2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Formal Verification by The Book: Error Detection and Correction Codes

K. Devarajegowda, V. Hiltl, T. Rabenalt, D. Stoffel, W. Kunz, W. Ecker

Infineon

Technische Universität Kaiserslautern

Technische Universität München

accellera SYSTEMS INITIATIVE

Outline

Error Correction Codes

Formal Verification of ECCs

- 1st try - Brute-force
- 2nd try - Divide & conquer
- Final try - Linearity

Results and Conclusion

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Outline

Error Correction Codes

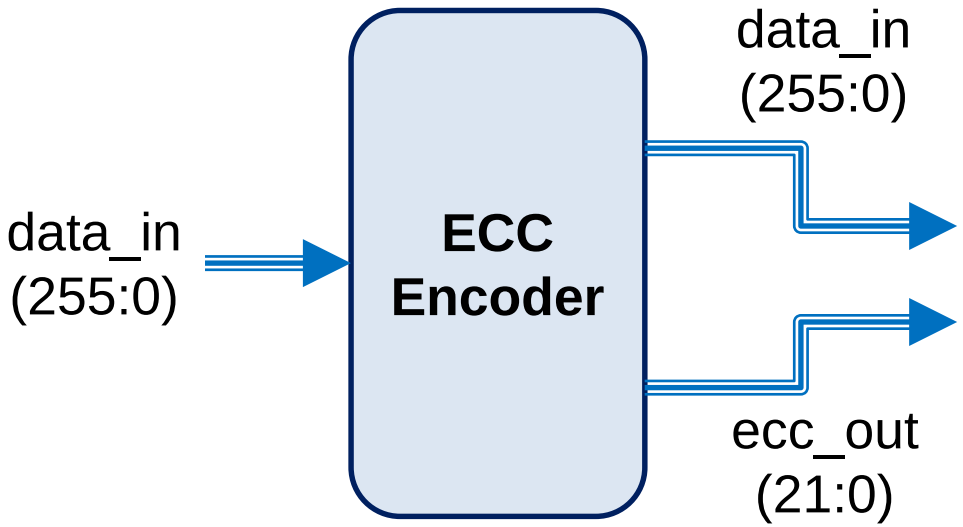
Formal Verification of ECCs

- 1st try - Brute-force
- 2nd try - Divide & conquer
- Final try - Linearity

Results and Conclusion

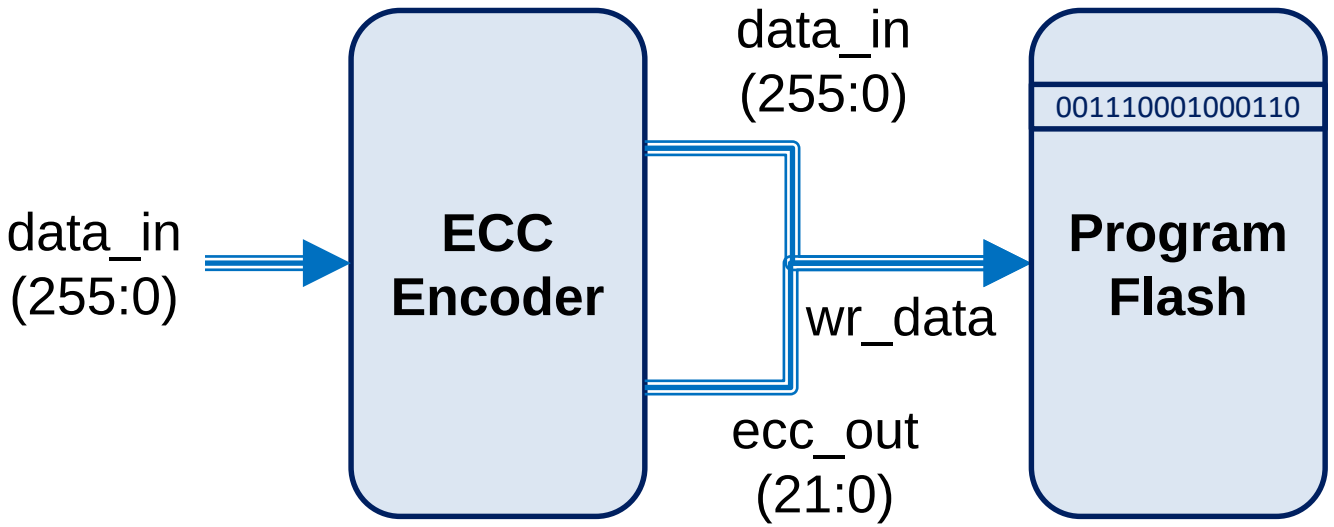
accellera SYSTEMS INITIATIVE

Error Correction Codes



- Error Detection and Correction Codes - ECCs
- 2 stages
  - encoding
  - decoding

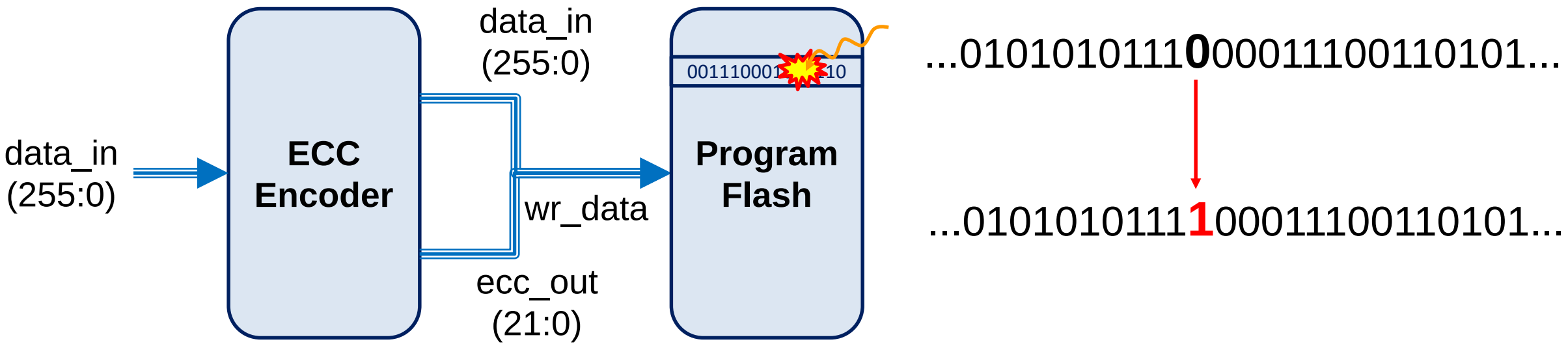
Error Correction Codes



ECC encoding

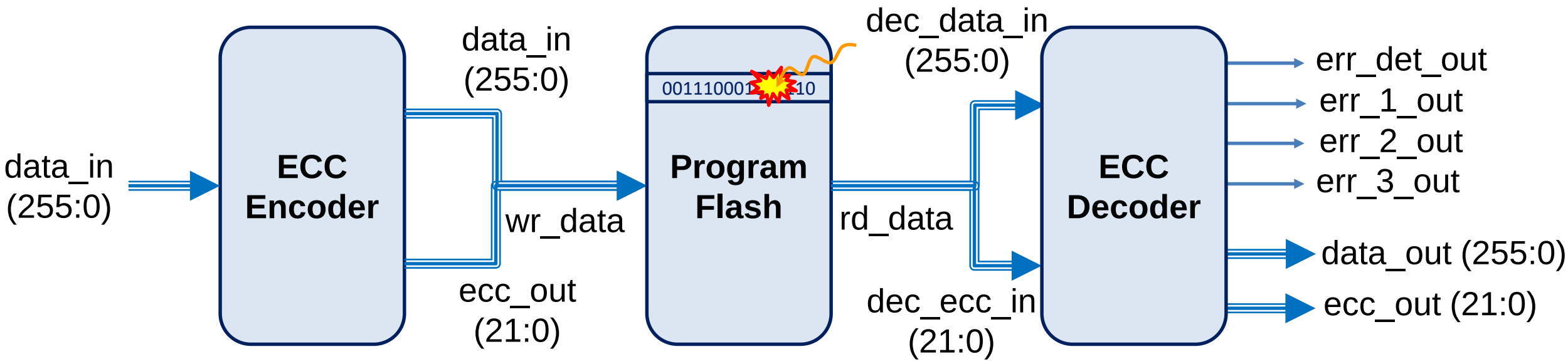
- data bits are encoded with additional ECC bits
- resulting codeword is written to the memory

Error Correction Codes



- Soft errors cause data corruption
  - o **0→1** or **1→0**
  - o codeword becomes a non-codeword

Error Correction Codes



ECC decoding

- error flags are set
- data is corrected *when #bit-errors <= correctable-range*

Outline

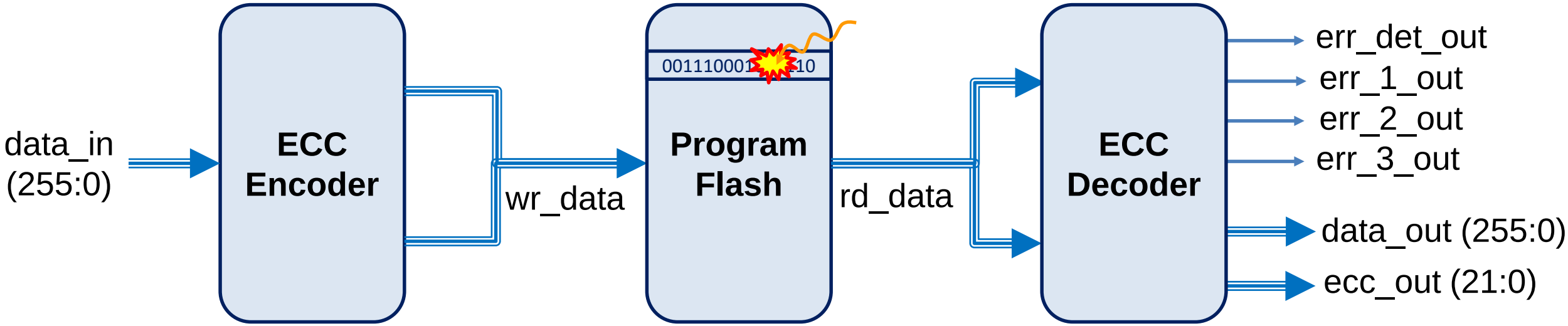
Error Correction Codes

Formal Verification of ECCs

- 1st try - Brute-force
- 2nd try - Divide & conquer
- Final try - Linearity

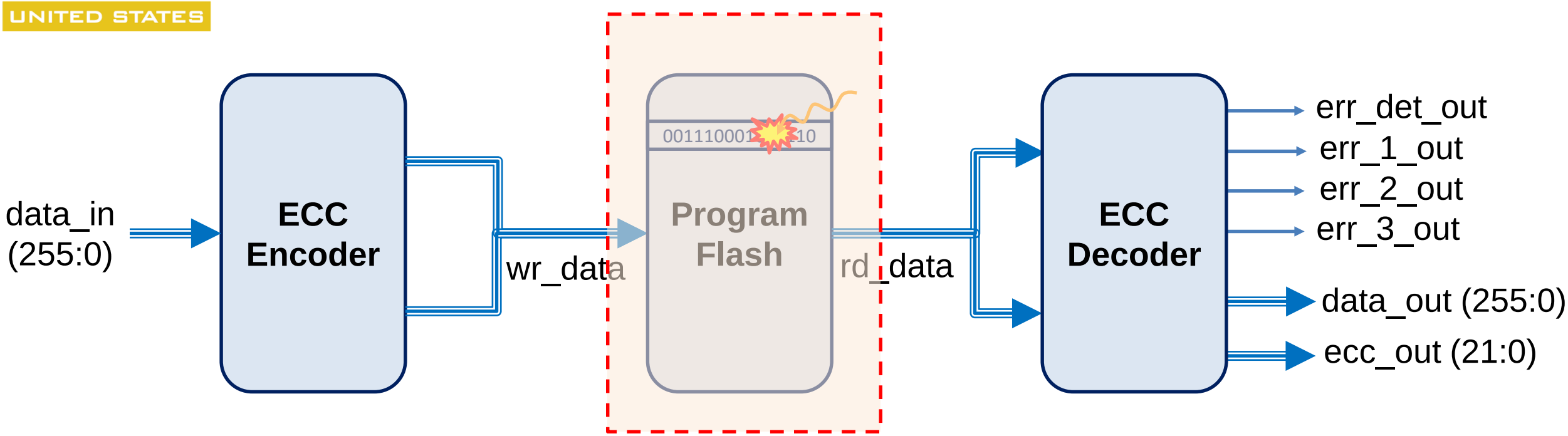
Results and Conclusion

Formal Verification of ECCs: Setup



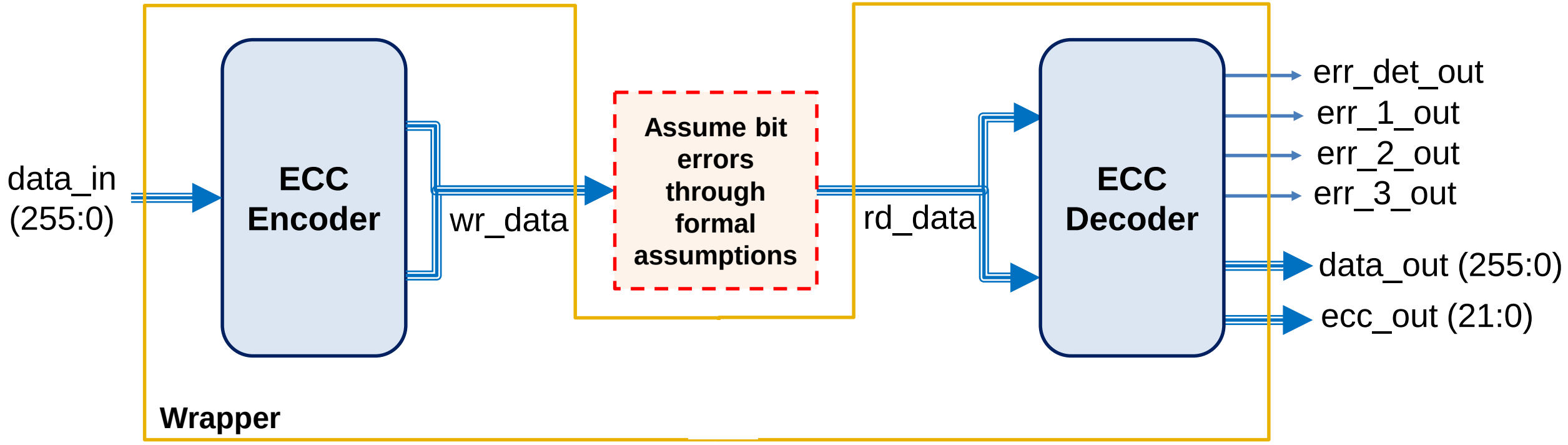
- Program flash interface is irrelevant for proving the correctness of ECCs

Formal Verification of ECCs: Setup



- Program flash interface is irrelevant for proving the correctness of ECCs

Formal Verification of ECCs: Setup



- An RTL wrapper is written instantiating only the encoder and decoder

Outline

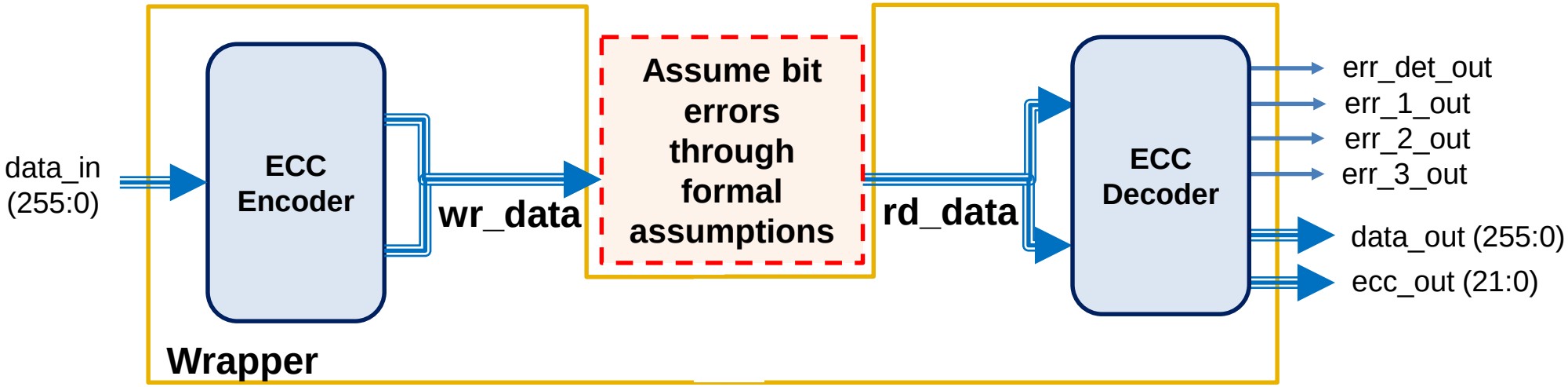
Error Correction Codes

Formal Verification of ECCs

- **1st try - Brute-force**
- 2nd try - Divide & conquer
- Final try - Linearity

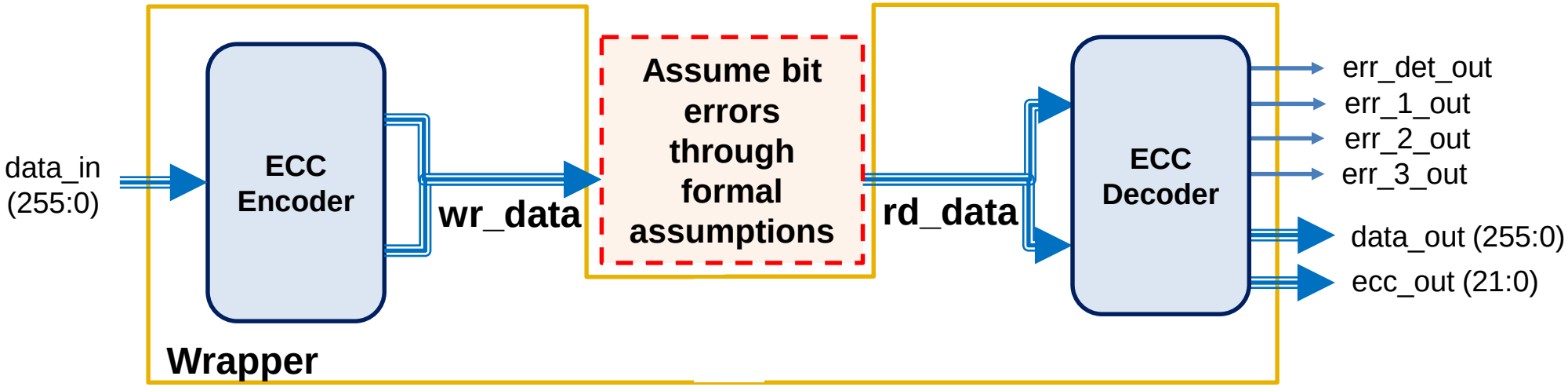
Results and Conclusion

Formal Verification of ECCs: Brute-force




```
property triple_bit_error_detect;
      $countones(wr_data ^ rd_data)==3
      |->
      err_3_out && err_det_out &&
      !err_1_out && !err_2_out;
endproperty
```

Formal Verification of ECCs: Brute-force

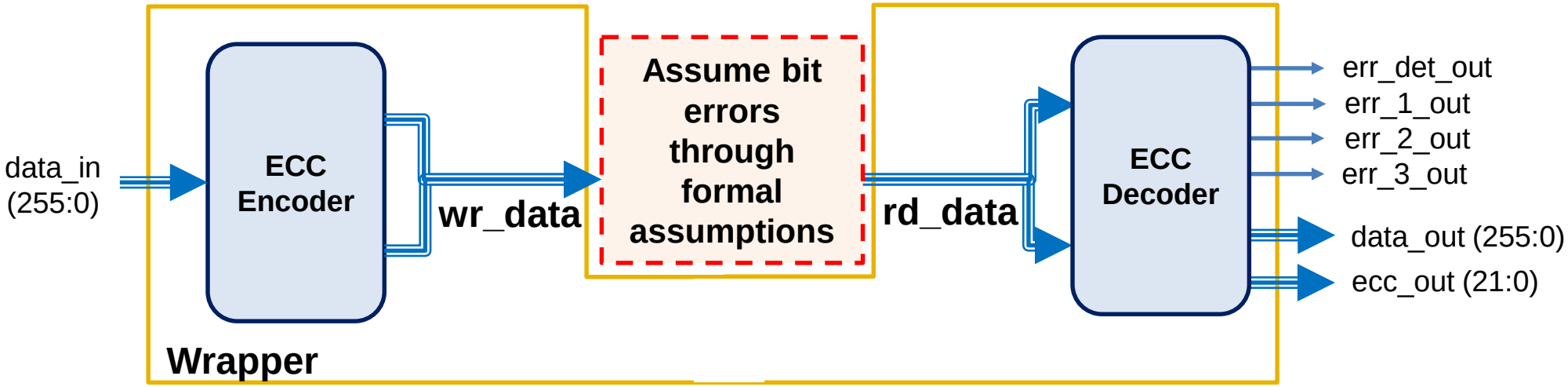



```
property triple_bit_error_detect;
      $countones(wr_data ^ rd_data)==3
      |->
      err_3_out && err_det_out &&
      !err_1_out && !err_2_out;
endproperty
```

analysis space:

$$2^{256} \times \binom{278}{3}$$

Formal Verification of ECCs: Brute-force



- Computation resources
  - o memory
  - o time
- ***Given up after 100 hours***

analysis space:

$$2^{256} \times \binom{278}{3}$$

Outline

Error Correction Codes

Formal Verification of ECCs

- 1st try - Brute-force
- **2nd try - Divide & conquer**
- Final try - Linearity

Results and Conclusion

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

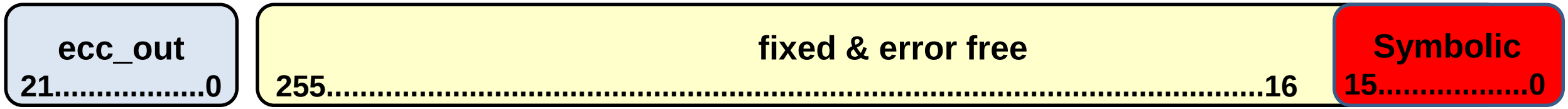
Formal Verification of ECCs: Brute-force

analysis space:

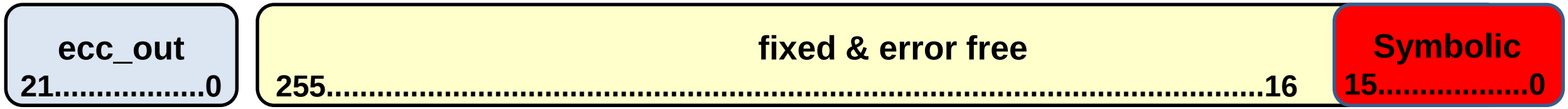
$$2^{256} \times \binom{278}{3}$$

accellera SYSTEMS INITIATIVE

Formal Verification of ECCs: Divide & conquer

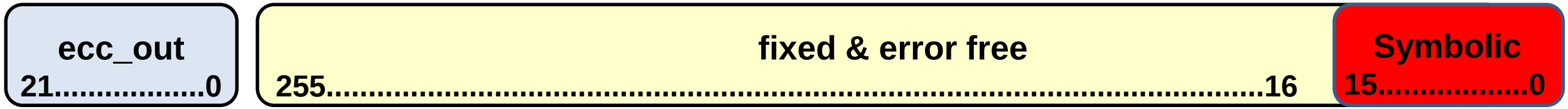


Formal Verification of ECCs: Divide & conquer




```
property triple_bit_error_detect;
      (wr_data[255:16] == 239'h43865af3c5dfa)  &&
      (wr_data[277:16] == rd_data[277:16])     &&
      $countones(wr_data[15:0] ^ rd_data[15:0])==3
      |->
      err_3_out && err_det_out &&
      !err_1_out && !err_2_out;
endproperty
```

Formal Verification of ECCs: Divide & conquer

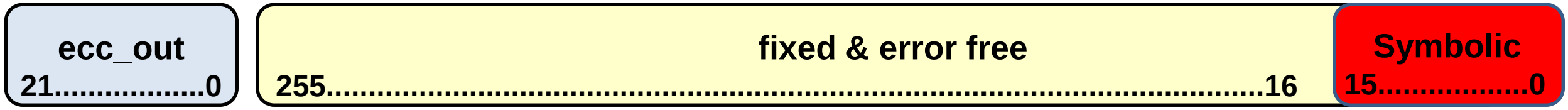



```
property triple_bit_error_detect;
      (wr_data[255:16] == 239'h43865af3c5dfa)  &&
      (wr_data[277:16] == rd_data[277:16])     &&
      $countones(wr_data[15:0] ^ rd_data[15:0])==3
      |->
      err_3_out && err_det_out &&
      !err_1_out && !err_2_out;
endproperty
```

analysis space:

$$2^{16} \times \binom{16}{3}$$

Formal Verification of ECCs: Divide & conquer



- Manageable complexity – full proof
- Large number of properties
- Not exhaustive
- Coverage?
- Verification gap?

analysis space:

$$2^{16} \times \binom{16}{3}$$

Outline

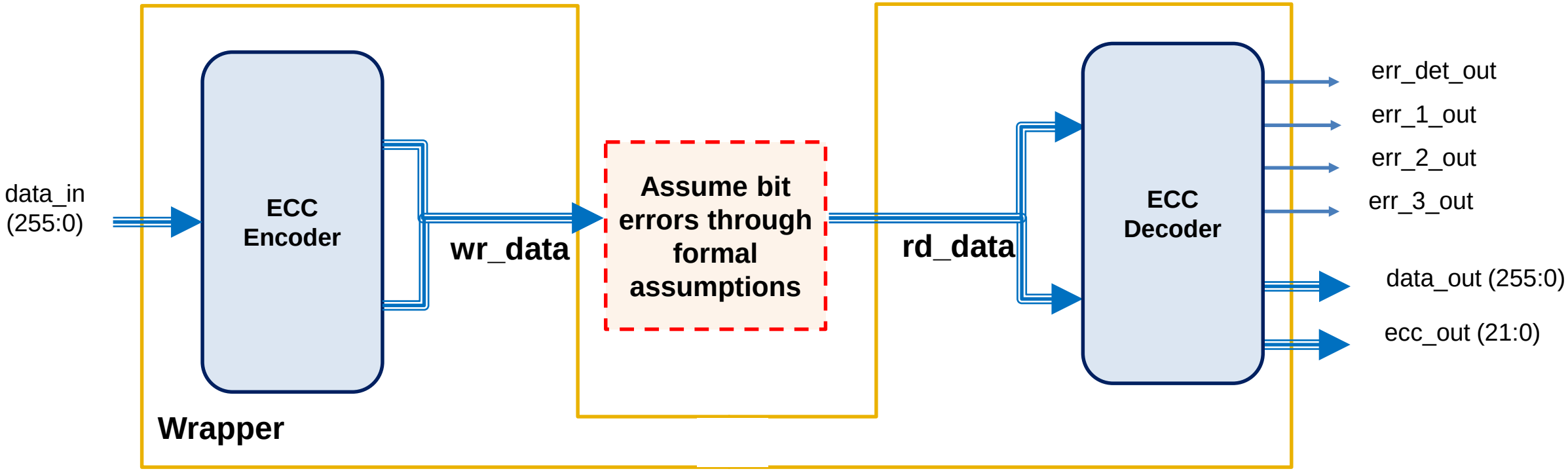
Error Correction Codes

Formal Verification of ECCs

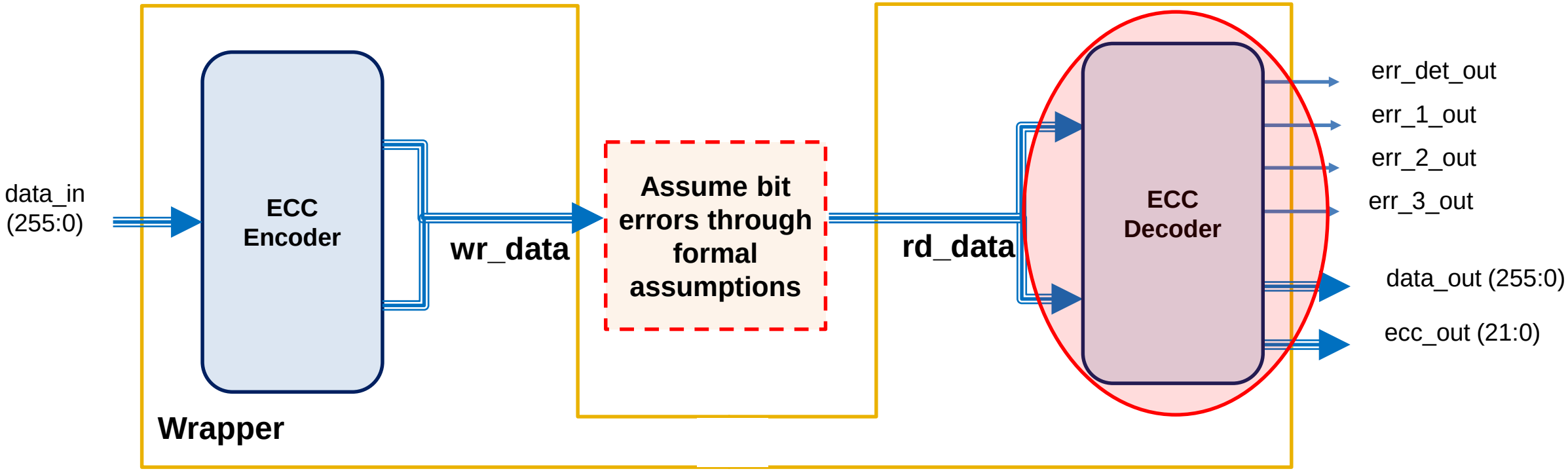
- 1st try - Brute-force
- 2nd try - Divide & conquer
- **Final try - Linearity**

Results and Conclusion

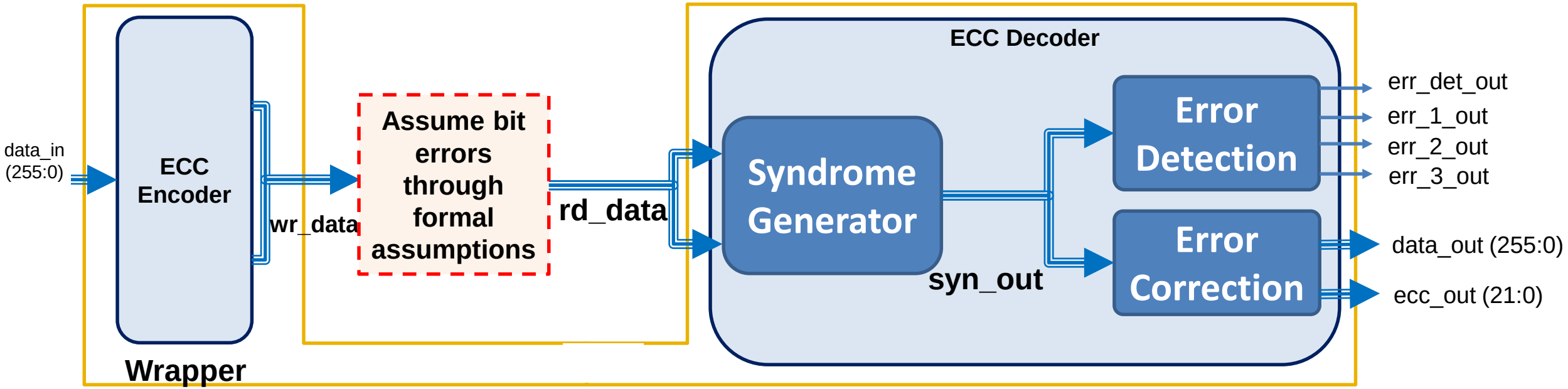
Formal Verification of ECCs: Linearity



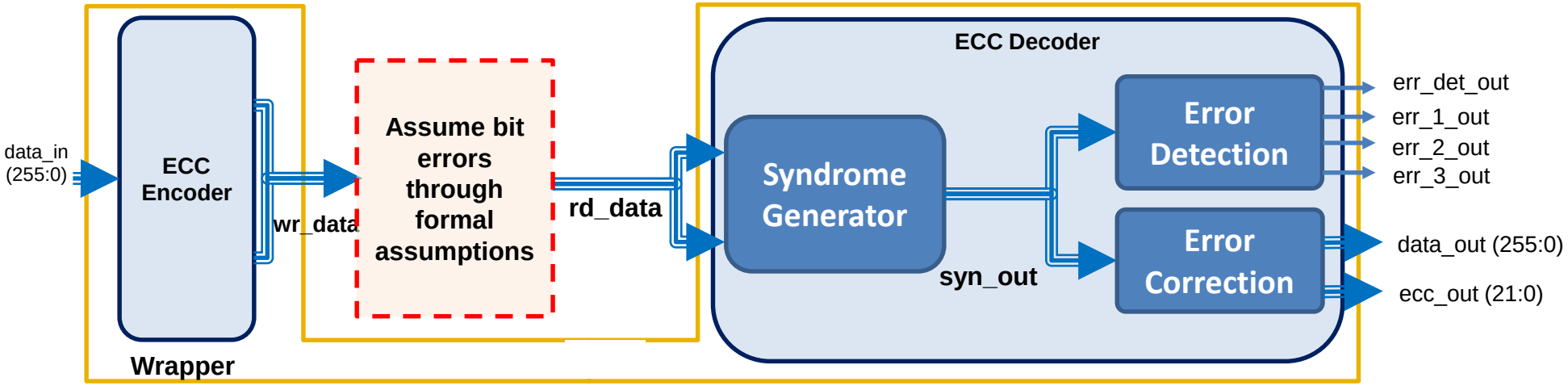
Formal Verification of ECCs: Linearity



Formal Verification of ECCs: Linearity

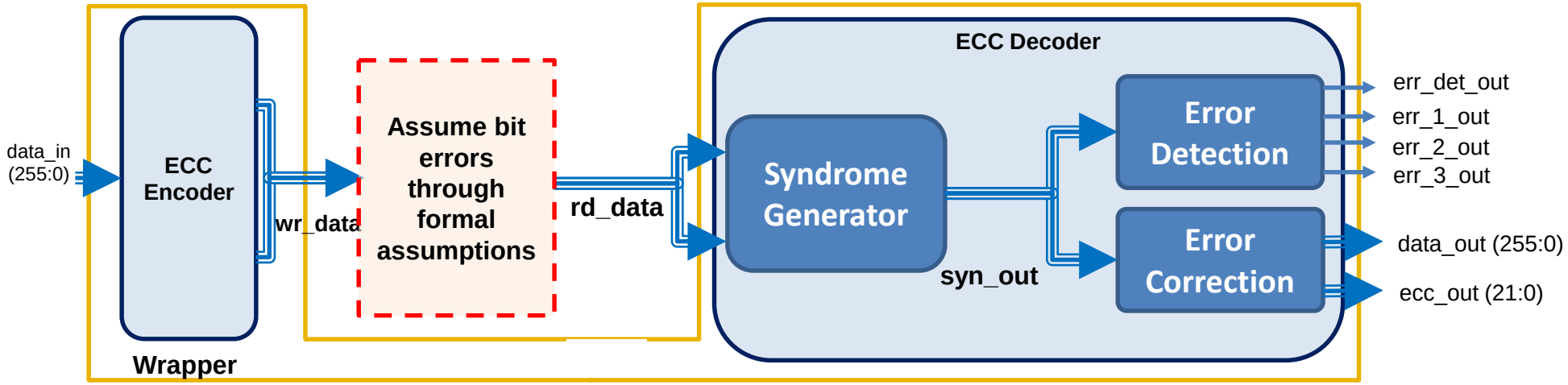


Formal Verification of ECCs: Linearity



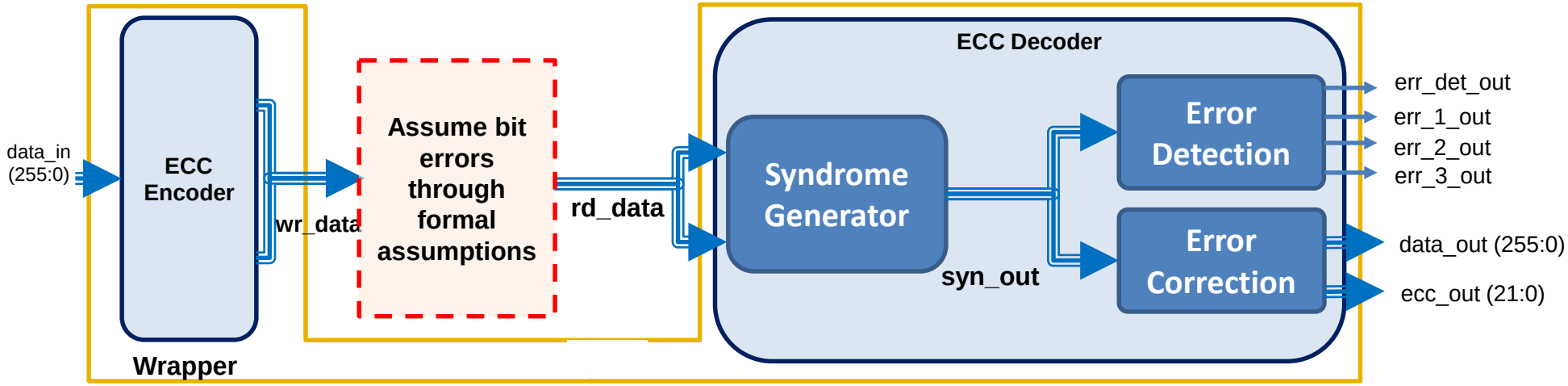
- Key findings:
  1. *(wr_data = rd_data) => (syn_out = 0)*

Formal Verification of ECCs: Linearity



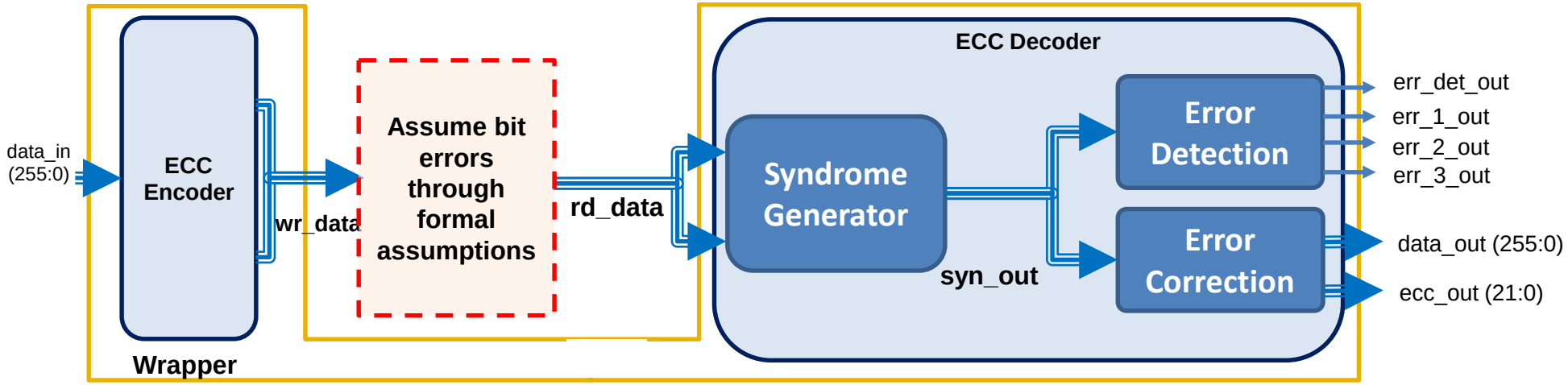
data_in	insert bit-error at	*syn_out*
256`h2874547623442	5th bit position	22`d0005

Formal Verification of ECCs: Linearity



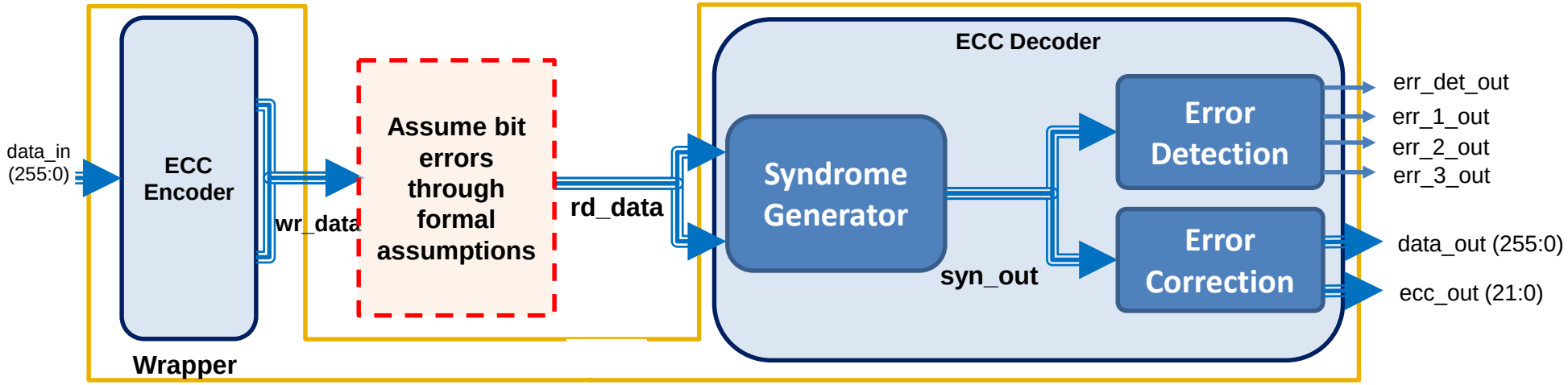
data_in	insert bit-error at	*syn_out*
256`h2874547623442	5th bit position	22`d0005
256`h644abcf5472034	5th bit position	22`d0005

Formal Verification of ECCs: Linearity



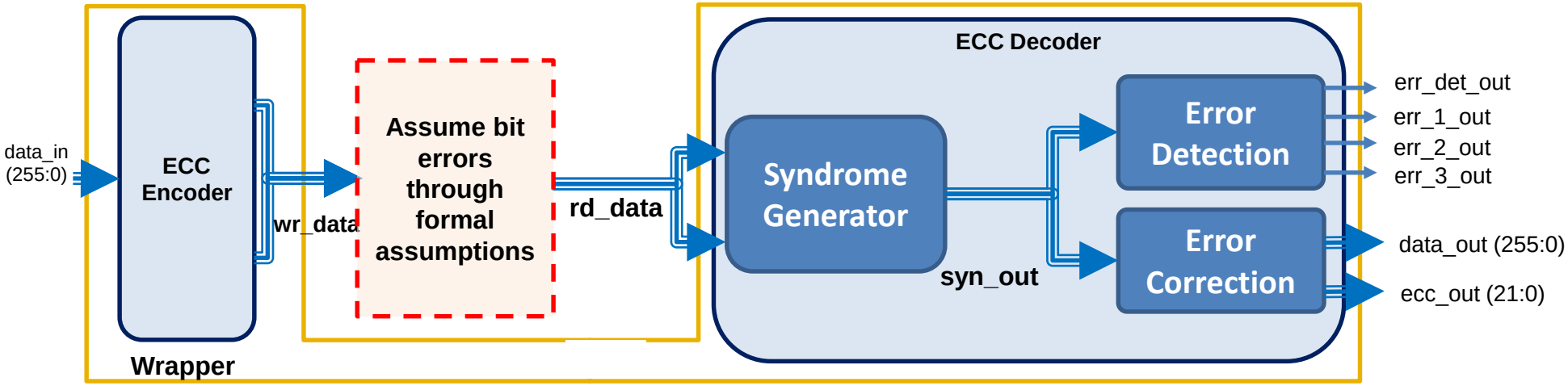
data_in	insert bit-error at	*syn_out*
256`h2874547623442	5th bit position	22`d0005
256`h644abcf5472034	5th bit position	22`d0005
256`h2874547623442	16th bit position	22`d0016

Formal Verification of ECCs: Linearity



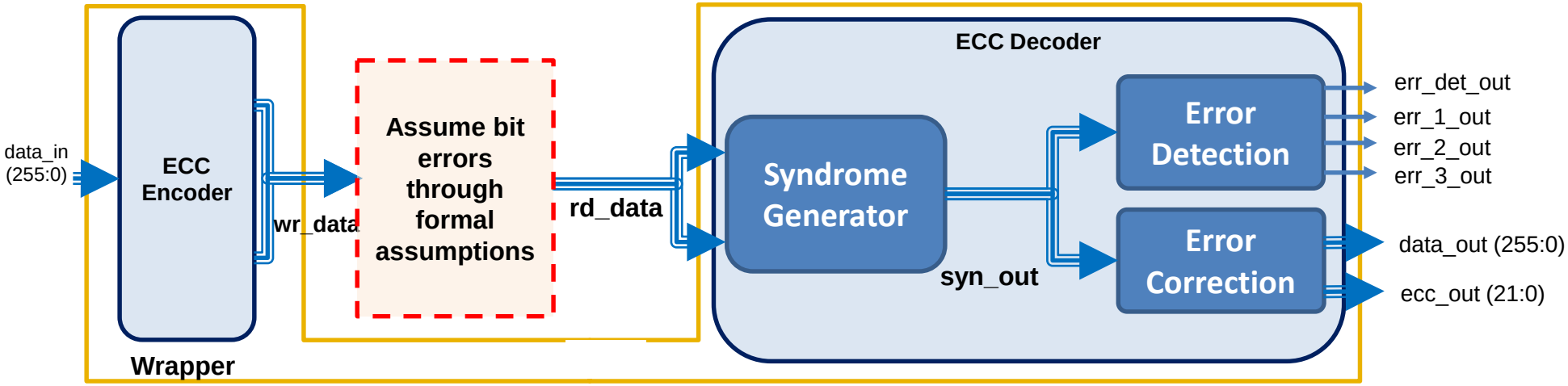
data_in	insert bit-error at	*syn_out*
256`h2874547623442	5th bit position	22`d0005
256`h644abcf5472034	5th bit position	22`d0005
256`h2874547623442	16th bit position	22`d0016
256`h644abcf5472034	16th bit position	22`d0016

Formal Verification of ECCs: Linearity



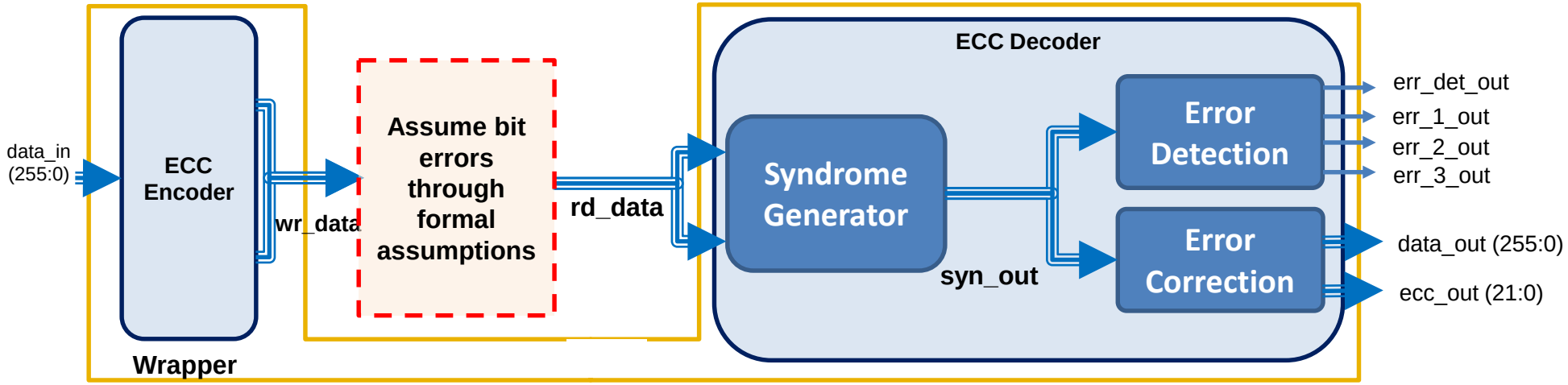
- Key findings:
  1. *(wr_data = rd_data) => (syn_out = 0)*
  2. *syn_out* is independent of *data_in* *and*
  3. *syn_out depends **only** on the erroneous bit positions*

Formal Verification of ECCs: Linearity



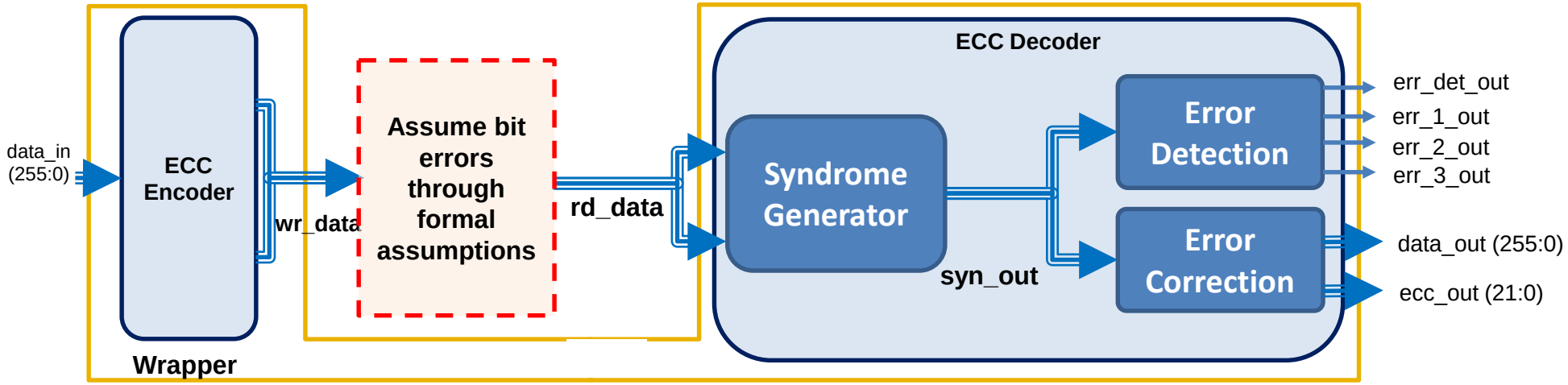
- *syn_out* is independent of *data_in*
  - *because, **Syndrome Generator** is a linear function*

Formal Verification of ECCs: Linearity



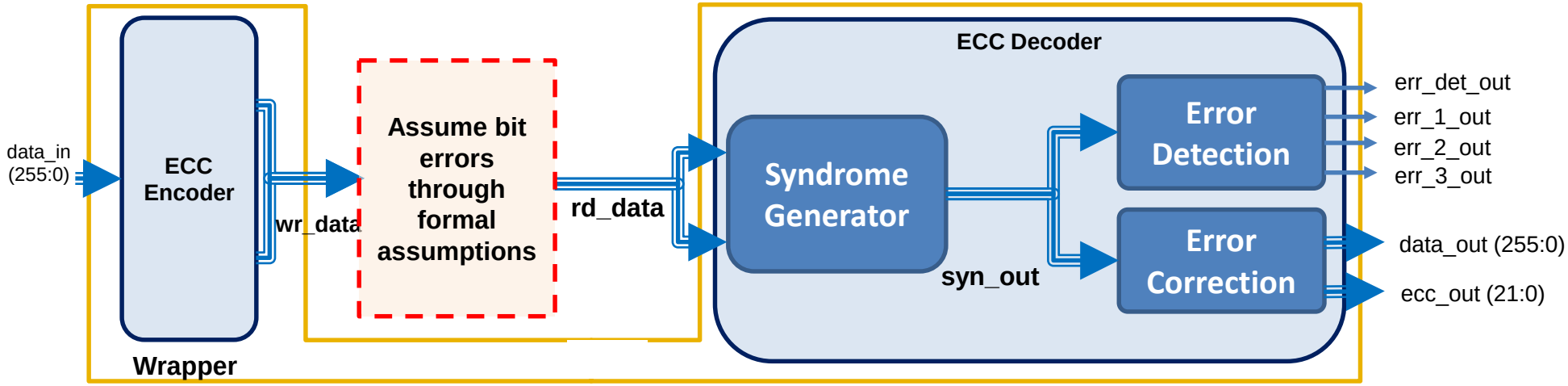
- *syn_out* is independent of *data_in*
  - *because,* ***Syndrome Generator*** *is a linear function*
- *Recall:* algebraic linear function
  - A function ***f*** is linear if, ***f(x) + f(y) = f(x+y)***
  - i.e., ***f*** preserves the addition operation

Formal Verification of ECCs: Linearity



- *syn_out* is independent of *data_in*
  - *because,* ***Syndrome Generator*** *is a linear function*
- ***syn(x ^ y) = syn(x) ^ syn(y)***
  - ***syn(x)*** *– rtl function computing the syndrome*
  - ***x*** *and* ***y*** *are arbitrary vectors*
  - `^` (***xor***) *is bitwise addition*

Formal Verification of ECCs: Linearity



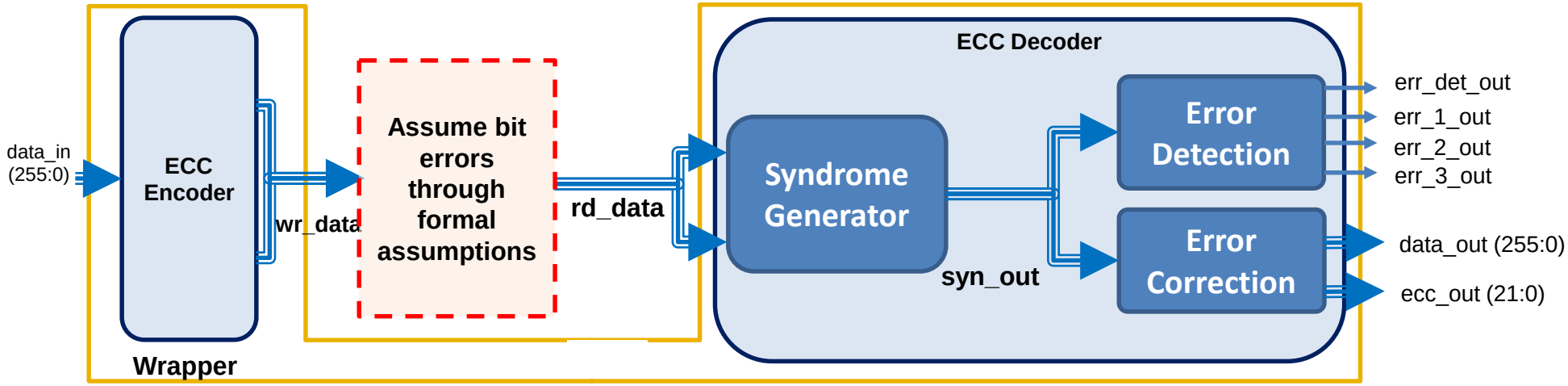

```
property syndrome_error_free_vector;
  wr_data == rd_data|-> syn_out == 0;
endproperty

property syndrome_linearity;
  syn(x) ^ syn(y) == syn(x ^ y);
endproperty
```

analysis space:

$$2^{278}$$

Formal Verification of ECCs: Linearity

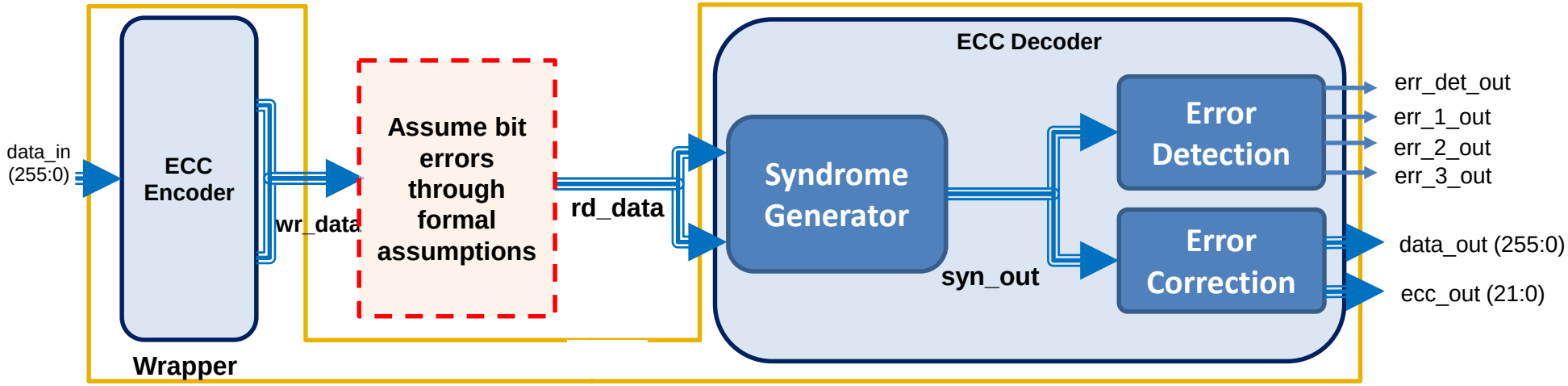



```
property triple_bit_error_detect;
  data_in = 256'b76324abfdc23a8db &&
  $countones(wr_data ^ rd_data)==3
  |->
  err_3_out && err_det_out &&
  !err_1_out && !err_2_out;
endproperty
```

analysis space:

$$1 \times \binom{278}{3}$$

Formal Verification of ECCs: Linearity



- *The property converges in 01:45:54 (hh:mm:ss)*

analysis space:

$$1 \times \binom{278}{3}$$

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Outline

- Error Correction Codes
- Formal Verification of ECCs
  - 1st try - Brute-force
  - 2nd try - Divide & conquer
  - Final try - Linearity
- **Results and Conclusion**

accellera SYSTEMS INITIATIVE

Experiments on 4 ECC Designs

Design	Data bits	ECC bits	Detection	Correction
SoC-1	256	22	1,2,3 bit errors	1,2 bit errors
SoC-1	64	22	1,2,3,4 bit errors	1,2,3 bit errors
SoC-2	26	6	1,2 bit errors	1 bit errors
SoC-3	16	6	1,2 bit errors	1 bit errors

Results

Design	Data bits	Detection	Correction	Brute-force
SoC-1	256	1,2,3	1,2	given up (100hours)
SoC-1	64	1,2,3,4	1,2,3	given up (100hours)
SoC-2	26	1,2	1	1minute
SoC-3	16	1,2	1	4minutes

- ***Formal tool: OneSpin 360 DV***

Results

Design	Data bits	Detection	Correction	Brute-force	Divide & Conquer
SoC-1	256	1,2,3	1,2	given up (100hrs)	40 hrs
SoC-1	64	1,2,3,4	1,2,3	given up (100hrs)	25 hrs
SoC-2	26	1,2	1	1min	---
SoC-3	16	1,2	1	4min	---

- ***Formal tool: OneSpin 360 DV***

Results

Design	Data bits	Detection	Correction	Brute-force	Divide & Conquer	Linearity
SoC-1	256	1,2,3	1,2	given up (100hrs)	40 hrs	9hrs
SoC-1	64	1,2,3,4	1,2,3	given up (100hrs)	25 hrs	7hrs
SoC-2	26	1,2	1	1min	---	10sec
SoC-3	16	1,2	1	4min	---	1min

- ***Formal tool: OneSpin 360 DV***

Formal verification of large ECCs

- ***Prove the linearity of syndrome generator***
- ***Prove the remaining properties with arbitrarily chosen, fixed data vector***

Summary

- Linearity approach
  - Significant reduction of property runtime
  - Previously **given-up** properties converge within **2hrs**
  - Properties become simpler and are generated
  - 10x – 15x productivity gain
- Formal Verification - real benefit comes with the right approach

2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Thank You for listening!
Questions!

accellera SYSTEMS INITIATIVE