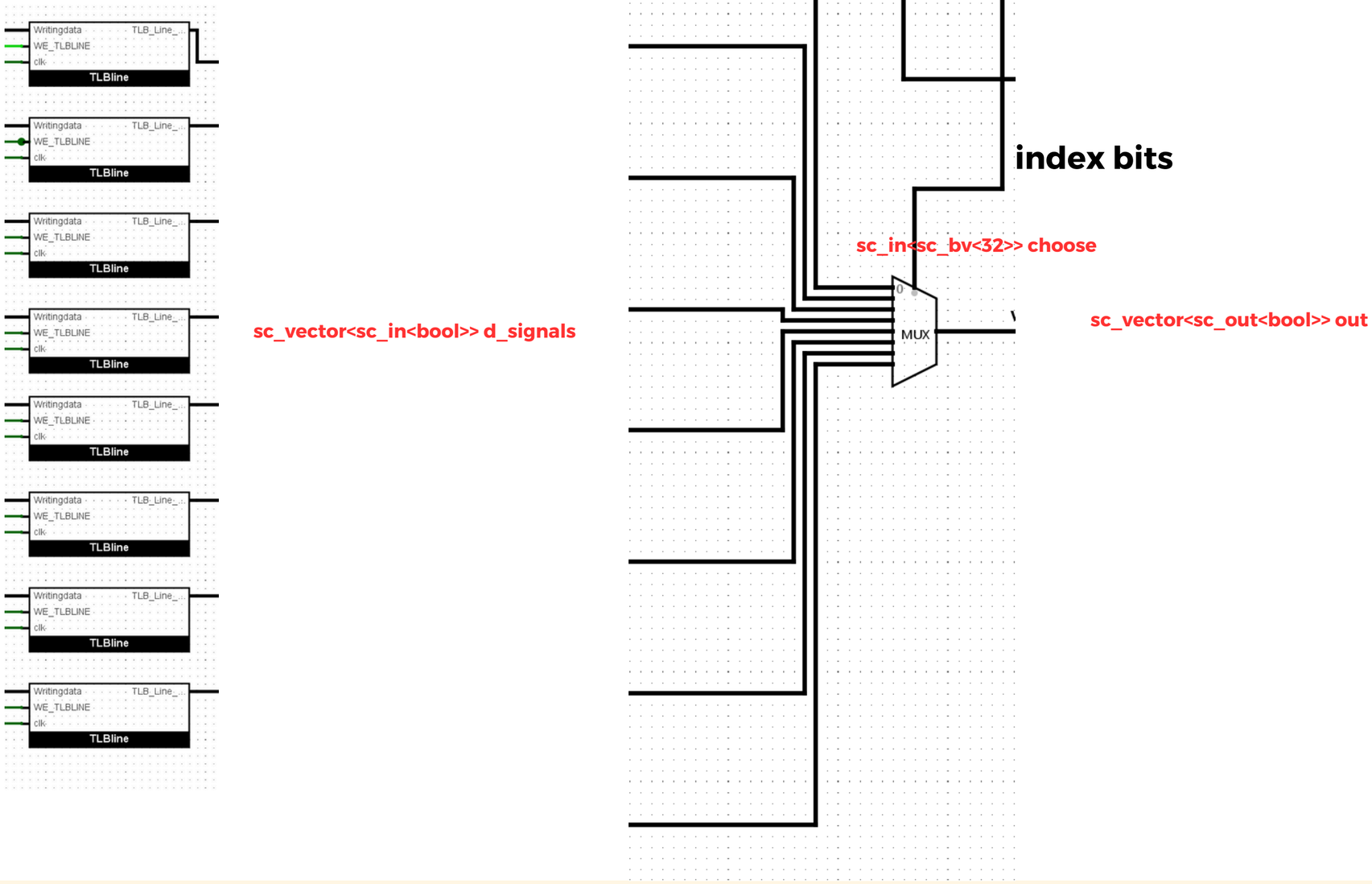

Writingdata
TLB_Line_...
WE_TLBLINE
clk
TLBline

TLB line

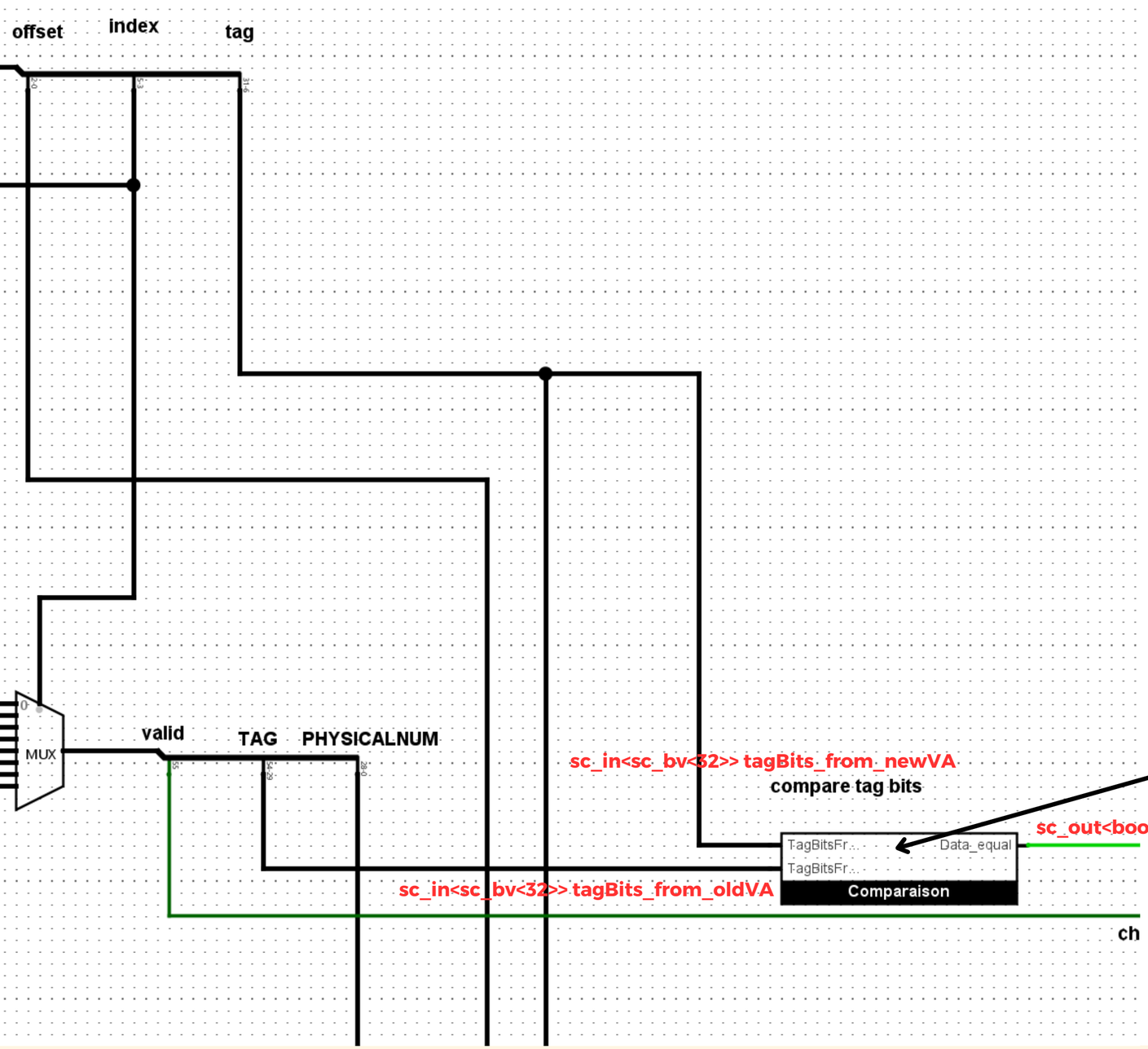
sc_vector<sc_signal<bool>> q_bar_signals
sc_vector<D_FLIP_FLOP> dflipflops
sc_signal<bool> tlb_line_clk
AND_GATE and_gate



TLB line

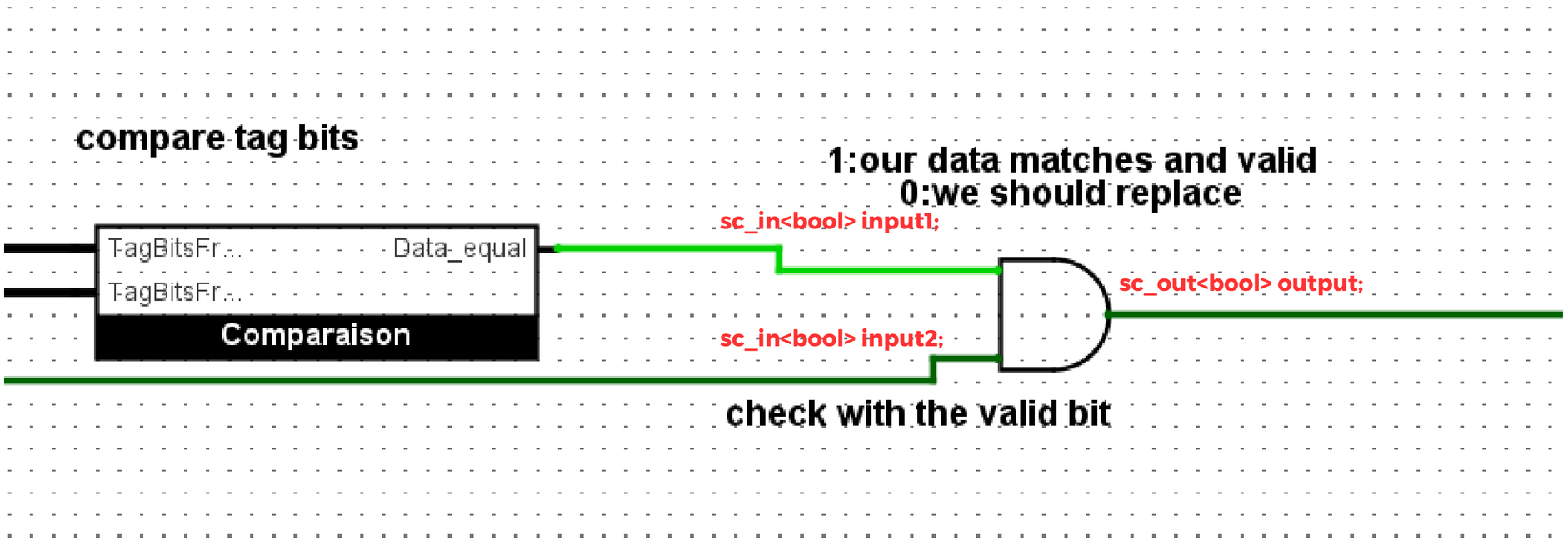


Multiplexer

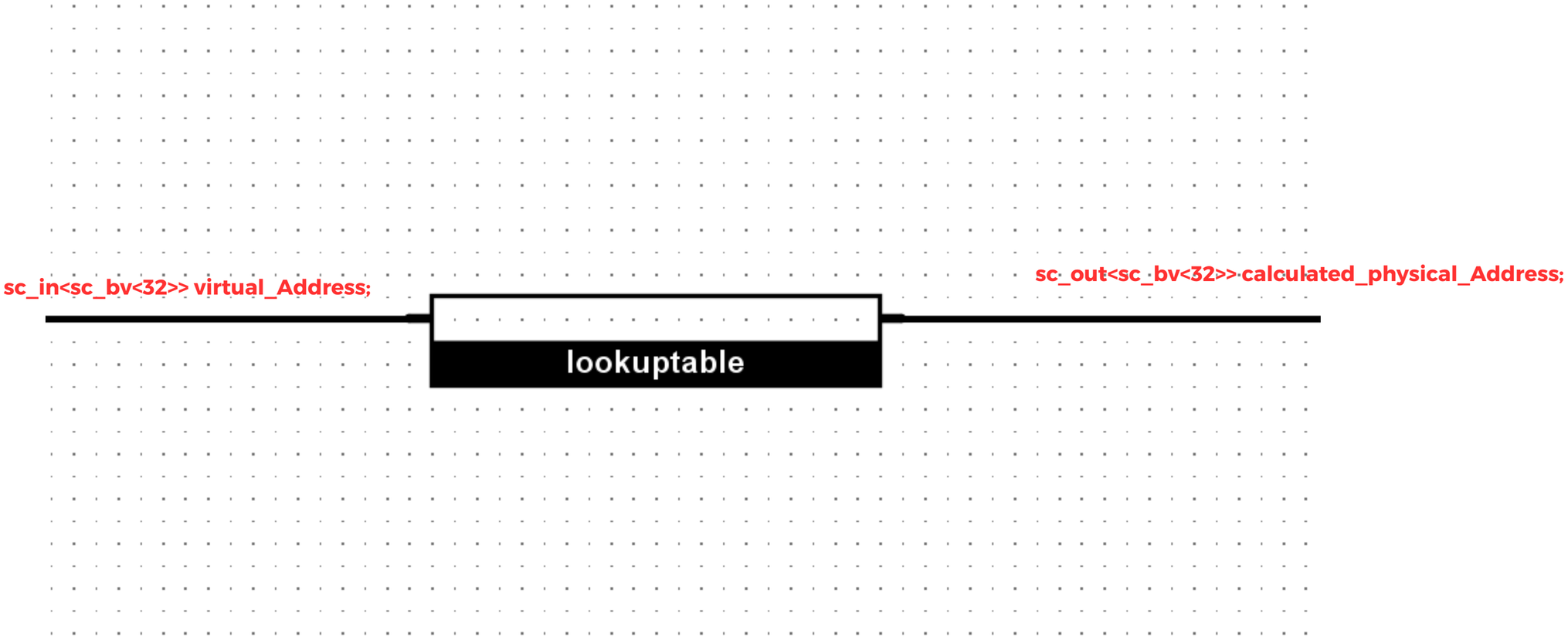

offset
index
tag
MUX
valid
TAG
PHYSICALNUM
sc_in<sc_bv<32>> tagBits_from_newVA
compare tag bits
TagBitsFr...
TagBitsFr...
Data_equal
Comparaison
sc_out<bool> out;
sc_in<sc_bv<32>> tagBits_from_oldVA
ch


TagBitsFr...
TagBitsFr...
Data_equal
Comparaison

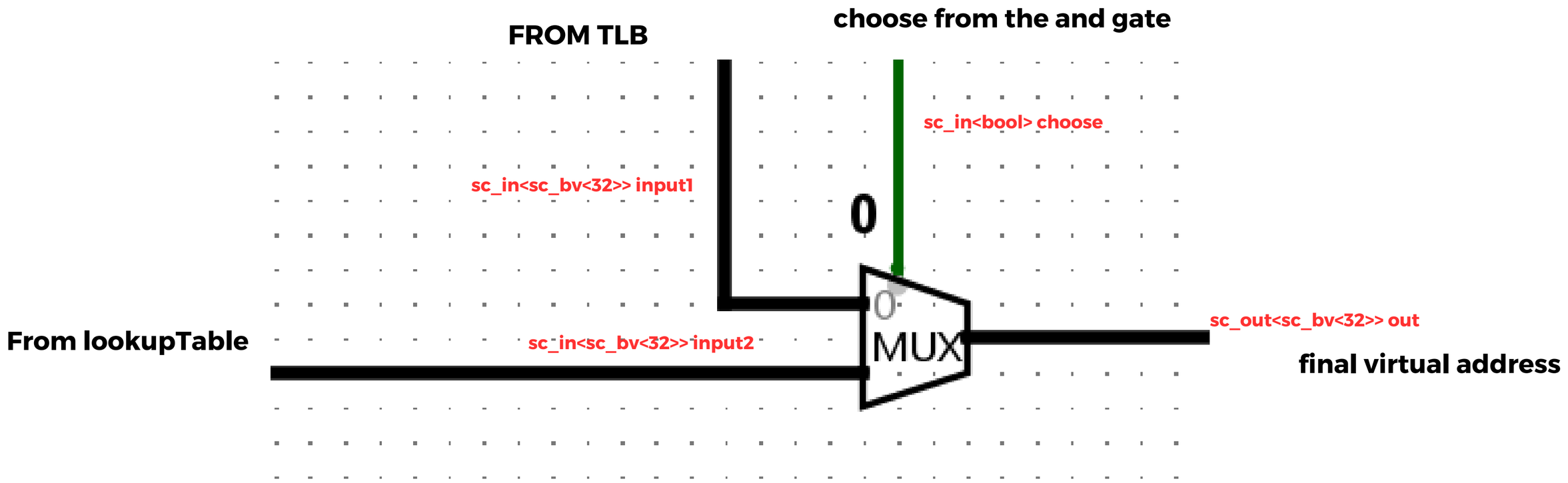
Comparator



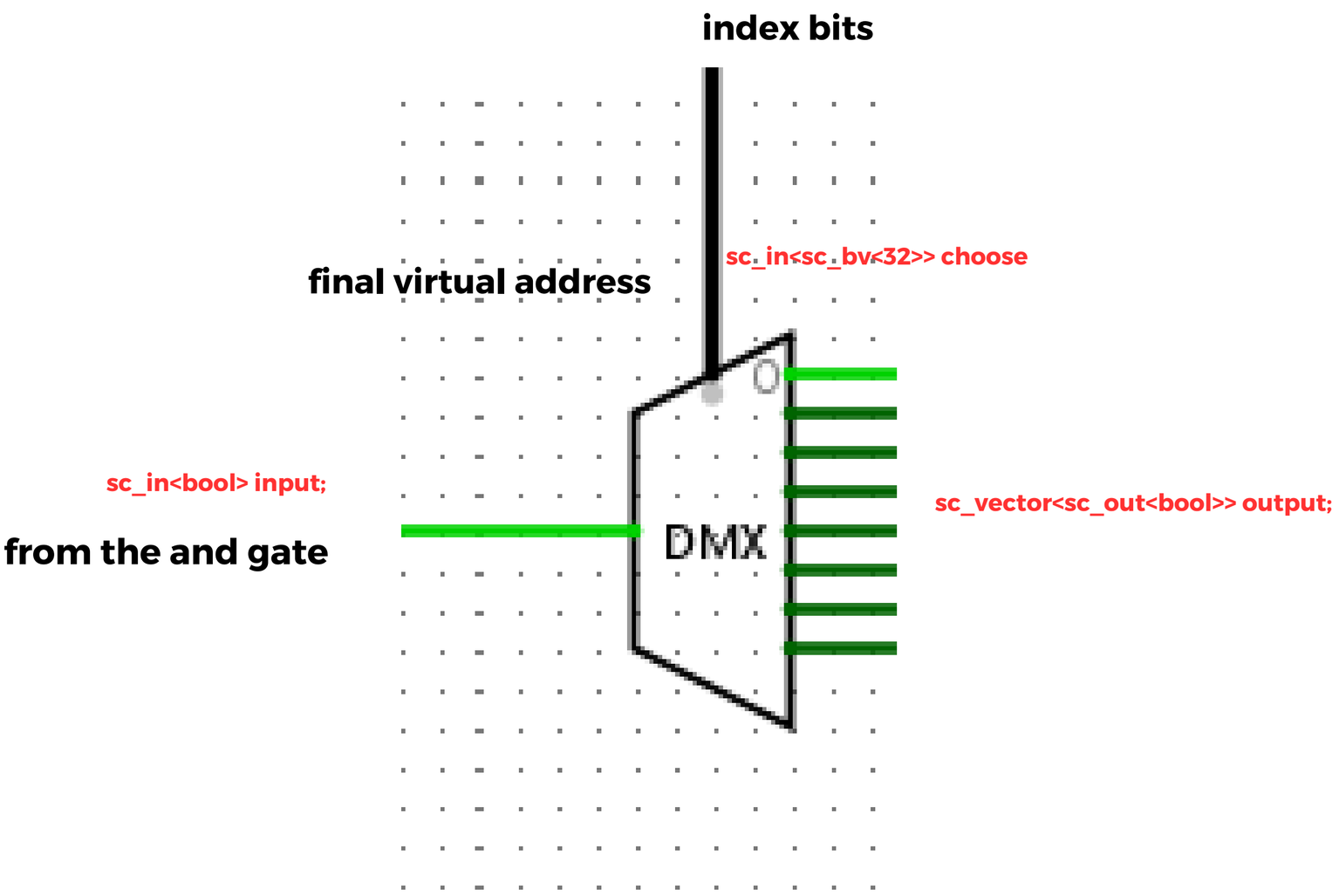
And gate and validity check

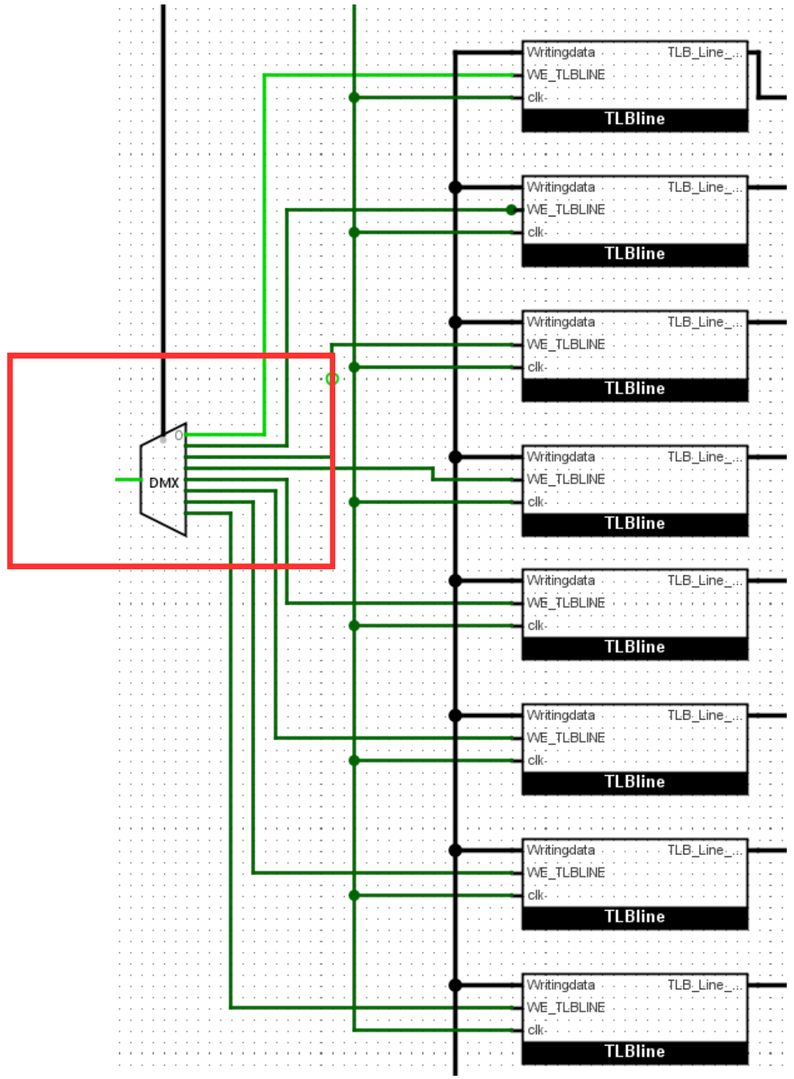


Lookuptable

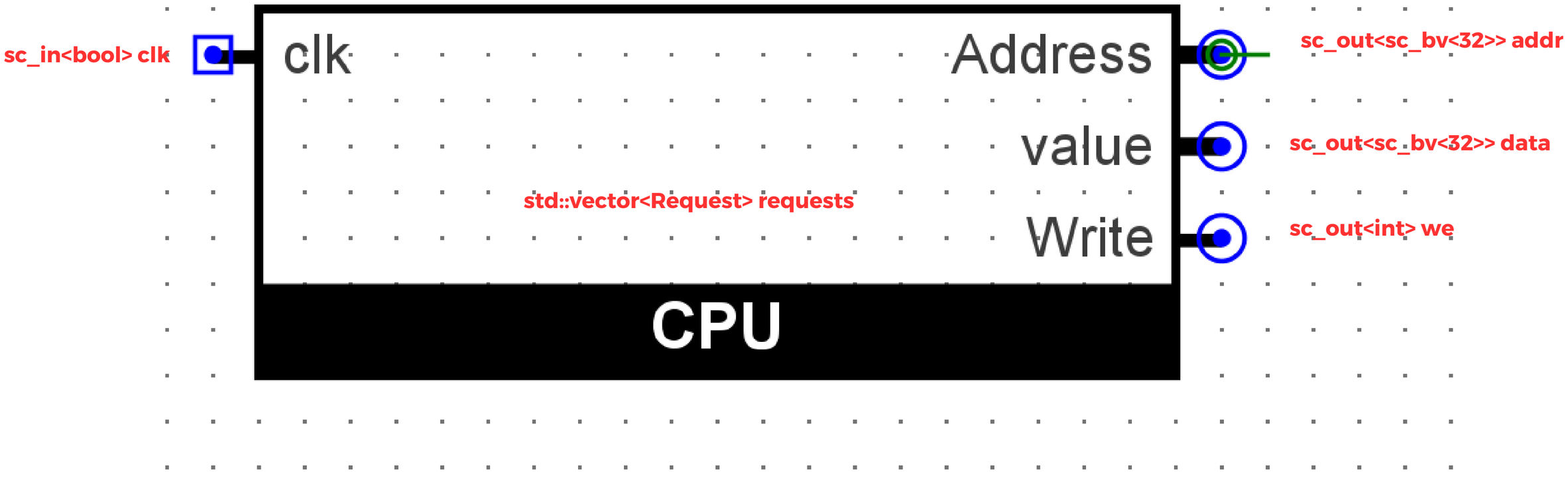


2nd Multiplexer

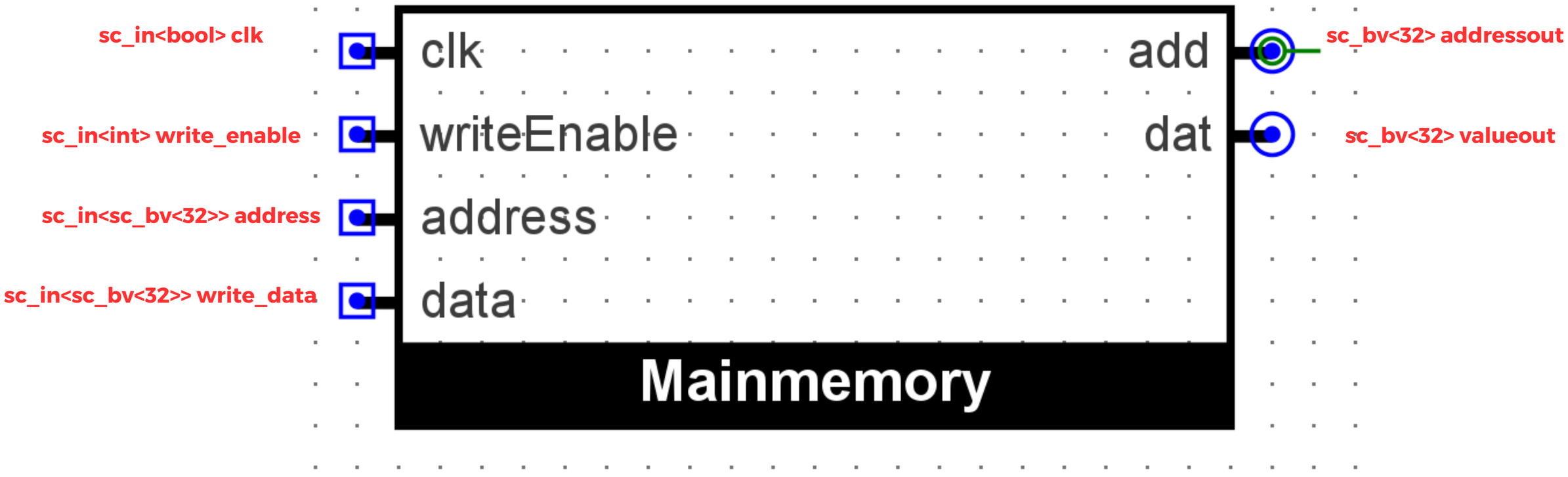




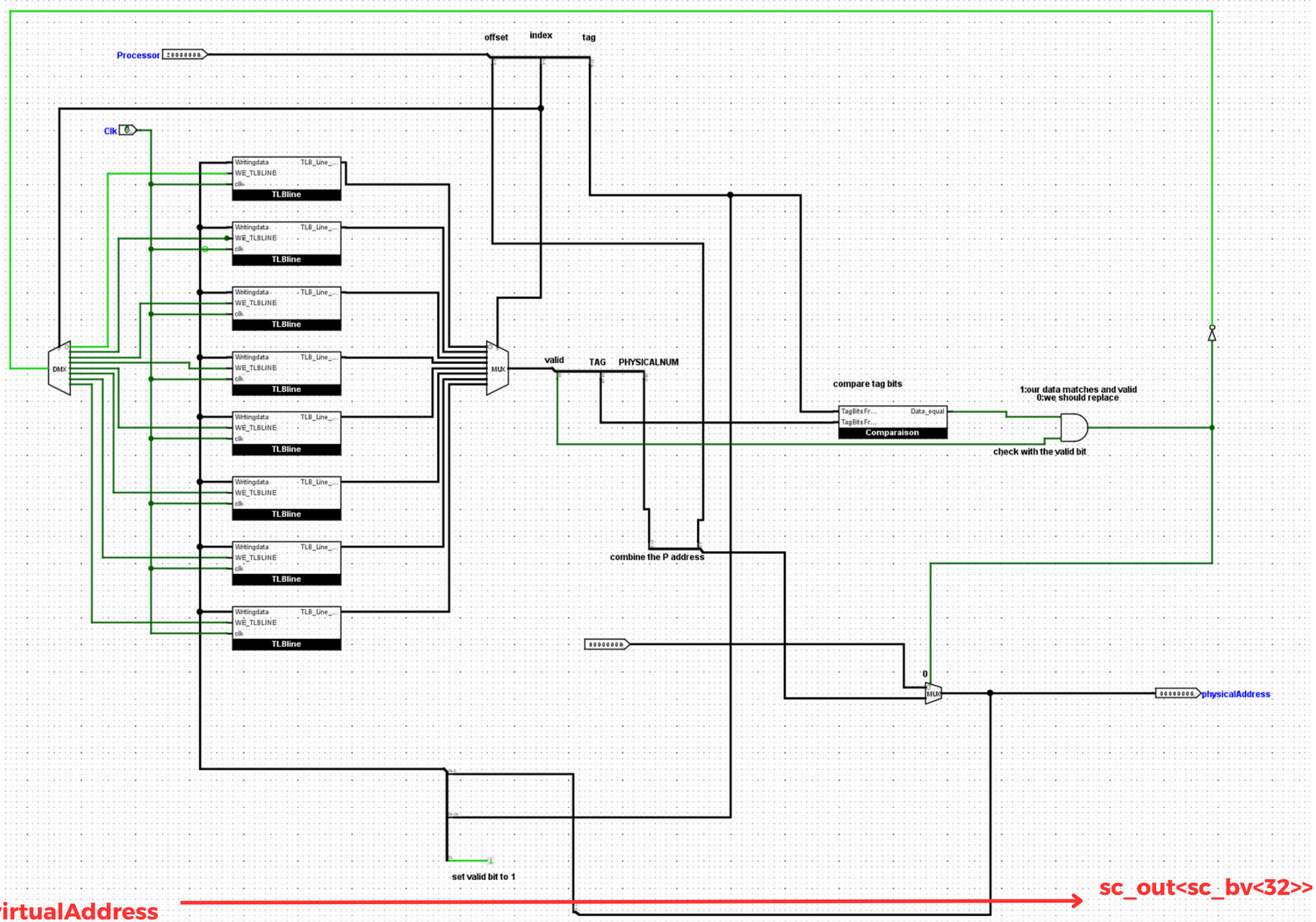
Demultiplexer


sc_in<bool> clk
clk
Address
sc_out<sc_bv<32>> addr
value
sc_out<sc_bv<32>> data
std::vector<Request> requests
Write
sc_out<int> we
CPU

CPU



Main Memory(fallende Flanke schreiben)



sc_in<bool> clk;
sc_in<sc_bv<32>> virtualAddress

sc_out<sc_bv<32>> physicalAddress

TLB mit allen Verbindungen

WEITERE IMPLEMENTIERUNGS DETAILS

Globale Konstanten

```
extern size hits;
extern size_t misses;
extern size_t numRequests;
extern size_t cycle_counter;
extern int max_cycles;
extern int tlb_line_length;
extern unsigned tlbSize;
extern unsigned blocksize;
extern unsigned tlbsLatency;
extern unsigned memoryLatency;
extern unsigned v2bBlockOffset;
extern unsigned number_of_tagBits;
extern unsigned number_of_offsetBits;
extern unsigned number_of_tlb_indexBits;
```

Namespaces :Namenskonflikte zu vermeiden und Code zu organisieren

WEITERE IMPLEMENTIERUNGS DETAILS

Latenz Berechnung

Hit latency= TLB latency+ memory latency

Hit latency= TLB latency+ 2*memory latency

Miss penalty= memory latency

WEITERE IMPLEMENTIERUNGS DETAILS

primitiv Gatteranzahl

DeMultiplexer= log_2(TLB_SIZE) + 2 * TLB_SIZE

D FlipFlop= (TLB_LINE_LENGTH * TLB_SIZE) * 4

Multiplexer= log_2(TLB_SIZE) + 64TLB_SIZE - 32

2nd Muxltiplexer= 97 gates

And Gate= 2

Not Gate = 1

Comparator = 63

WEITERE IMPLEMENTIERUNGS DETAILS

Optimierungen

Page zu Page-Zuordnung:

Nutzung einer TLB-Entry für mehrere Adressen oder Adressgruppen.

Verwendung dynamischer Vektoren :

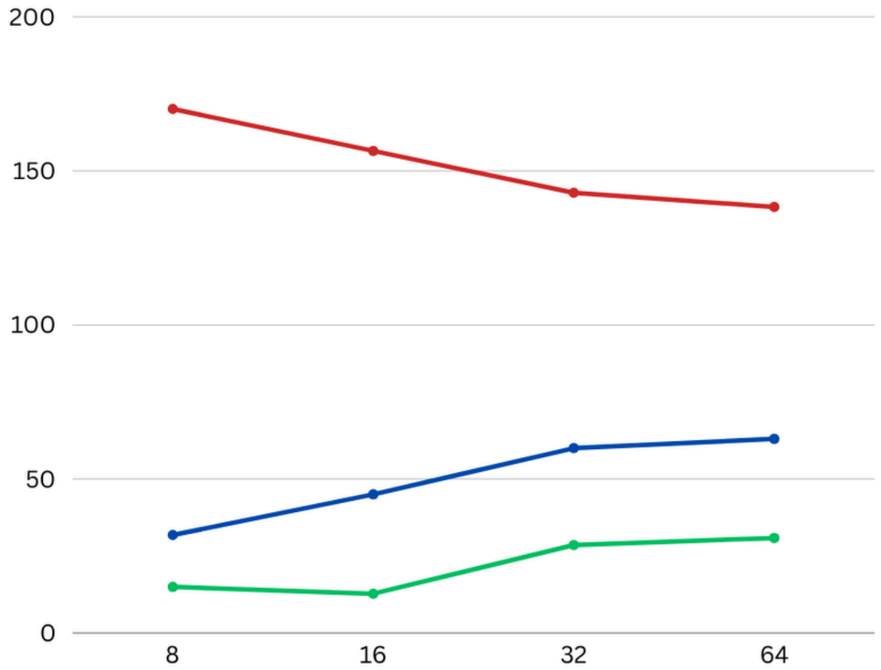
Erreichung der Flexibilität von Hardwarekomponenten.

Testing Units

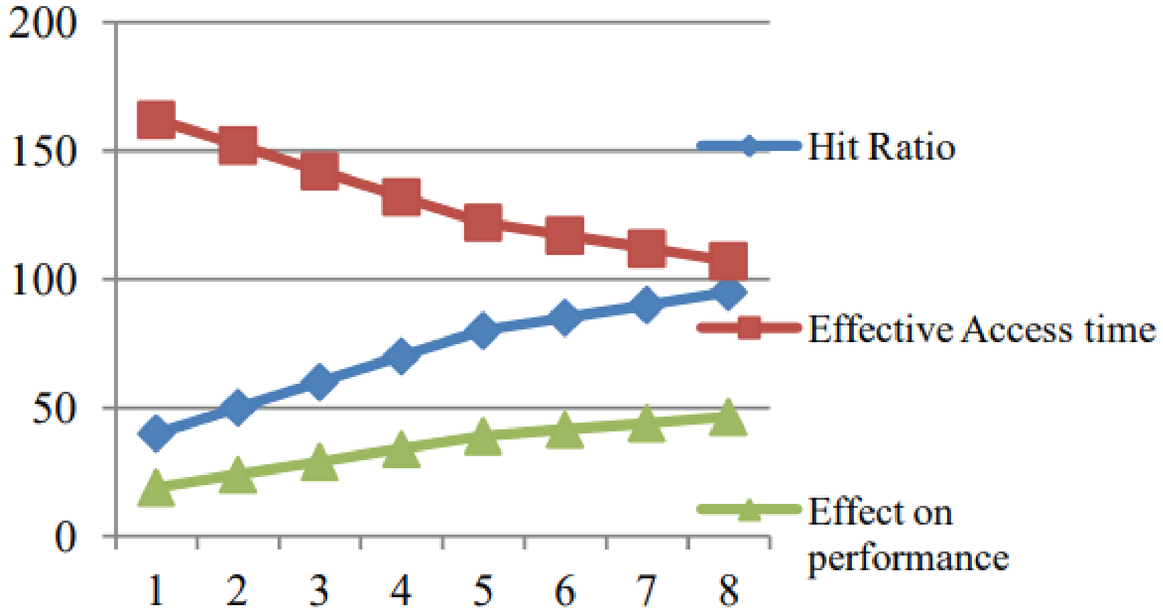
KORREKTHEIT

das Speicherzugriffsverhalten einer Summe über einer verketteten Liste.

Unsere Ergebnisse



Ergebnisse von Banasthali University



KORREKTHEIT

DANKE FÜR IHRE AUFMERKSAMKEIT