Chip Verification

with

Constraint Integer Programming

Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB)

Tobias Achterberg
achterberg@zib.de

Part I

Problem Overview
and
Results

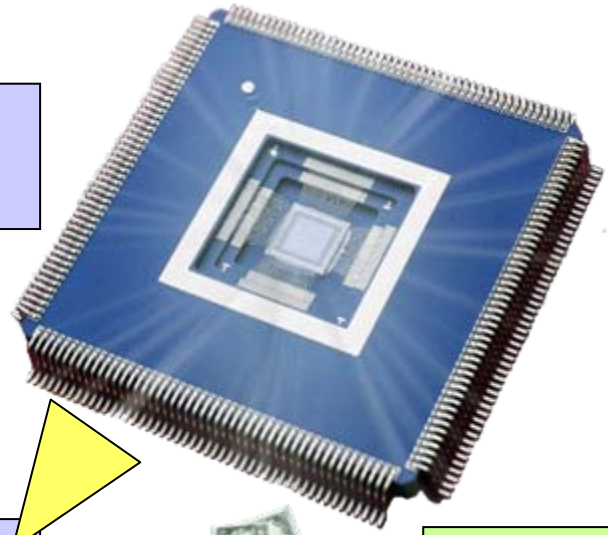
Tobias Achterberg
achterberg@zib.de

Chip Development

```
module minus_test3(a,b,c,o);

   input [7:0] a;
   input [7:0] b;
   input [7:0] c;
   output      o;

   wire [7:0]   t1;
   wire [7:0]   t2;

   assign t1 = a - (b << 2'd1) + (c << 2'd2);
   assign t2 = a - 4'd2*(b + 4'd2*c);
   assign o = !(t1 == t2);


endmodule
```

Chip Design

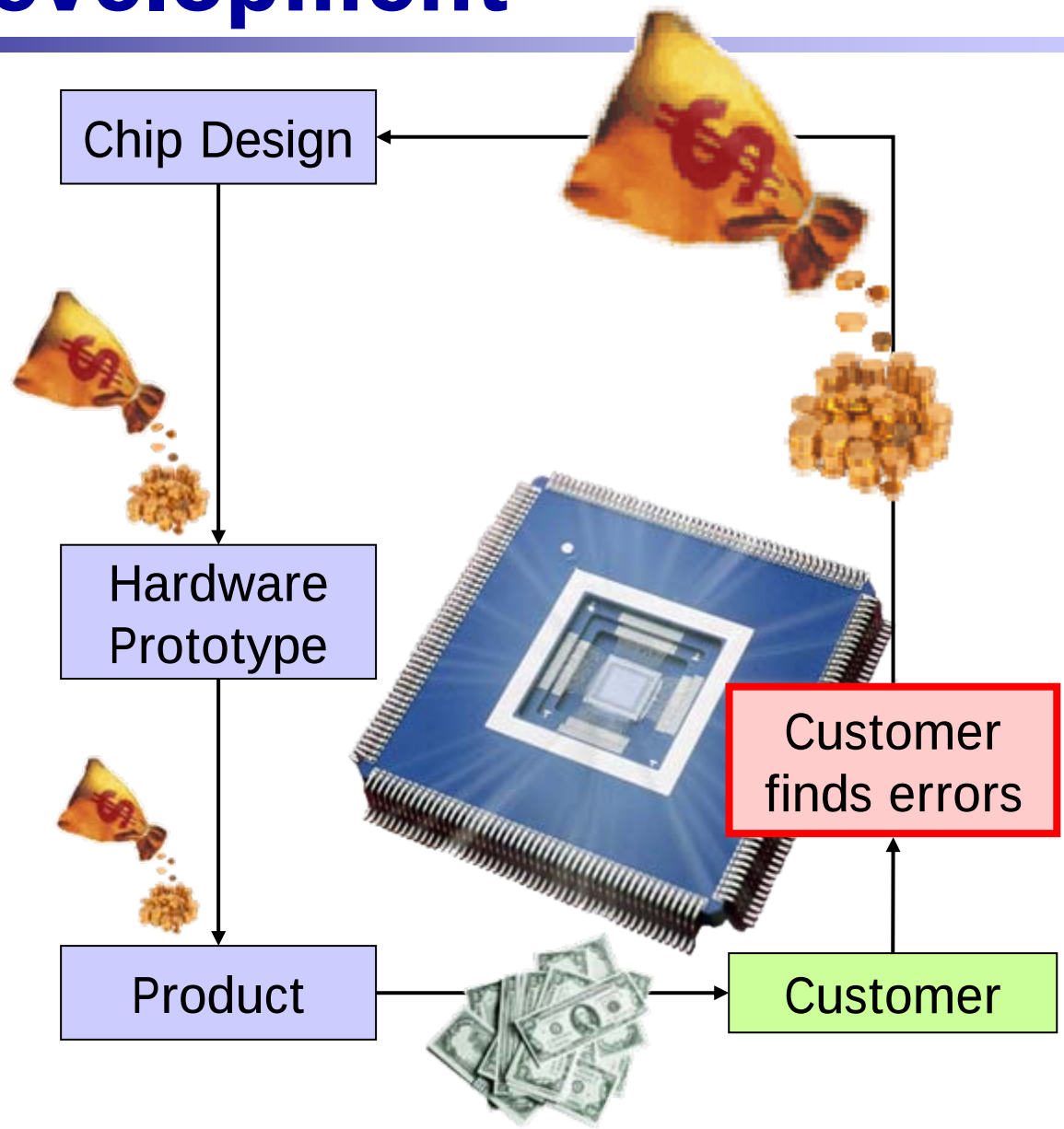
Hardware Prototype



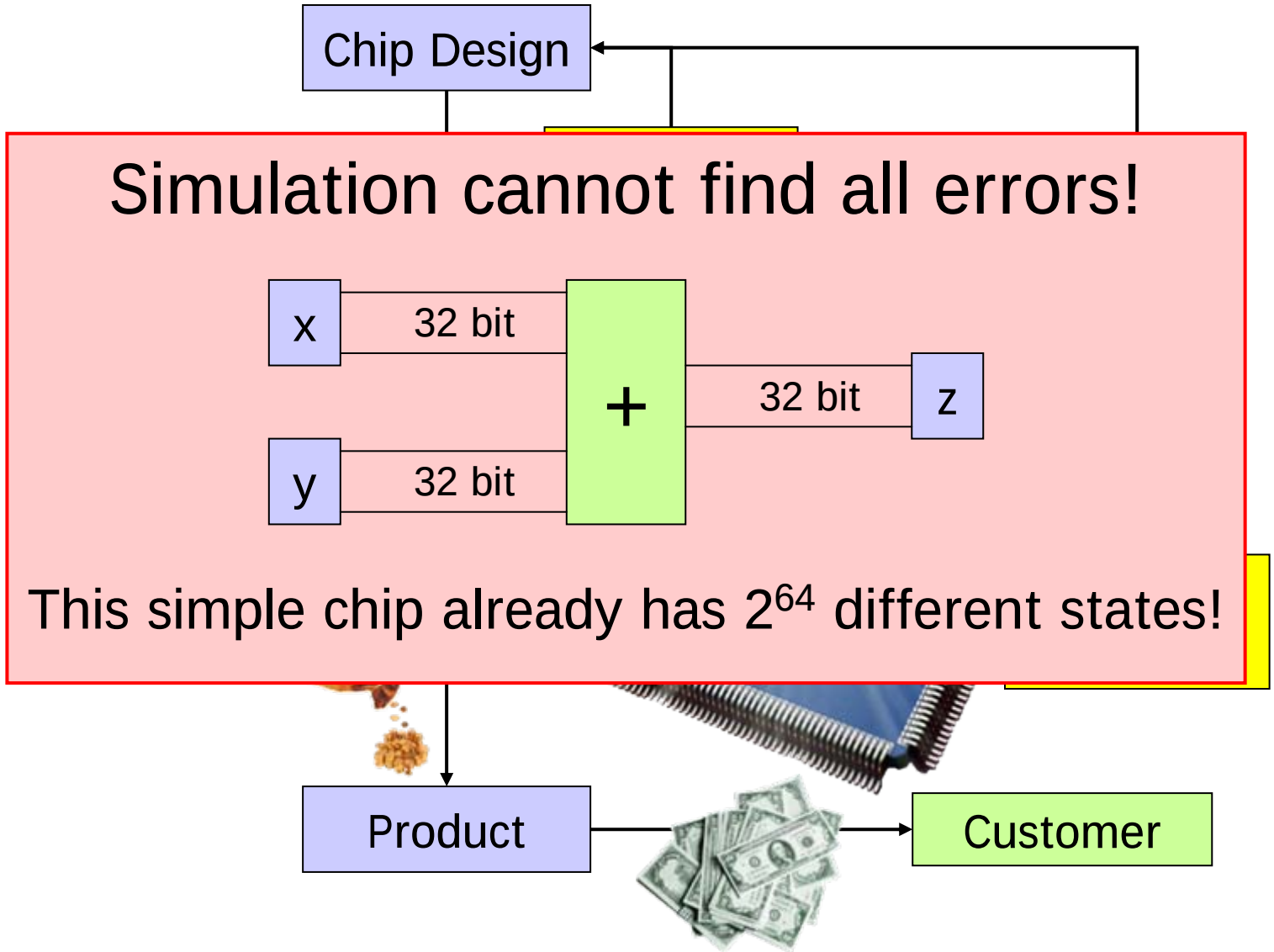
Product

Customer

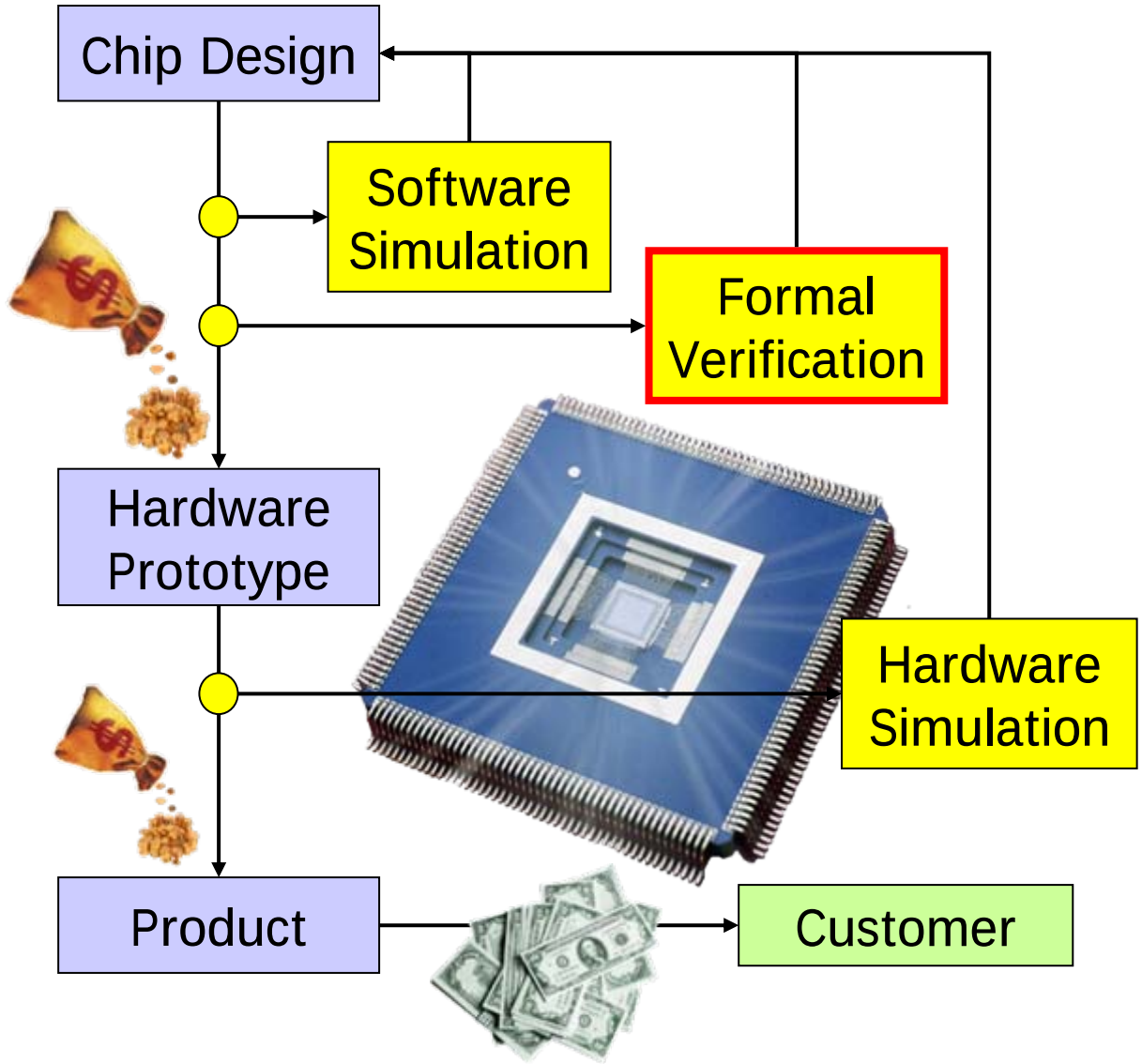
Chip Development



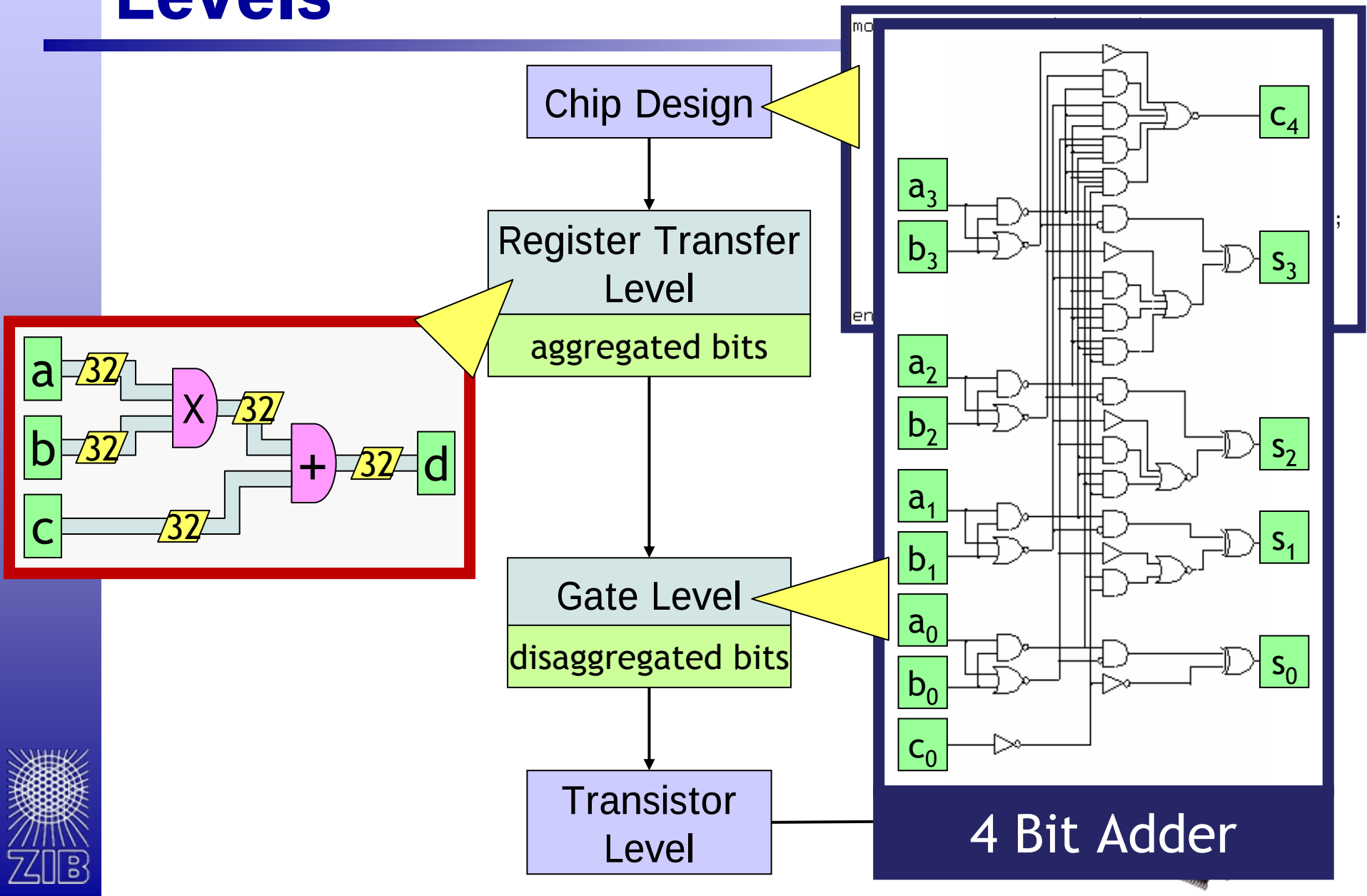
Chip Verification: Simulation



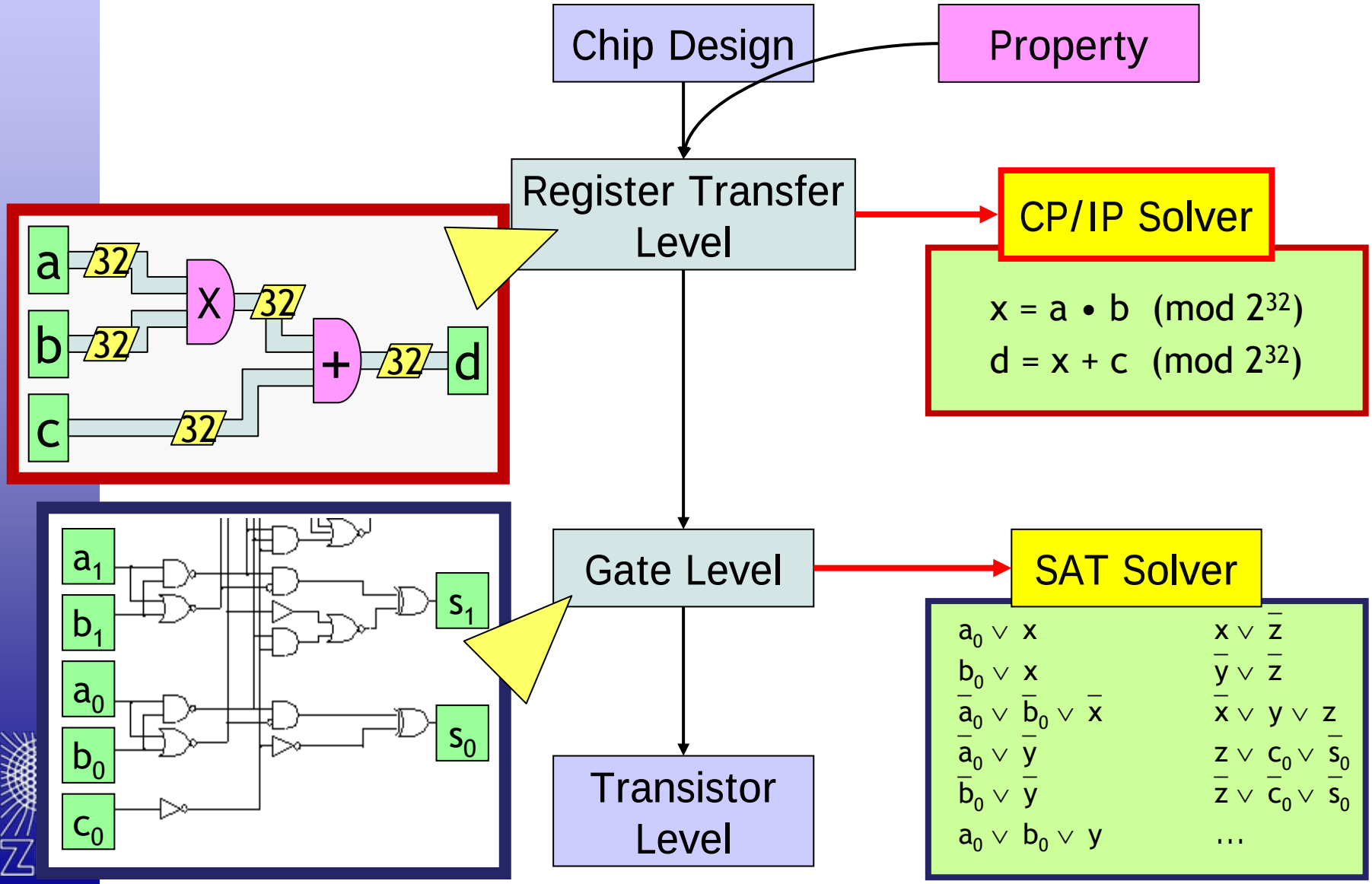
Formal Chip Verification



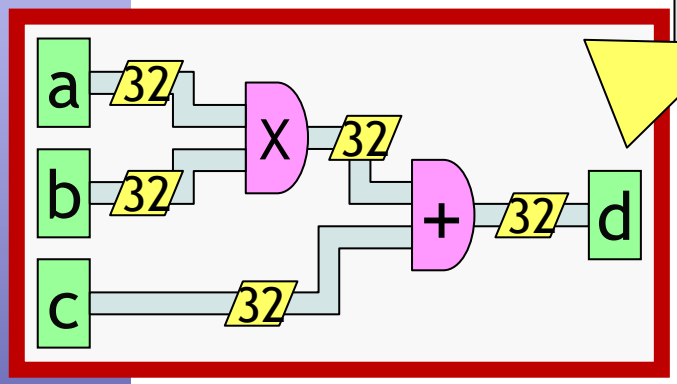
Chip Design: Transformation Levels

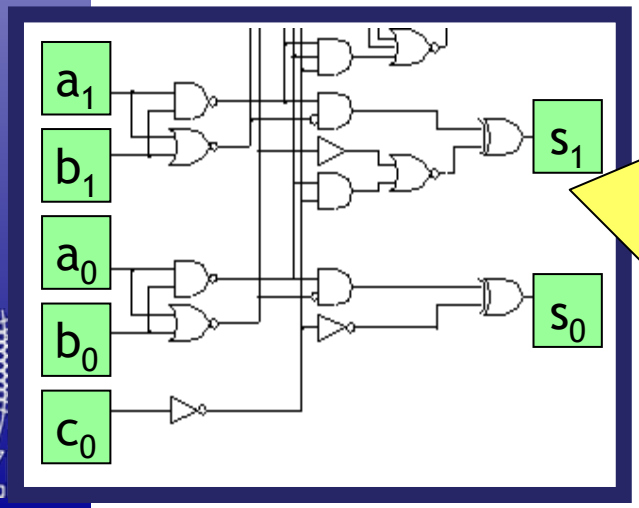


Chip Verification: Property Checking



Chip Verification: Property Checking





Chip Design

Property

Register Transfer Level

CP/IP Solver

4-Mult

186 register variables
153 complex constraints
15.8 seconds

Gate Level

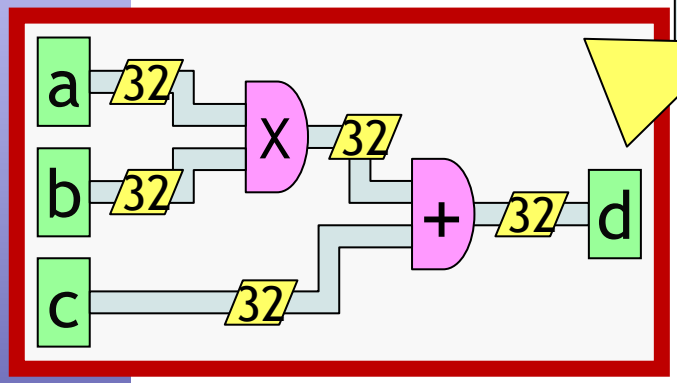
SAT Solver

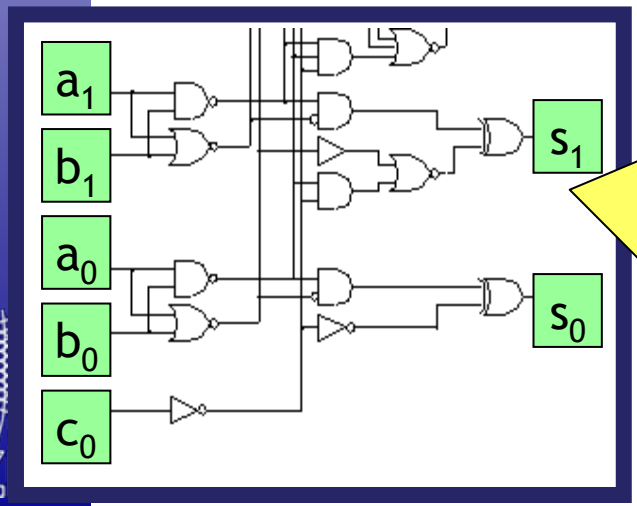
4-Mult

74 025 bit variables
219 485 simple constraints
not solved in 24 hours

Transistor Level

Chip Verification: Property Checking





Chip Design

Property

Register Transfer Level

CP/IP Solver

Gate Level

Transistor Level

4-Mult	186 register variables 153 complex constraints 15.8 seconds
Filter	1 579 register variables 1 493 complex constraints 202.3 seconds
Tricore	252 586 register variables 250 853 complex constraints

Part II

Integer Programming Model

Tobias Achterberg
achterberg@zib.de

Problem Definition

- variables: $(x_1, \ldots, x_n) \in X$

  $X = X_1 \times \ldots \times X_n$ with $X_i = \{0, \ldots, 2^{w_i} - 1\}$

- constraints: $C_1,\ldots,C_m : X \to \{0,1\}$
- property: $P : X \to \{0,1\}$

- prove: $\forall x \in X : C_1(x) \wedge \ldots \wedge C_m(x) \to P(x)$

- negation: $\exists x \in X : C_1(x) \wedge \ldots \wedge C_m(x) \wedge \neg P(x)$

Problem Definition

- negation: $\exists x \in X : C_1(x) \wedge \ldots \wedge C_m(x) \wedge \neg P(x)$
- Constraint Satisfaction Problem (CSP):

$C_1(x)$

...

$C_m(x)$

$\neg P(x)$

$x_1 \in X_1$

...

$x_n \in X_n$

(CSP) is feasible:

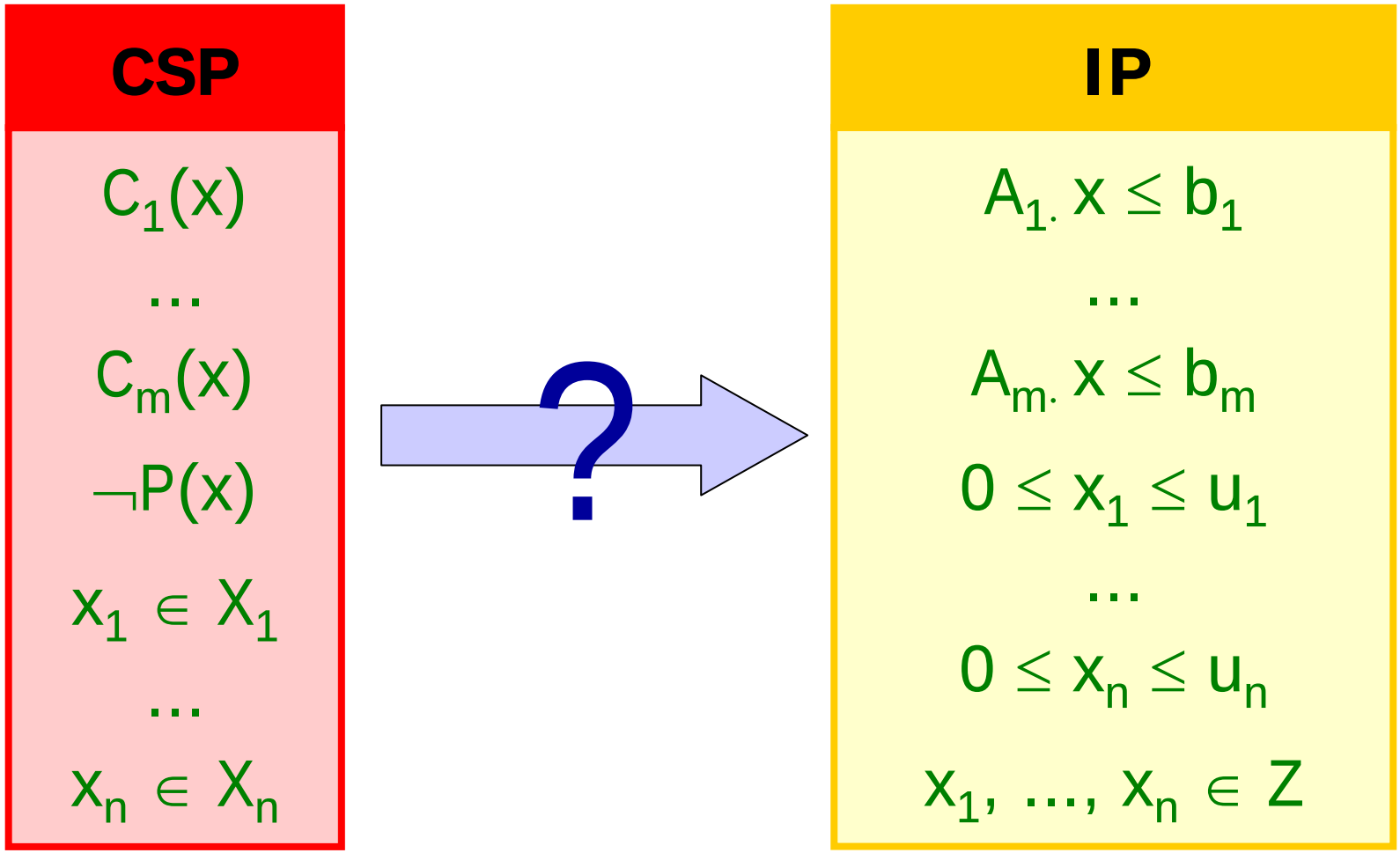
→ counter-example for property

→ property is invalid

(CSP) is infeasible:

→ property is valid

Integer Programming Model



Chip Design Constraints

- bit representation: $X = 2^0 x_0 + \ldots + 2^{w-1} x_{w-1}$
- R = add(X,Y): $R + 2^w o = X + Y$
- R = sub(X,Y): $R - 2^w o = X - Y$
- R = and(X,Y): $x_i + y_i - r_i \leq 1$
  $-x_i + r_i \leq 0$
  $-y_i + r_i \leq 0$
- or, xor: analog

Chip Design Constraints

- r = eq(X,Y):

$$S - T = X - Y$$
$$S \le p\,(U_X - L_Y)$$
$$p \le S$$
$$\Big\} \; p = (S > 0)$$
$$T \le q\,(U_Y - L_X)$$
$$q \le T$$
$$\Big\} \; q = (T > 0)$$
$$p + q + r = 1$$

- r = lt(X,Y):

$$S - T = X - Y$$
$$S \le p\,(U_X - L_Y)$$
$$p \le S$$
$$T \le r\,(U_Y - L_X)$$
$$r \le T$$
$$p + r \le 1$$

Chip Design Constraints

- R = ite(x,Y,Z):

$$R - Y \leq (U_Z - L_Y)\,(1 - x)$$
$$R - Y \geq (L_Z - U_Y)\,(1 - x)$$
$$R - Z \leq (U_Y - L_Z)\,x$$
$$R - Z \geq (L_Y - U_Z)\,x$$

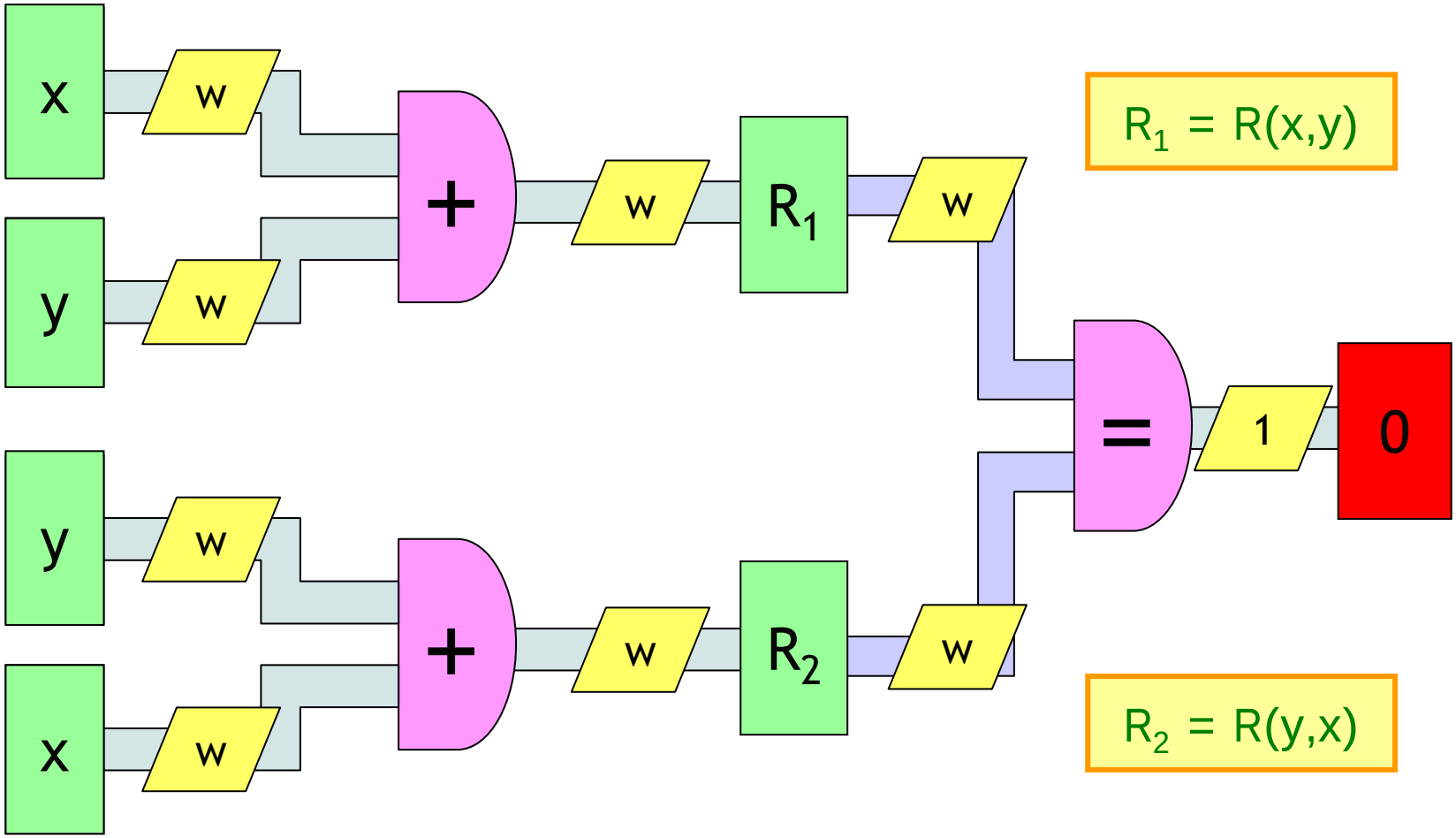
- R = mult(X,Y):
  - use bit representation of X
  - introduce artificial variables for partial products of x_i with nibbles of Y
  - adding partial product variables gives R and overflows

Chip Design Constraints

- R = read(X,Y):
- R = shl(X,Y):
- R = shr(X,Y):
- R = slice(X,Y):
- R = write(X,Y,Z):

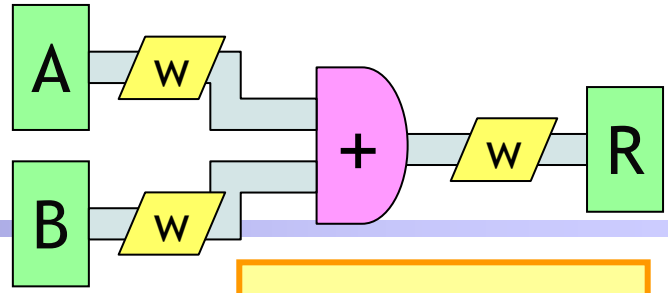
use Constraint Programming techniques

Example 1: 2-Adder



- negated property: $\exists x,y: E(x,y) = 0$

Example 1: 2-Adder



R(x,y) = R(y,x)

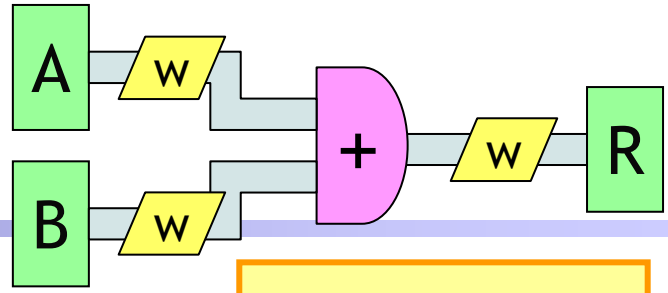
```
param   w       := 8;
set     C       := {1, 2};
var     a[C]    integer >= 0 <= 2^w-1;
var     b[C]    integer >= 0 <= 2^w-1;
var     r[C]    integer >= 0 <= 2^w-1;
var     rovr[C] binary;
var     x       integer >= 0 <= 2^w-1;
var     y       integer >= 0 <= 2^w-1;
var     s       integer >= 0 <= 2^w-1;
var     t       integer >= 0 <= 2^w-1;
var     p       binary;
var     q       binary;
var     e       binary;
```

circuit registers

property variables

auxiliary variables for eq constraint

Example 1: 2-Adder



R(x,y) = R(y,x)

```
minimize          0*x;

subto adder:      forall <i> in C:
                  r[i]          == a[i] + b[i] - 2^w * rovr[i];

subto input_a_1:  a[1]          == x;
subto input_b_1:  b[1]          == y;

subto input_a_2:  a[2]          == y;
subto input_b_2:  b[2]          == x;

subto eq_1:       s - t         == r[2] - r[1];
subto eq_2:       s             <= 2^w * p;
subto eq_3:       p             <= s;
subto eq_4:       t             <= 2^w * q;
subto eq_5:       q             <= t;
subto eq_6:       p + q + e     == 1;

subto prop:       e             == 0;
```

input of 1st adder

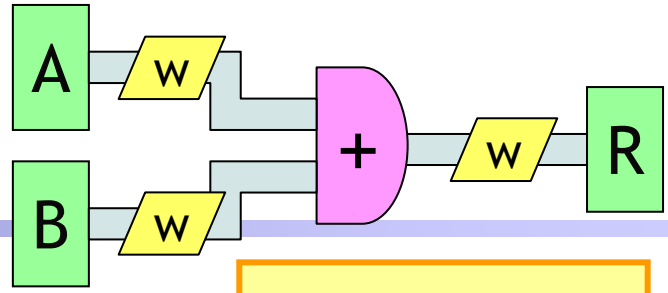
adder (2 copies)

input of 2nd adder

Extended Functions:

subto prop:
vabs(r[2]-r[1]) >= 1;

Example 1: 2-Adder




```
param   w          := 8;
set     C          := {1, 2};
var     a[C]       integer >= 0 <= 2^w-1;
var     b[C]       integer >= 0 <= 2^w-1;
var     r[C]       integer >= 0 <= 2^w-1;
var     rovr[C]    binary;
var     x          integer >= 0 <= 2^w-1;
var     y          integer >= 0 <= 2^w-1;
minimize           0*x;
subto   adder:     forall <i> in C:  r[i]          == a[i] + b[i] - 2^w * rovr[i];
subto   in_a_1:                      a[1]          == x;
subto   in_b_1:                      b[1]          == y;
subto   in_a_2:                      a[2]          == y;
subto   in_b_2:                      b[2]          == x;
subto   prop:                        vabs(r[2]-r[1]) >= 1;
```

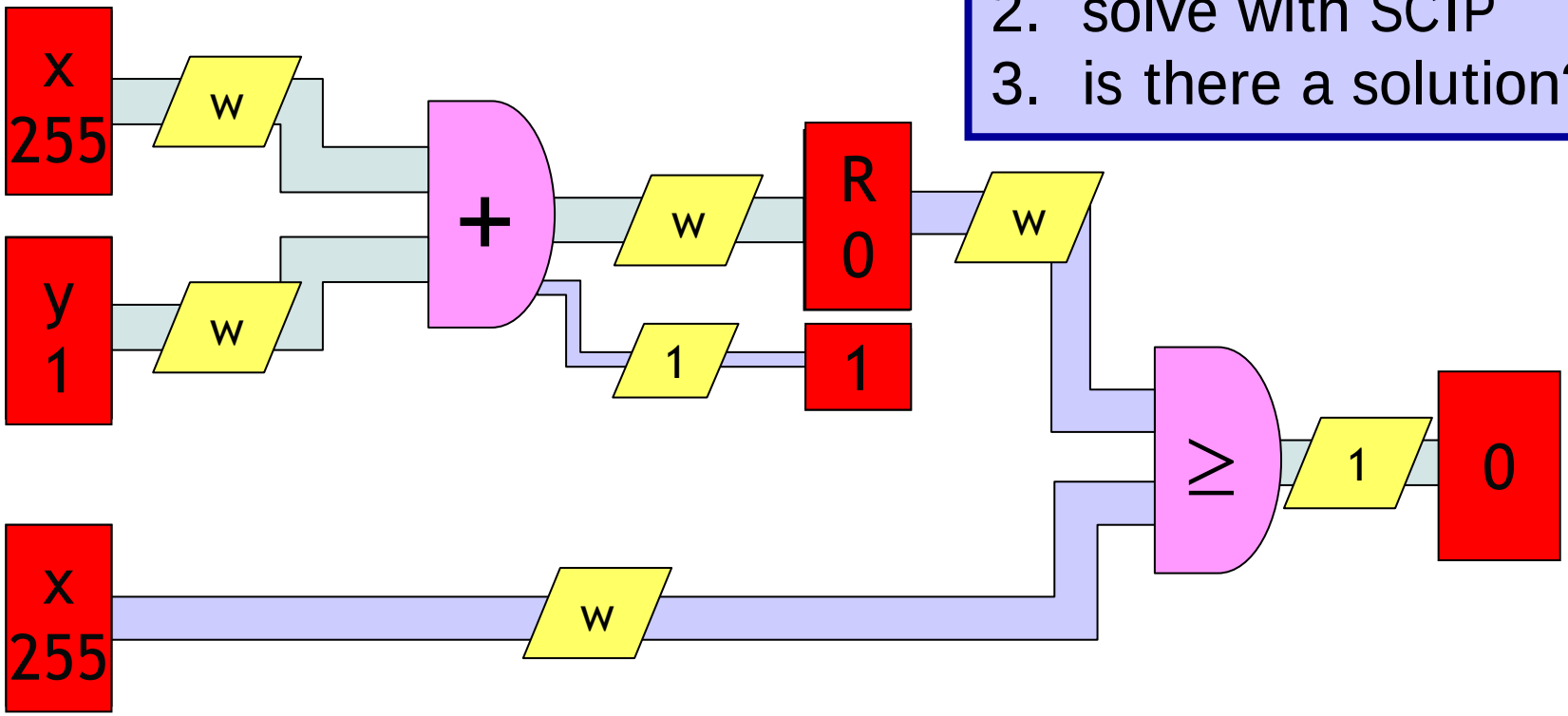
Excercise

1. enter ZIMPL model
2. run SCIP
3. load ZIMPL file
4. optimize
5. is there a solution?

Example 2: 2-Adder

Excercise

1. enter ZIMPL model
2. solve with SCIP
3. is there a solution?



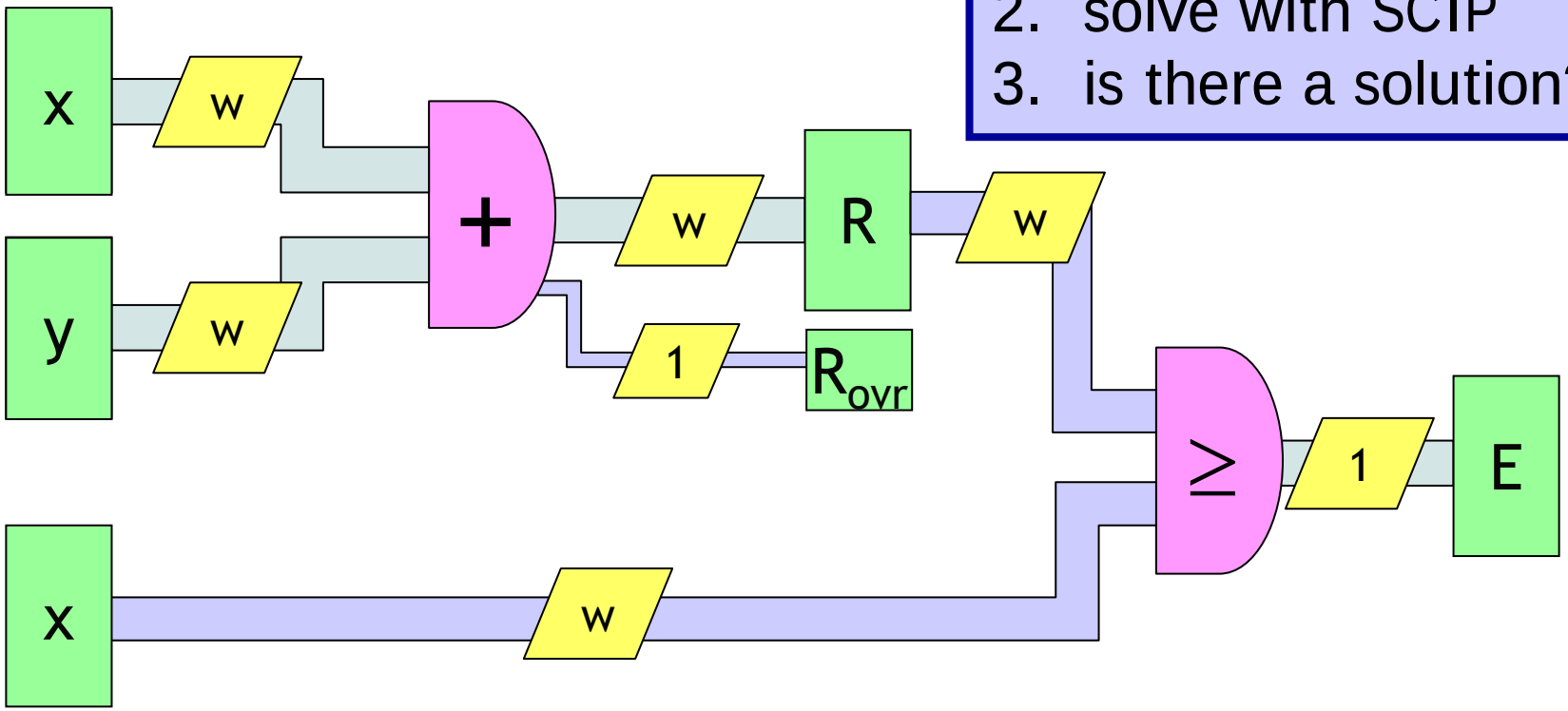
counterexample: property fails!

- negated property: $\exists x,y: E(x,y) = 0$

Example 2: 2-Adder

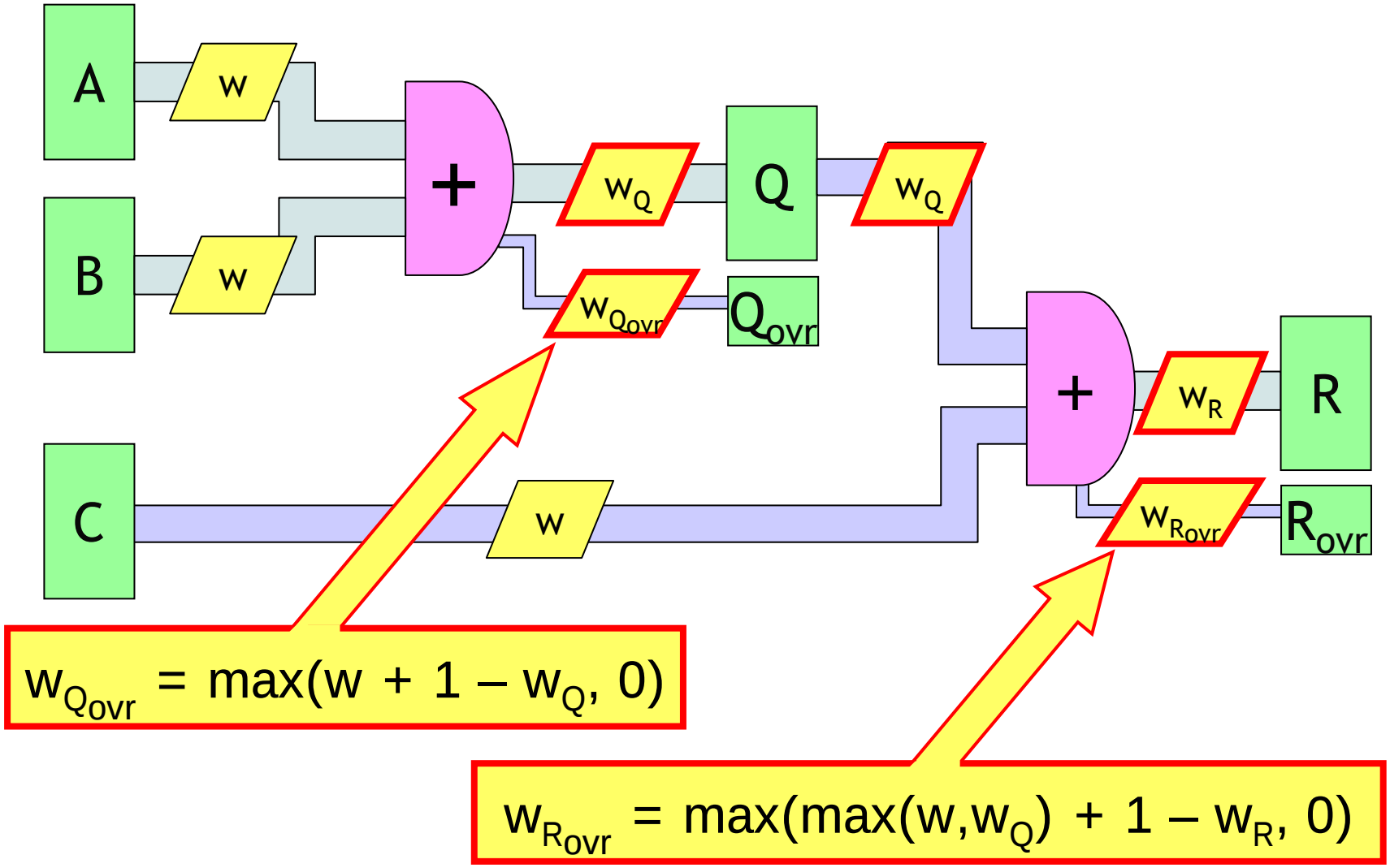
Excercise

1. modify model
2. solve with SCIP
3. is there a solution?



- new property: $R_{ovr} = 0 \rightarrow \forall x,y: R(x,y) \geq x$

Example 3: 3-Adder

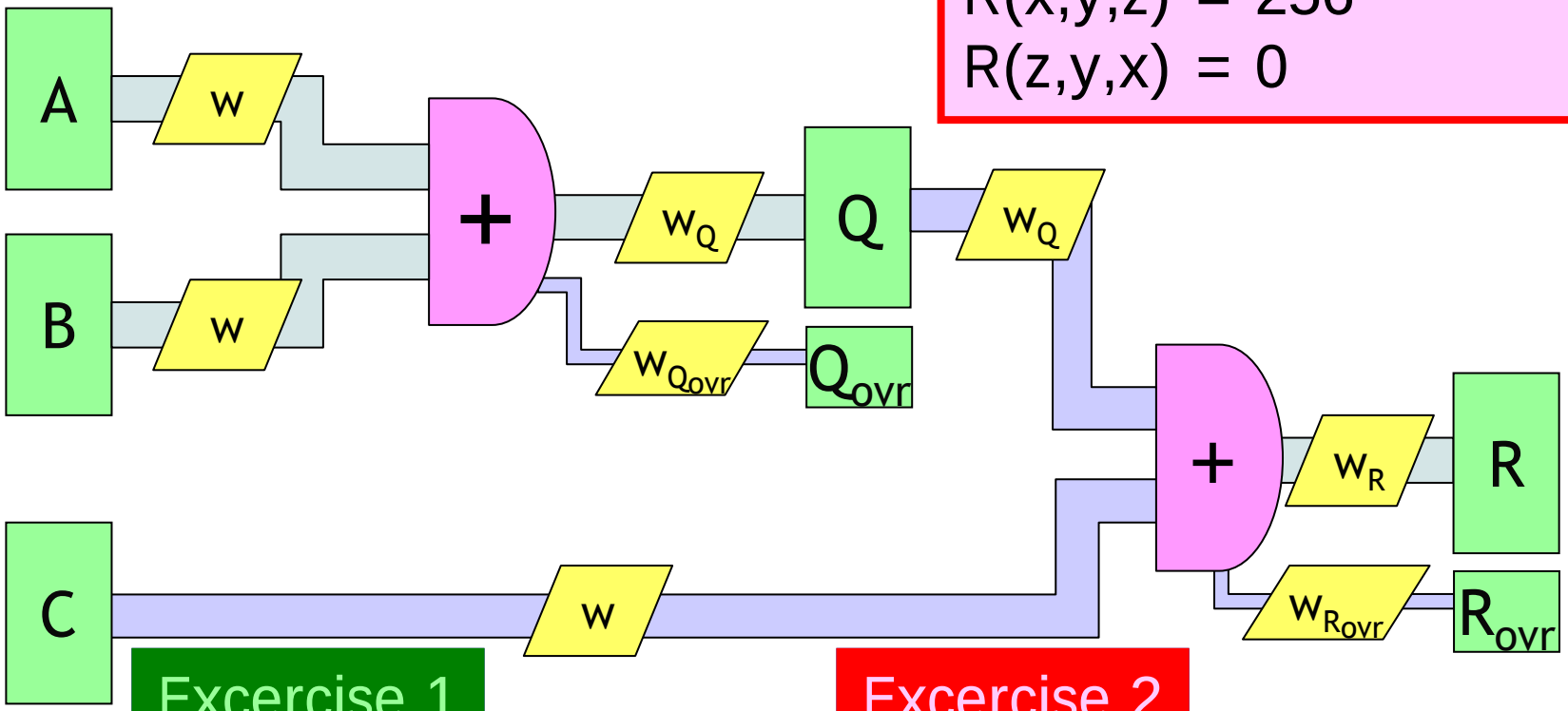


- property: $\forall x,y,z: R(x,y,z) = R(z,y,x)$

Example 3: 3-Adder

Counter-Example

x = 0, y = 1, z = 255
R(x,y,z) = 256
R(z,y,x) = 0



Excercise 1

1. $w = w_Q = w_R = 8$
2. solve with SCIP
3. is there a solution?

Excercise 2

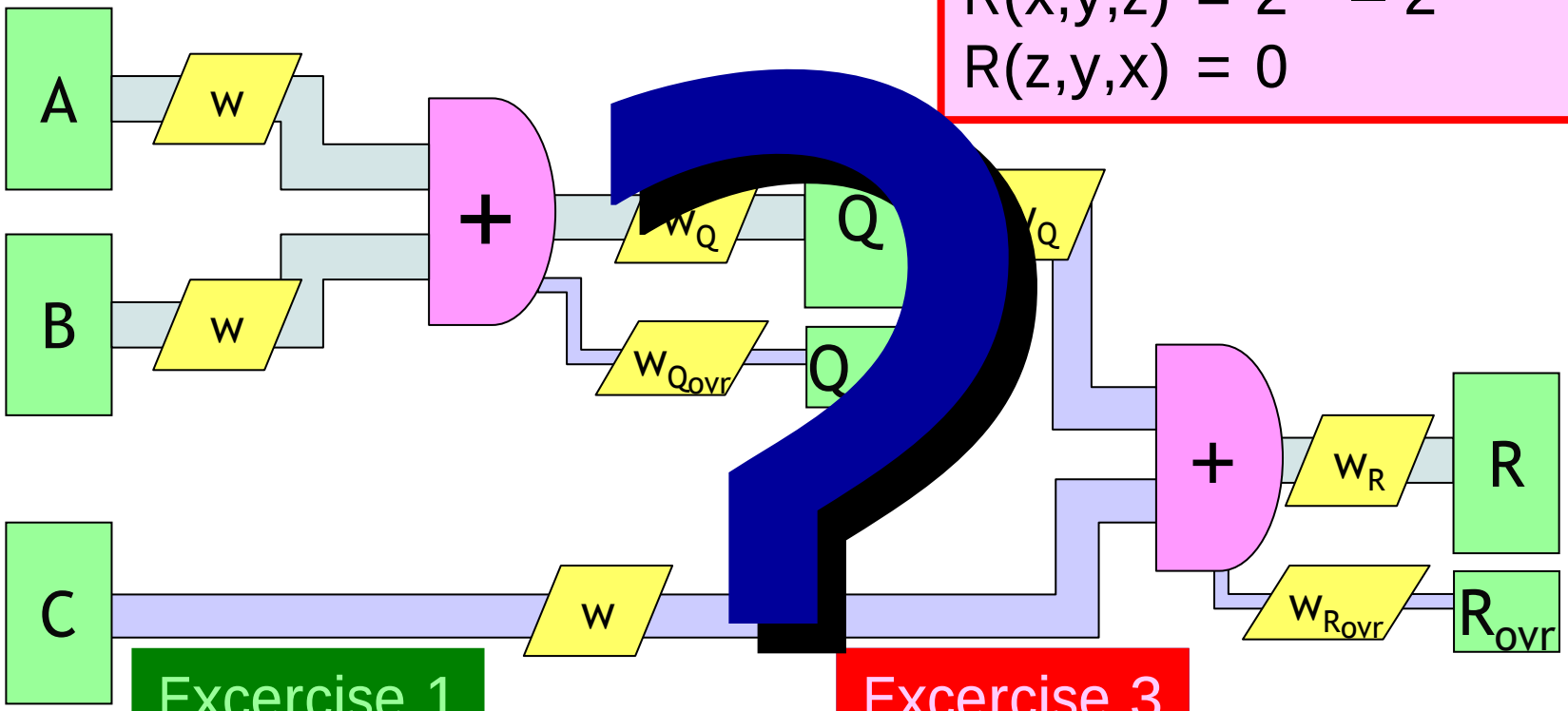
1. $w = w_Q = 8, w_R = 9$
2. solve with SCIP
3. is there a solution?

- property: $\forall x,y,z: R(x,y,z) = R(z,y,x)$

Example 3: 3-Adder

„Counter-Example“

$x = 0,\ y = z = 2^{20} - 1$
$R(x,y,z) = 2^{20} - 2$
$R(z,y,x) = 0$



Excercise 1

1. $w = w_Q = w_R = 8$
2. solve with SCIP
3. is there a solution?

Excercise 3

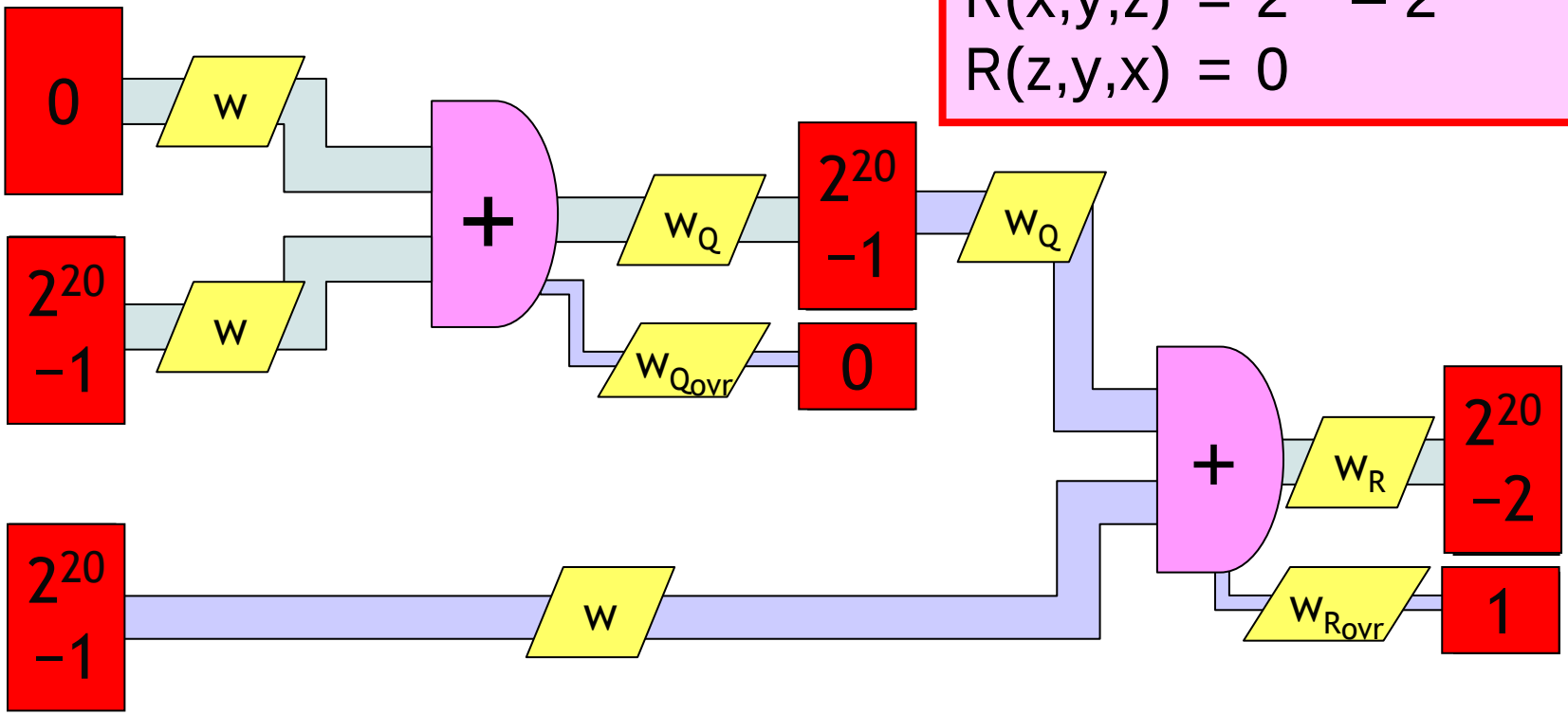
1. $w = w_Q = w_R = 20$
2. solve with SCIP
3. is there a solution?

- property: $\forall x,y,z: R(x,y,z) = R(z,y,x)$

Example 3: 3-Adder

„Counter-Example“

$x = 0,\ y = z = 2^{20} - 1$
$R(x,y,z) = 2^{20} - 2$
$R(z,y,x) = 0$



$R(x,y,z) = 2^{20} - 2$

- property: $\forall x,y,z: R(x,y,x) = R(z,y,x)$

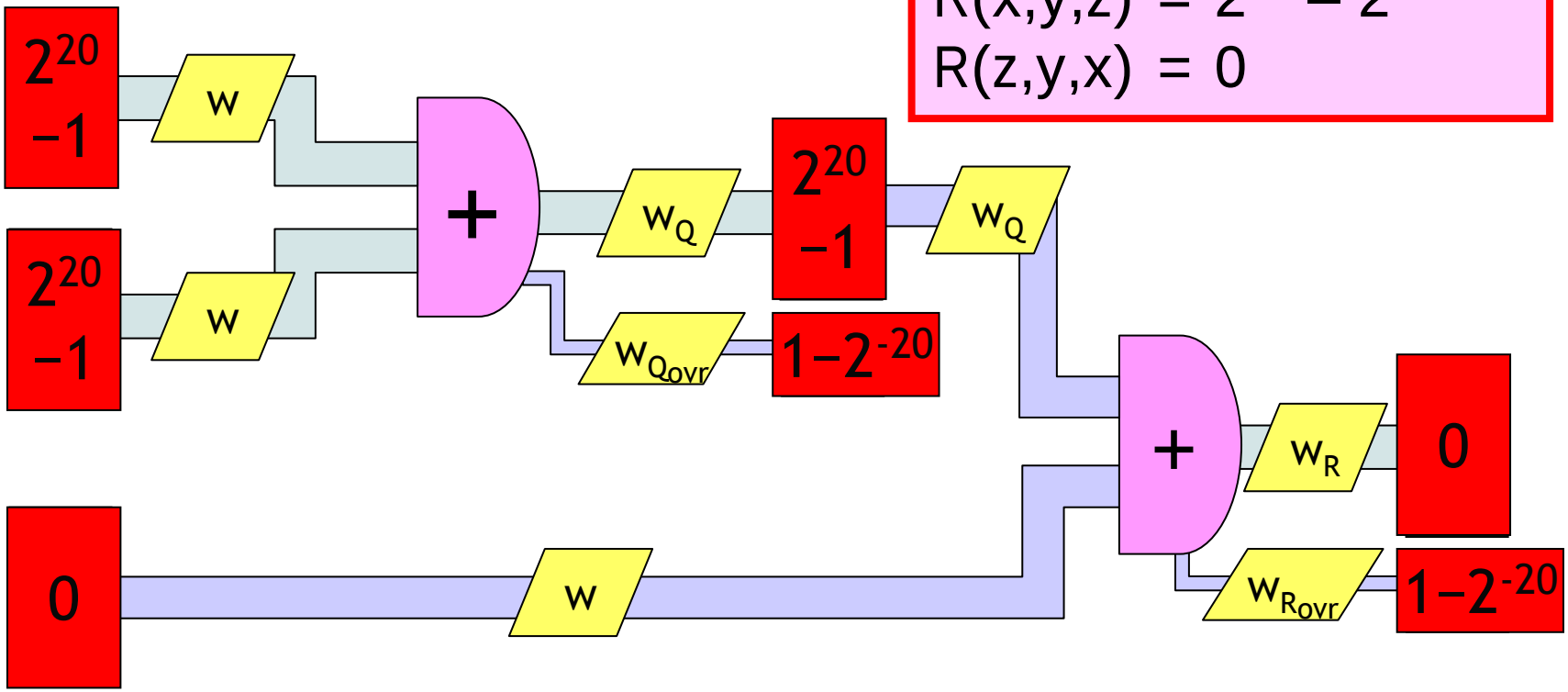
Example 3: 3-Adder

„Counter-Example“

$x = 0,\ y = z = 2^{20} - 1$

$R(x,y,z) = 2^{20} - 2$

$R(z,y,x) = 0$



$R(x,y,z) = 2^{20} - 2$ ✓

$R(z,y,x) = 0$ ☠

- property: $\forall x,y,z: R(x,y,z) = R(z,y,x)$

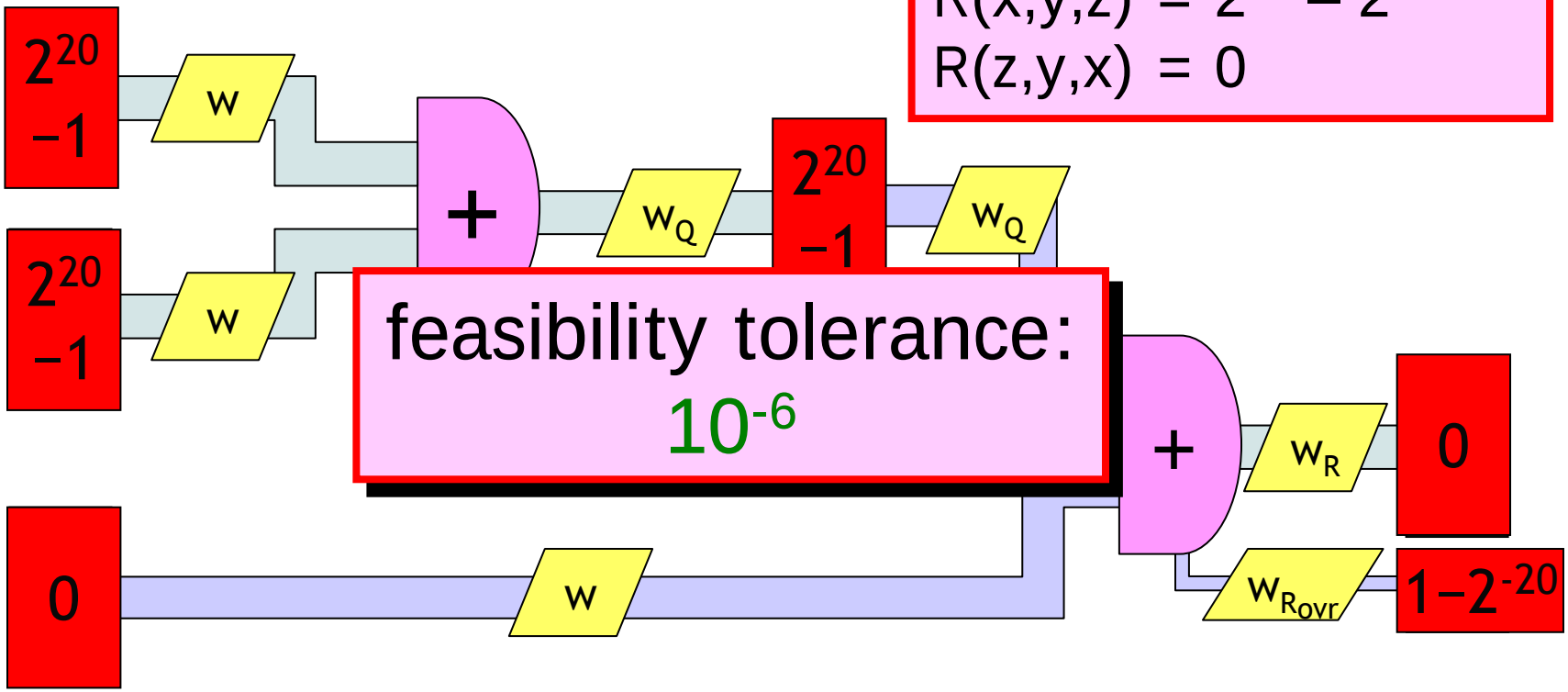
Example 3: 3-Adder

„Counter-Example“

$x = 0,\ y = z = 2^{20} - 1$

$R(x,y,z) = 2^{20} - 2$

$R(z,y,x) = 0$

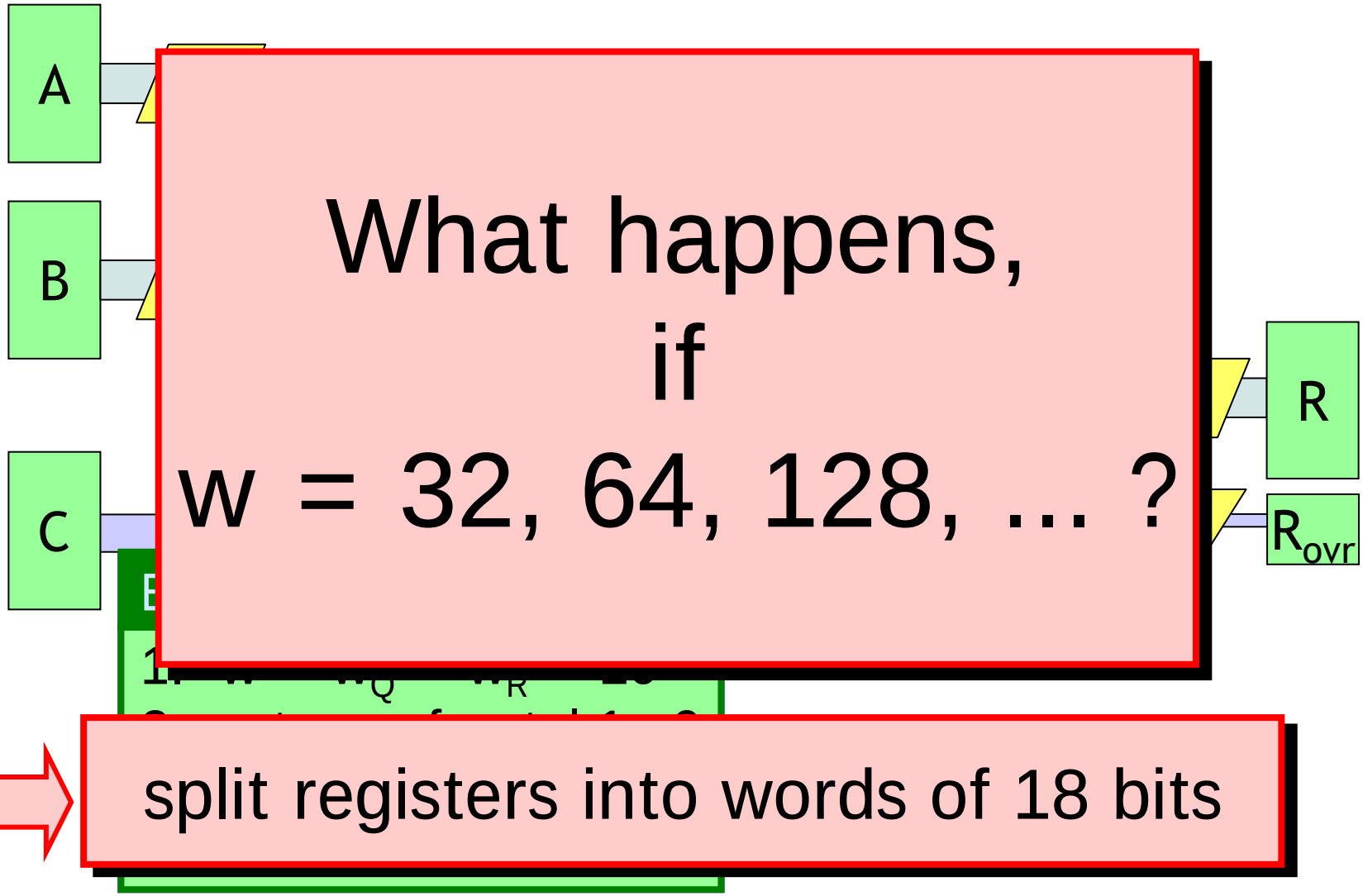


$$Q_{ovr} = R_{ovr} = 1 - 2^{-20} \in \mathbf{Z} \ ?$$

$$1 - 2^{-20} = 0{,}99999904632568359375...$$

$$1 - 2^{-20} \approx 1 - 9{,}54 \cdot 10^{-7}$$

Example 3: 3-Adder



- property: $\forall x,y,z: R(x,y,z) = R(z,y,x)$