An Efficient Parallel Hardware Scheme for Solving the N-Queens Problem

Yuuma Azuma, Hayato Sakagami, Kenji Kise

Tokyo Institute of Technology, Japan

Outline

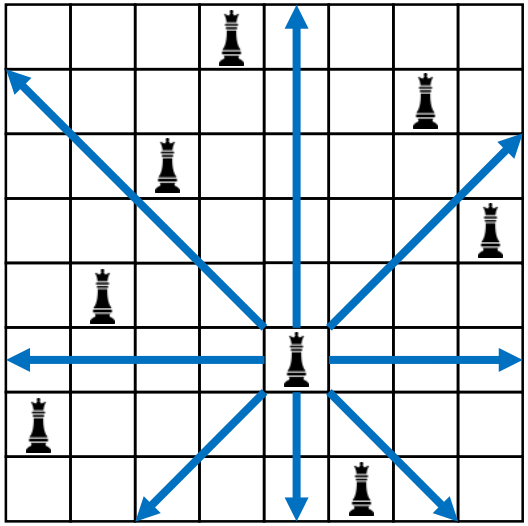
1. Introduction
2. Proposal
3. Evaluation
4. Conclusion

Outline

1. Introduction
    1. Background
    2. Previous Works
2. Proposal
3. Evaluation
4. Conclusion

The N-Queens Problem

- The N-Queens problem is a generalized problem of the 8-Queens puzzle

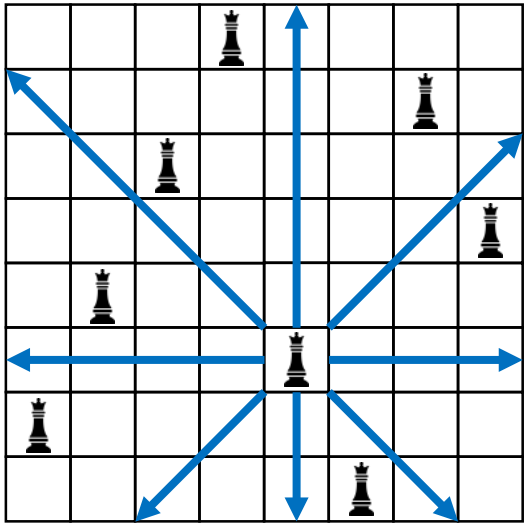


One example of solution with N=8

- placing N chess queens on an N x N board
- no two queens threaten each other

The N-Queens Problem

- The computational complexity increases drastically with increasing N

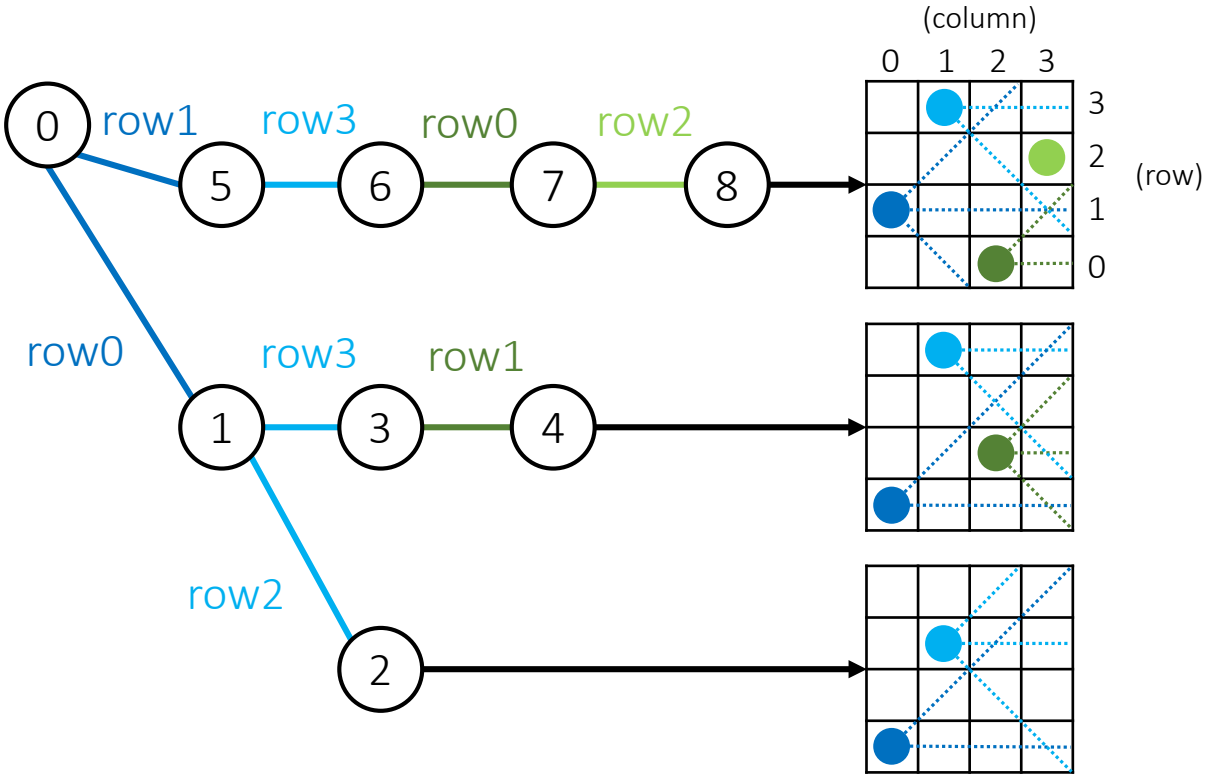


N	The total number of solutions
23	24,233,937,684,440
24	227,514,171,973,736
25	2,207,893,435,808,352
26	22,317,699,616,364,044
27	234,907,967,154,122,528
28	unsolved

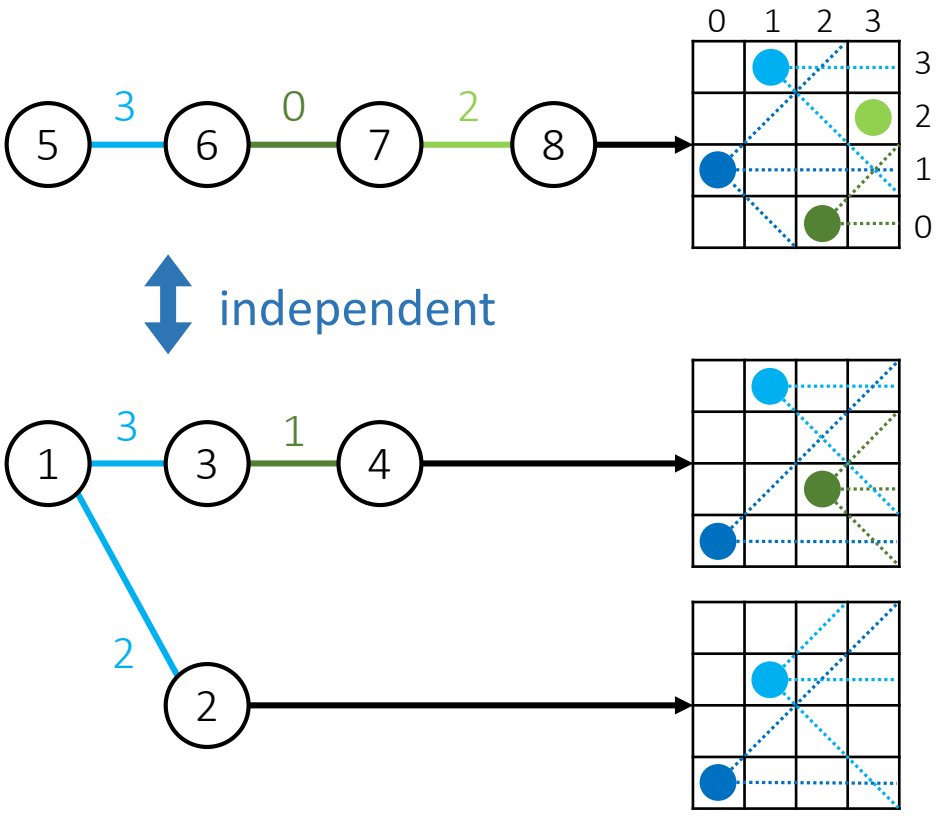
We aim to solve the unsolved problems

Backtracking

- The backtracking method is one of typical algorithms to search the solutions



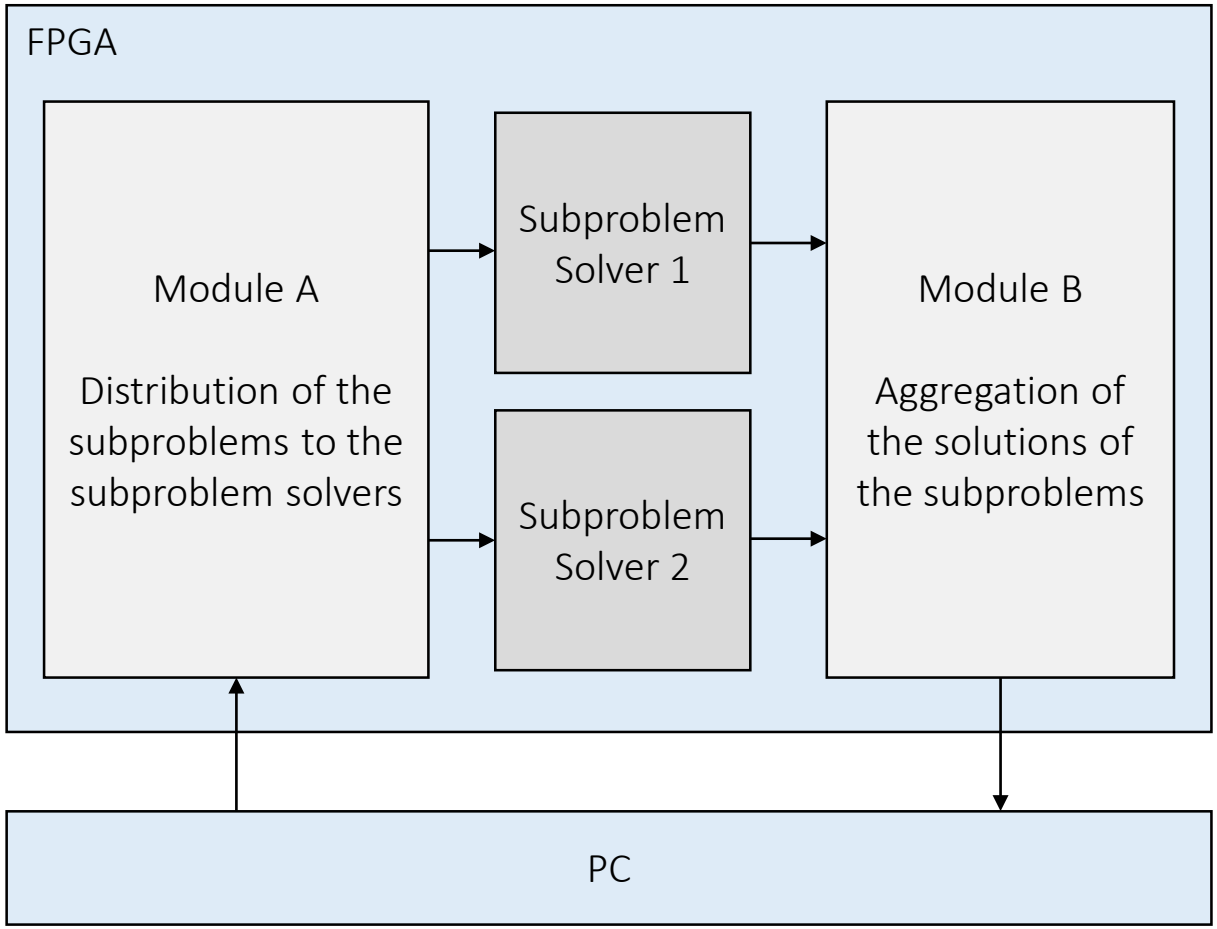
Subproblem



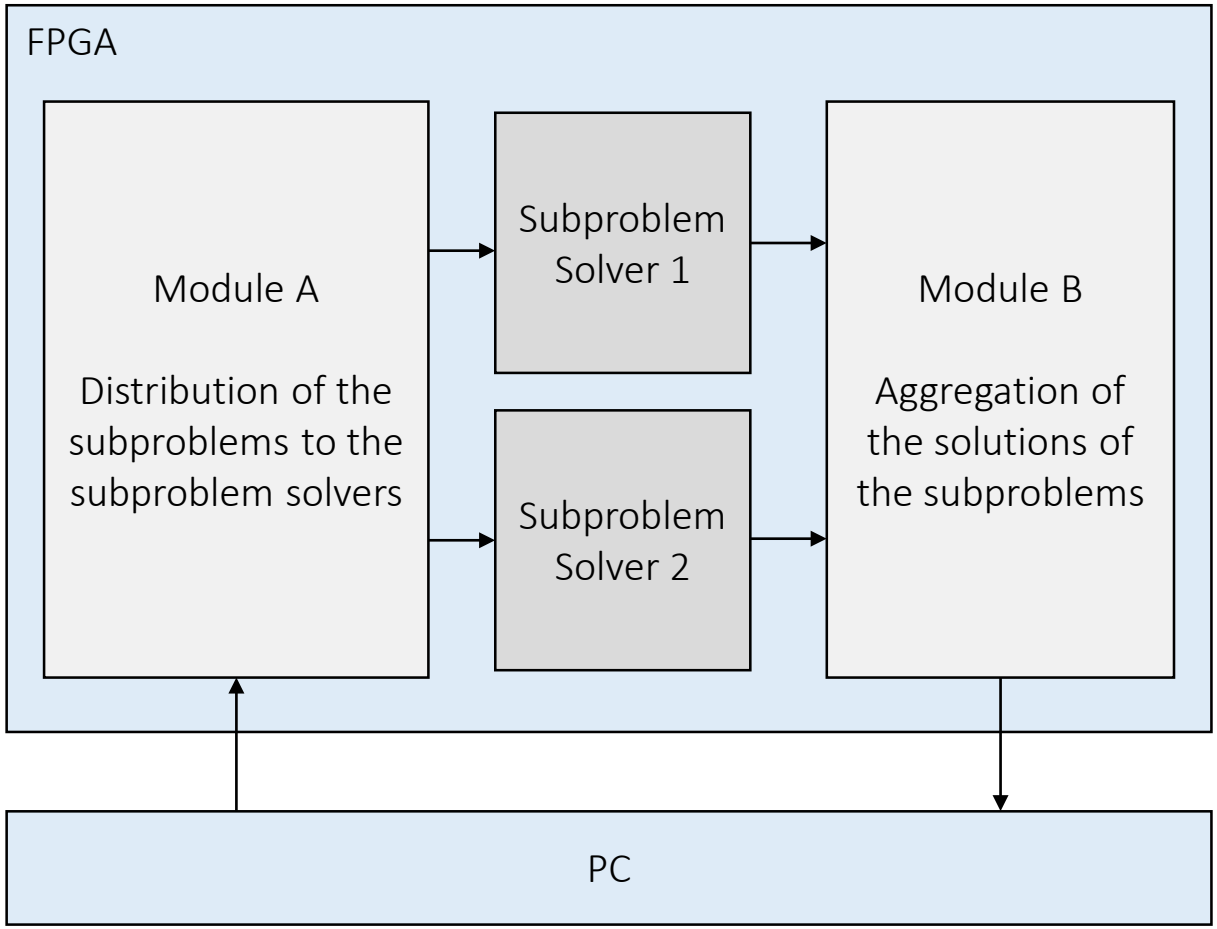
- *Subproblem*: Place some queens before searching the solution
- The search spaces of different subproblems are independent

Searching solutions in parallel is possible

The N-Queens Problem on an FPGA



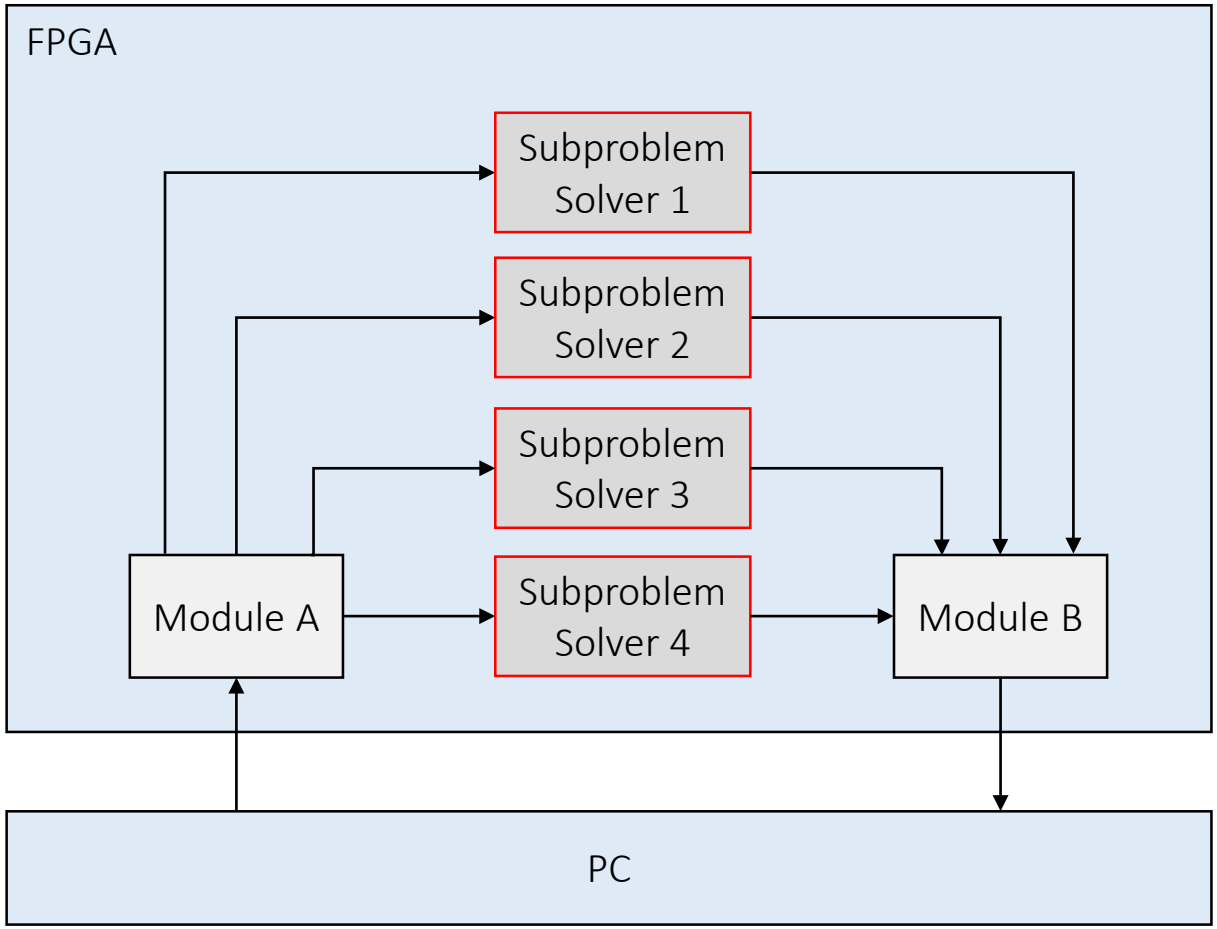
The N-Queens Problem on an FPGA



Implementation of the small solver

Optimization of distribution / aggregation

The N-Queens Problem on an FPGA



Implementation of more solvers on FPGA

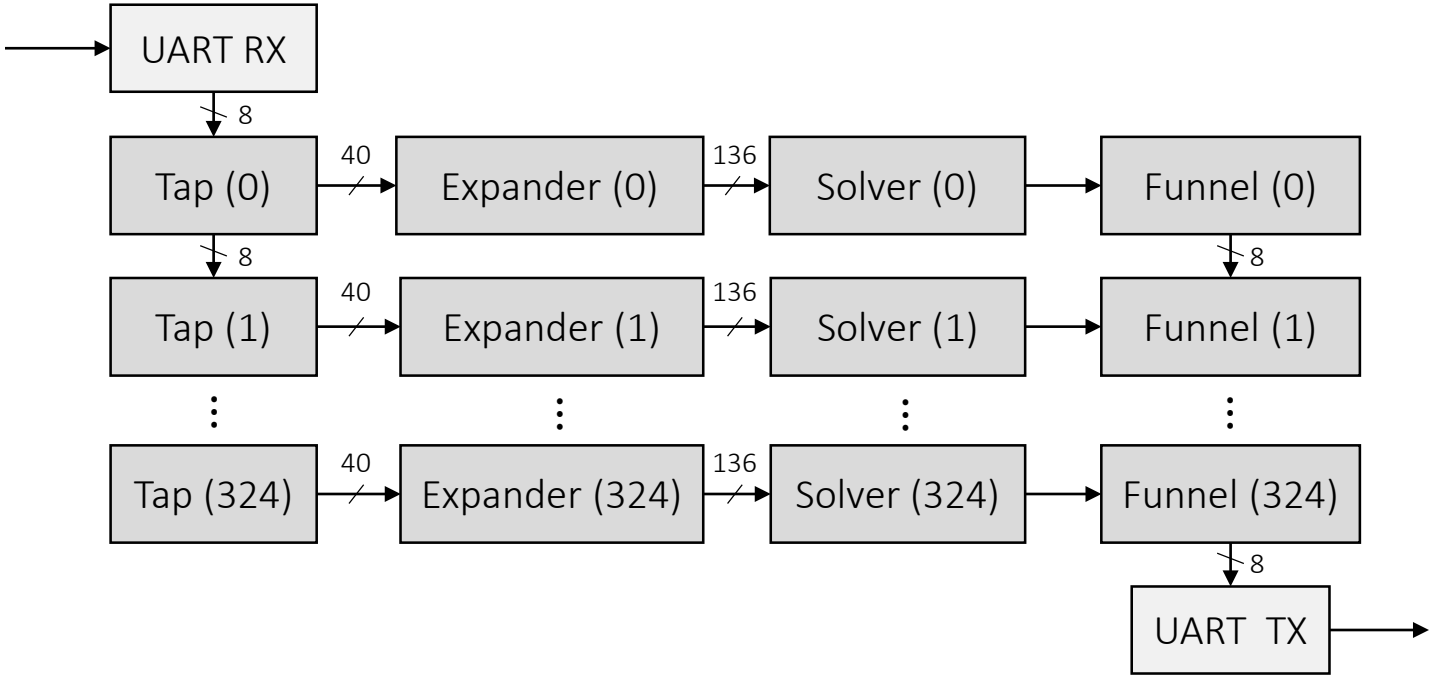
Operation on high frequency

Implementation of the small solver

Optimization of distribution / aggregation

The Previous Work [1]

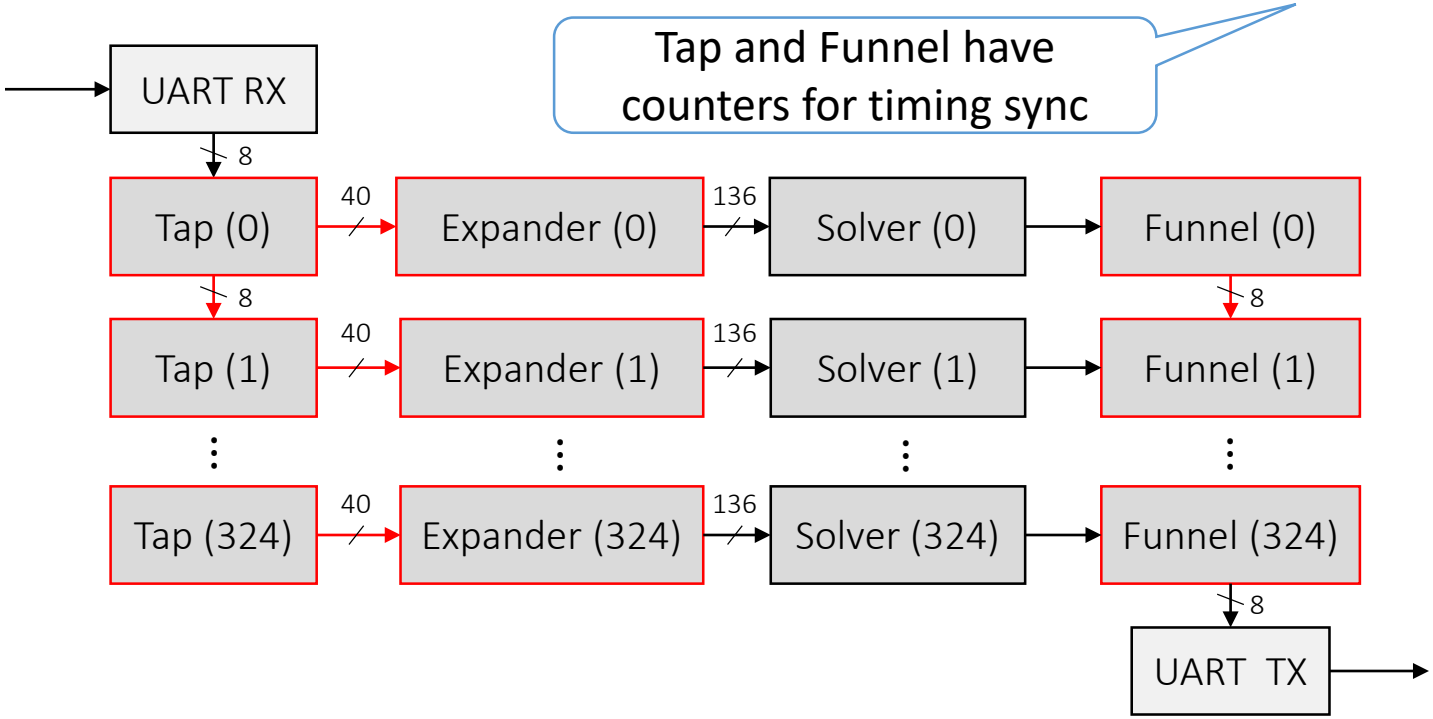
- This FPGA system solved the 27-Queens problem



[1] Thomas B Preußer and Matthias R Engelhardt. Putting queens in carry chains, no 27. Journal of Signal Processing Systems, Vol. 88, No. 2, pp. 185-201, 2017.

The Previous Work [1]

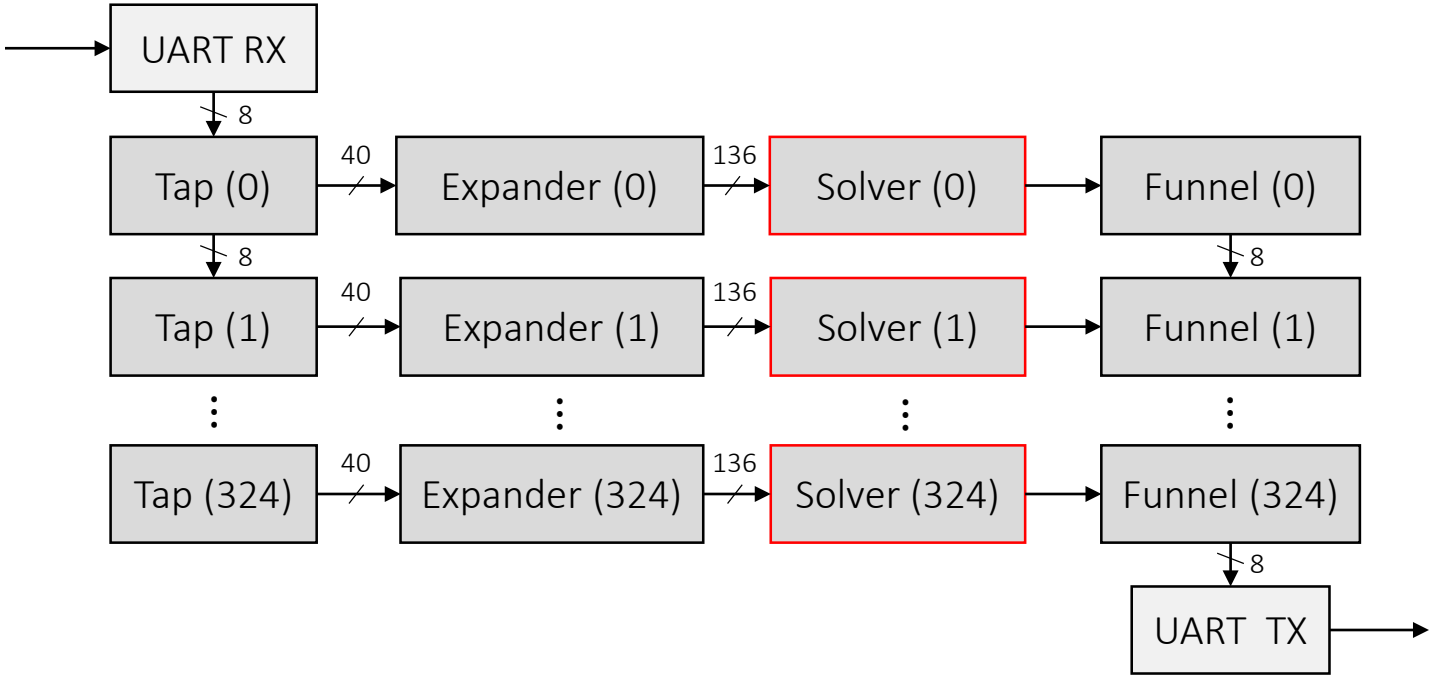
- The distribution / aggregation system has the critical path



[1] Thomas B Preußer and Matthias R Engelhardt. Putting queens in carry chains, no 27. Journal of Signal Processing Systems, Vol. 88, No. 2, pp. 185-201, 2017.

The Previous Work [1]

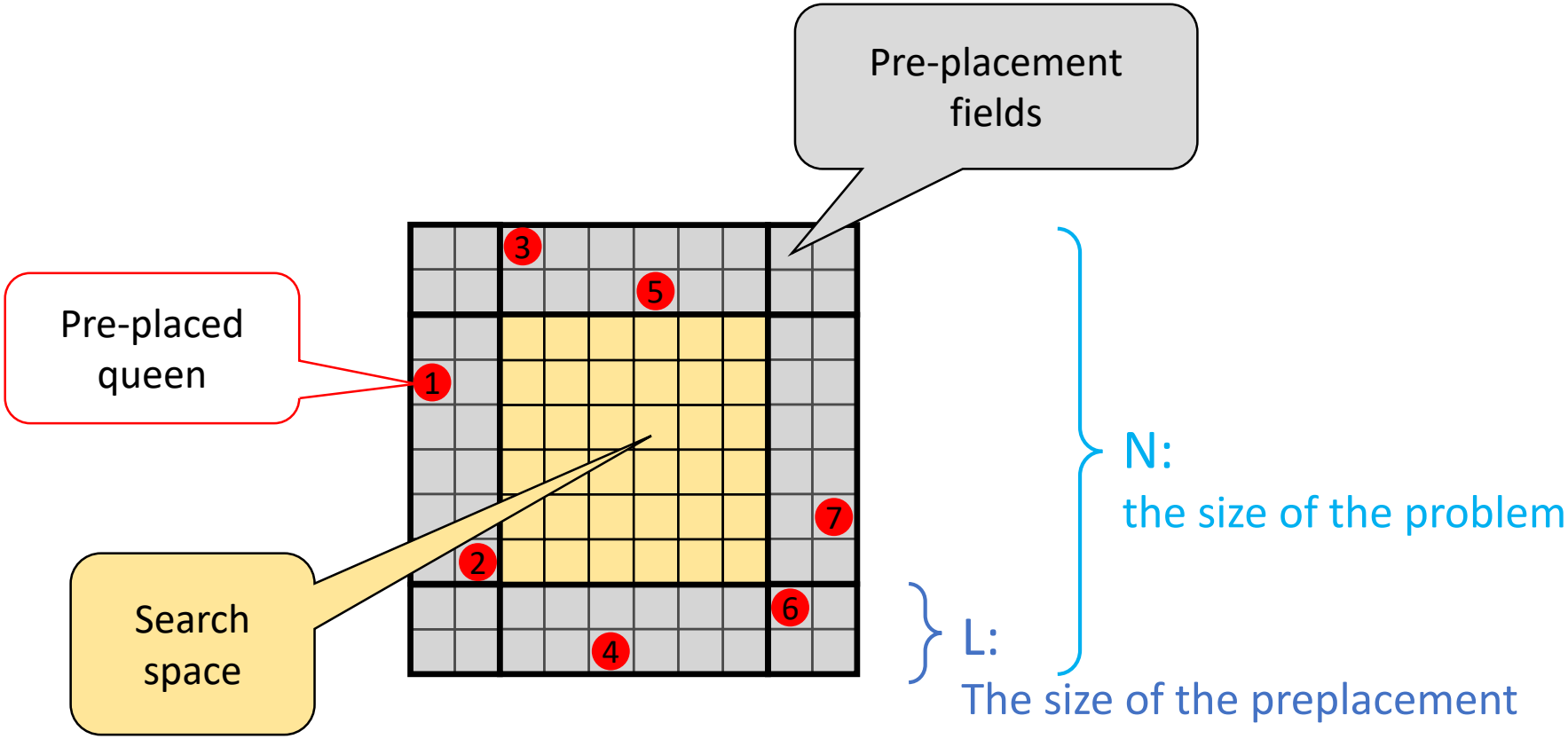
- A Solver solves a subploblem



[1] Thomas B Preußer and Matthias R Engelhardt. Putting queens in carry chains, no 27. Journal of Signal Processing Systems, Vol. 88, No. 2, pp. 185-201, 2017.

The annular pre-placement

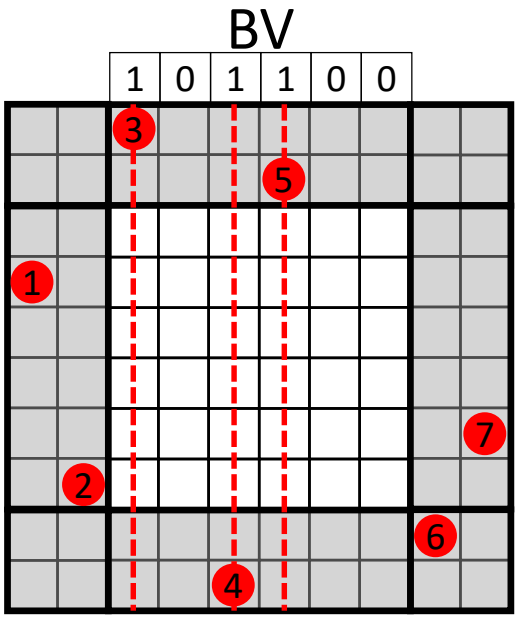
in the previous work [1]



[1] Thomas B Preußer and Matthias R Engelhardt. Putting queens in carry chains, no 27. Journal of Signal Processing Systems, Vol. 88, No. 2, pp. 185-201, 2017.

The annular pre-placement

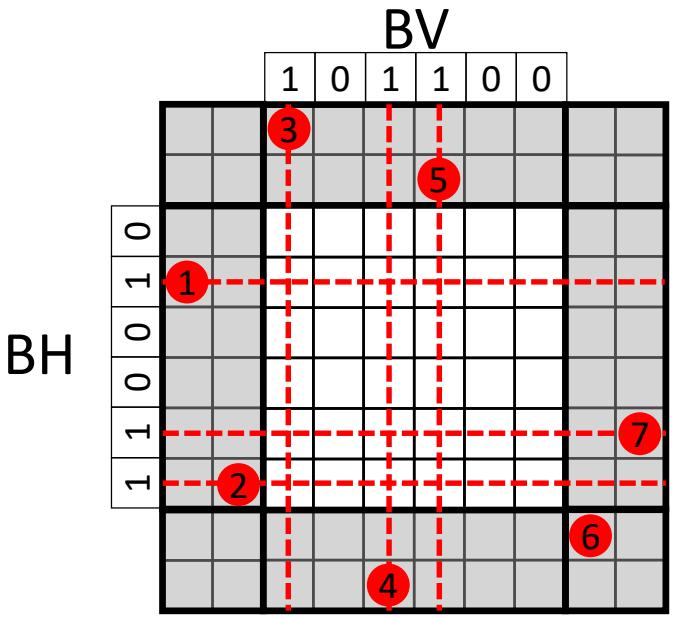
in the previous work [1]



BV: 101100

The annular pre-placement

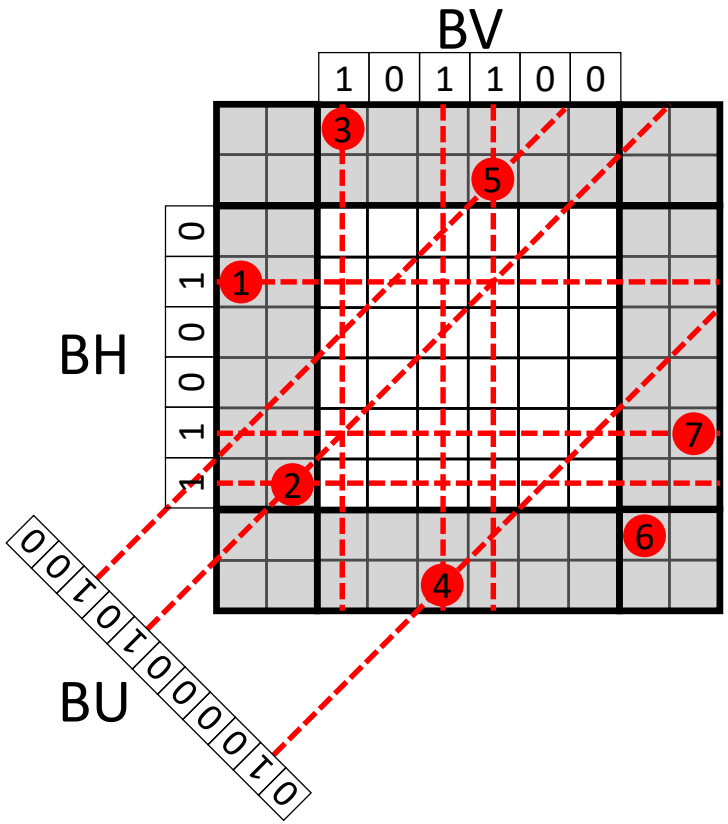
in the previous work [1]



BV: 101100
BH: 110010

The annular pre-placement

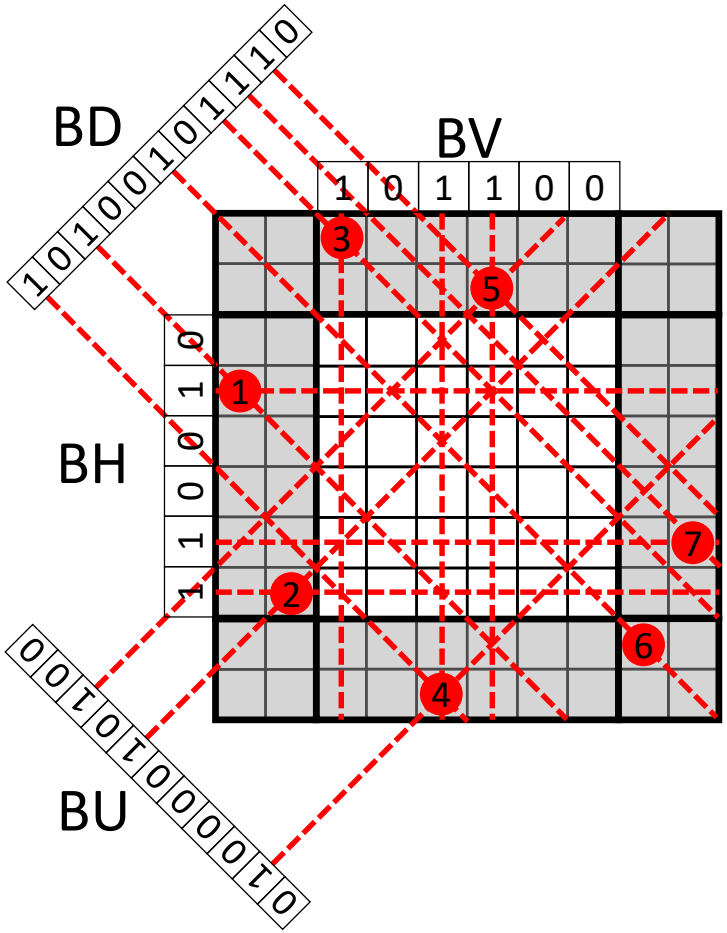
in the previous work [1]



BV: 101100
BH: 110010
BU: 01000010100

The annular pre-placement

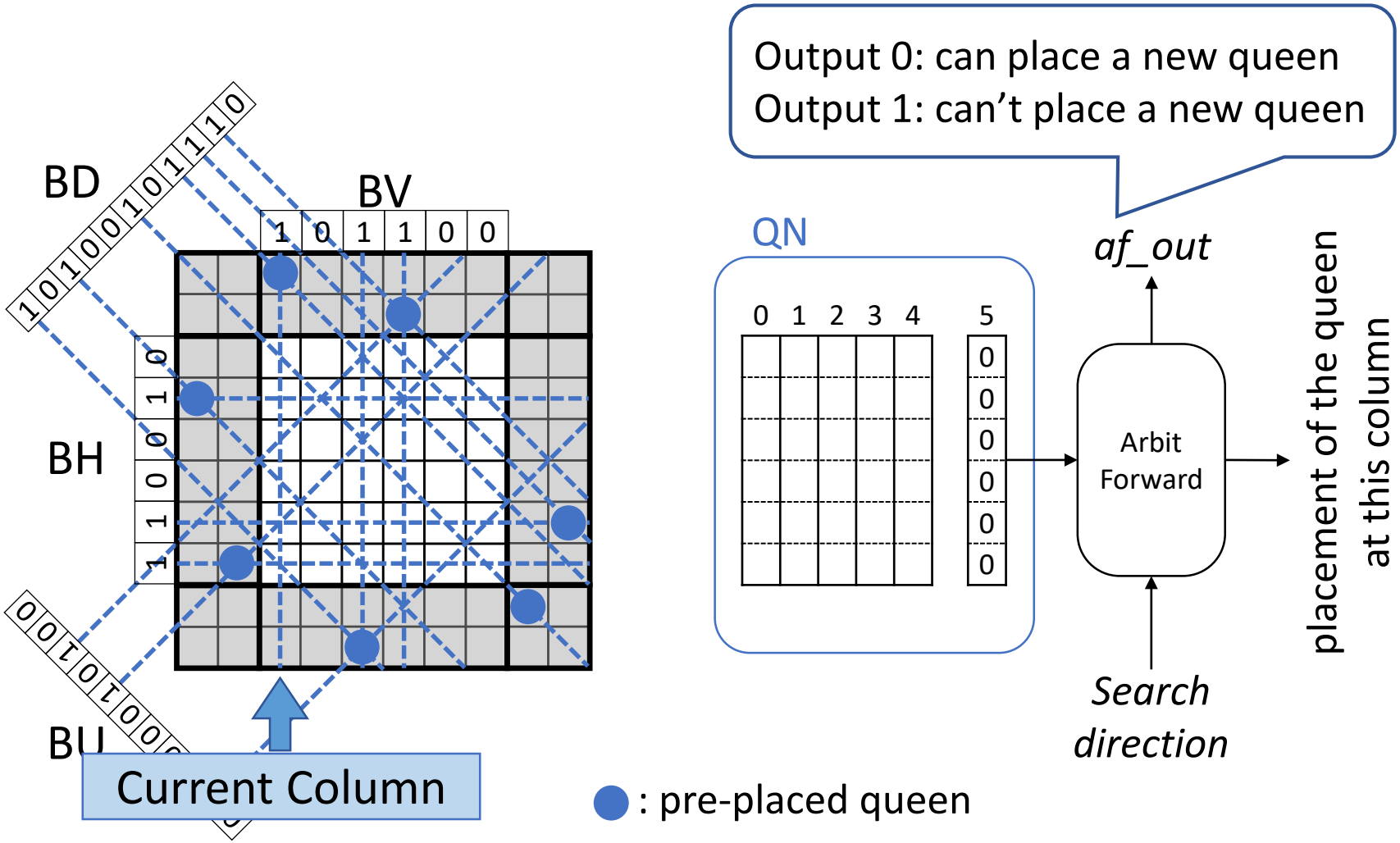
in the previous work [1]



Blocking Information

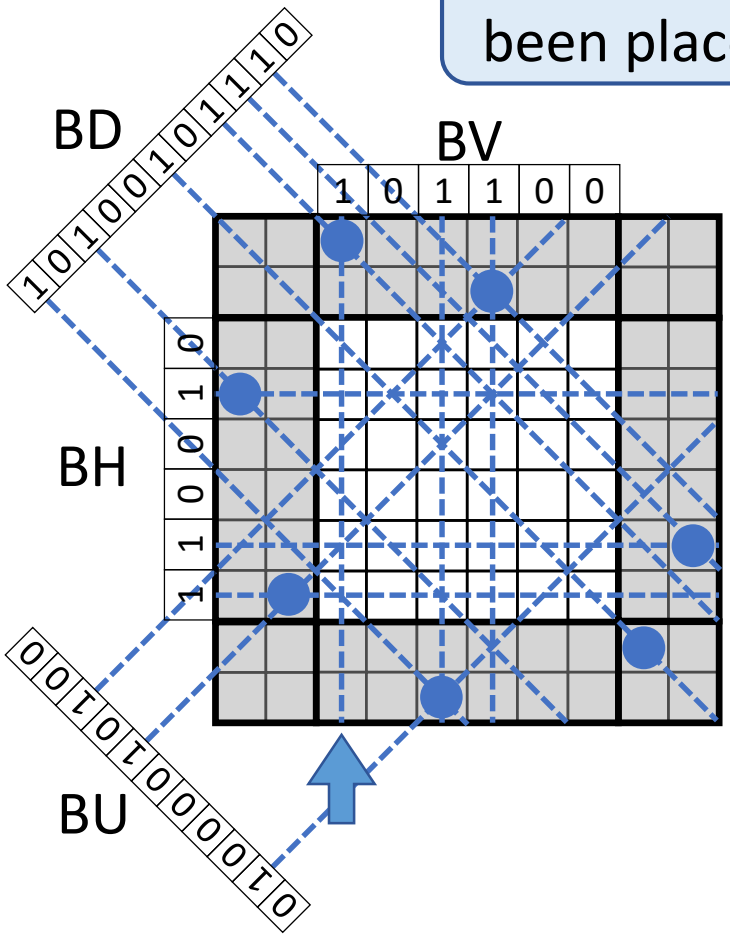
```
BV:      101100
BH:      110010
BU:01000010100
BD:10100101110
```

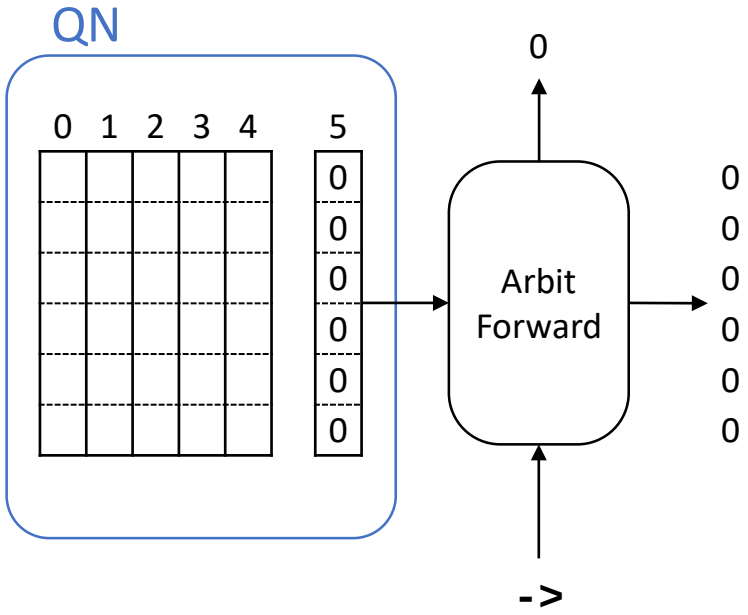
Solving a subproblem on a solver



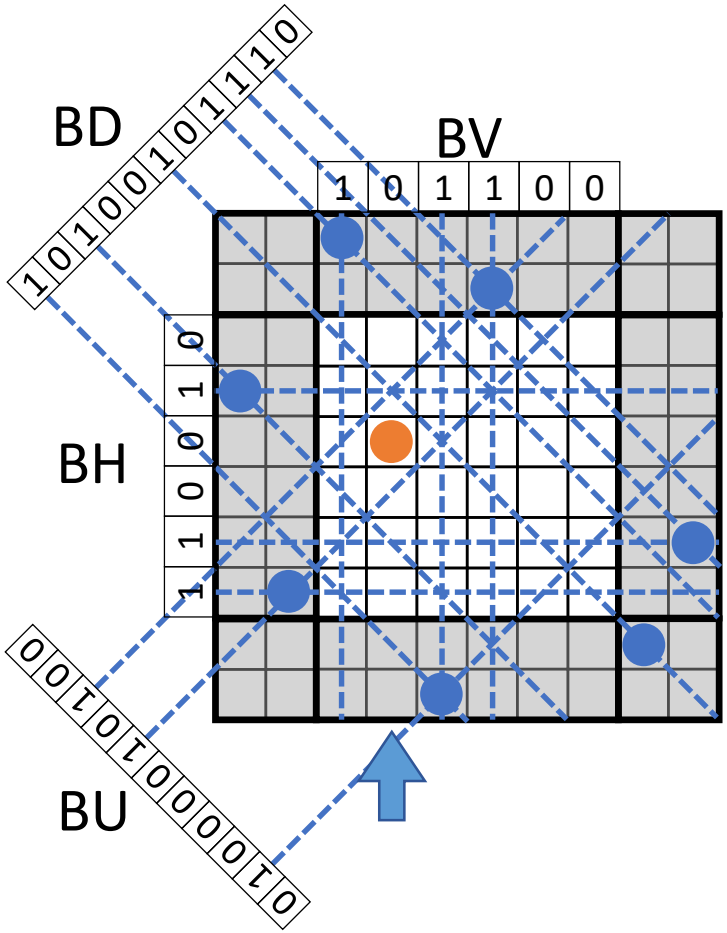
Solving a subproblem on a solver

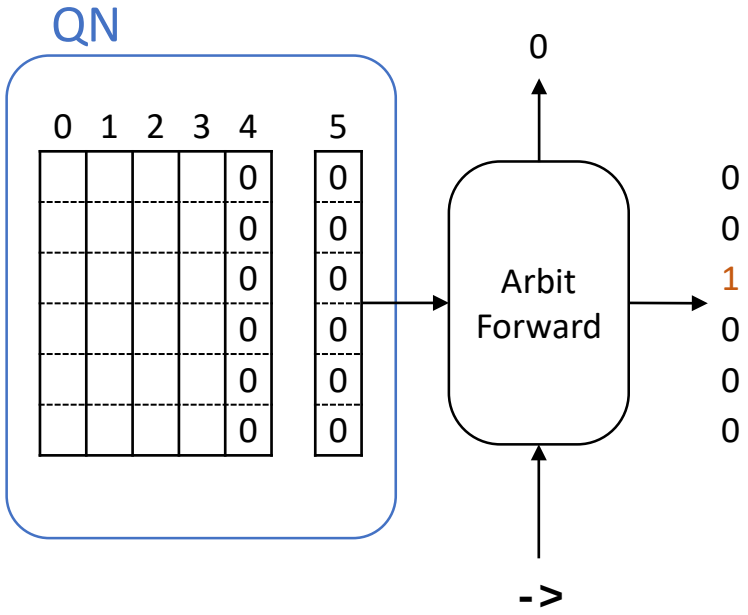
The *af_out* is 0 because a queen has already been placed in this column by pre-placement



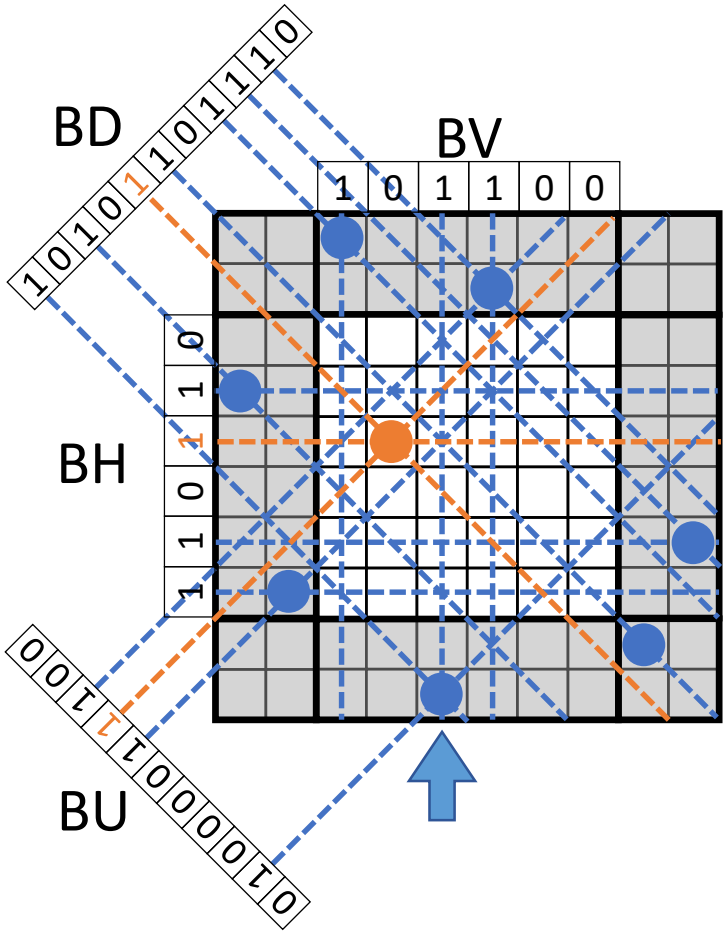


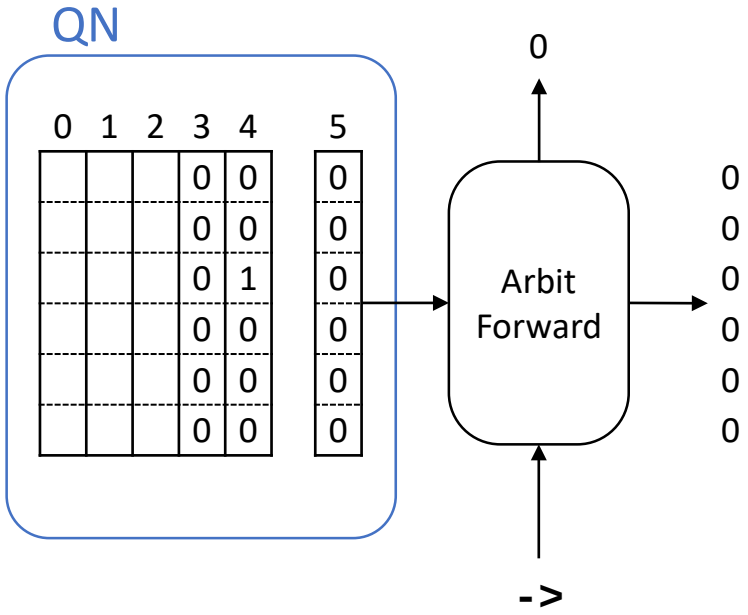
Solving a subproblem on a solver



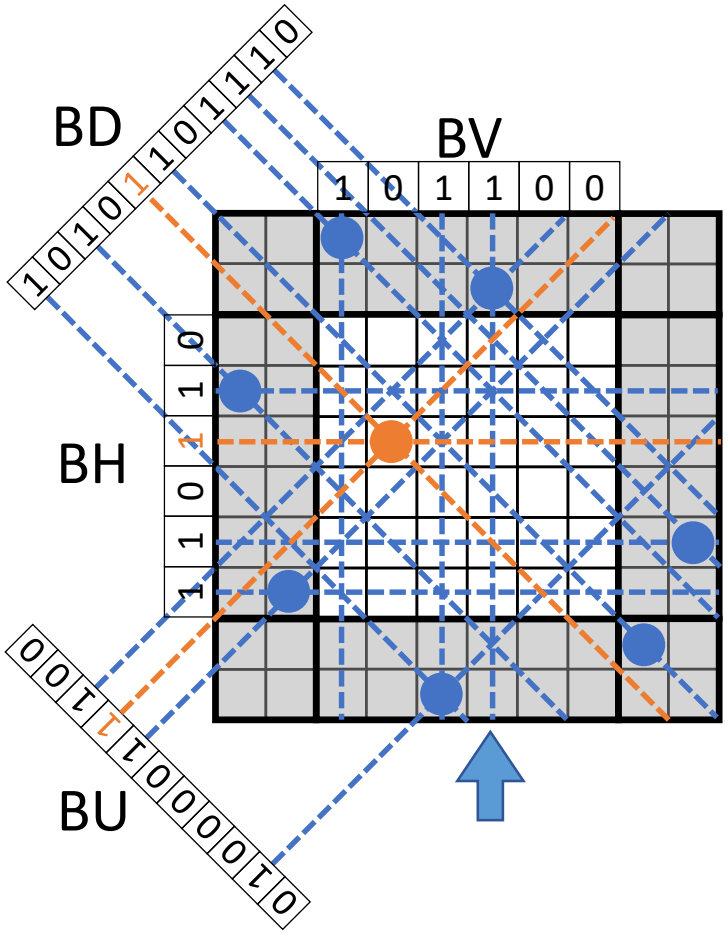


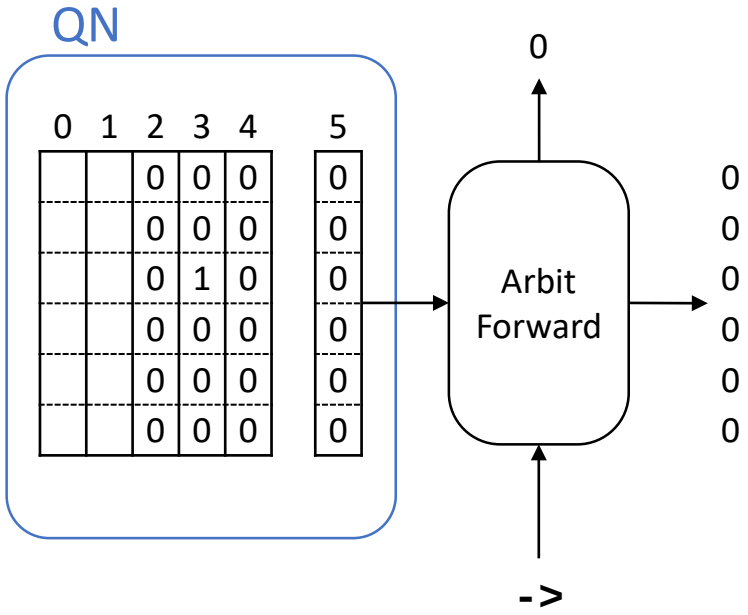
Solving a subproblem on a solver



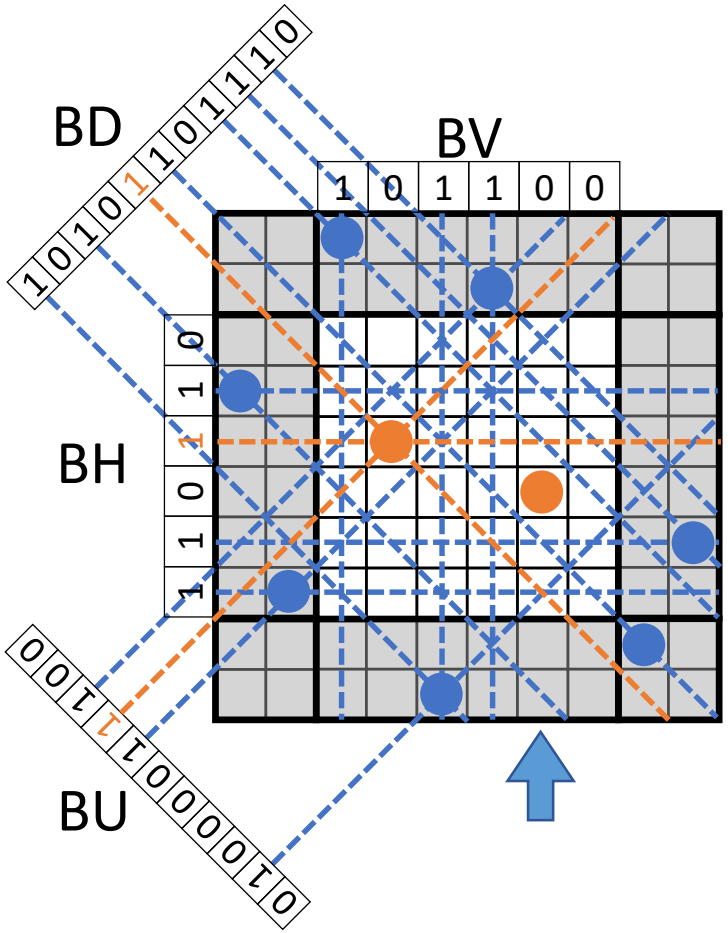


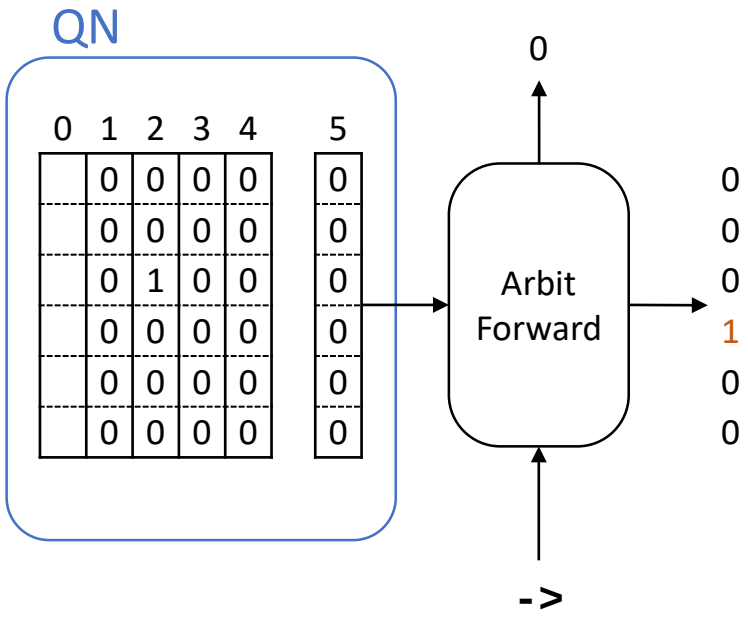
Solving a subproblem on a solver





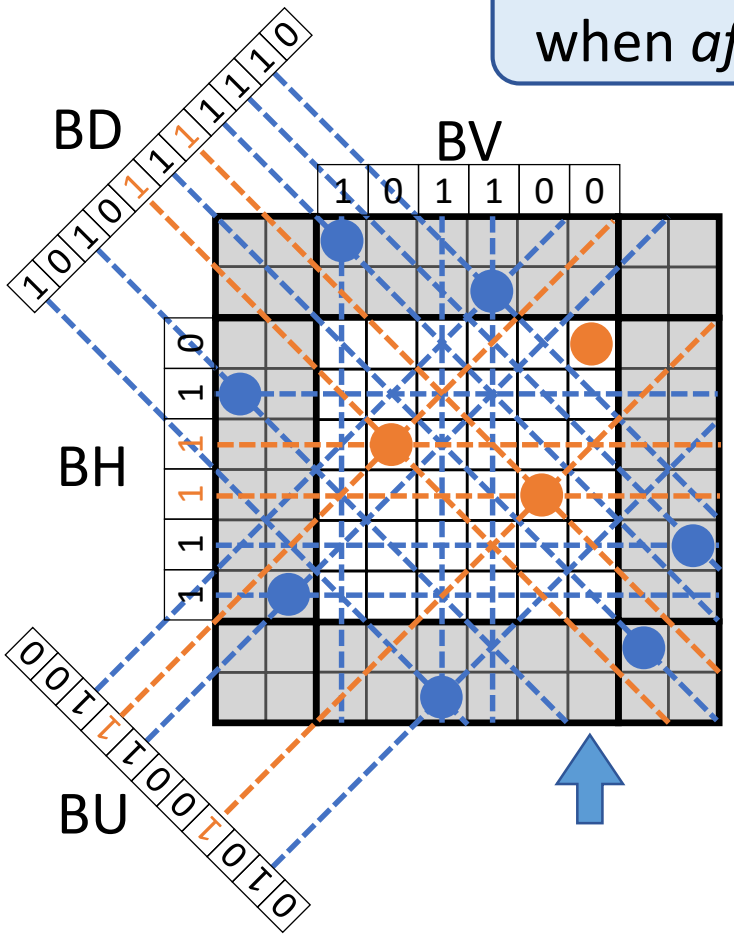
Solving a subproblem on a solver

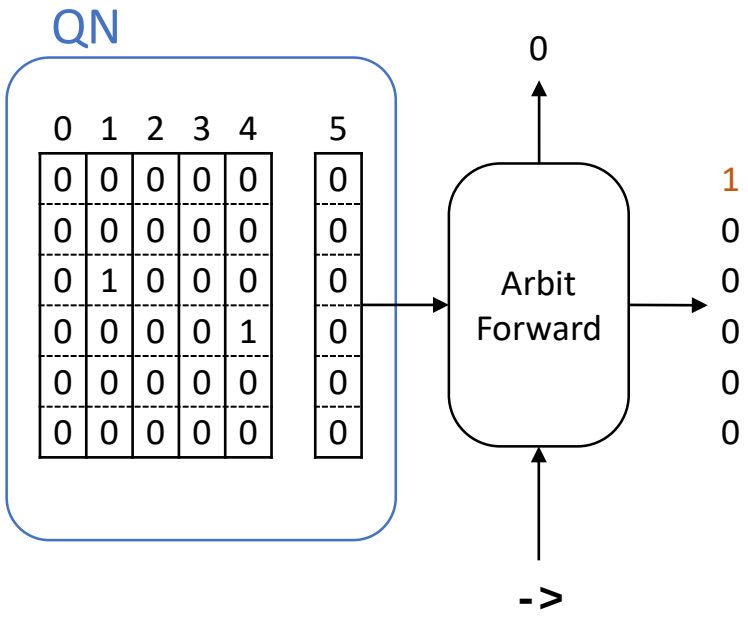




Solving a subproblem on a solver

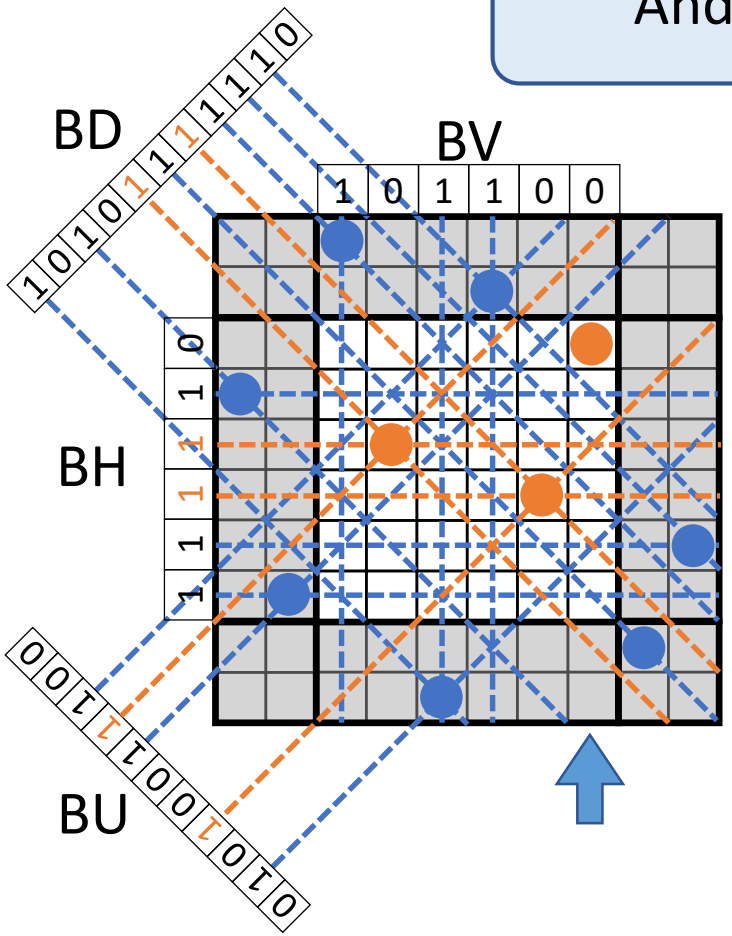
The Solver asserts the signal *sol_bit* when *af_out* is 0 in the rightmost column

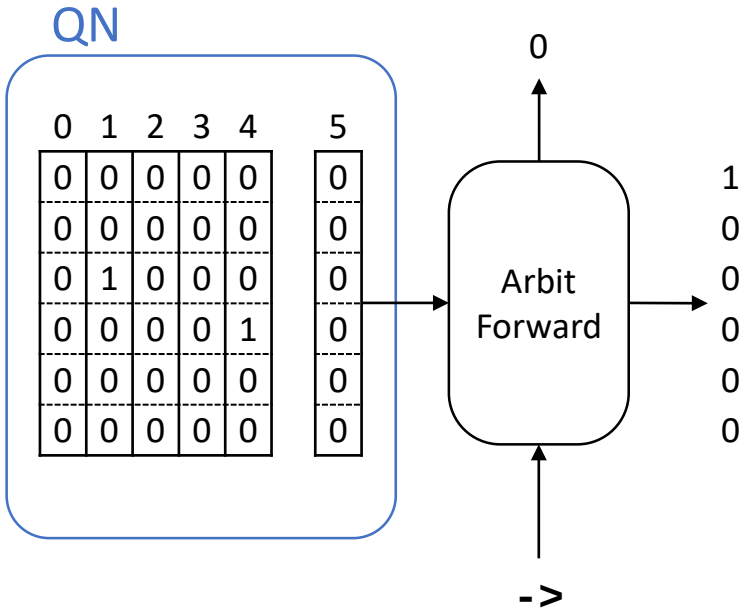




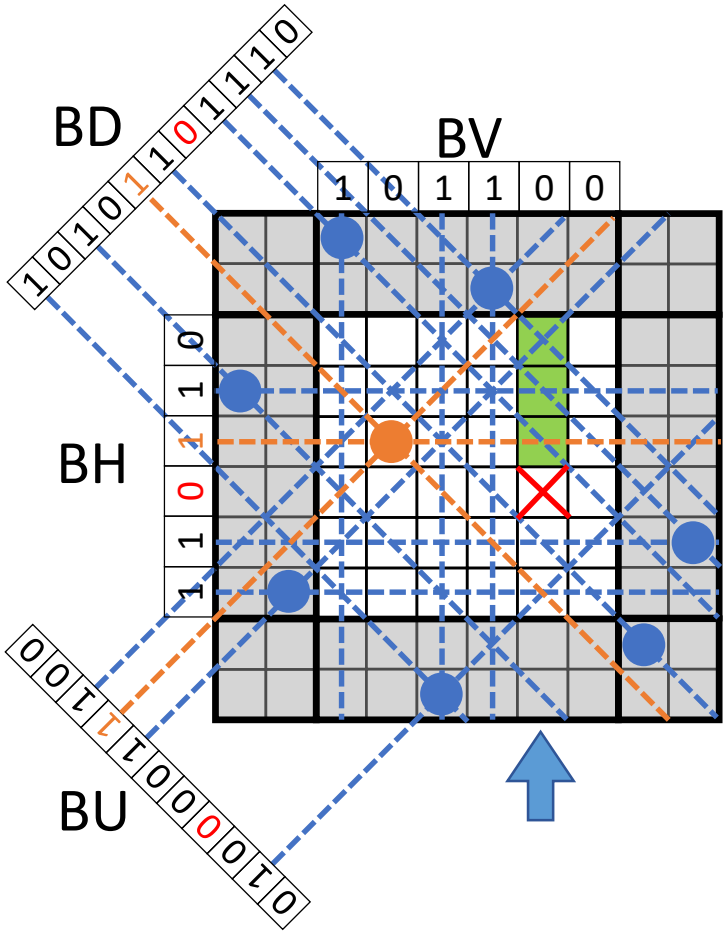
Solving a subproblem on a solver

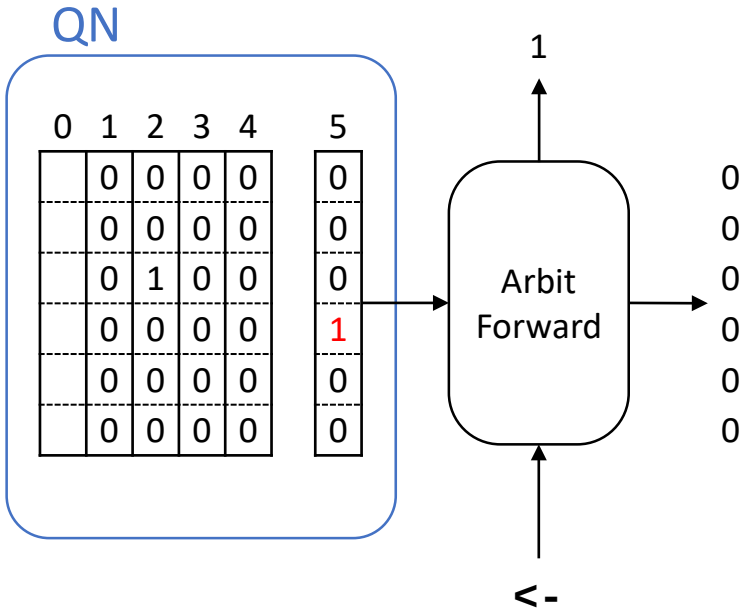
And the solver starts to backtrack



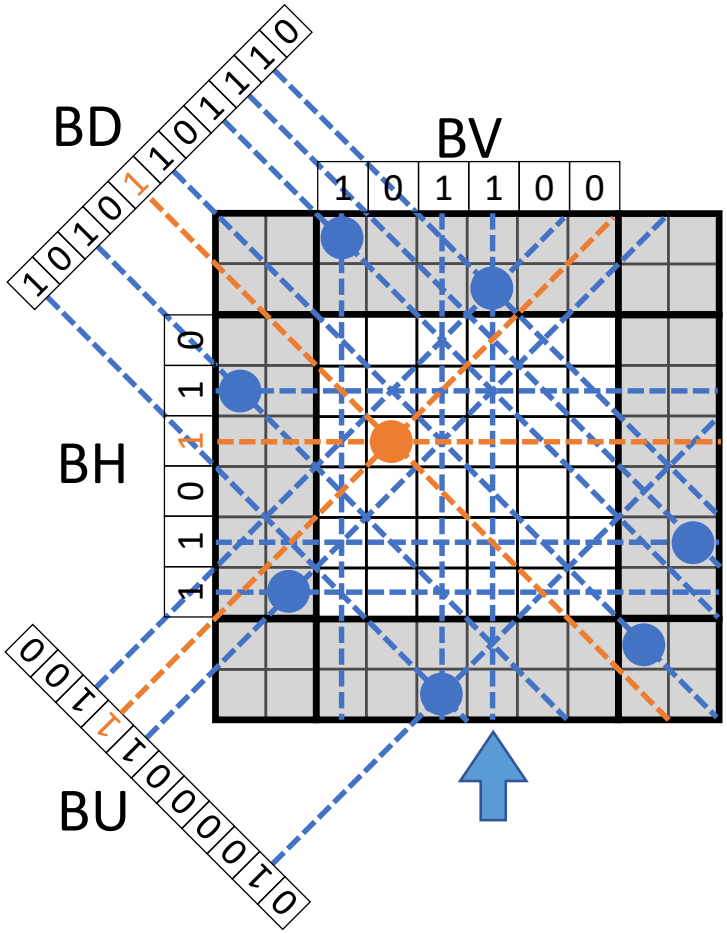


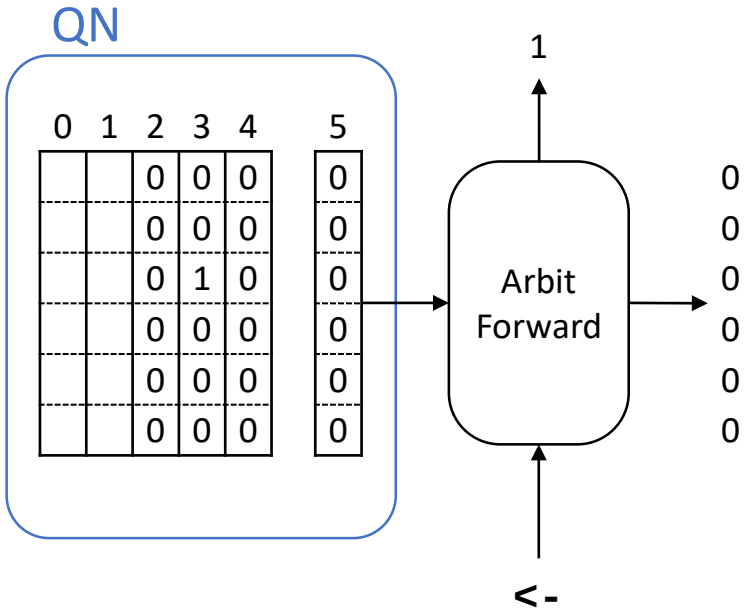
Solving a subproblem on a solver



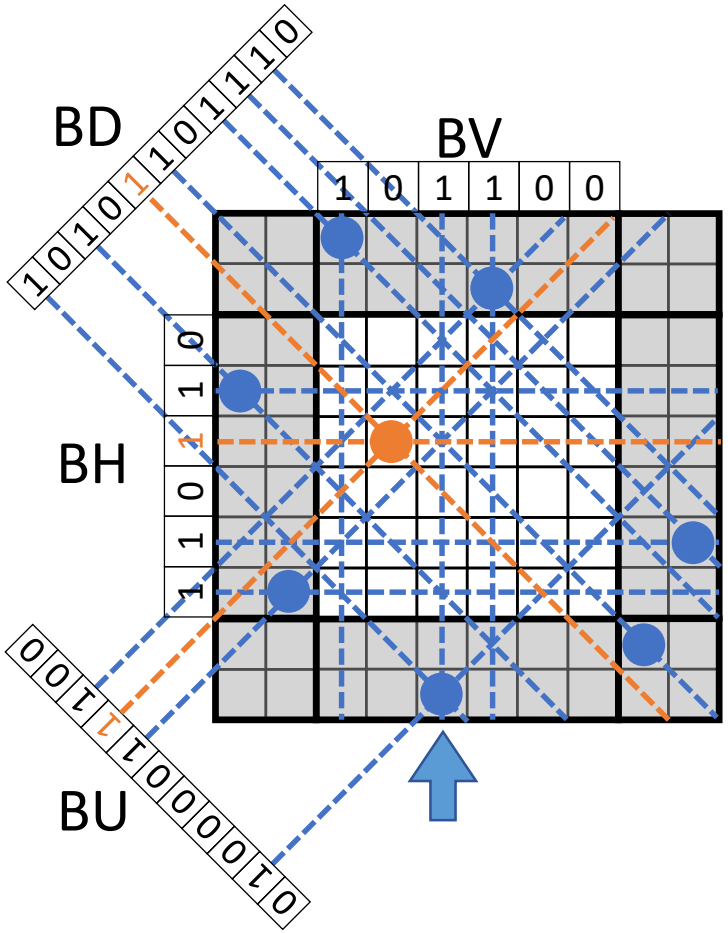


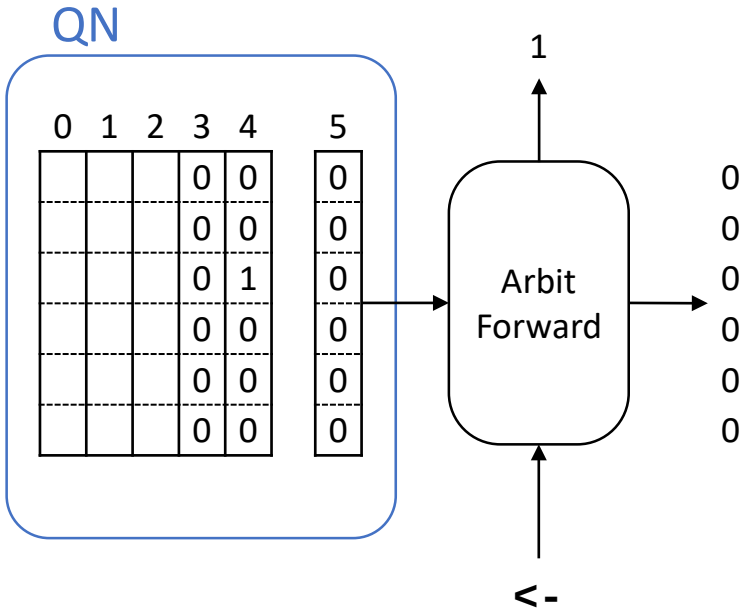
Solving a subproblem on a solver



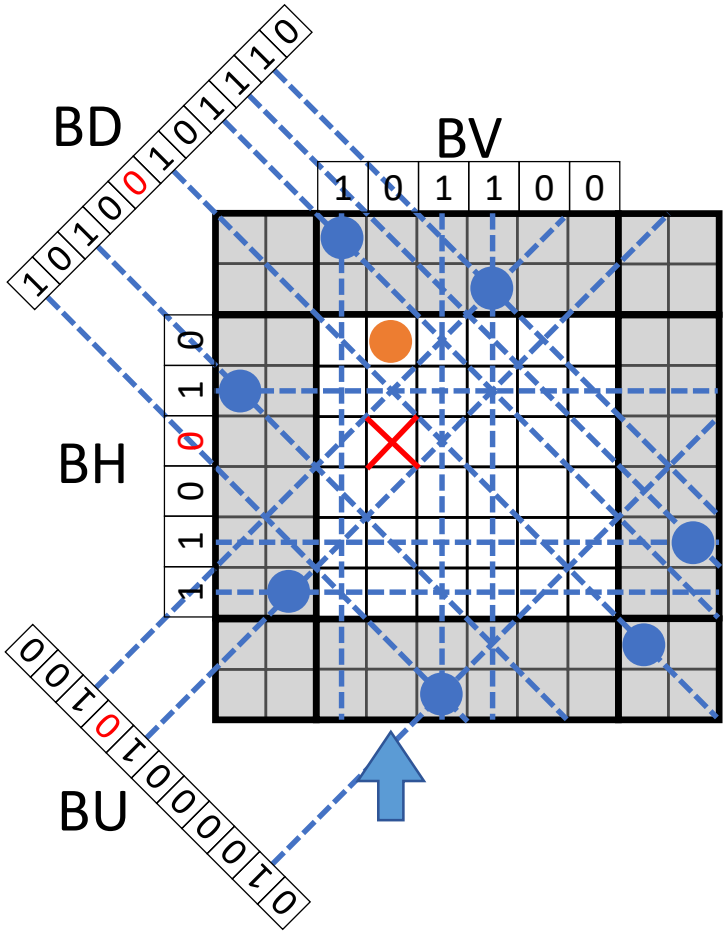


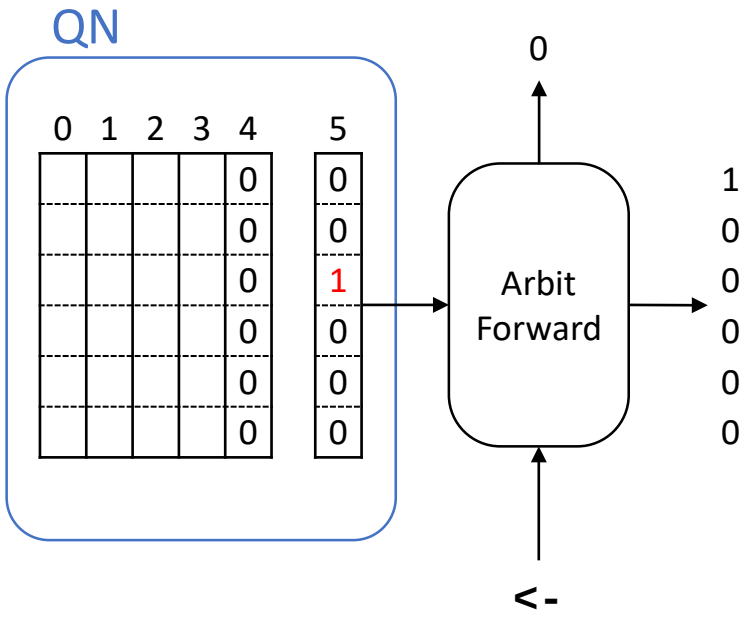
Solving a subproblem on a solver



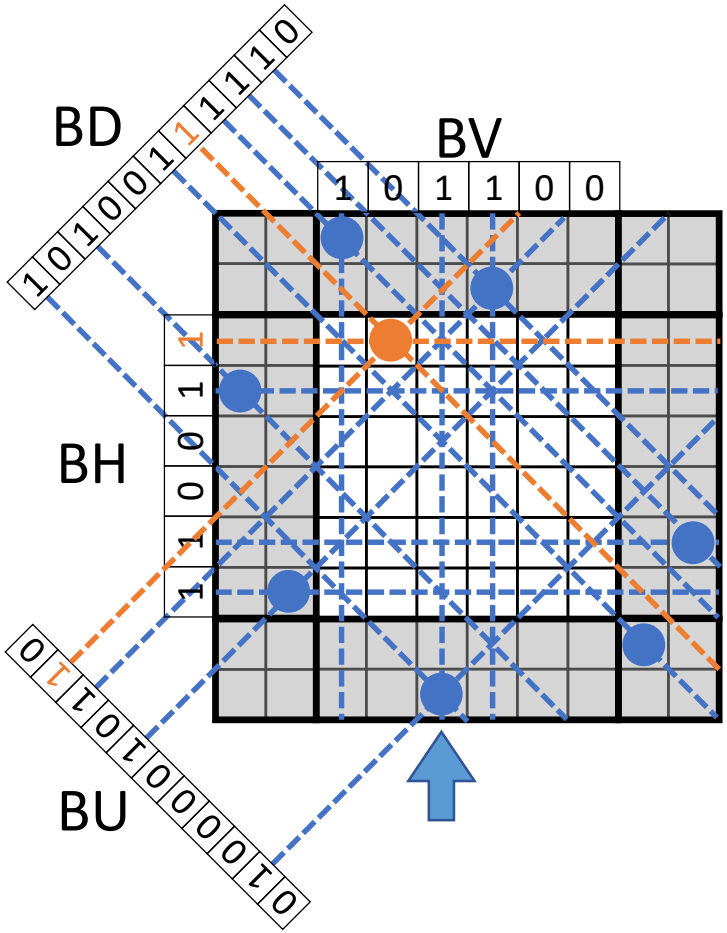


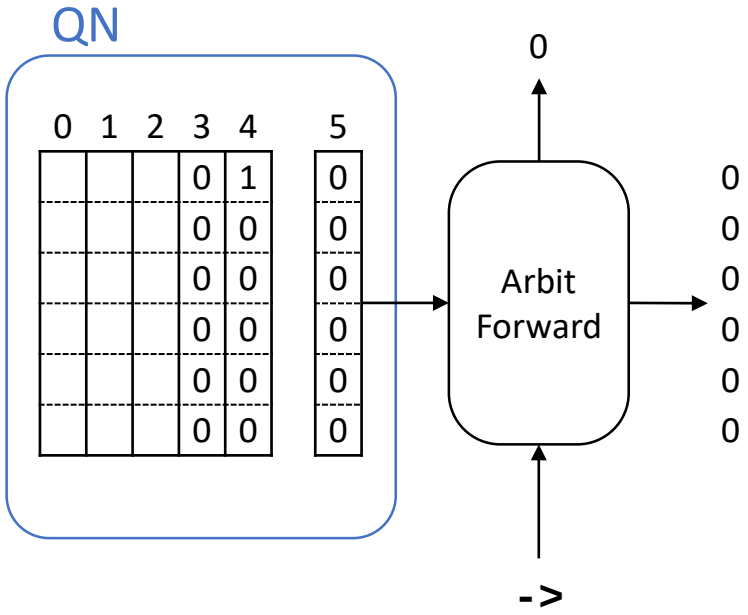
Solving a subproblem on a solver





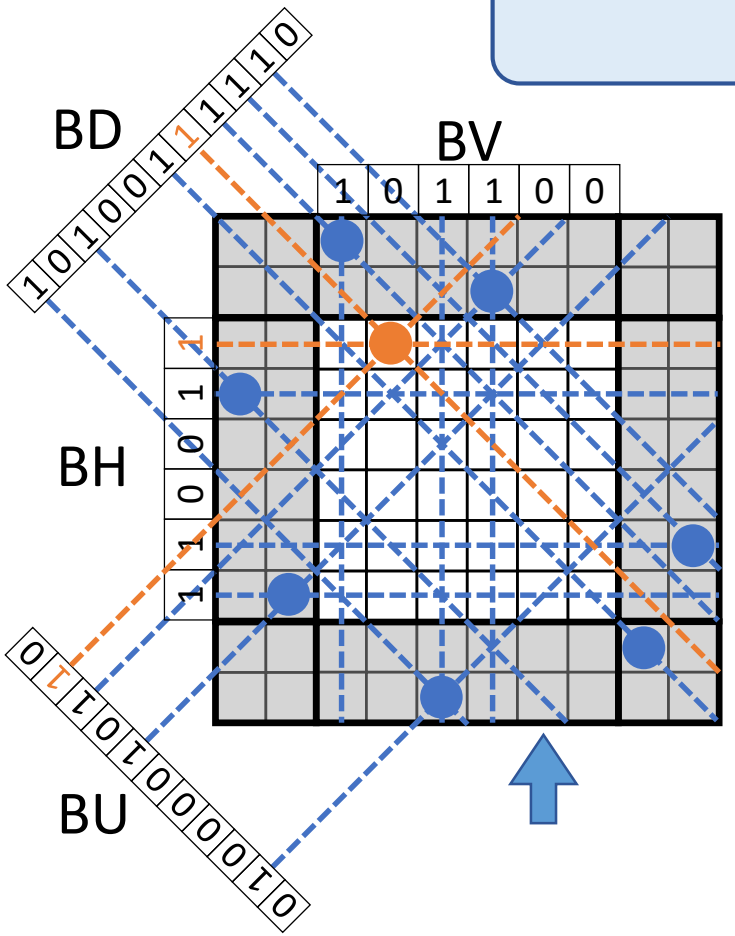
Solving a subproblem on a solver

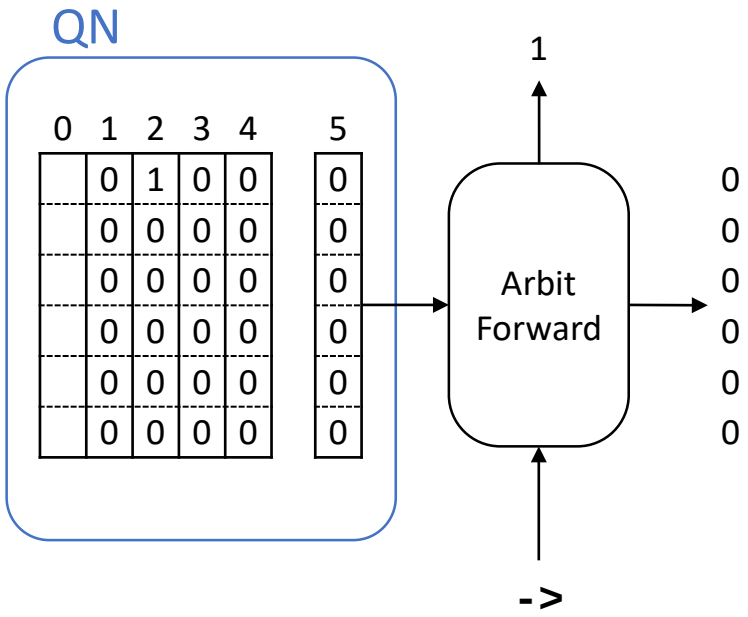




Solving a subproblem on a solver

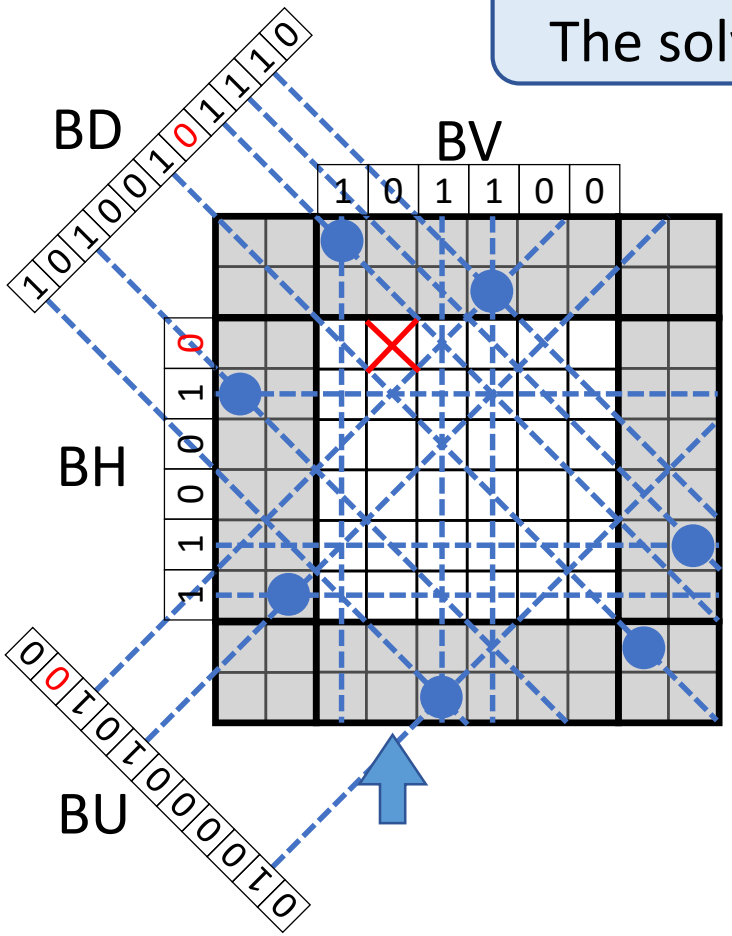
2 cycles later…
A queen can't be placed

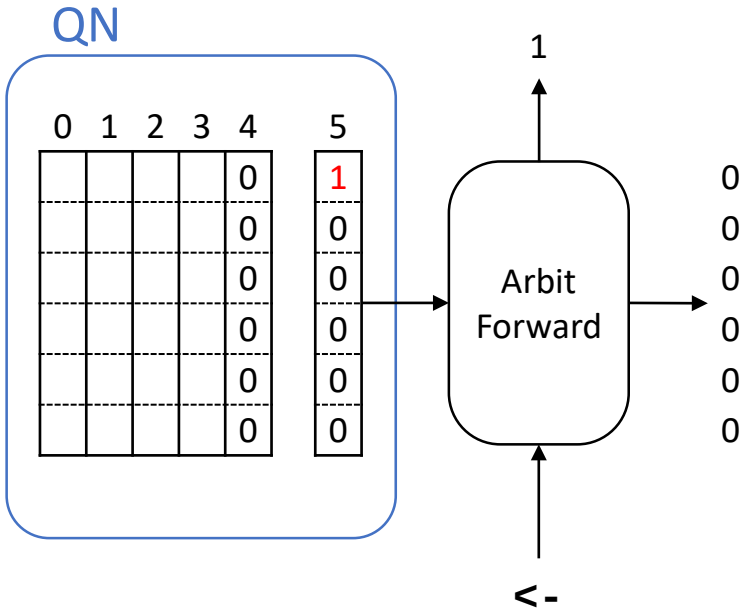




Solving a subproblem on a solver

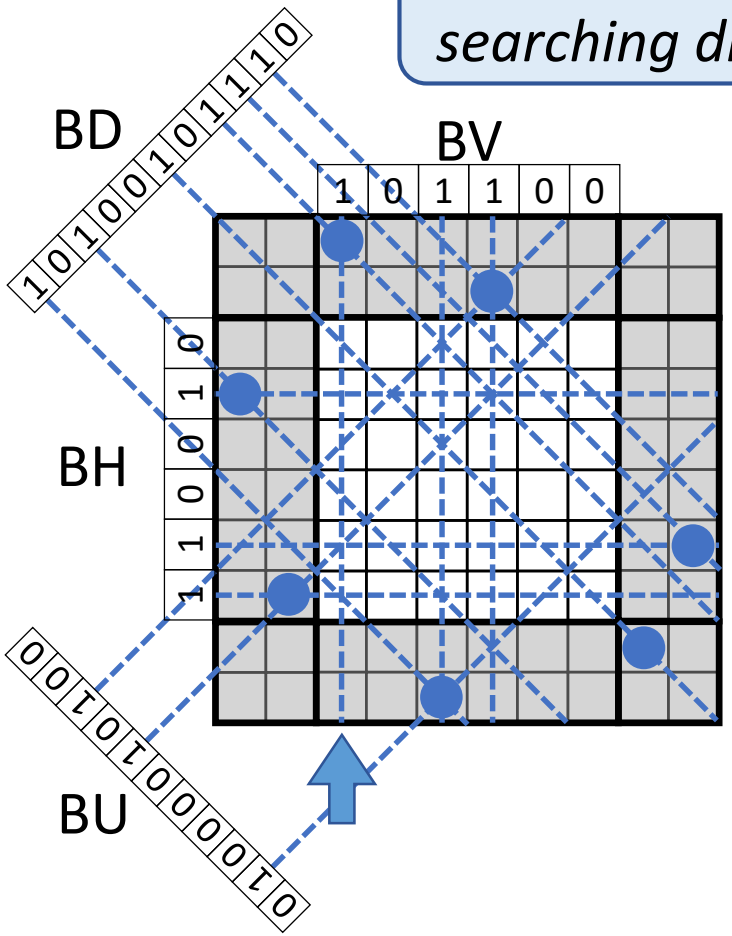
3 cycles later…
The solver have already reached the top

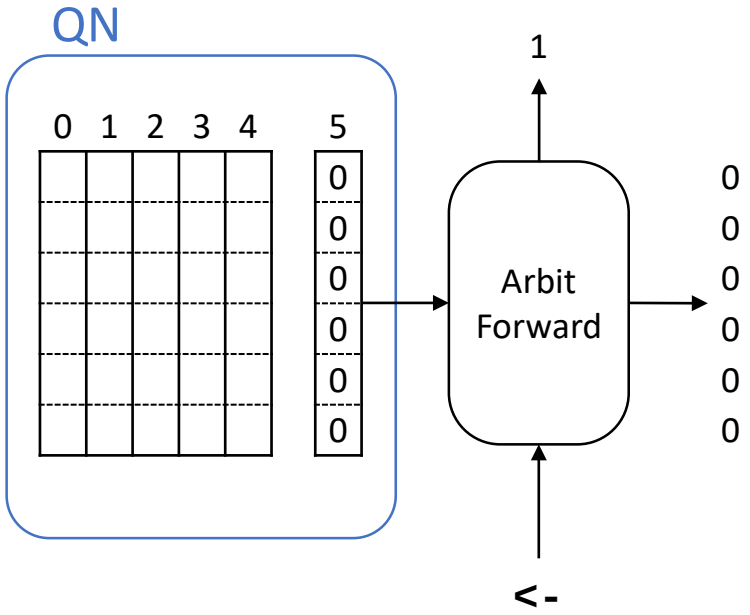




Solving a subproblem on a solver

The Solver asserts the signal *done_bit* when *searching direction* is left in the leftmost column

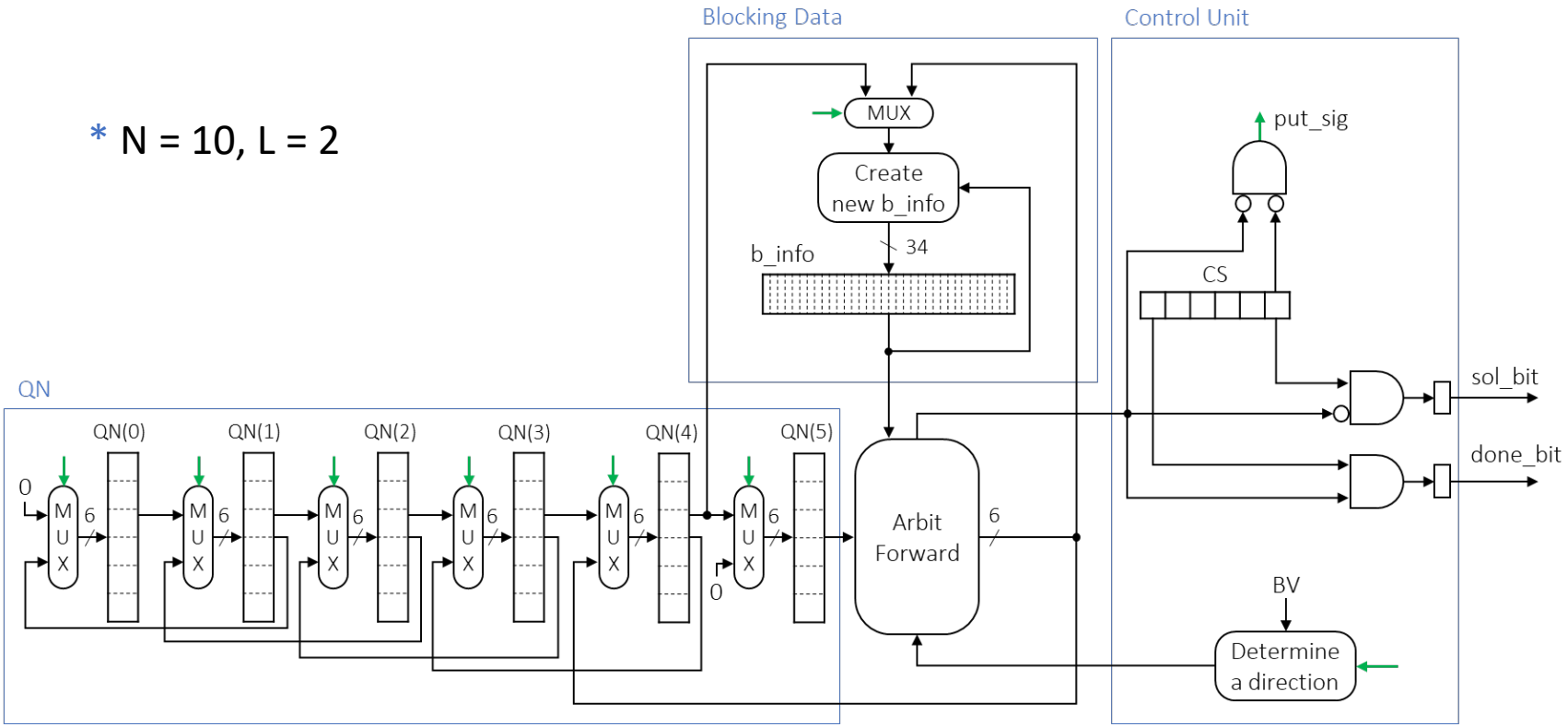




The Solver Architecture

- The solver architecture on the previous work

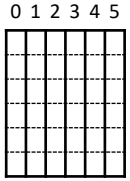
* N = 10, L = 2



Operation of QN on Solver

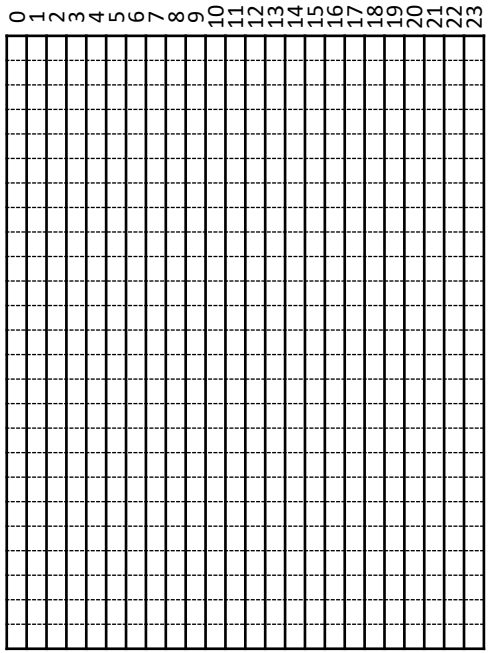
QN uses too many registers when N increases

QN with N=10, L=2



$6 \times 6 = 36\ bits$

QN with N=28, L=2



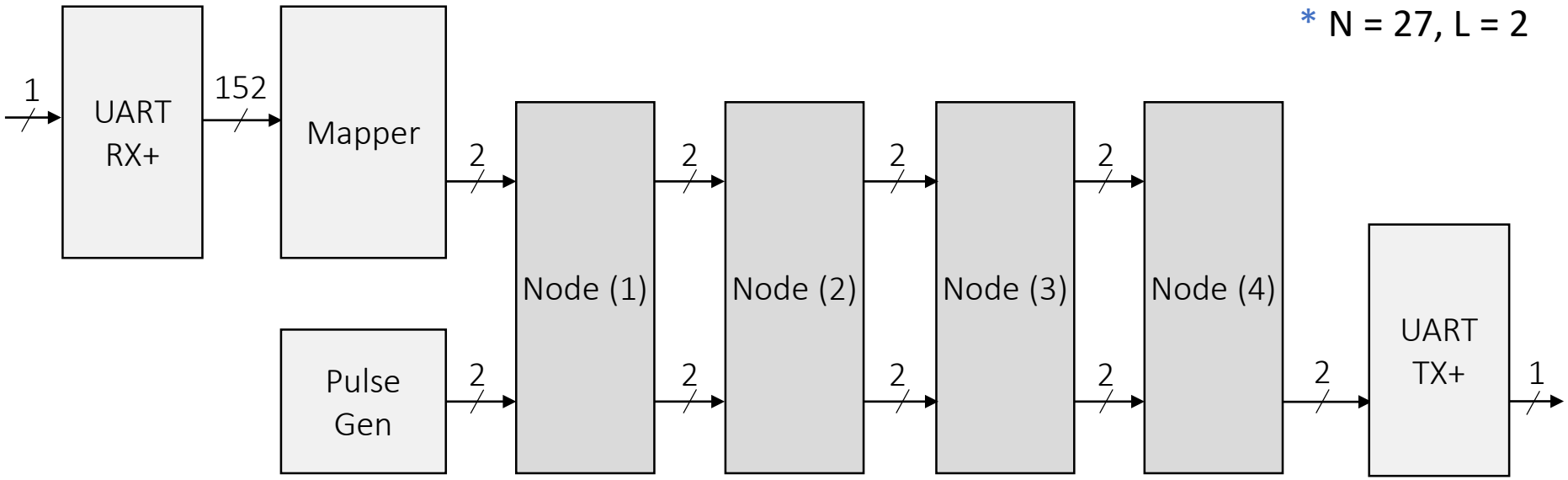
$24 \times 24 = 576\ bits$

Outline

1. Introduction
2. Proposal
    1. The FPGA System
    2. The Solver
3. Evaluation
4. Conclusion

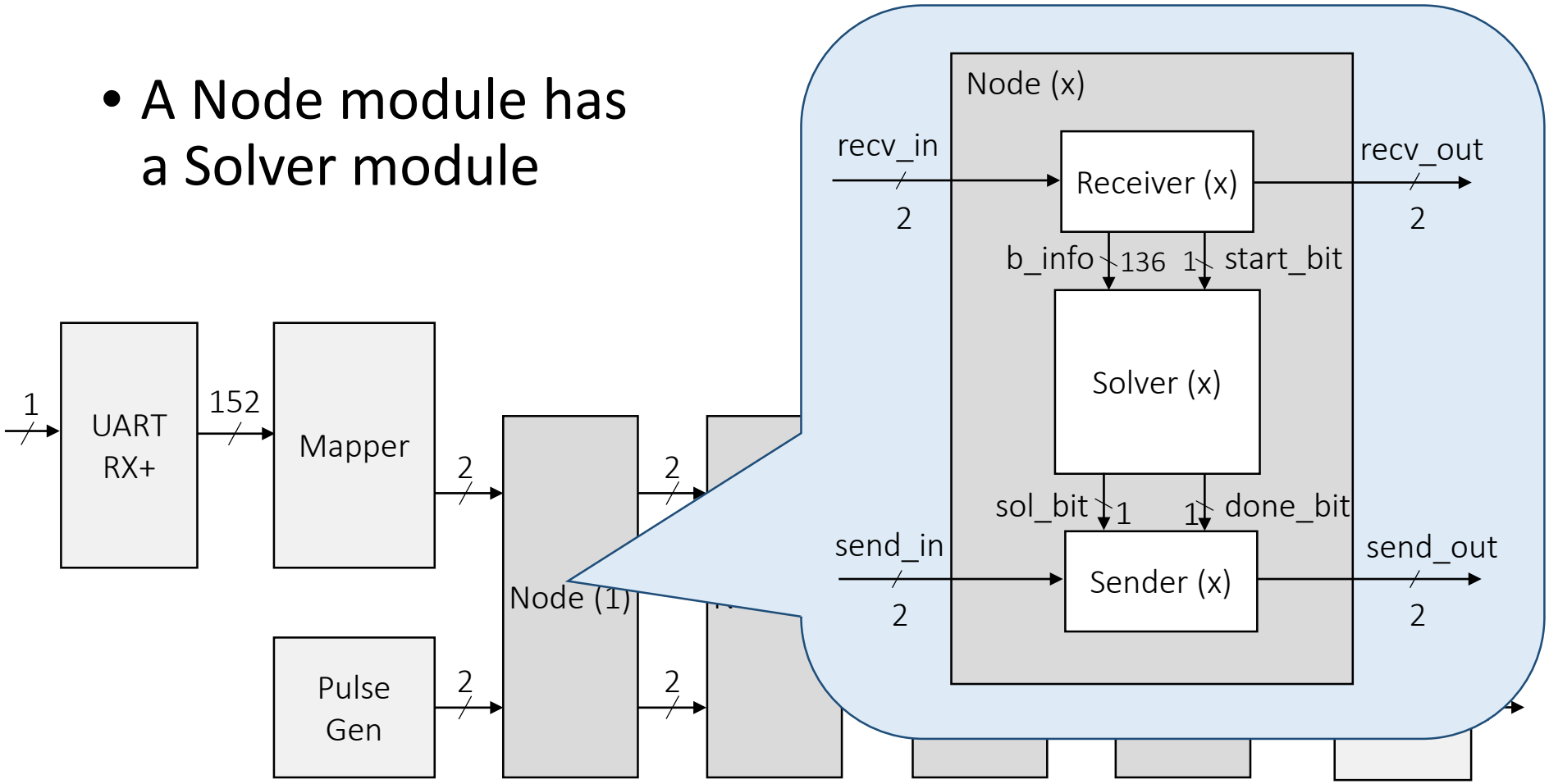
The Proposed FPGA System

- A cost-effective FPGA architecture to distribute subproblems and to collect obtained solutions



The Proposed FPGA System

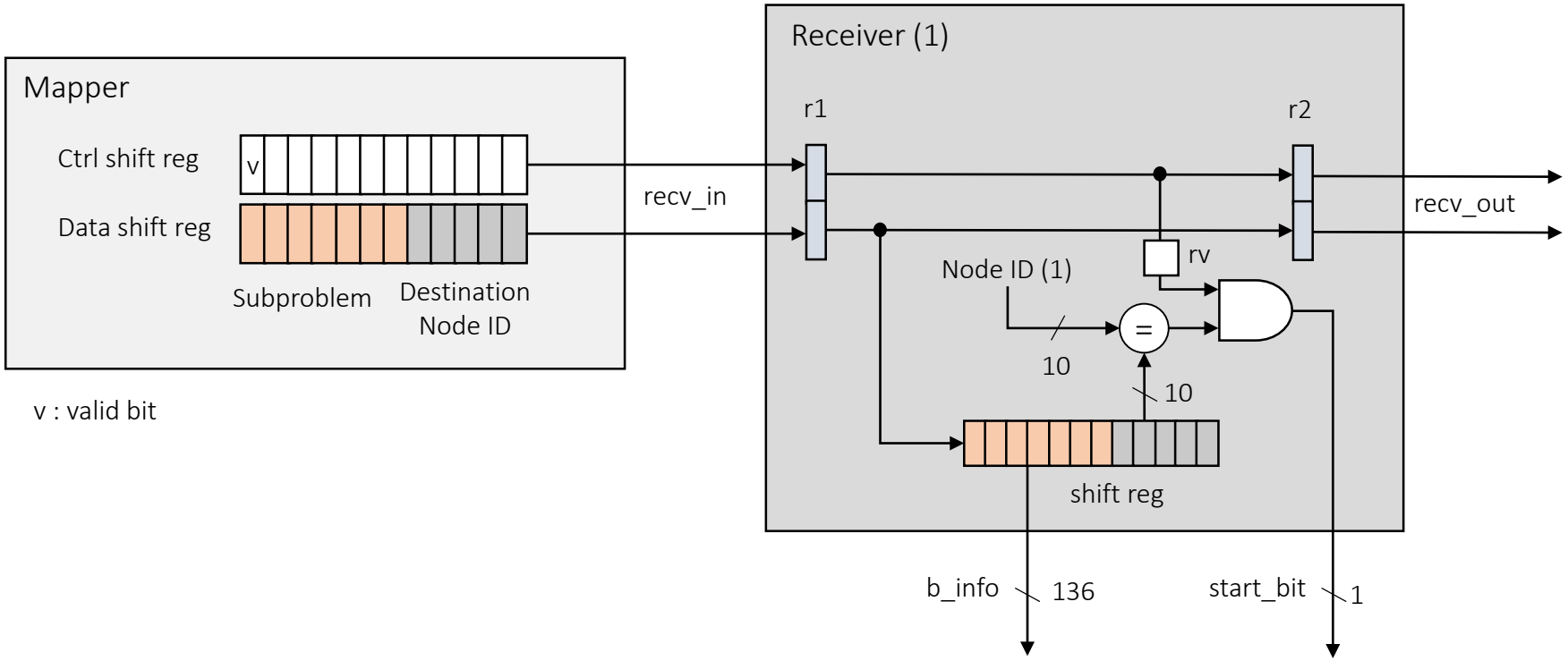
- A Node module has a Solver module



* N = 27, L = 2

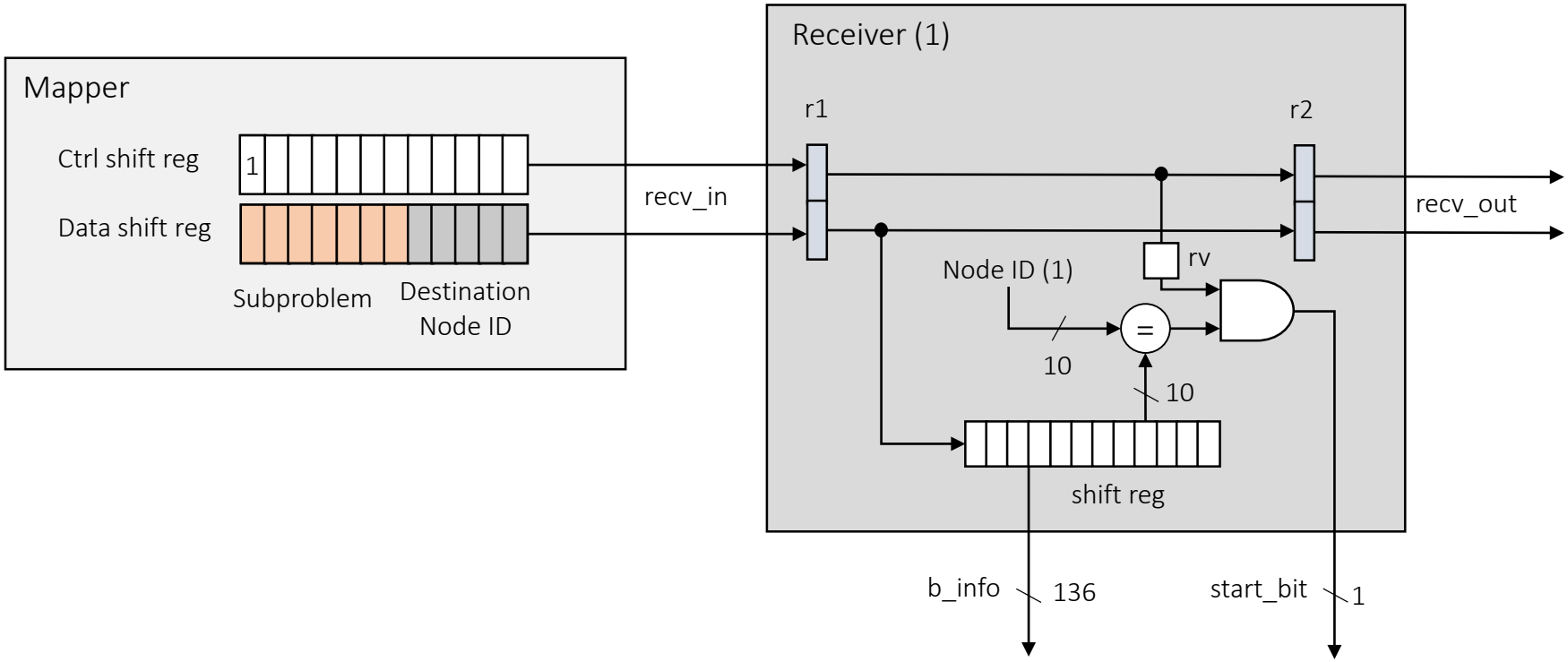
Receiver

- The receiver module sends the data of subproblems received from PC to solvers

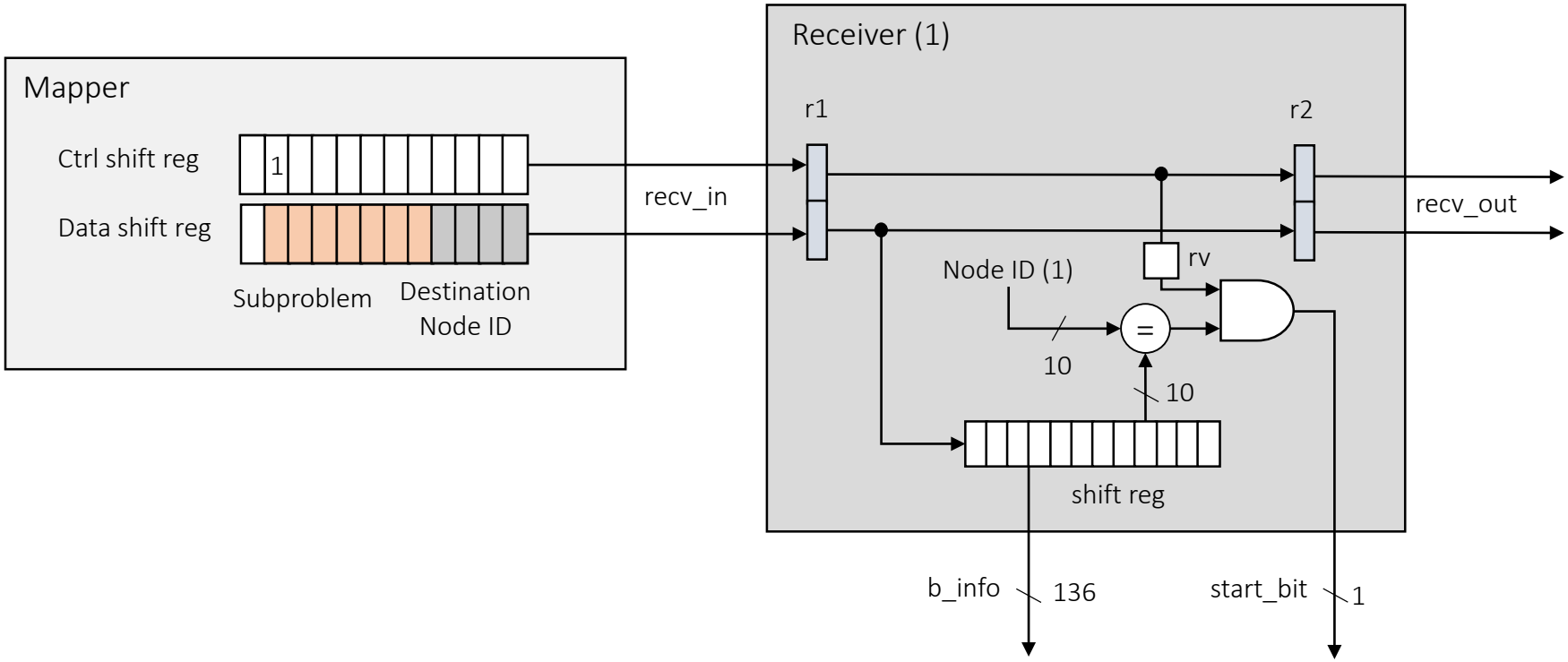


Receiver

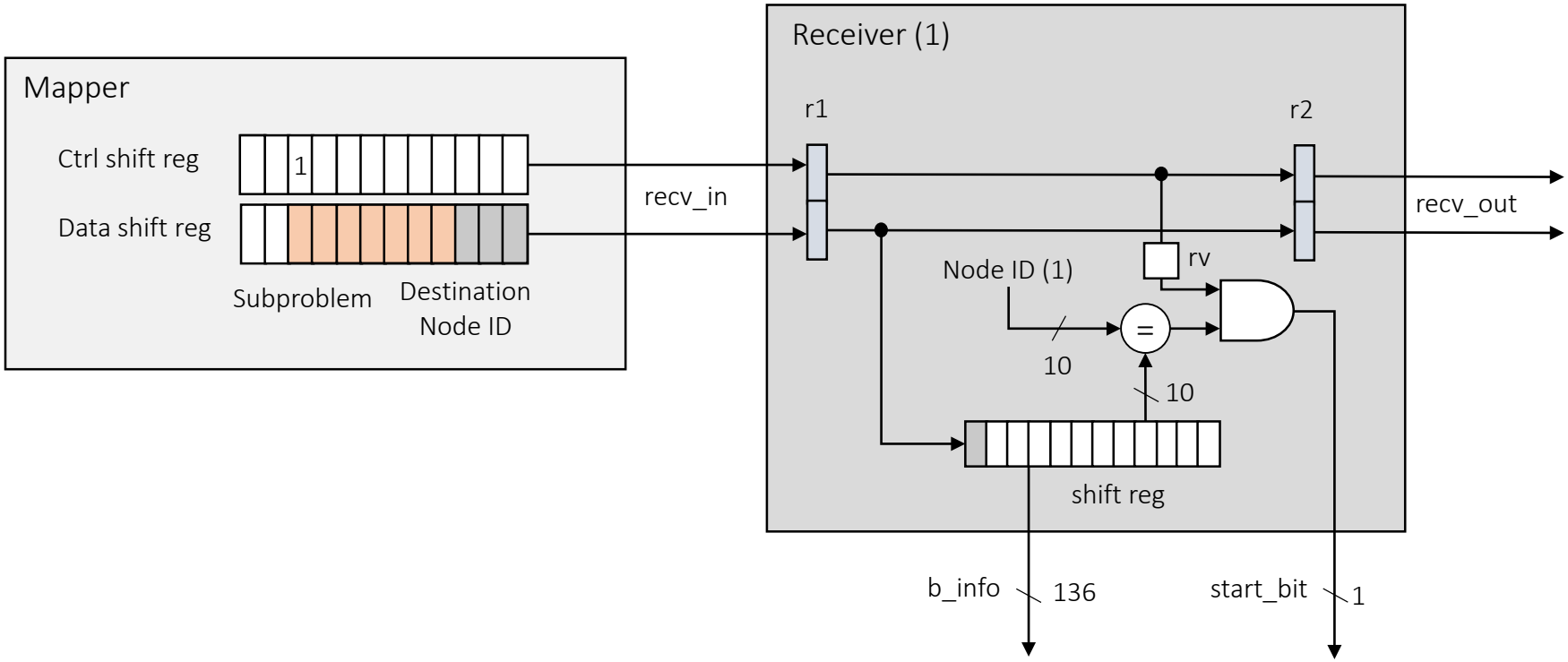
- Example of operation



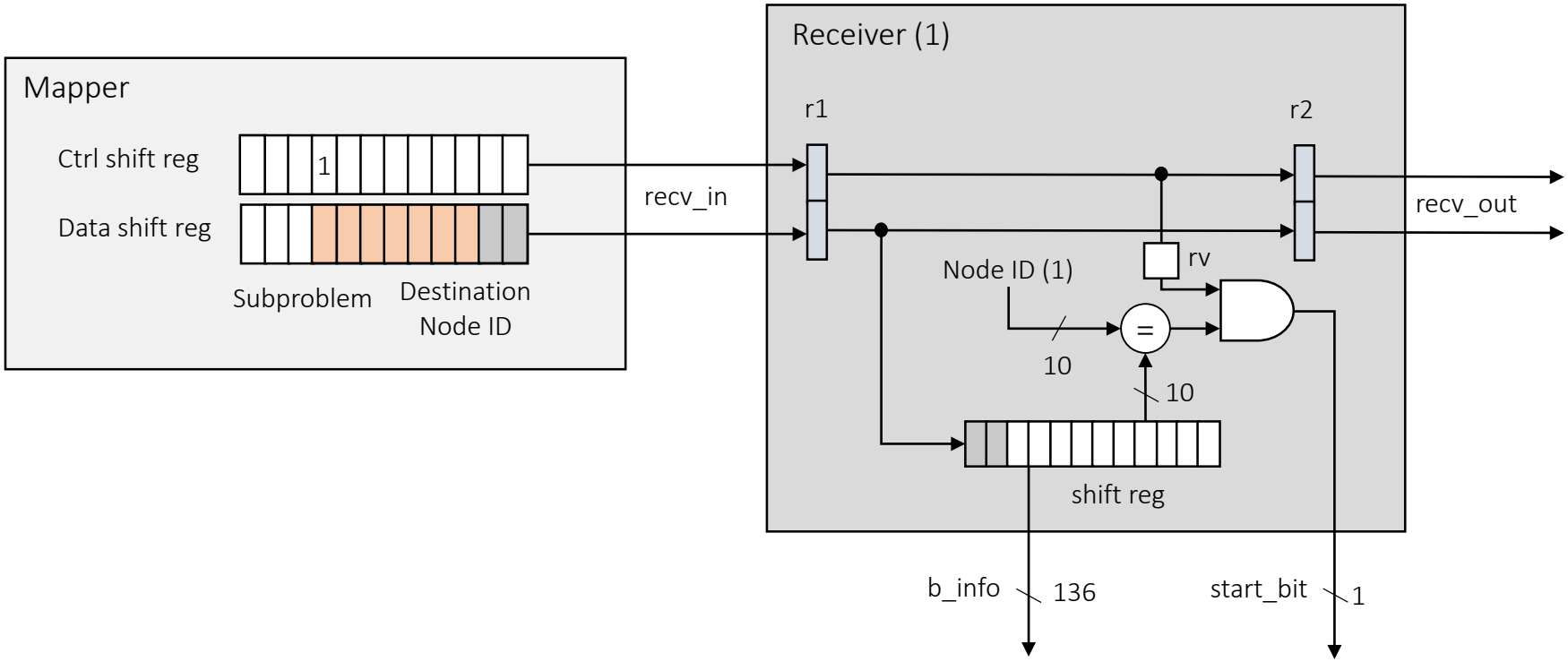
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
10
=
10
shift reg
b_info
136
start_bit
1

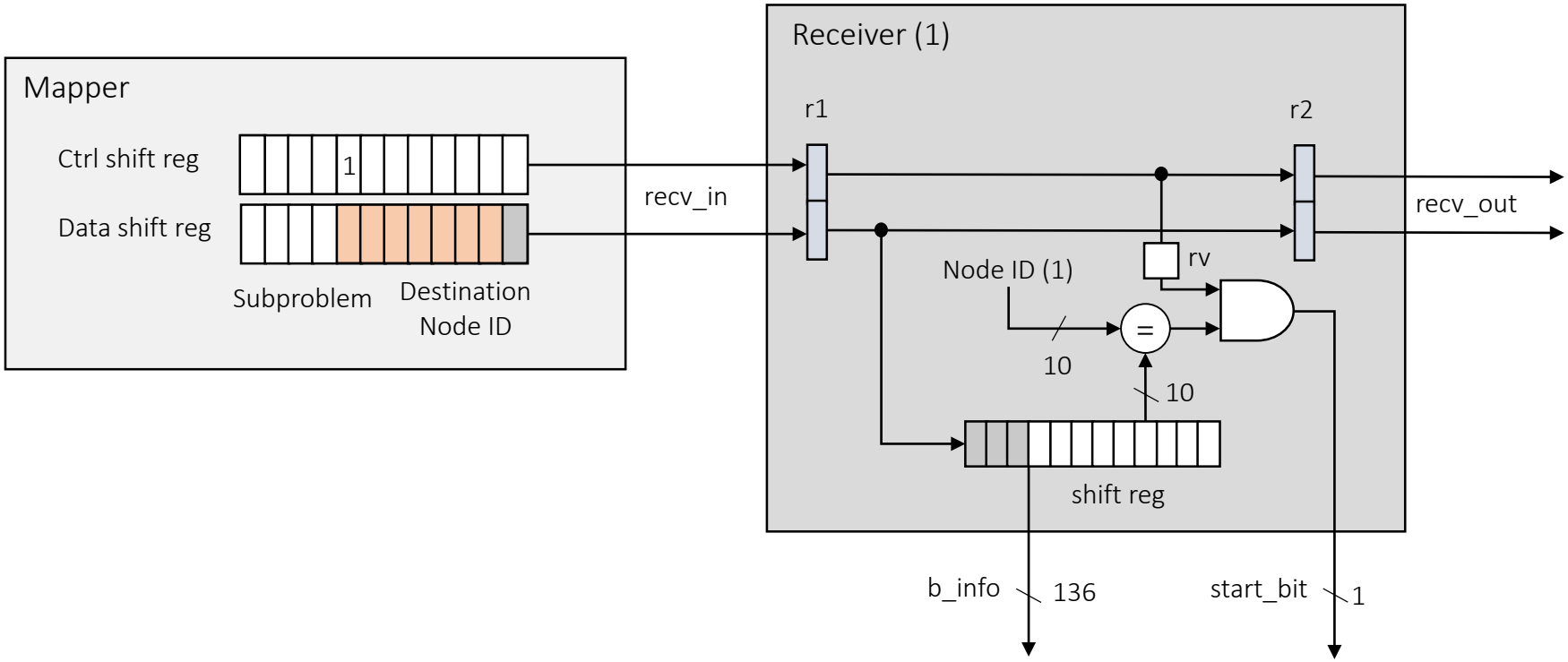
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

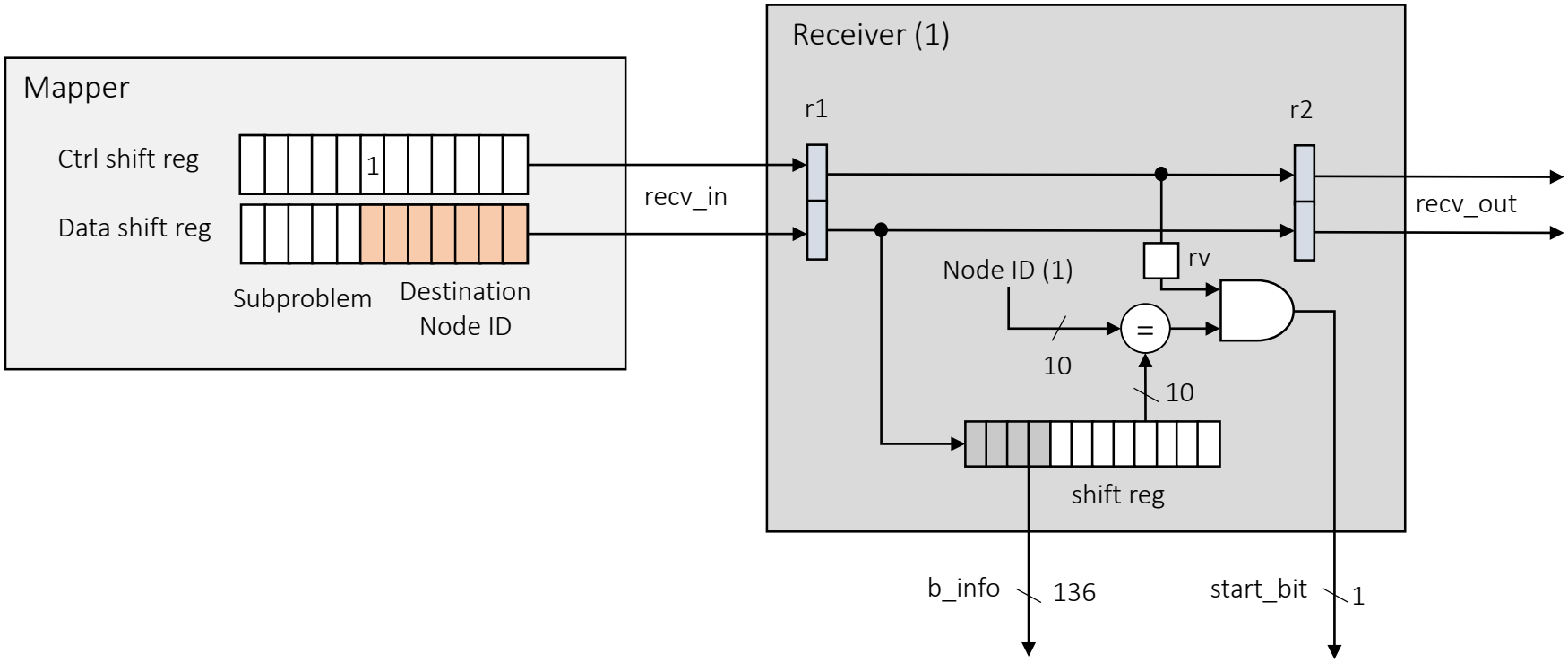
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

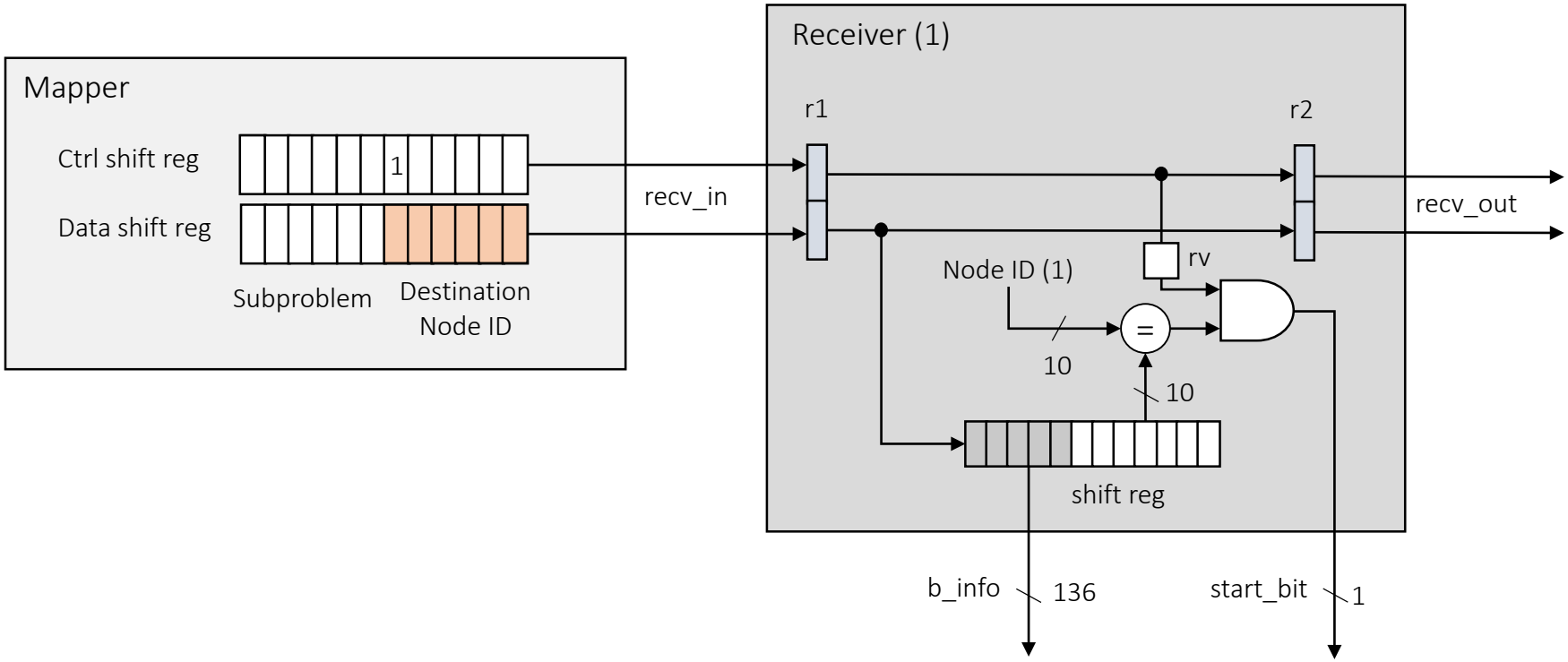
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

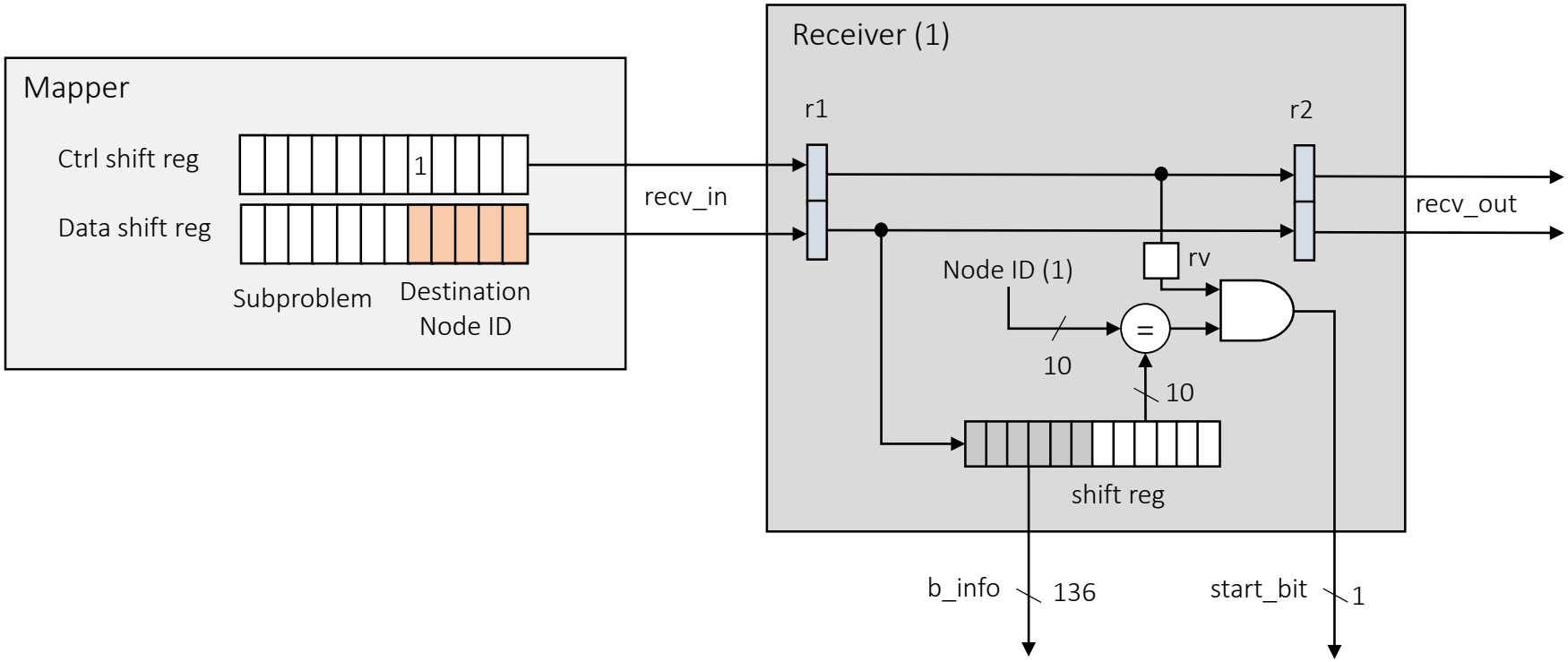
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

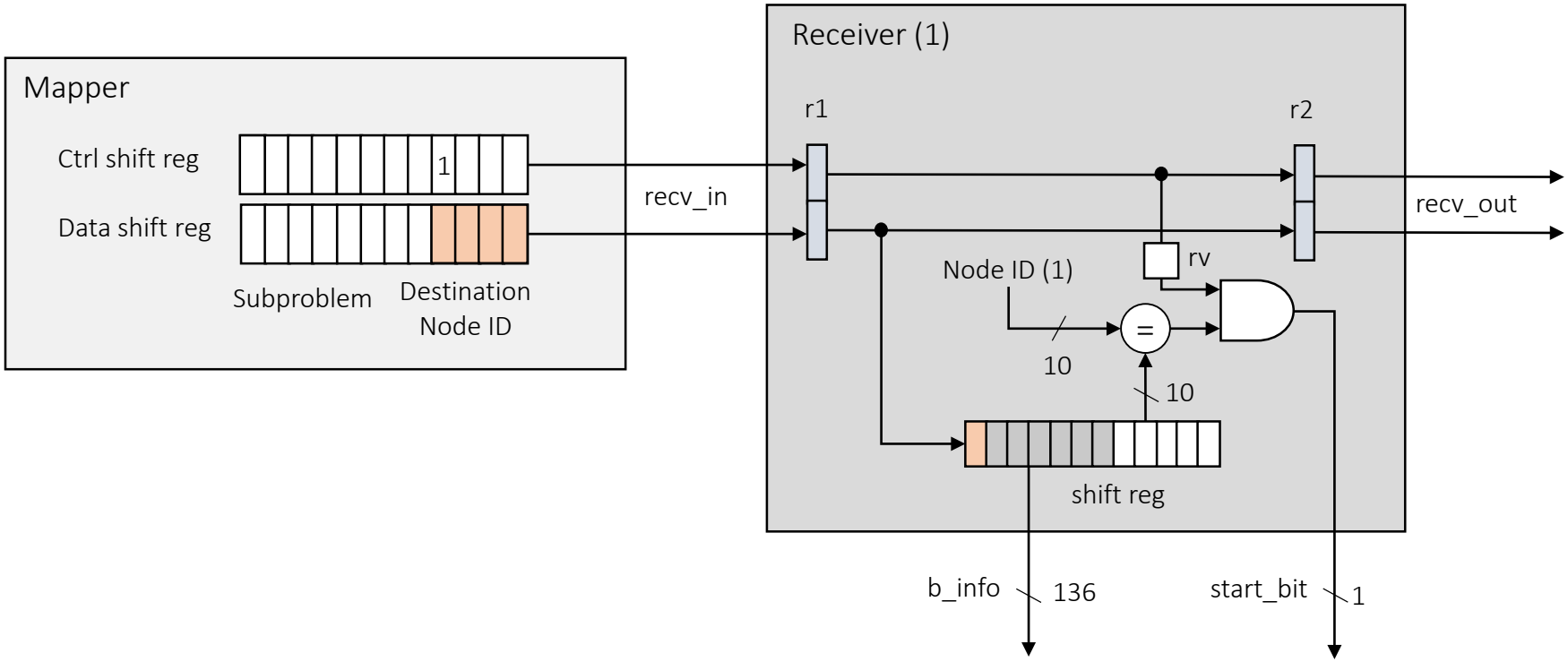
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

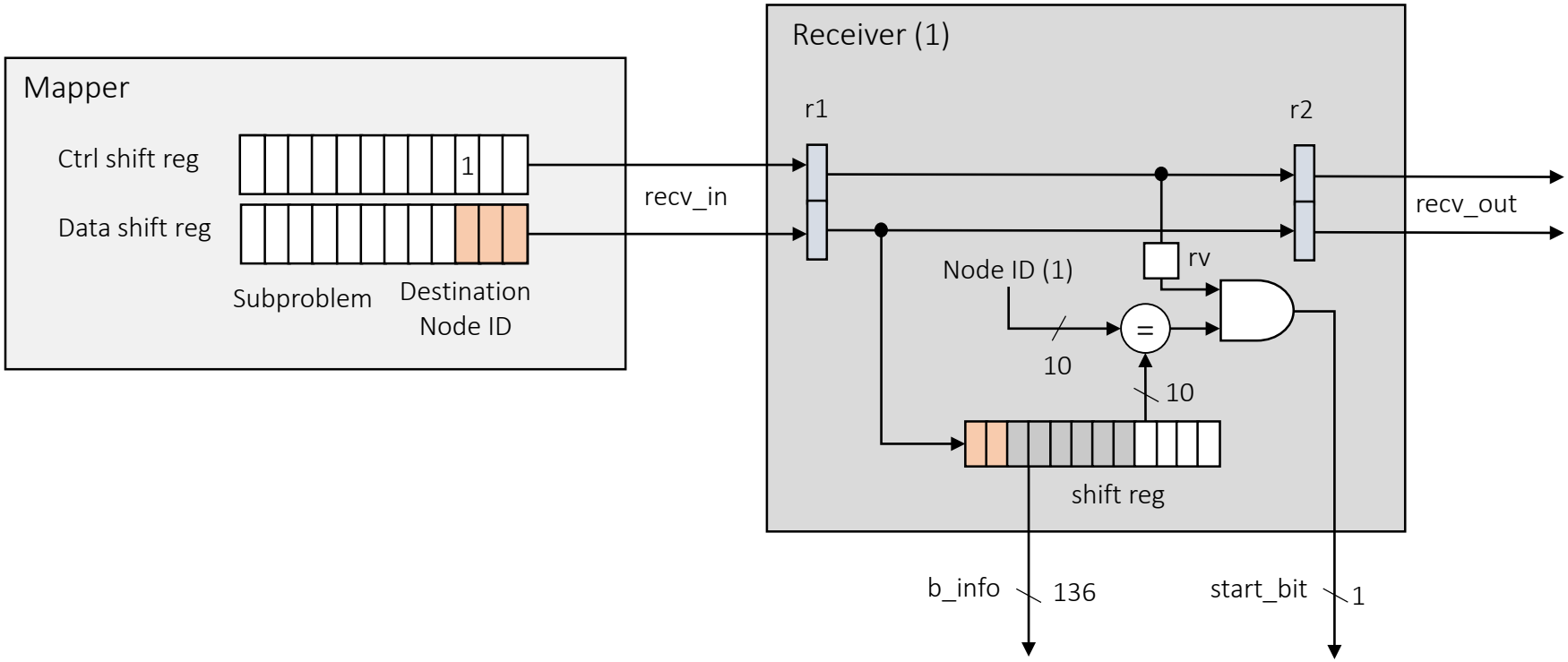
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

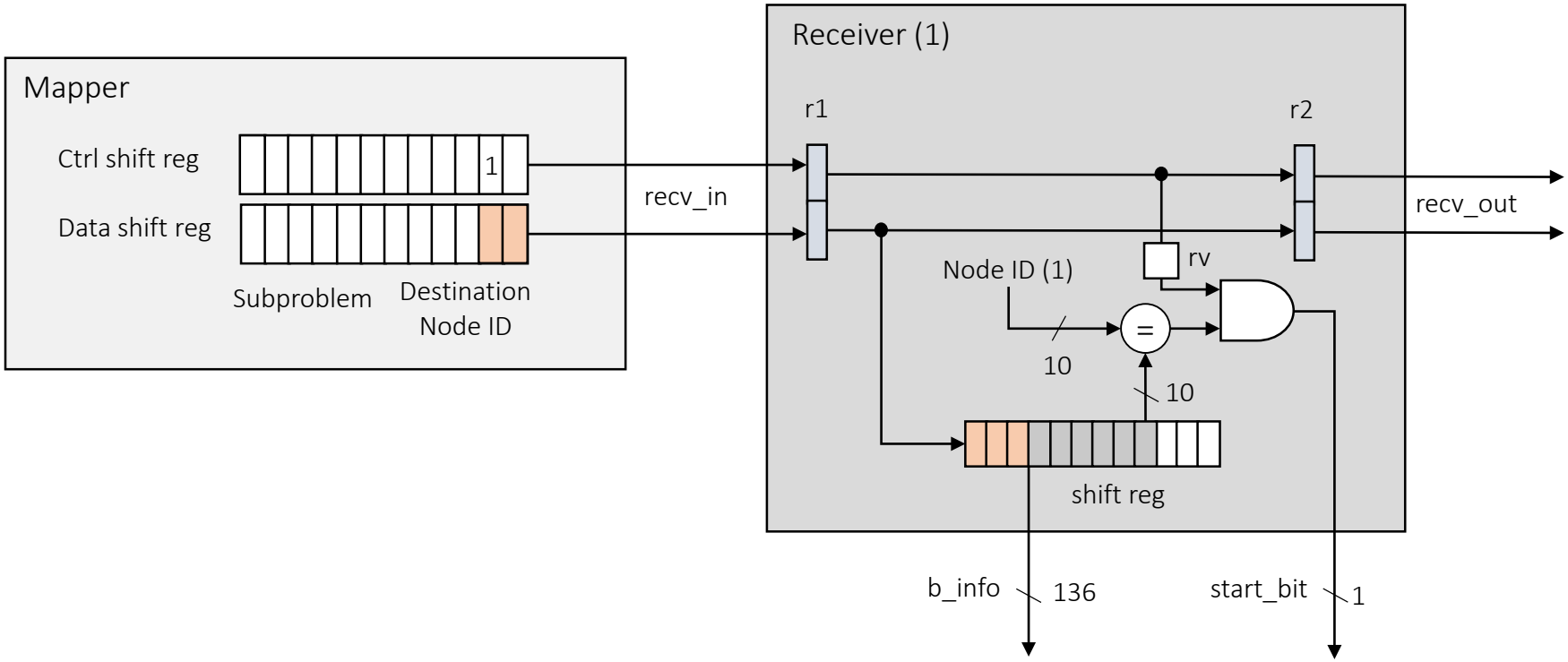
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

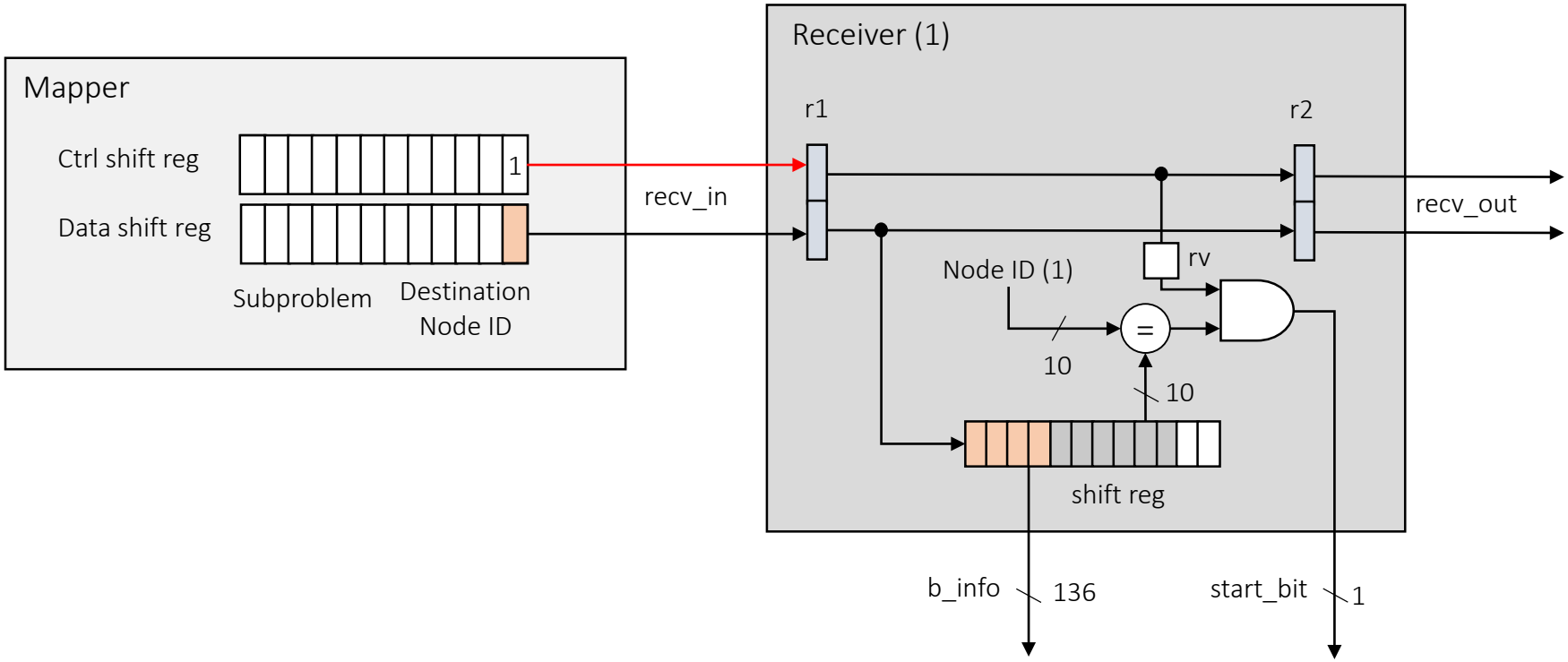
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

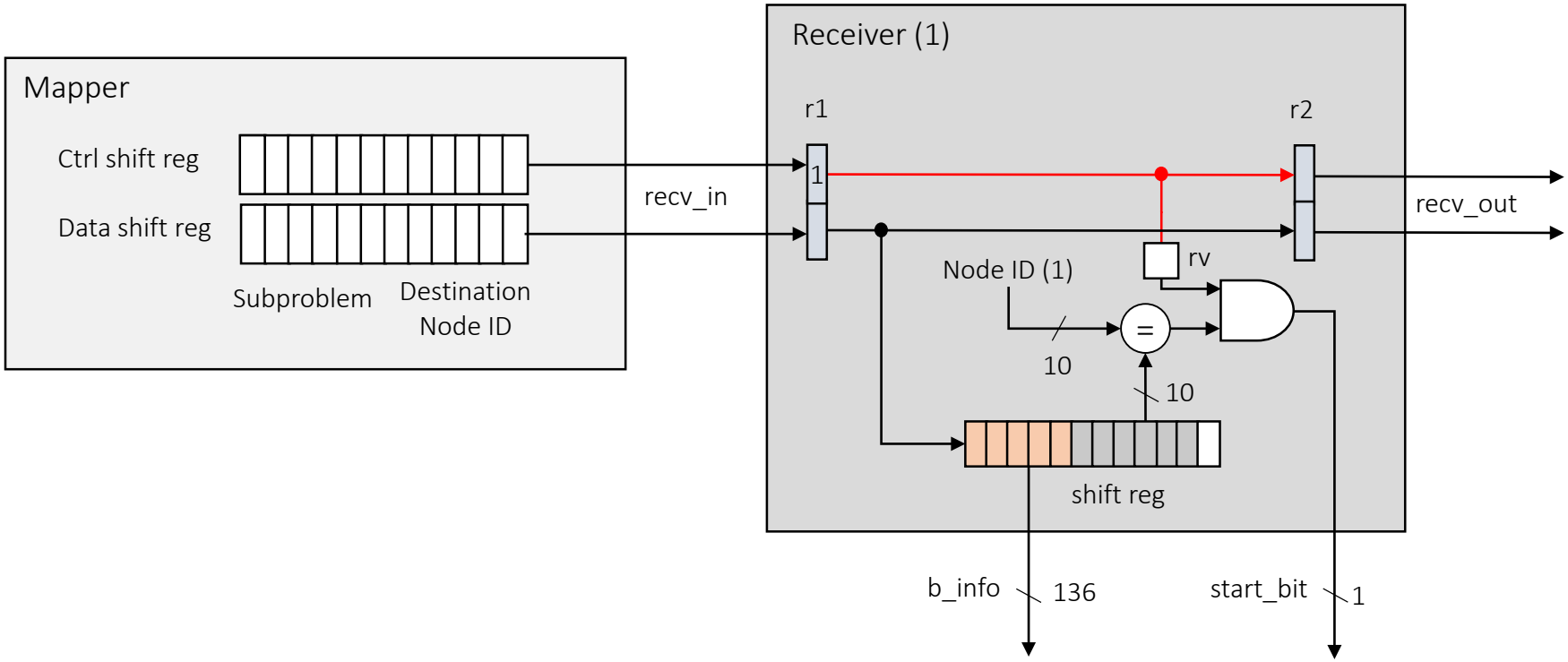
Receiver


Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

Receiver

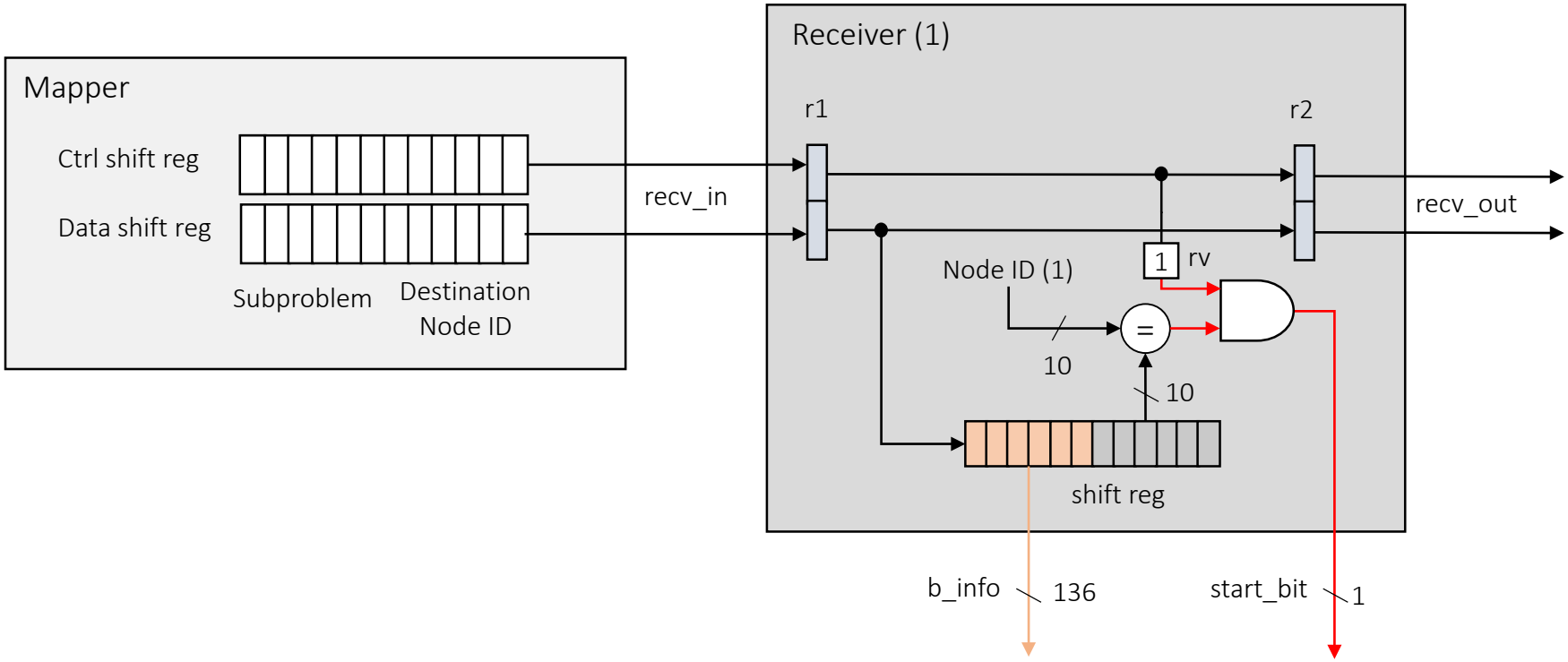

Mapper
Ctrl shift reg
1
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

Receiver


Mapper
Ctrl shift reg
Data shift reg
Subproblem
Destination Node ID
recv_in
Receiver (1)
r1
r2
1
recv_out
rv
Node ID (1)
=
10
10
shift reg
b_info
136
start_bit
1

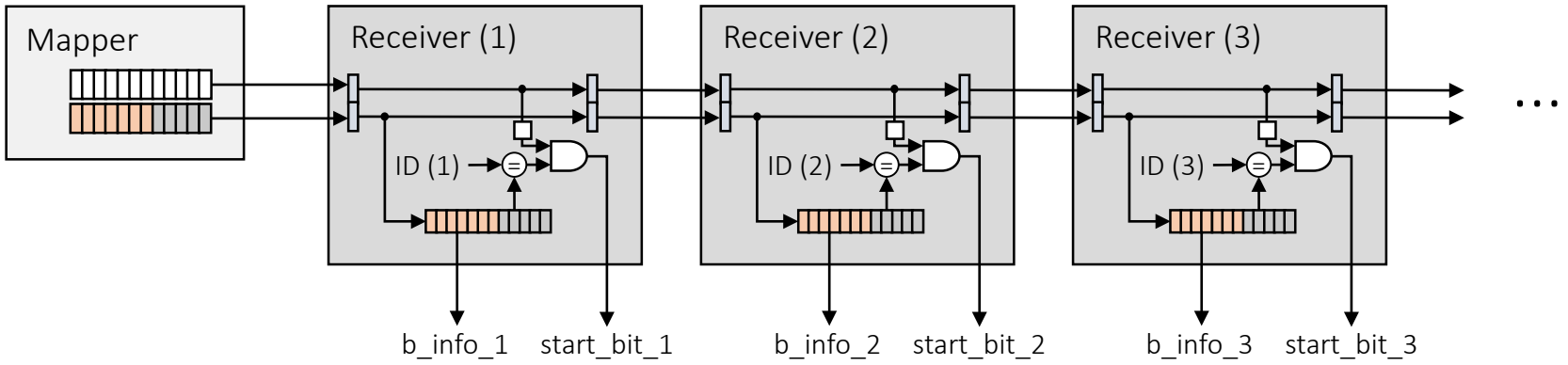
Receiver

- The receiver does NOT have any counters
  - The critical path is removed



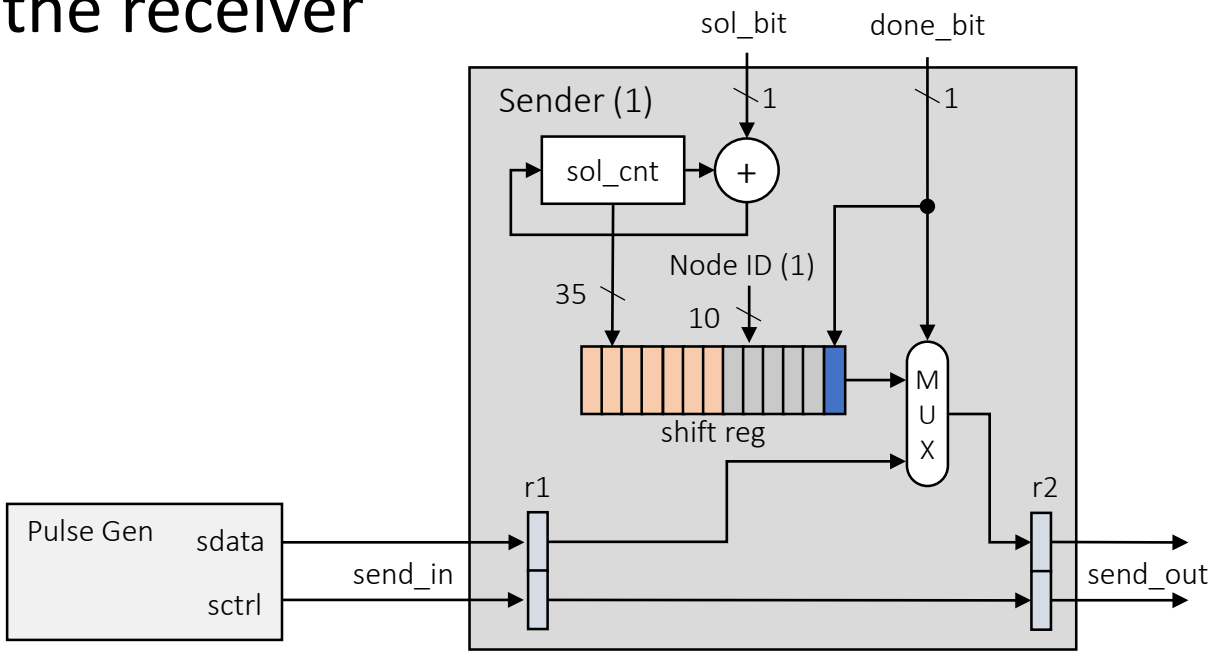
Receiver

- This mechanism is scalable



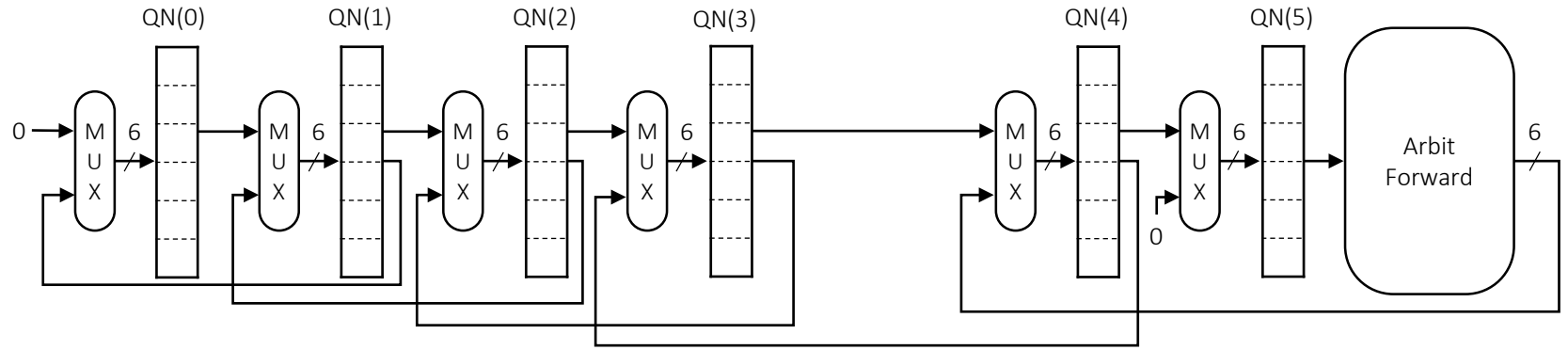
Sender

- The sender module sends data to UART-TX+
- A clock counter can be omitted by a method similar to the receiver

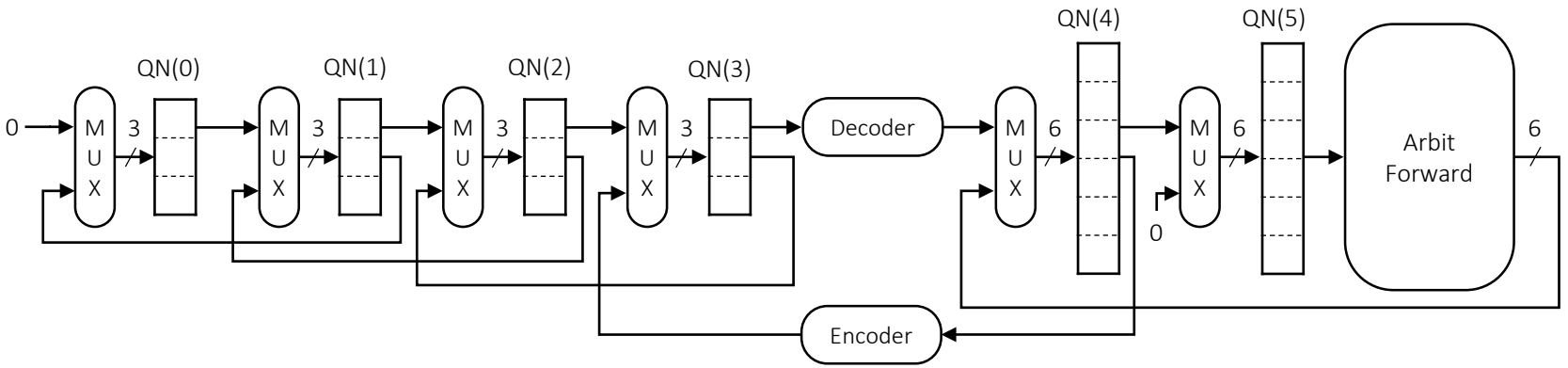


The Proposed Solver

* N = 10, L = 2



(a) Q27 Solver

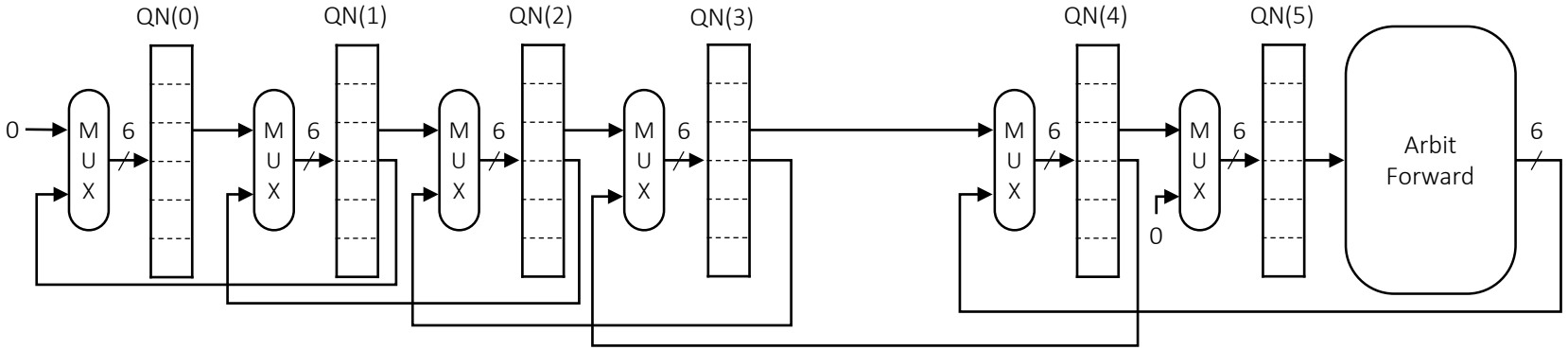


(b) Proposal

The Proposed Solver

- QN have the information that where queen is placed in that column
- QN is one-hot
- So that information can be compressed

* N = 10, L = 2



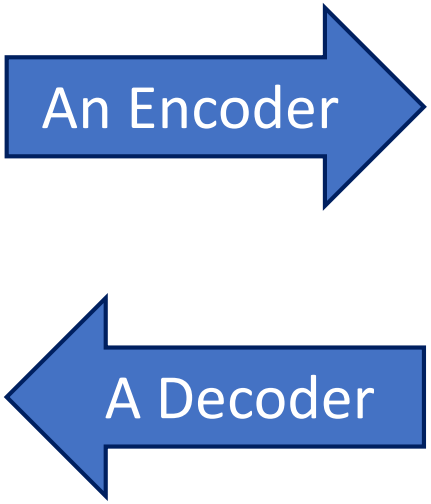
(a) Q27 Solver

The Proposed Solver

* N = 27, L = 2

- The Encoding/Decoding logic

Original QN
000000000000000000000 (not place)
000000000000000000001
000000000000000000010
000000000000000000100
000000000000000001000
000000000000000010000
000000000000000100000
⋮

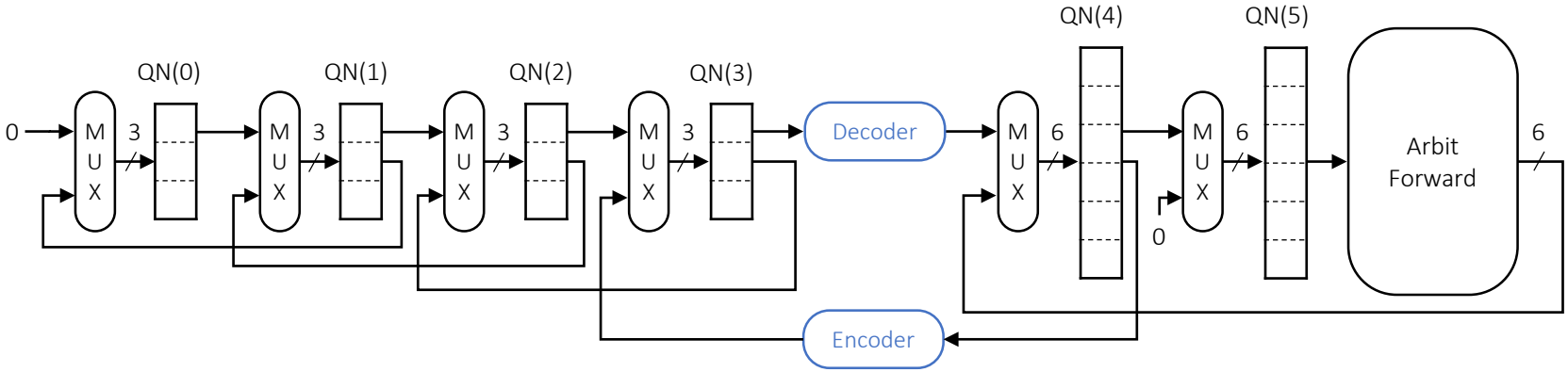


Encoded QN
00000
00001
00010
00011
00100
00101
00110
⋮

The Proposed Solver

- We insert an encoder and a decoder
- An encoder and a decoder is simple

* N = 10, L = 2

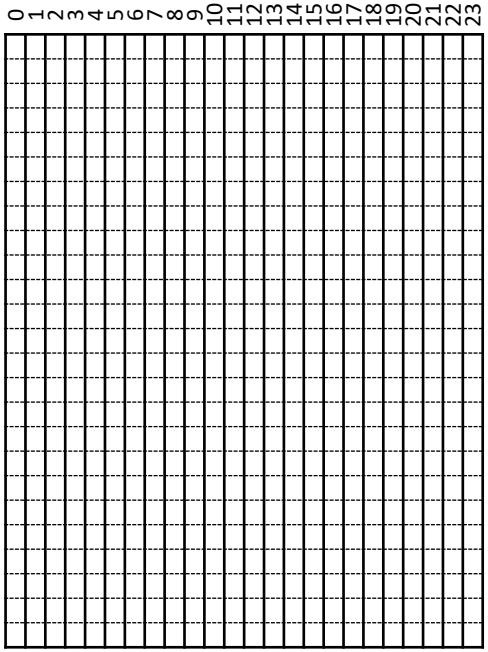


(b) Proposal

QN on Proposed Solver

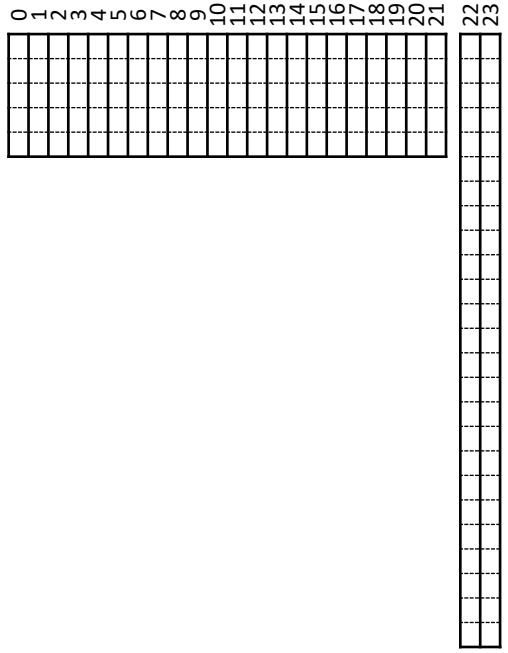
The register usage can be significantly reduced

Previous QN with N=28, L=2



$24 \times 24 = 576\ bits$

Proposed QN with N=28, L=2



$5 \times 21 + 24 \times 2 = 153\ bits$

Outline

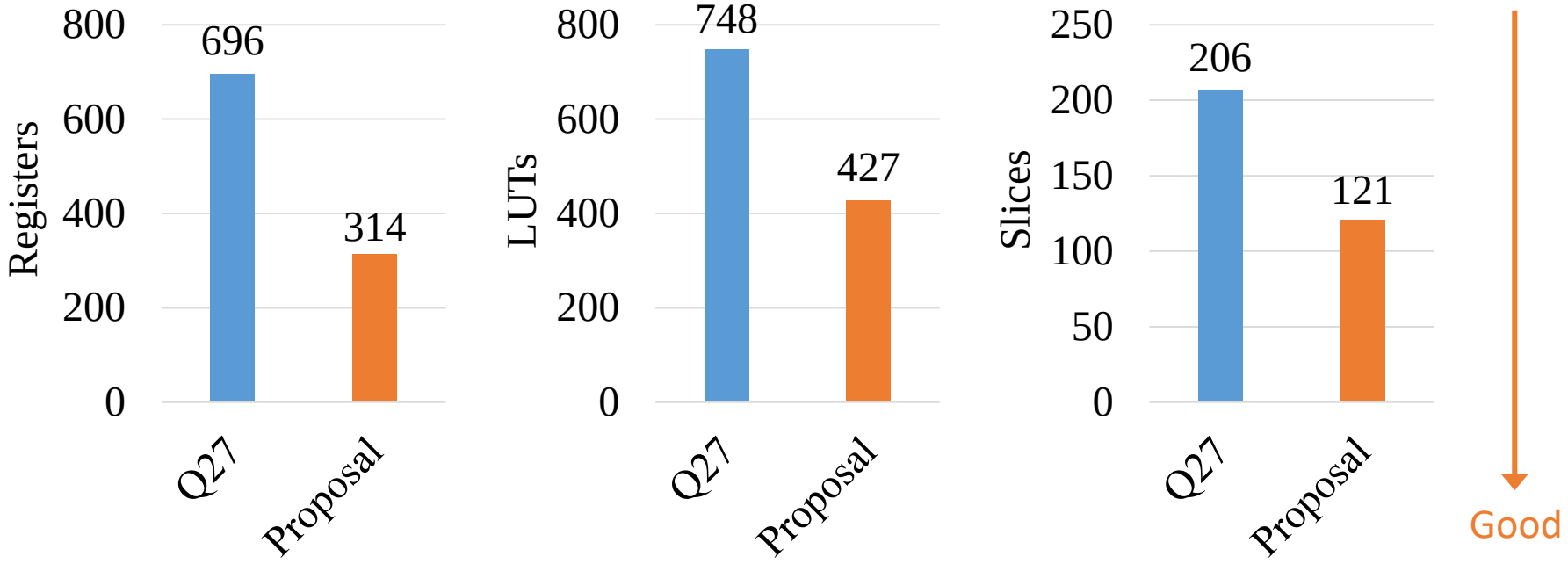
1. Introduction
2. Proposal
3. Evaluation
4. Conclusion

Environment

- Parameter:
  - N=27, L=2
- Design Tool:
  - Xilinx Vivado Design Suite 2017.2
- Target Device:
  - Xilinx VC707 Evaluation Board (Virtex-7 VX485T)

Evaluation Results (1/3)

- The HW resource usage of one solver module
  - The proposed solver uses less resources



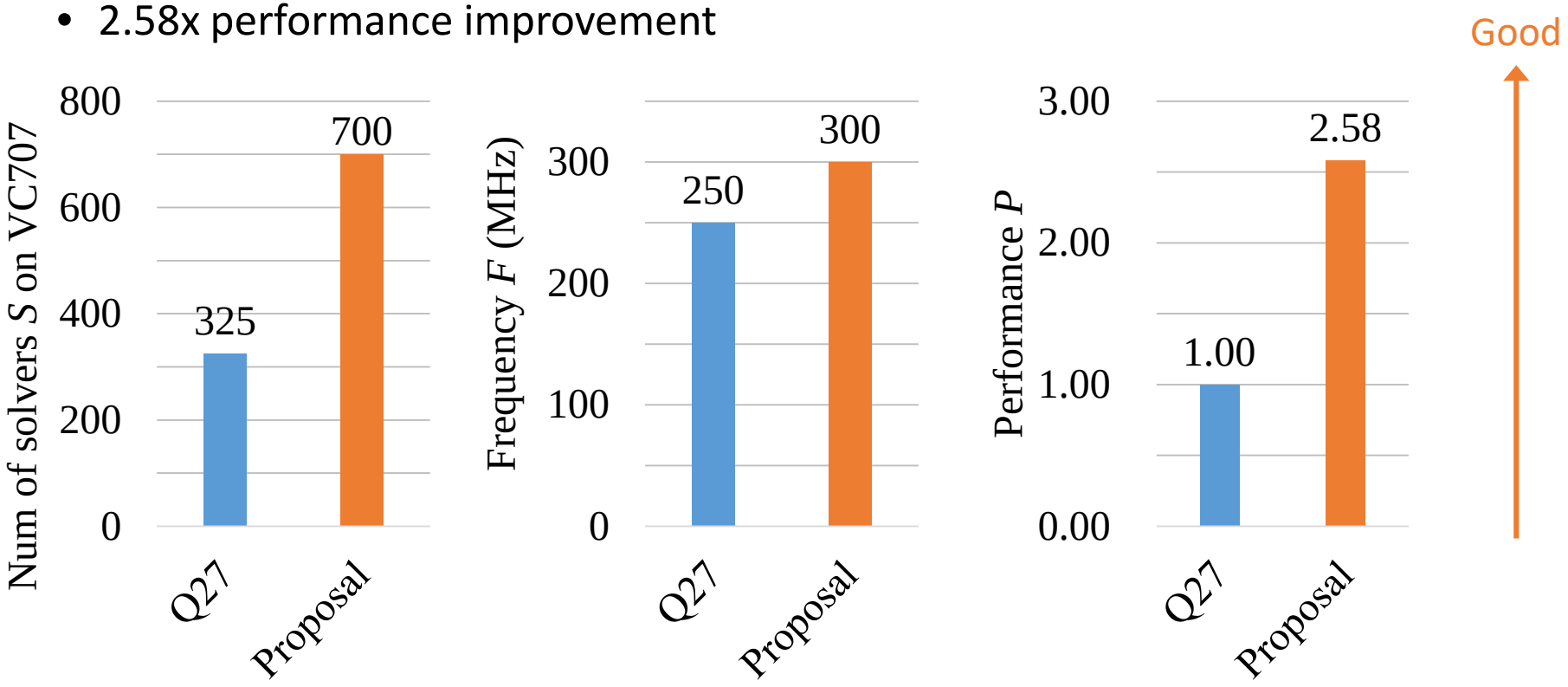
Evaluation Results (2/3)

- These modules consume less than 0.3% of the available slices of the target FPGA

	UART RX+	UART TX+	Mapper	Pulse gen	The whole system
Registers	184	126	164	16	287,331
LUTs	100	66	14	4	270,277
Slices	77	45	43	7	75,099

Evaluation Results (3/3)

- We can implement 700 solvers running at 300MHz
- *P* is normalized by the previous work $(P \propto S \cdot F)$
- 2.58x performance improvement



Outline

1. Introduction
2. Proposal
3. Evaluation
4. Conclusion

Conclusion

- We propose two methods to enable further large-scale parallelization
  - A method to reduce the hardware usage of a solver module using an encoder and a decoder for crucial data structures
  - An efficient method for distributing and collecting the resulting counts
- The performance of the proposed system implementing achieves 2.58x of the previous work
- Future work
  - Further improvement of the performance to solve the 28-Queens problem in a realistic time
  - Implementation of the necessary software

Q&A

- Thank you for listening

Evaluation Items

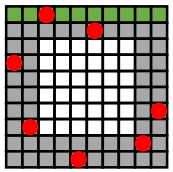
- Registers, LUTs, Slices:
  - Each usage amount of HW resources
- Number of solvers S:
  - The number of solvers implemented on a target FPGA
- Frequency F:
  - The maximum operating frequency
- Performance P:
  - P is normalized by the previous work
  - $P = \frac{F}{F_{previous}} \times \frac{S}{S_{previous}}$

Reduction of the search space

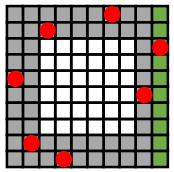
in the previous work [1]

- The number of solutions obtained from the pre-placed board (a) is multiplied by 8
- The sum of the resulting counts for eight subproblems for the pre-placed board (a) to (h)

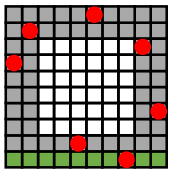
equals



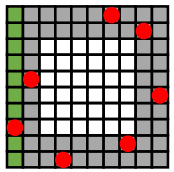
(a) Original board



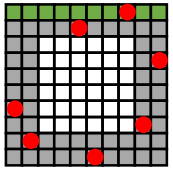
(b) Clockwise 90 degree rotation



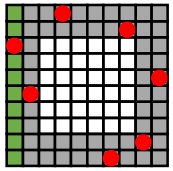
(c) Clockwise 180 degree rotation



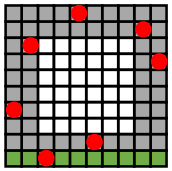
(d) Clockwise 270 degree rotation



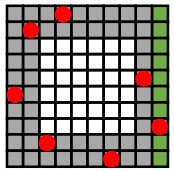
(e) Reflection



(f) Clockwise 90 degree rotation and reflection



(g) Clockwise180 degree rotation and reflection



(h) Clockwise 270 degree rotation and reflection