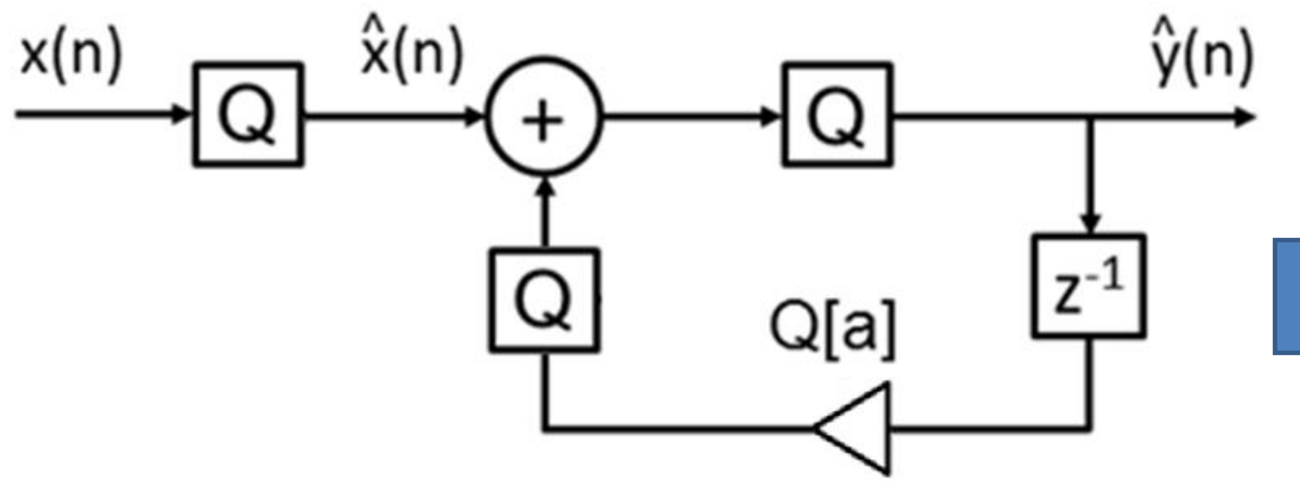
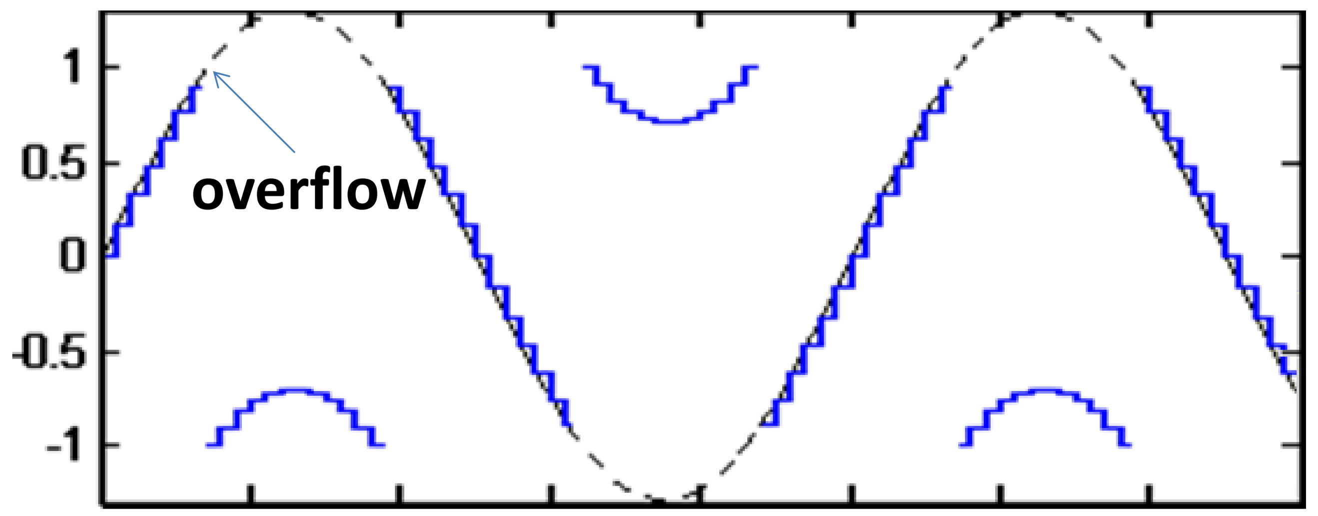
Verifying Fixed-Point Digital Filters using SMT-Based Bounded Model Checking

Renato B. Abreu, Lucas Cordeiro and Eddie B. L. Filho
renato.abreu@indt.org.br, lucascordeiro@ufam.edu.br, eddie@ctpim.org.br

1. Introduction

- Fixed-point implementation leads to quantization nonlinearities, round off errors, and overflows due to operations with finite word-length
- Testing and simulation can lead to a limited number of scenarios, which do not exploit all possible behaviors of the system

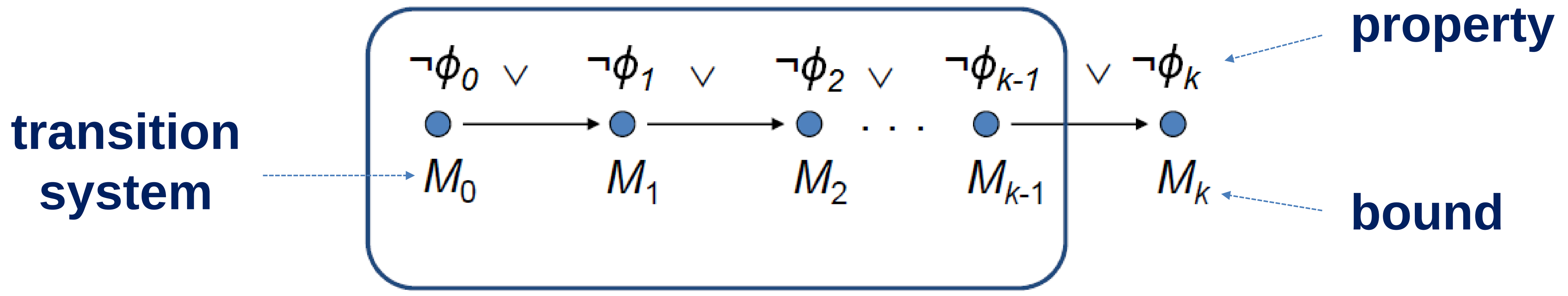




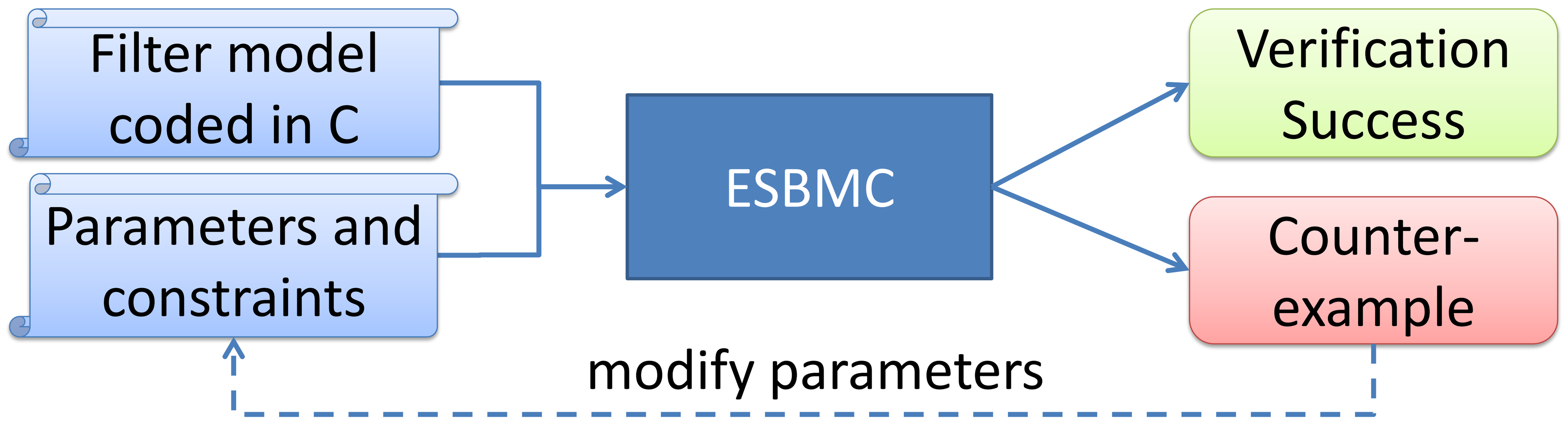
$$y(n) = -\sum_{k=1}^{N} a_k\, y(n-k) + \sum_{k=0}^{M} b_k\, x(n-k)$$

2. SMT-based BMC of Digital Filter

- Check the negation of a given property at a given depth



3. Proposed Approach



4. Experimental Setup

- Environment*: Fedora 64 bits, ESBMC v1.21, SMT Z3 v3.2
- Filters tested (from Matlab design toolbox and from literature)
  - Low Pass, High Pass, Band Pass, Band Stop
  - Up to 6th order IIR and up to 30th order FIR
  - Word-length up to 16 bits

5. Conclusions

- The method can detect overflow, limit cycle and time constraint failures in filters of different types and orders
- Can find problems that are hard to detect using tests and simulations
- Verification time tends to be higher for high order filters and for longest word-length formats since these lead to harder verification conditions
- Exploits the advantages of an state of art model checker ESBMC over a model coded in C

* Tools and benchmarks available in www.esbmc.org

Verifying Fixed-Point Digital Filters using SMT-Based Bounded Model Checking

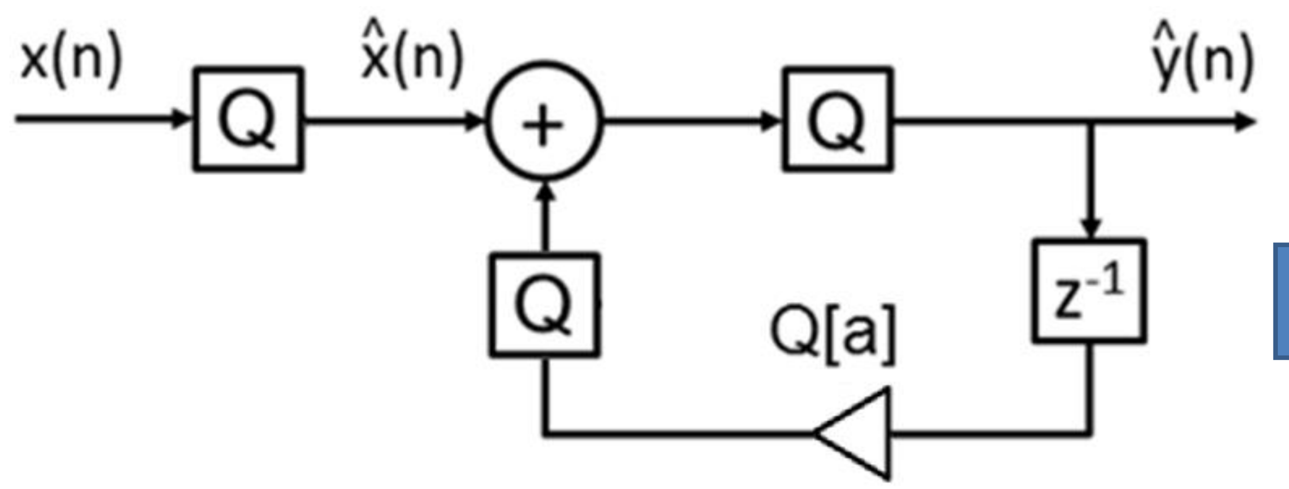
UFAM

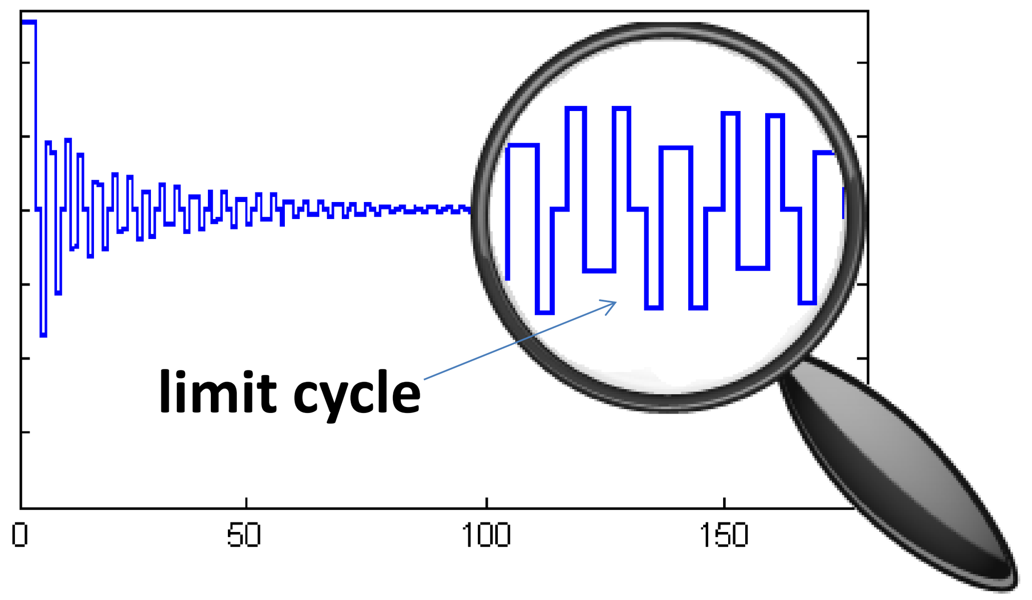
Renato B. Abreu, Lucas Cordeiro and Eddie B. L. Filho

renato.abreu@indt.org.br, lucascordeiro@ufam.edu.br, eddie@ctpim.org.br

1. Introduction

- Fixed-point implementation leads to quantization nonlinearities, round off errors, and overflows due to operations with finite word-length
- Testing and simulation can lead to a limited number of scenarios, which do not exploit all possible behaviors of the system

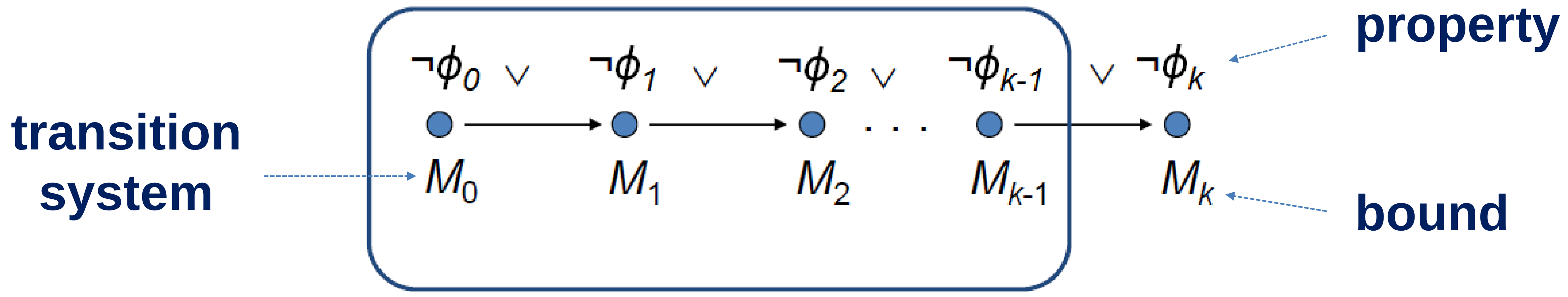




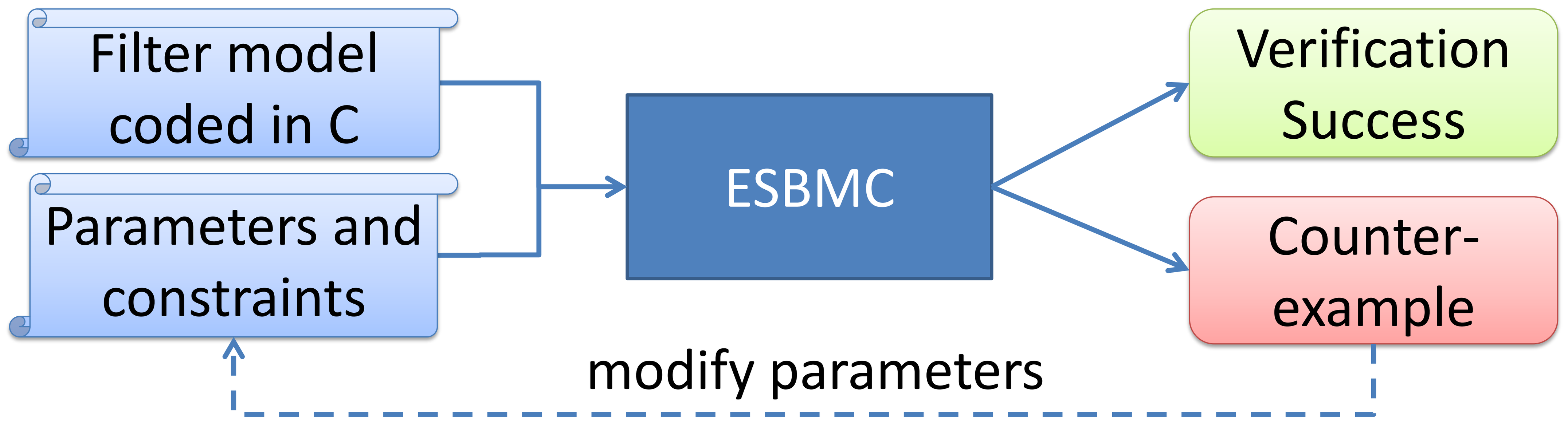
$$y(n) = -\sum_{k=1}^{N} a_k\, y(n-k) + \sum_{k=0}^{M} b_k\, x(n-k)$$

2. SMT-based BMC of Digital Filter

- Check the negation of a given property at a given depth



3. Proposed Approach



4. Experimental Setup

- Environment*: Fedora 64 bits, ESBMC v1.21, SMT Z3 v3.2
- Filters tested (from Matlab design toolbox and from literature)
  - Low Pass, High Pass, Band Pass, Band Stop
  - Up to 6th order IIR and up to 30th order FIR
  - Word-length up to 16 bits

5. Conclusions

- The method can detect overflow, limit cycle and time constraint failures in filters of different types and orders
- Can find problems that are hard to detect using tests and simulations
- Verification time tends to be higher for high order filters and for longest word-length formats since these lead to harder verification conditions
- Exploits the advantages of an state of art model checker ESBMC over a model coded in C

* Tools and benchmarks available in www.esbmc.org

Verifying Fixed-Point Digital Filters using SMT-Based Bounded Model Checking

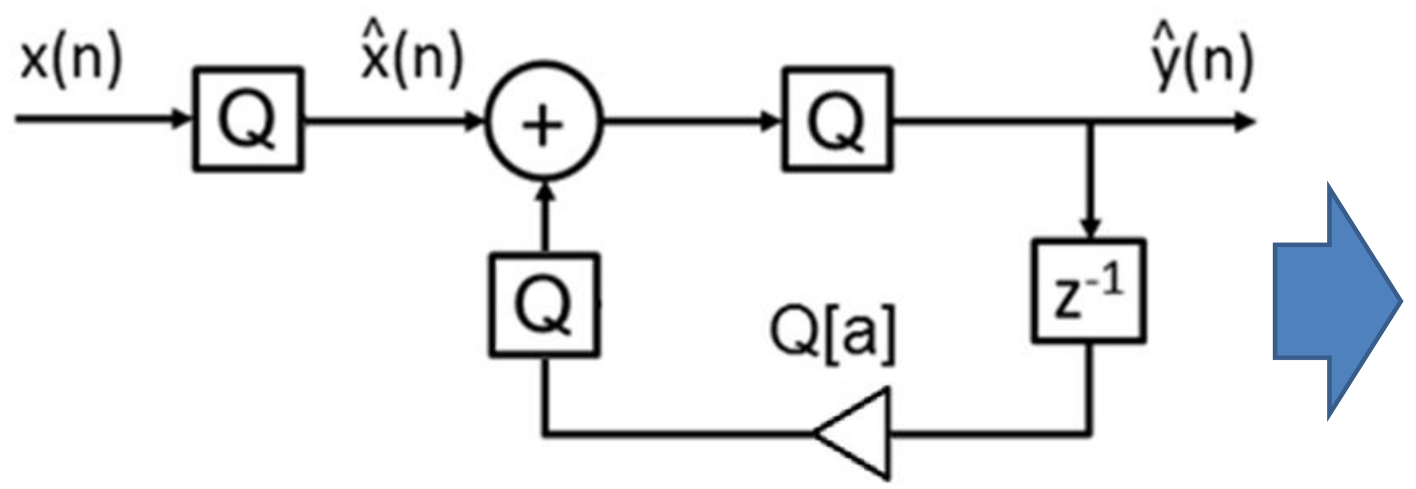
UFAM

Renato B. Abreu, Lucas Cordeiro and Eddie B. L. Filho

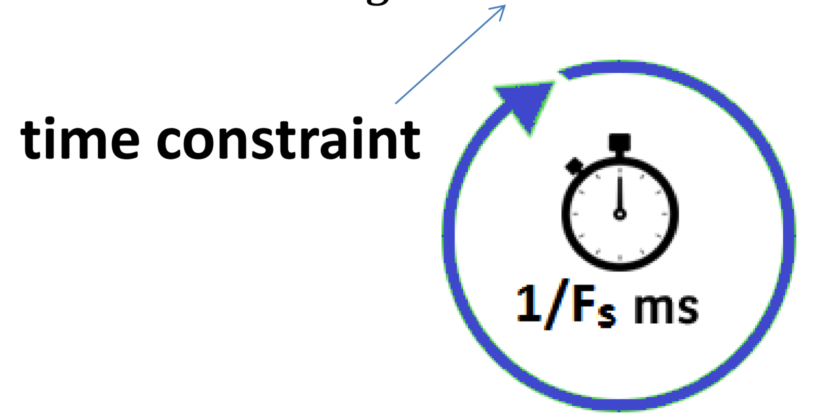
renato.abreu@indt.org.br, lucascordeiro@ufam.edu.br, eddie@ctpim.org.br

1. Introduction

- Fixed-point implementation leads to quantization nonlinearities, round off errors, and overflows due to operations with finite word-length
- Testing and simulation can lead to a limited number of scenarios, which do not exploit all possible behaviors of the system



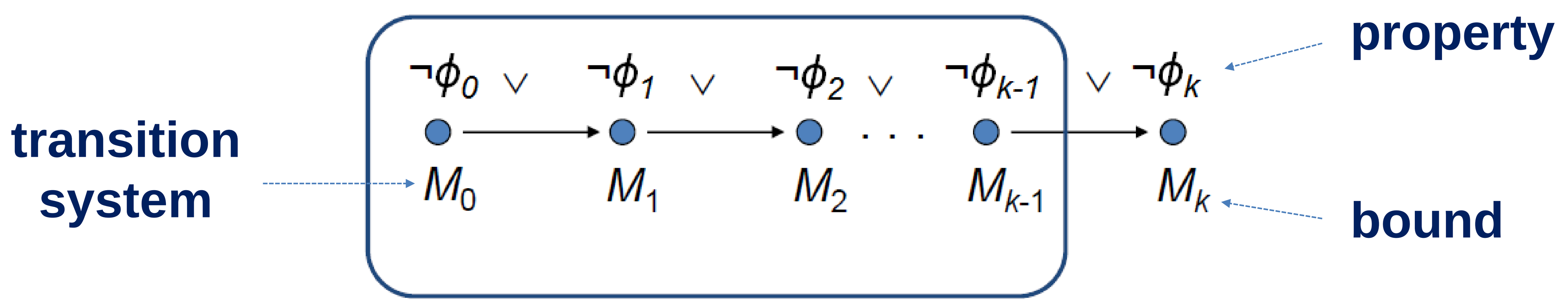
$$l_{timing} \Leftrightarrow ((N \times T) \le D)$$



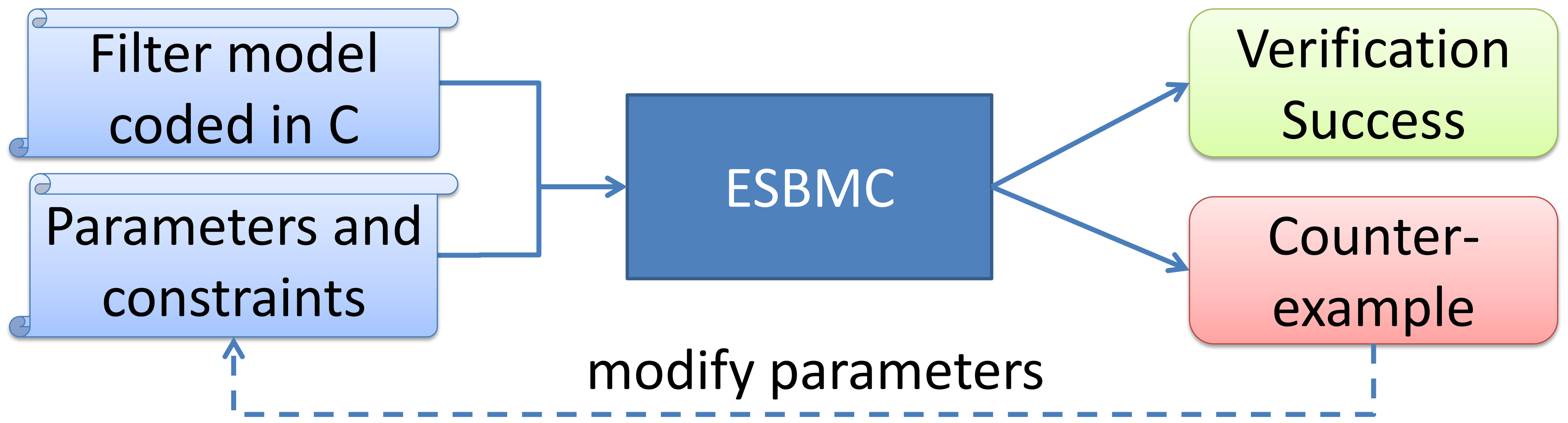
$$y(n) = -\sum_{k=1}^{N} a_k\, y(n-k) + \sum_{k=0}^{M} b_k\, x(n-k)$$

2. SMT-based BMC of Digital Filter

- Check the negation of a given property at a given depth



3. Proposed Approach



4. Experimental Setup

- Environment*: Fedora 64 bits, ESBMC v1.21, SMT Z3 v3.2
- Filters tested (from Matlab design toolbox and from literature)
  - Low Pass, High Pass, Band Pass, Band Stop
  - Up to 6th order IIR and up to 30th order FIR
  - Word-length up to 16 bits

5. Conclusions

- The method can detect overflow, limit cycle and time constraint failures in filters of different types and orders
- Can find problems that are hard to detect using tests and simulations
- Verification time tends to be higher for high order filters and for longest word-length formats since these lead to harder verification conditions
- Exploits the advantages of an state of art model checker ESBMC over a model coded in C

* Tools and benchmarks available in www.esbmc.org

Satisfiability Modulo Theories

- An SMT solver decides about the satisfiability of a first order formula using different background theories so it generalizes the propositional satisfiability

Satisfiability Modulo Theories

- An SMT solver decides about the satisfiability of a first order formula using different background theories so it generalizes the propositional satisfiability

Theory	Example
Equality	$x_1 = x_2 \land \neg(x_1 = x_3) \Rightarrow \neg(x_1 = x_3)$
Bit vector	$(b \gg i)\&1 = 1$
Linear arithmetic	$(4y_1 + 3y_2 \geq 4) \lor (y_2 - 3y_3 \leq 3)$
Arrays	$(j = k \land a[k] = 2) \Rightarrow a[j] = 2$
Combined theories	$(j \leq k \land a[j] = 2) \Rightarrow a[i] < 3$

Satisfiability Modulo Theories

- An SMT solver decides about the satisfiability of a first order formula using different background theories so it generalizes the propositional satisfiability

Theory	Example
Equality	$x_1 = x_2 \land \neg(x_1 = x_3) \Rightarrow \neg(x_1 = x_3)$
Bit vector	$(b \gg i) \& 1 = 1$
Linear arithmetic	$(4y_1 + 3y_2 \geq 4) \lor (y_2 - 3y_3 \leq 3)$
Arrays	$(j = k \land a[k] = 2) \Rightarrow a[j] = 2$
Combined theories	$(j \leq k \land a[j] = 2) \Rightarrow a[i] < 3$

$$\langle k, l \rangle \times \langle k, l \rangle = \langle 2k, 2l \rangle$$

$$\langle 2k, 2l \rangle \gg f = \langle 2k, f \rangle$$

Satisfiability Modulo Theories

- An SMT solver decides about the satisfiability of a first order formula using different background theories so it generalizes the propositional satisfiability

Theory	Example
Equality	$x_1 = x_2 \land \neg(x_1 = x_3) \Rightarrow \neg(x_1 = x_3)$
Bit vector	$(b \gg i)\&1 = 1$
Linear arithmetic	$(4y_1 + 3y_2 \geq 4) \lor (y_2 - 3y_3 \leq 3)$
Arrays	$(j = k \land a[k] = 2) \Rightarrow a[j] = 2$
Combined theories	$(j \leq k \land a[j] = 2) \Rightarrow a[i] < 3$

$$\langle k, l \rangle \times \langle k, l \rangle = \langle 2k, 2l \rangle$$

$$\langle 2k, 2l \rangle \gg f = \langle 2k, f \rangle$$

Overflow verification

BMC using the ESBMC

BMC using the ESBMC

Code
C/C++

```
int main() {
  int a[2], i, x;
  if (x==0)
    a[i]=0;
  else
    a[i+2]=1;
  assert(a[i+1]==1);
}
```

BMC using the ESBMC

Code C/C++ $\xrightarrow{\text{Control flow}}$ GOTO program

```
int main() {
  int a[2], i, x;
  if (x==0)
    a[i]=0;
  else
    a[i+2]=1;
  assert(a[i+1]==1);
}
```

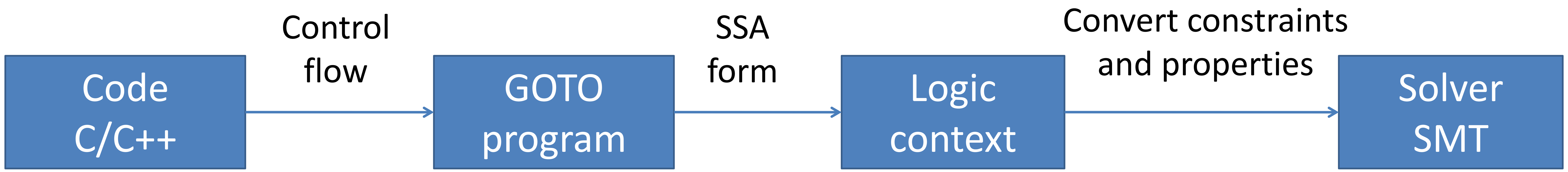
BMC using the ESBMC

Code C/C++ $\xrightarrow{\text{Control flow}}$ GOTO program $\xrightarrow{\text{SSA form}}$ Logic context

```
int main() {
  int a[2], i, x;
  if (x==0)
    a[i]=0;
  else
    a[i+2]=1;
  assert(a[i+1]==1);
}
```

$$
\begin{aligned}
g_1 &= x_1 == 0 \\
a_1 &= a_0 \text{ WITH } [i_0{:=}0] \\
a_2 &= a_0 \\
a_3 &= a_2 \text{ WITH } [2+i_0{:=}1] \\
a_4 &= g_1 \; ? \; a_1 : a_3 \\
t_1 &= a_4\,[1+i_0] == 1
\end{aligned}
$$

BMC using the ESBMC




```
int main() {
  int a[2], i, x;
  if (x==0)
    a[i]=0;
  else
    a[i+2]=1;
  assert(a[i+1]==1);
}
```

$$C := \left[\begin{array}{l} g_1 := (x_1 = 0) \\ \wedge\, a_1 := store(a_0, i_0, 0) \\ \wedge\, a_2 := a_0 \\ \wedge\, a_3 := store(a_2, 2+i_0, 1) \\ \wedge\, a_4 := ite(g_1, a_1, a_3) \end{array}\right]$$

$$P := \left[\begin{array}{l} i_0 \geq 0 \wedge i_0 < 2 \\ \wedge\, 2+i_0 \geq 0 \wedge 2+i_0 < 2 \\ \wedge\, 1+i_0 \geq 0 \wedge 1+i_0 < 2 \\ \wedge\, select(a_4, i_0+1) = 1 \end{array}\right]$$

BMC using the ESBMC


```
int main() {
  int a[2], i, x;
  if (x==0)
    a[i]=0;
  else
    a[i+2]=1;
  assert(a[i+1]==1);
}
```

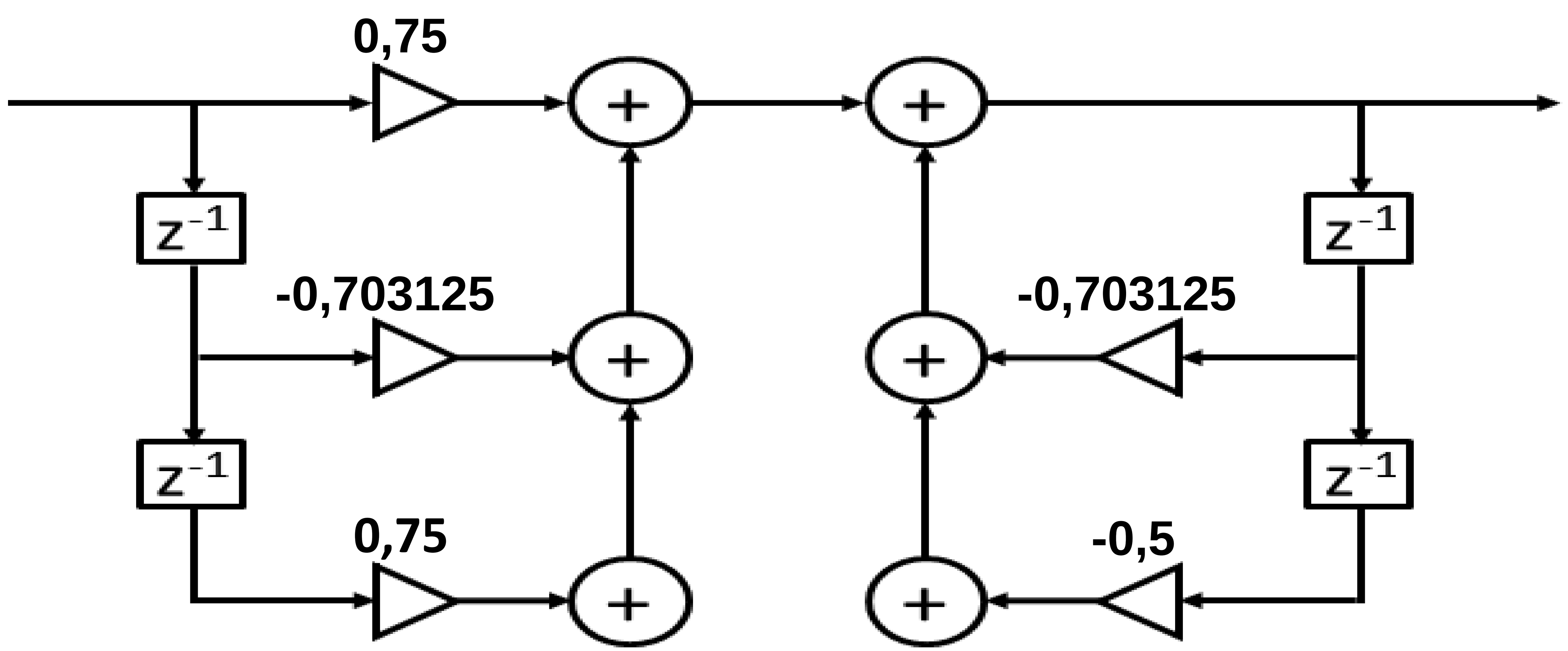
$$C := \begin{bmatrix} g_1 := (x_1 = 0) \\ \wedge\, a_1 := store(a_0, i_0, 0) \\ \wedge\, a_2 := a_0 \\ \wedge\, a_3 := store(a_2, 2 + i_0, 1) \\ \wedge\, a_4 := ite(g_1, a_1, a_3) \end{bmatrix}$$

$$P := \begin{bmatrix} i_0 \geq 0 \wedge i_0 < 2 \\ \wedge\, 2 + i_0 \geq 0 \wedge 2 + i_0 < 2 \\ \wedge\, 1 + i_0 \geq 0 \wedge 1 + i_0 < 2 \\ \wedge\, select(a_4, i_0 + 1) = 1 \end{bmatrix}$$

Example 1: Overflow Verification

Example 1: Overflow Verification

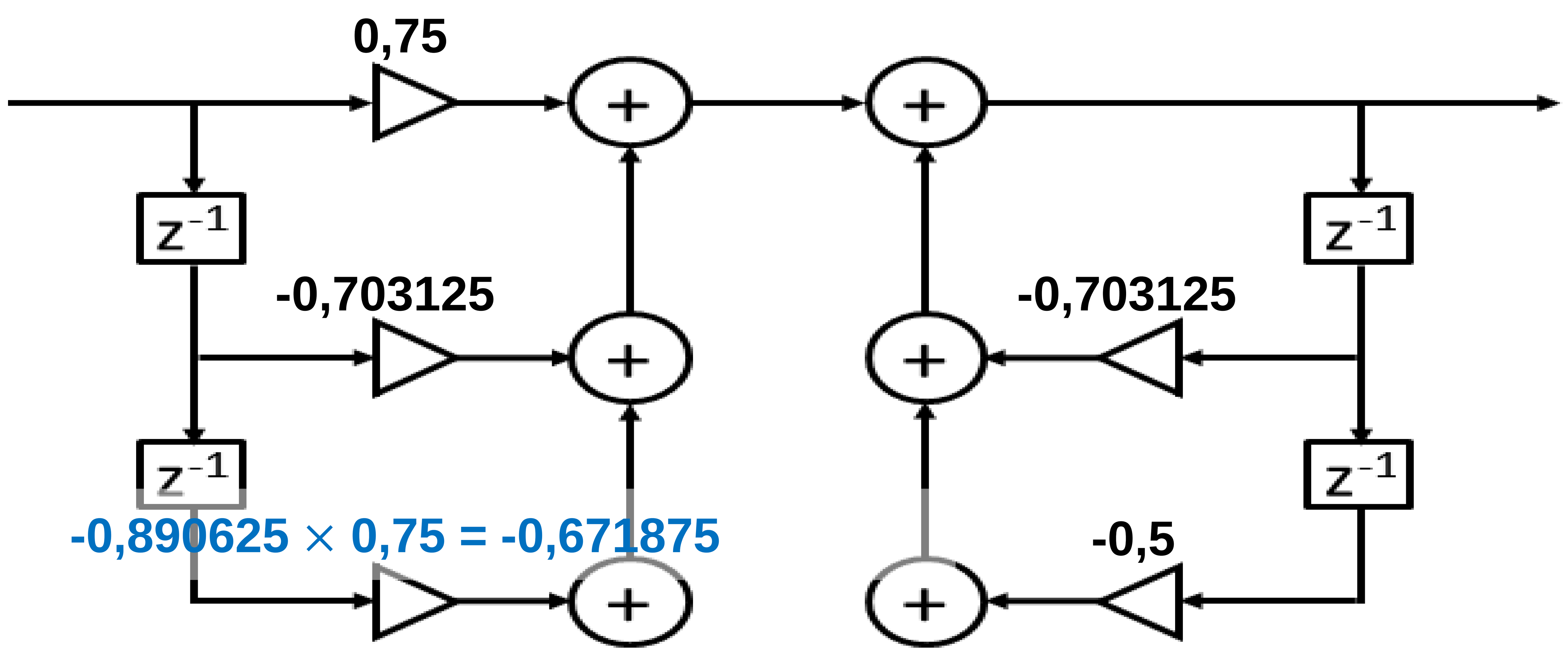
- The model checker applies non-deterministic inputs in the interval [-1,1] searching the negation of:
  - $l_{overflow} \Leftrightarrow (-2 \leq FP) \wedge (FP \leq 1{,}984375)$
- Here the verification returns the following couter-example:
  - x = { 0.0f, 0.015625f, 0.0f, -0.890625f, 0.96875f, -0.890625f }
  - xaux = { -0.890625f, 0.96875f, -0.890625f }
  - yaux = { 0f, -0.671875f, 0.890625f }



- $\sum_{k=0}^{100}|h_k|$ = 1,8178
- For x between [-1,1], $|y(n)| \leq 1{,}82$
- Using format $\langle 2, 6 \rangle$
  - Interval [-2, 1,984375]
  - Error ±0,0078125

Example 1: Overflow Verification

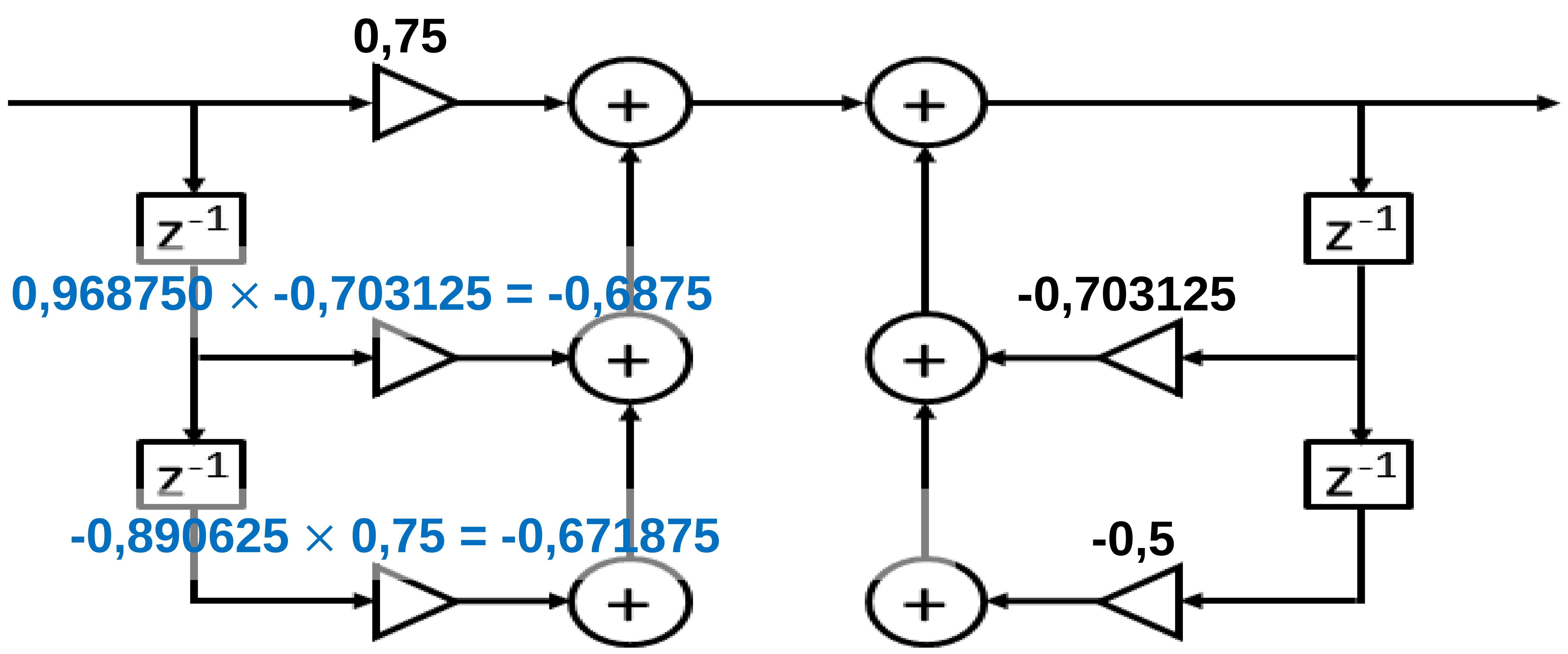
- The model checker applies non-deterministic inputs in the interval [-1,1] searching the negation of:
  - $l_{overflow} \Leftrightarrow (-2 \leq FP) \wedge (FP \leq 1{,}984375)$
- Here the verification returns the following couter-example:
  - x = { 0.0f, 0.015625f, 0.0f, -0.890625f, 0.96875f, -0.890625f }
  - xaux = { -0.890625f, 0.96875f, -0.890625f }
  - yaux = { 0f, -0.671875f, 0.890625f }



- $\sum_{k=0}^{100}|h_k|$ = 1,8178
- For x between [-1,1], $|y(n)| \leq 1{,}82$
- Using format $\langle 2, 6 \rangle$
  - Interval [-2, 1,984375]
  - Error ±0,0078125

Example 1: Overflow Verification

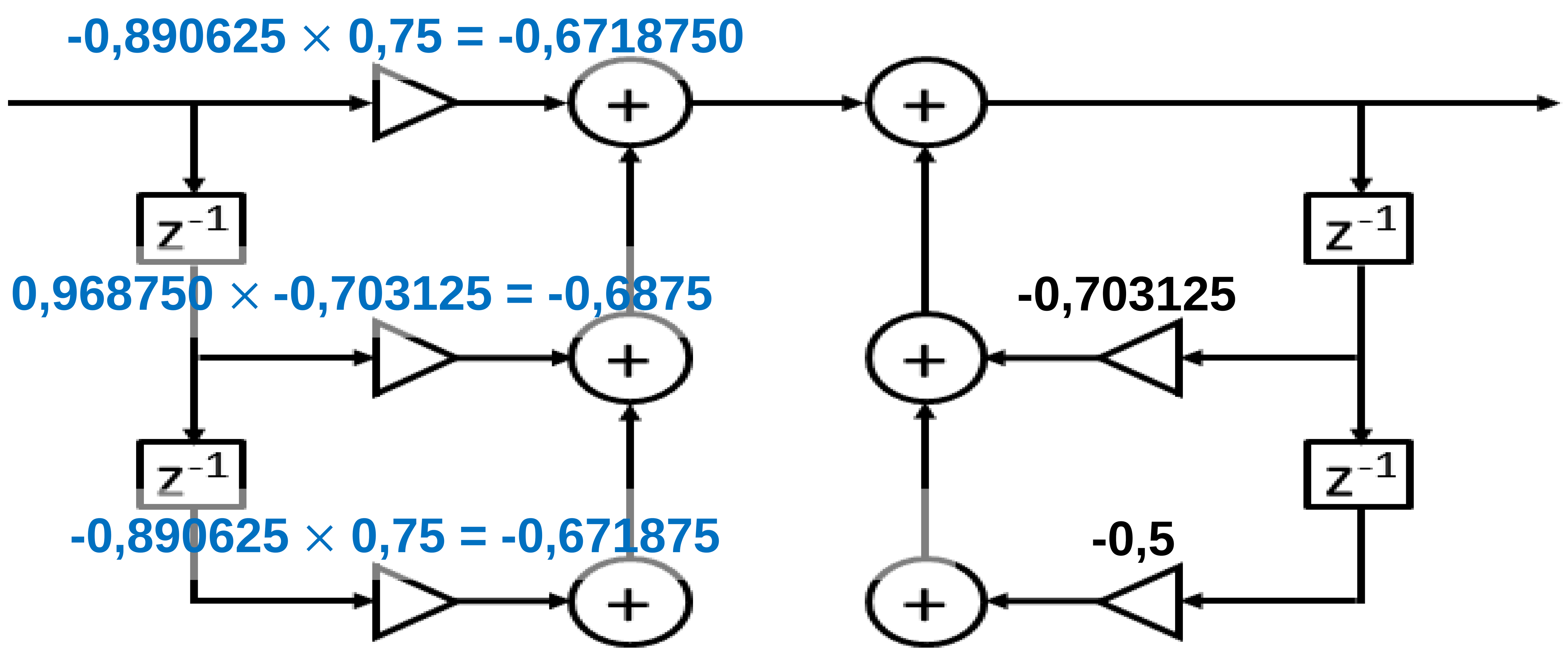
- The model checker applies non-deterministic inputs in the interval [-1,1] searching the negation of:
  - $l_{overflow} \Leftrightarrow (-2 \leq FP) \wedge (FP \leq 1{,}984375)$
- Here the verification returns the following couter-example:
  - x = { 0.0f, 0.015625f, 0.0f, -0.890625f, 0.96875f, -0.890625f }
  - xaux = { -0.890625f, 0.96875f, -0.890625f }
  - yaux = { 0f, -0.671875f, 0.890625f }



- $\sum_{k=0}^{100} |h_k|$ = 1,8178
- For x between [-1,1], $|y(n)| \leq 1{,}82$
- Using format $\langle 2, 6 \rangle$
  - Interval [-2, 1,984375]
  - Error ±0,0078125

Example 1: Overflow Verification

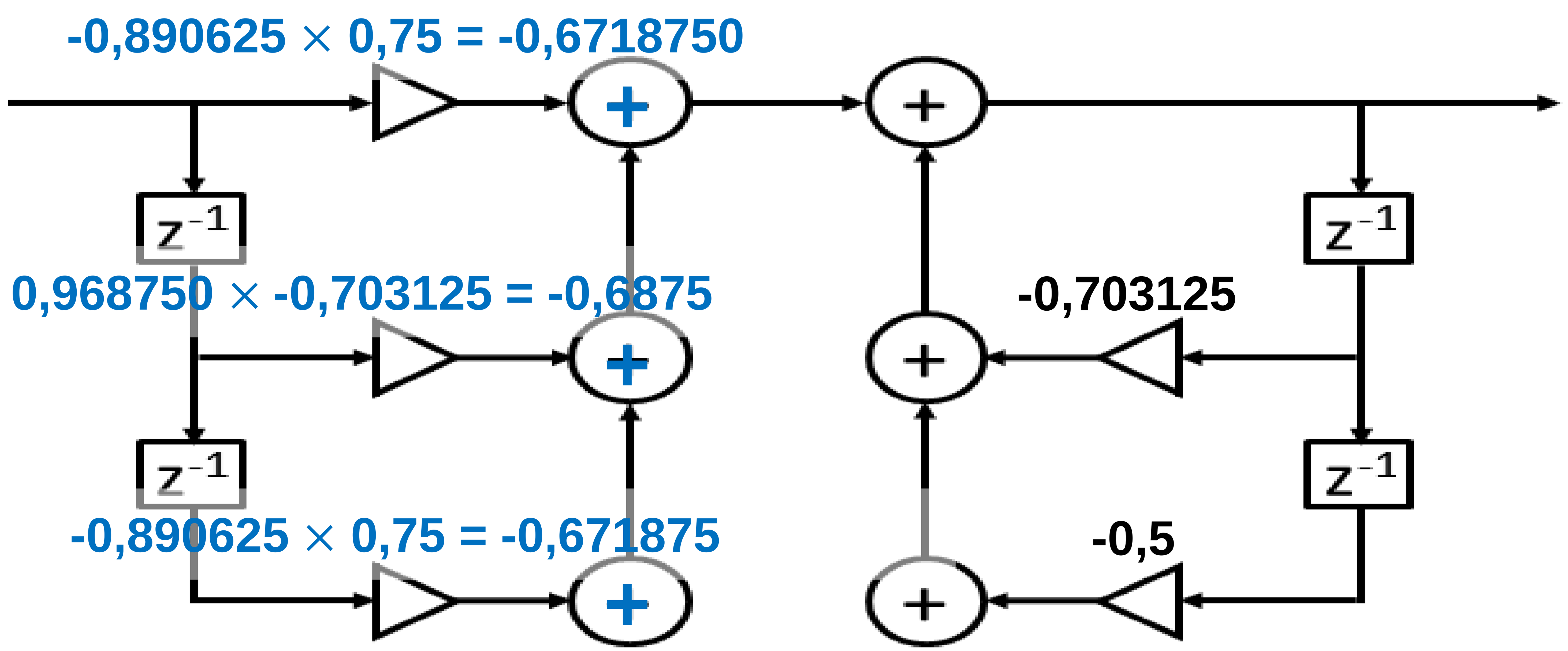
- The model checker applies non-deterministic inputs in the interval [-1,1] searching the negation of:
  - $l_{overflow} \Leftrightarrow (-2 \leq FP) \wedge (FP \leq 1{,}984375)$
- Here the verification returns the following couter-example:
  - x = { 0.0f, 0.015625f, 0.0f, -0.890625f, 0.96875f, -0.890625f }
  - xaux = { -0.890625f, 0.96875f, -0.890625f }
  - yaux = { 0f, -0.671875f, 0.890625f }



- $\sum_{k=0}^{100} |h_k|$ = 1,8178
- For x between [-1,1], $|y(n)| \leq 1{,}82$
- Using format $\langle 2, 6 \rangle$
  - Interval [-2, 1,984375]
  - Error ±0,0078125

Example 1: Overflow Verification

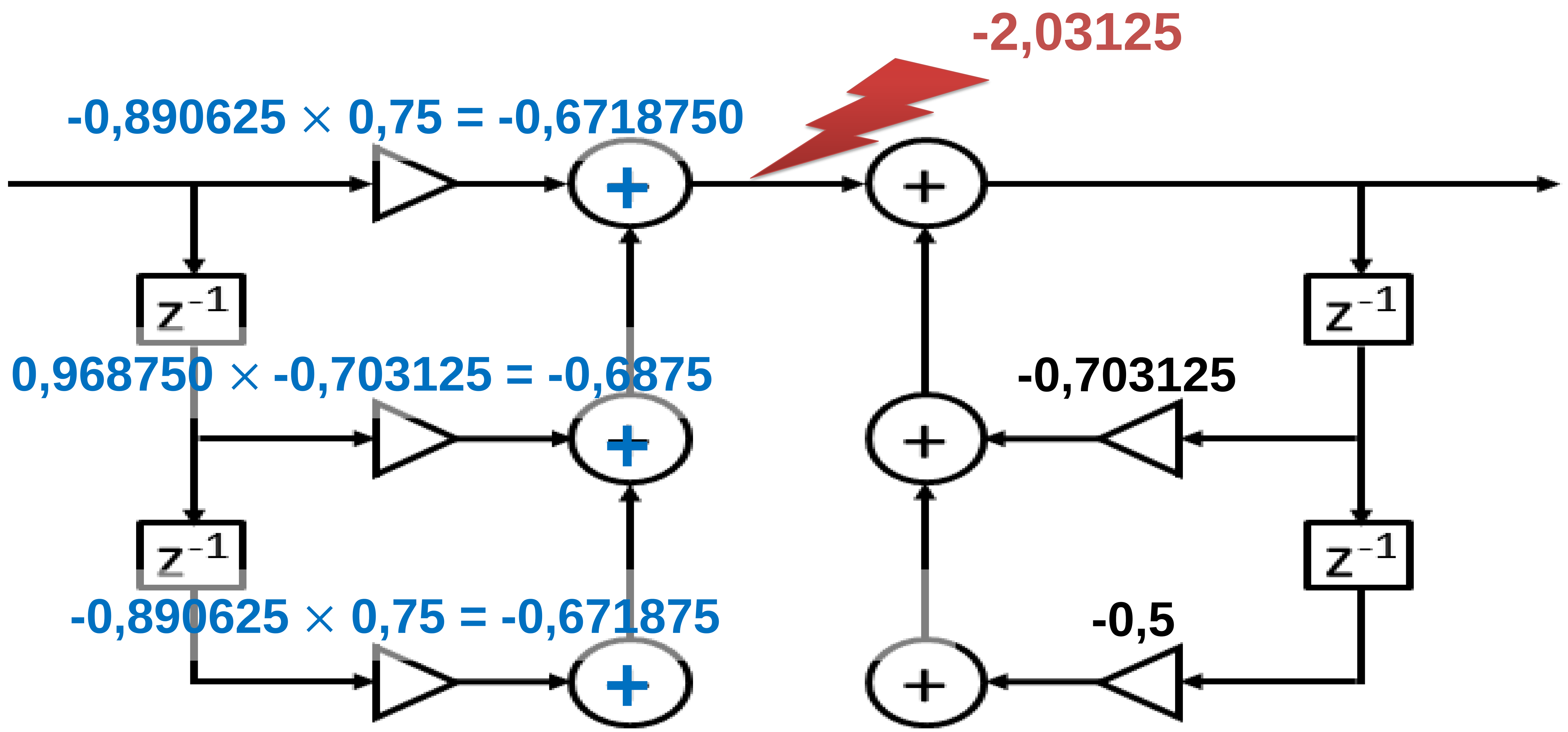
- The model checker applies non-deterministic inputs in the interval [-1,1] searching the negation of:
  - $l_{overflow} \Leftrightarrow (-2 \leq FP) \wedge (FP \leq 1{,}984375)$
- Here the verification returns the following couter-example:
  - x = { 0.0f, 0.015625f, 0.0f, -0.890625f, 0.96875f, -0.890625f }
  - xaux = { -0.890625f, 0.96875f, -0.890625f }
  - yaux = { 0f, -0.671875f, 0.890625f }



- $\sum_{k=0}^{100}|h_k|$ = 1,8178
- For x between [-1,1], $|y(n)| \leq 1{,}82$
- Using format $\langle 2, 6\rangle$
  - Interval [-2, 1,984375]
  - Error ±0,0078125

Example 1: Overflow Verification

- The model checker applies non-deterministic inputs in the interval [-1,1] searching the negation of:
  - $l_{overflow} \Leftrightarrow (-2 \leq FP) \wedge (FP \leq 1{,}984375)$
- Here the verification returns the following couter-example:
  - x = { 0.0f, 0.015625f, 0.0f, -0.890625f, 0.96875f, -0.890625f }
  - xaux = { -0.890625f, 0.96875f, -0.890625f }
  - yaux = { 0f, -0.671875f, 0.890625f }

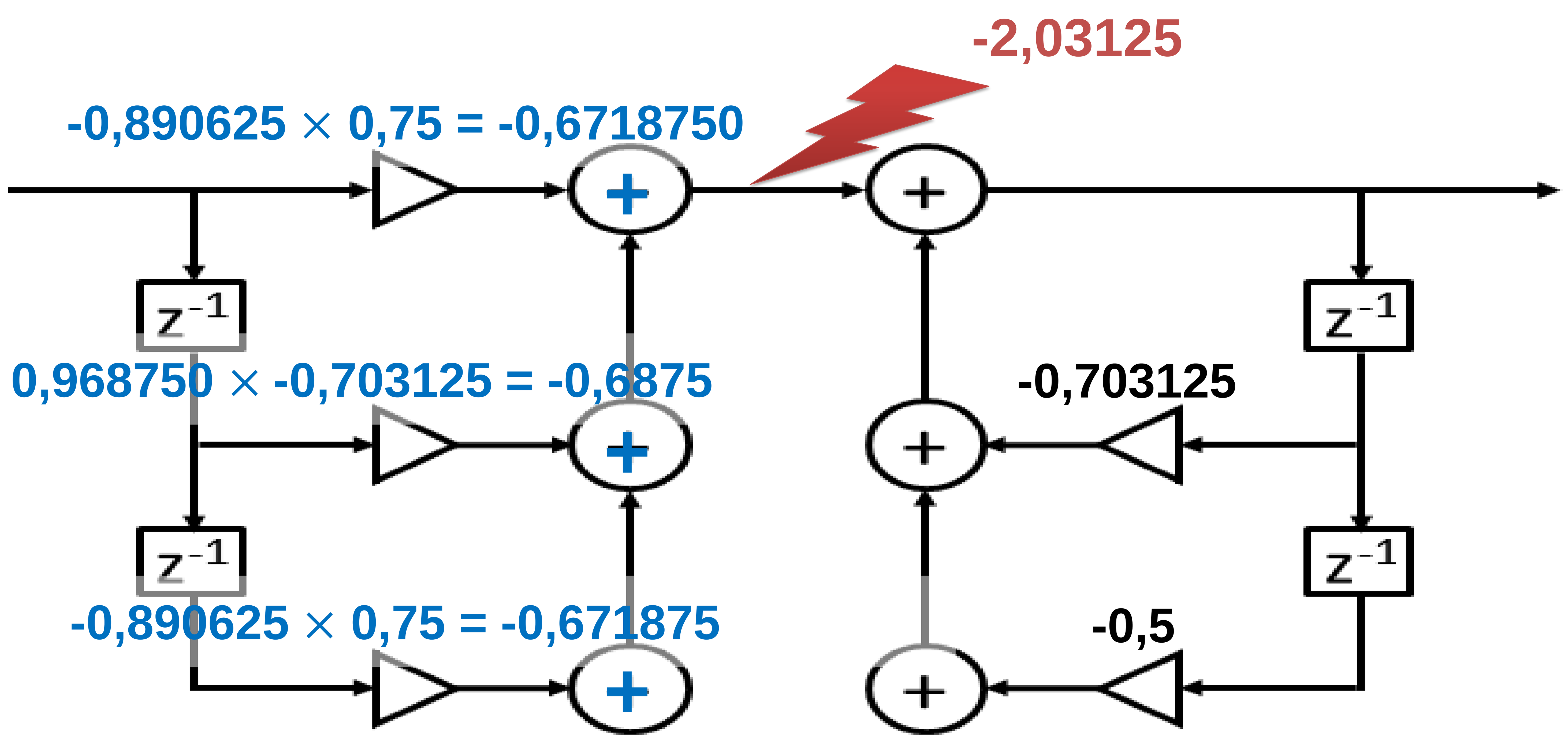


- $\sum_{k=0}^{100} |h_k|$ = 1,8178
- For x between [-1,1], $|y(n)| \leq 1{,}82$
- Using format $\langle 2, 6 \rangle$
  - Interval [-2, 1,984375]
  - Error ±0,0078125

UFAM

Example 1: Overflow Verification

- The model checker applies non-deterministic inputs in the interval [-1,1] searching the negation of:
  - $l_{overflow} \Leftrightarrow (-2 \leq FP) \wedge (FP \leq 1{,}984375)$
- Here the verification returns the following couter-example:
  - x = { 0.0f, 0.015625f, 0.0f, -0.890625f, 0.96875f, -0.890625f }
  - xaux = { -0.890625f, 0.96875f, -0.890625f }
  - yaux = { 0f, -0.671875f, 0.890625f }

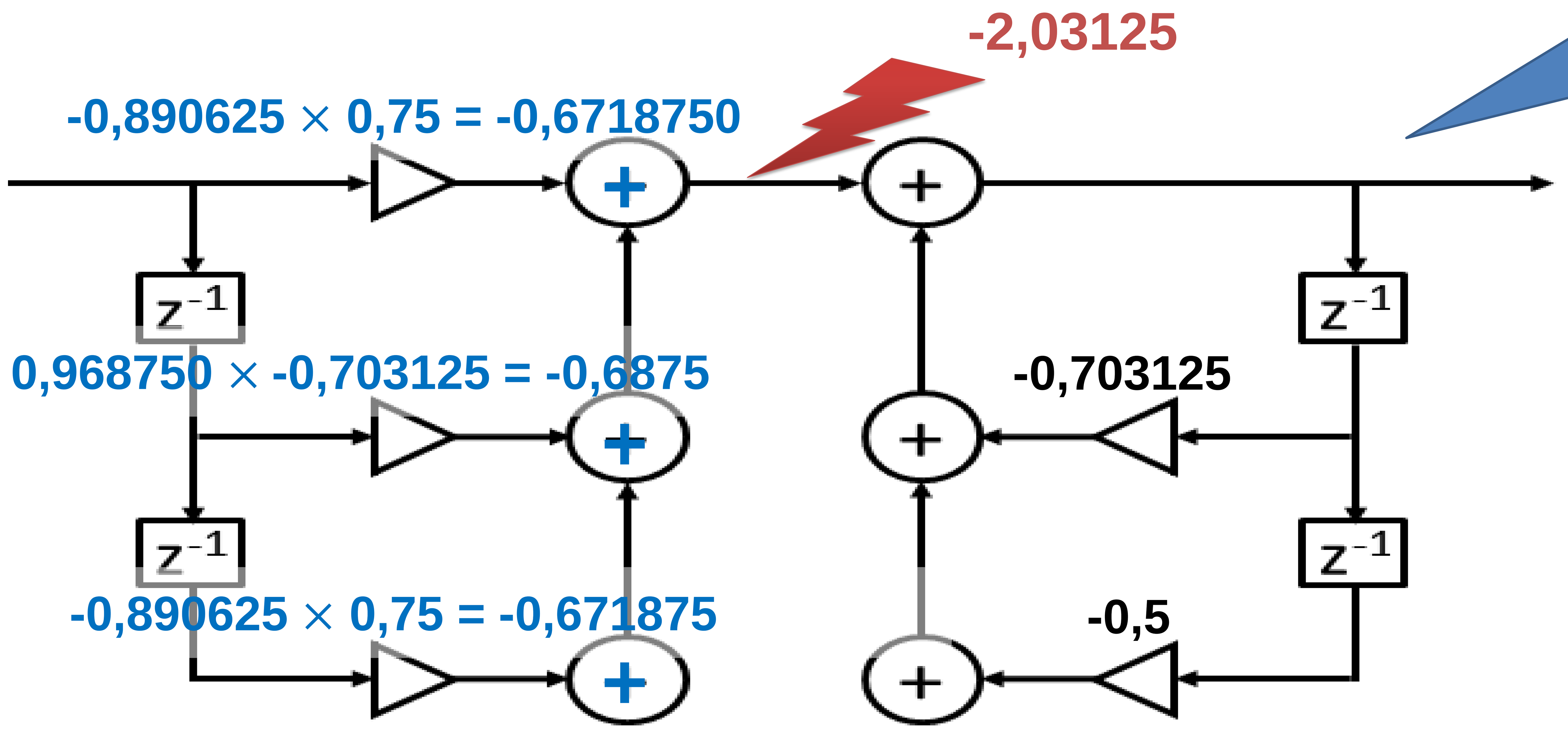


- $\sum_{k=0}^{100} |h_k|$ = 1,8178
- For x between [-1,1], $|y(n)| \leq 1{,}82$
- Using format $\langle 2, 6 \rangle$
  - Interval [-2, 1,984375]
  - Error ±0,0078125

Example 1: Overflow Verification

- The model checker applies non-deterministic inputs in the interval [-1,1] searching the negation of:
  - $l_{overflow} \Leftrightarrow (-2 \leq FP) \land (FP \leq 1{,}984375)$
- Here the verification returns the following couter-example:
  - x = { 0.0f, 0.015625f, 0.0f, -0.890625f, 0.96875f, -0.890625f }
  - xaux = { -0.890625f, 0.96875f, -0.890625f }
  - yaux = { 0f, -0.671875f, 0.890625f }

The filter also fails in Direct Form II, but does not fail for the Transposed Form II due to the order of operations



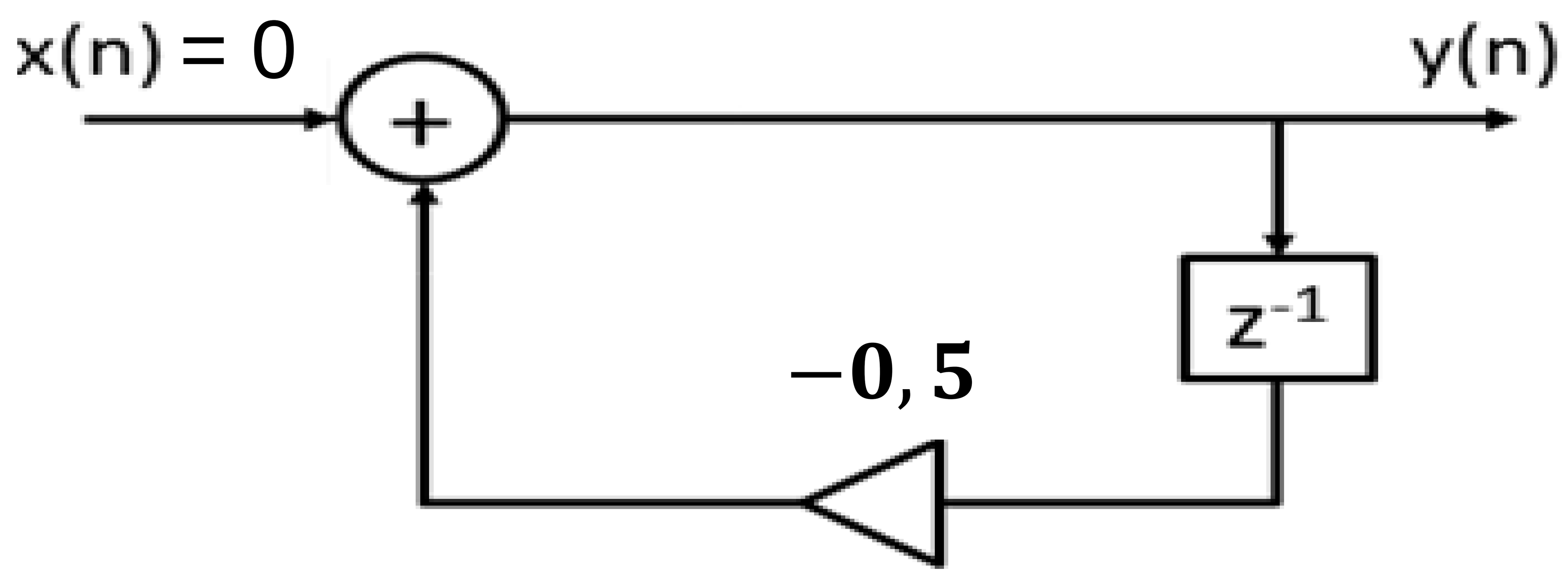
- $\sum_{k=0}^{100} |h_k|$ = 1,8178
- For x between [-1,1], $|y(n)| \leq 1{,}82$
- Using format $\langle 2, 6 \rangle$
  - Interval [-2, 1,984375]
  - Error ±0,0078125

Example 2: Limit Cycle Verification

- We use a null input and non-deterministic values to previous outputs
- An *assert* detects a failure if the set of previous states of the outputs repeats

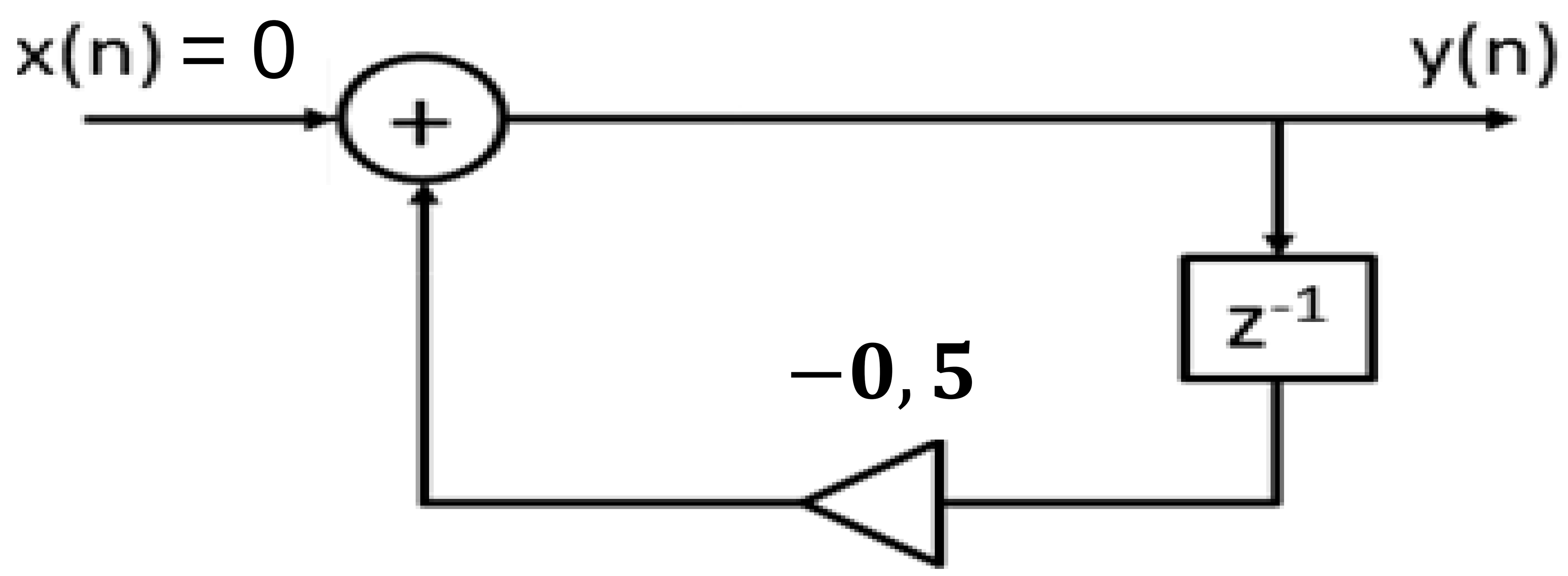
Example 2: Limit Cycle Verification

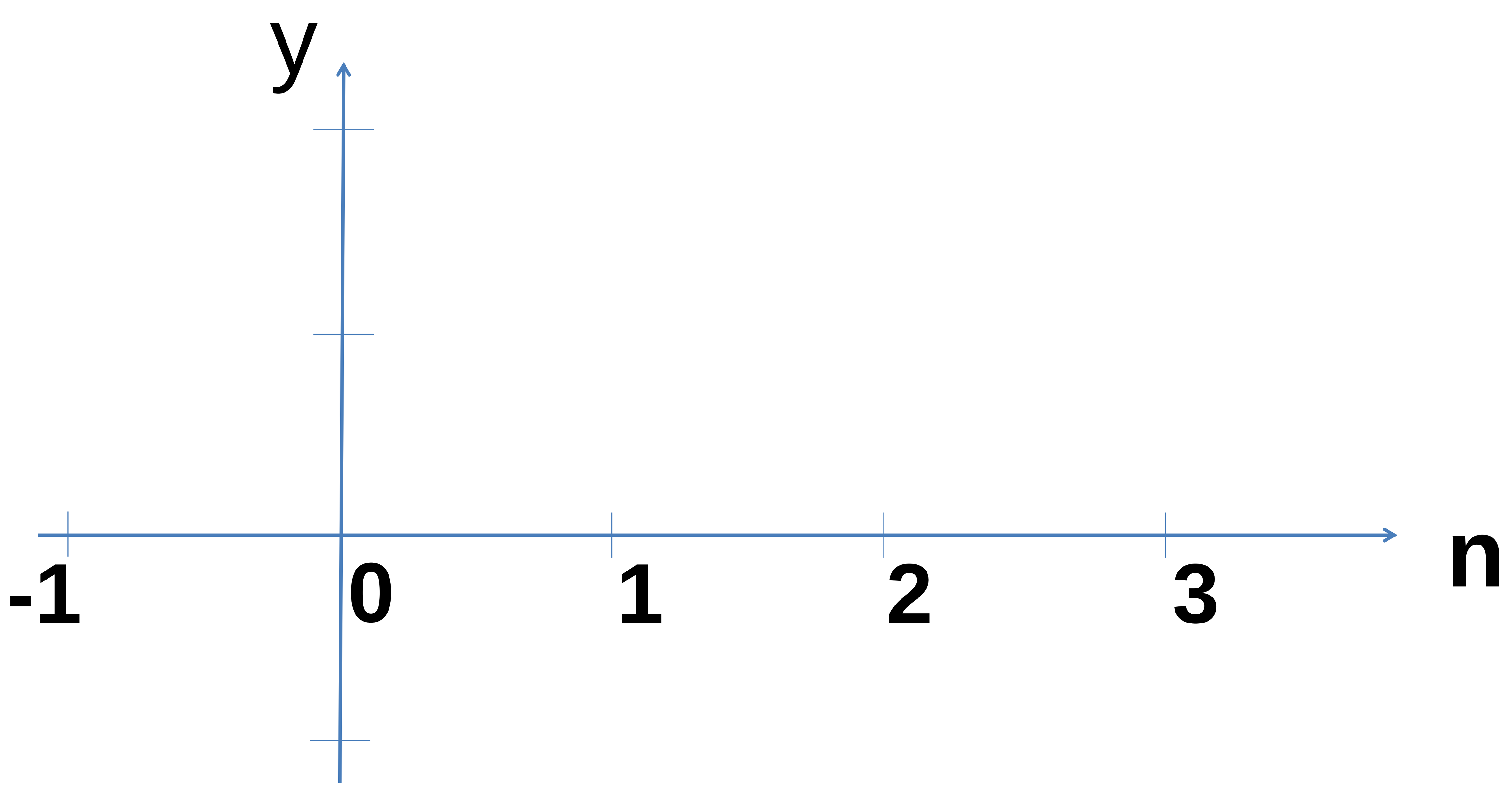
- We use a null input and non-deterministic values to previous outputs
- An *assert* detects a failure if the set of previous states of the outputs repeats



Example 2: Limit Cycle Verification

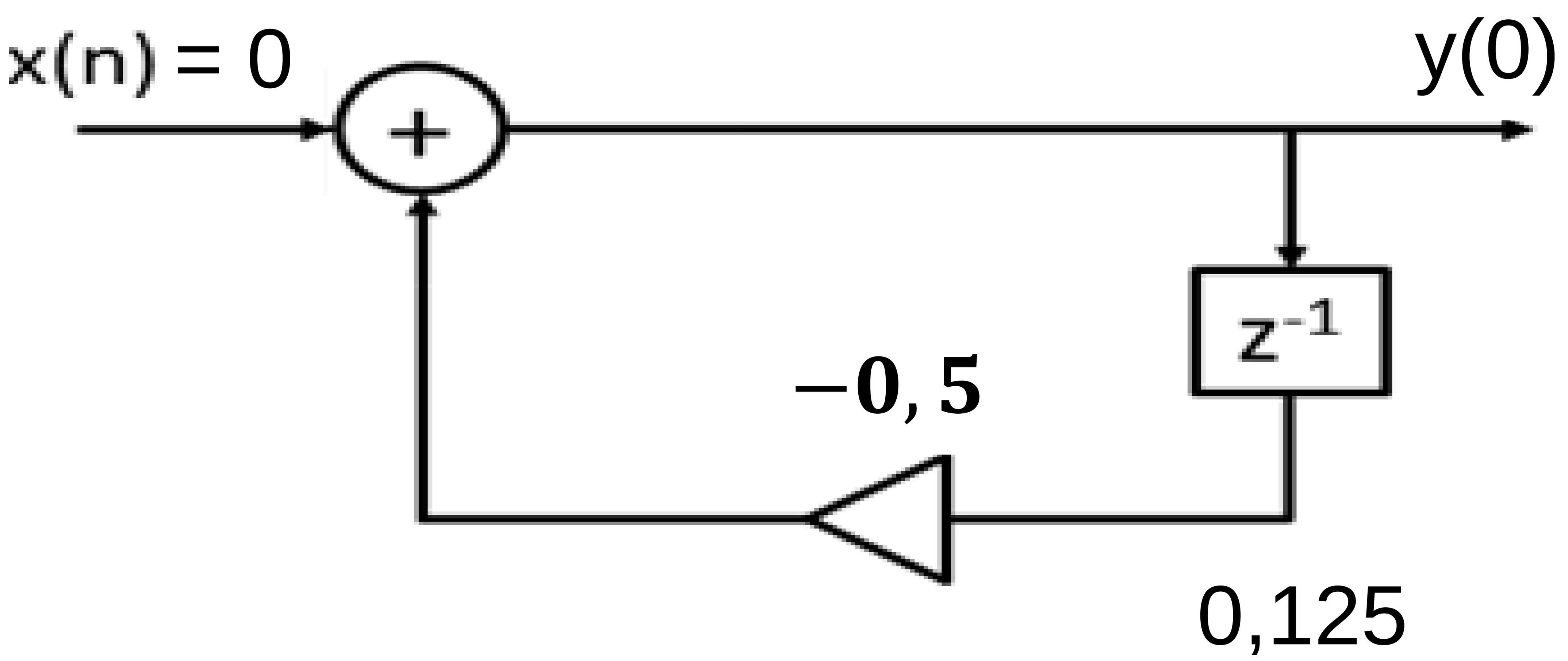
- We use a null input and non-deterministic values to previous outputs
- An *assert* detects a failure if the set of previous states of the outputs repeats

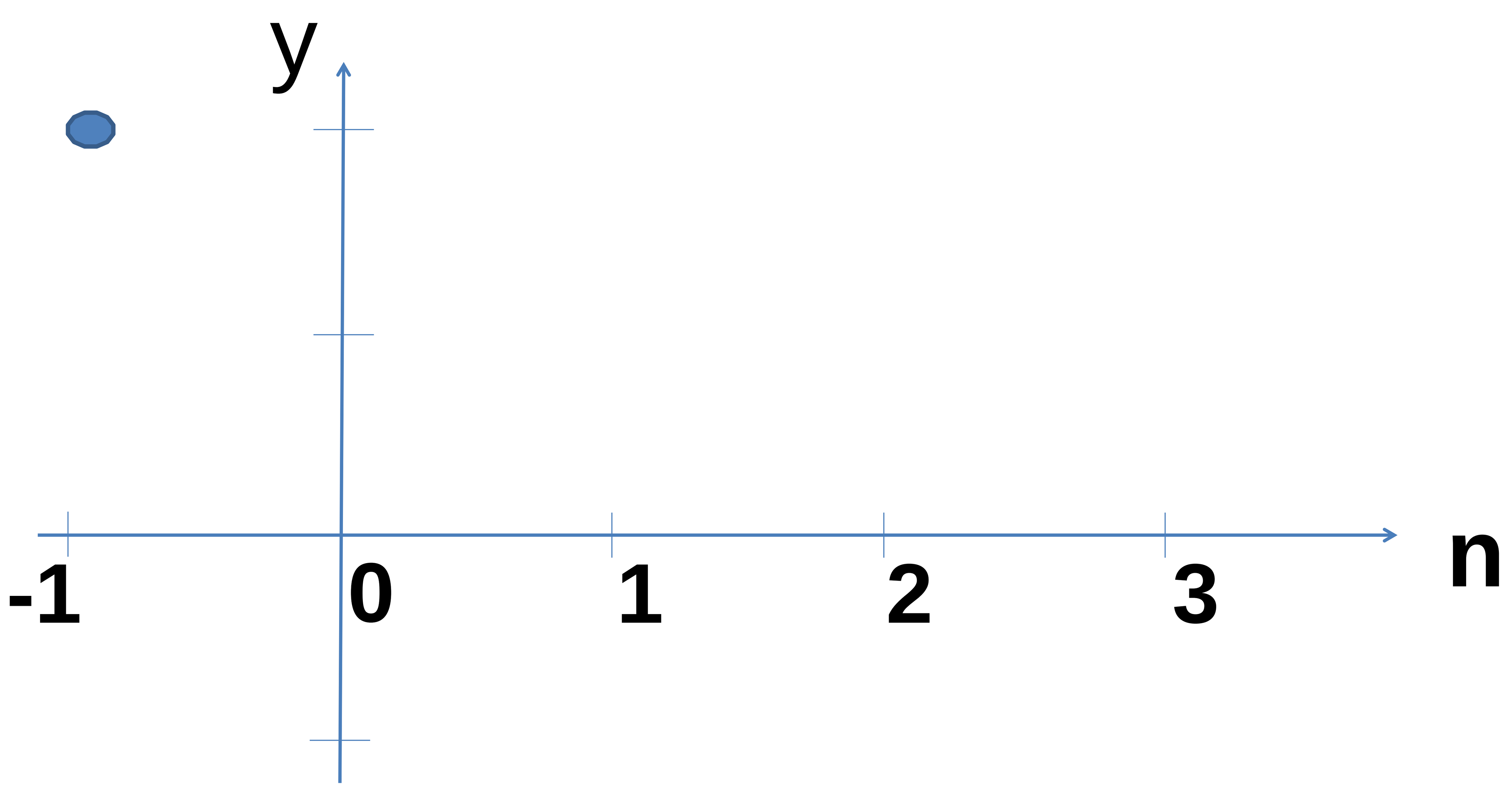




Example 2: Limit Cycle Verification

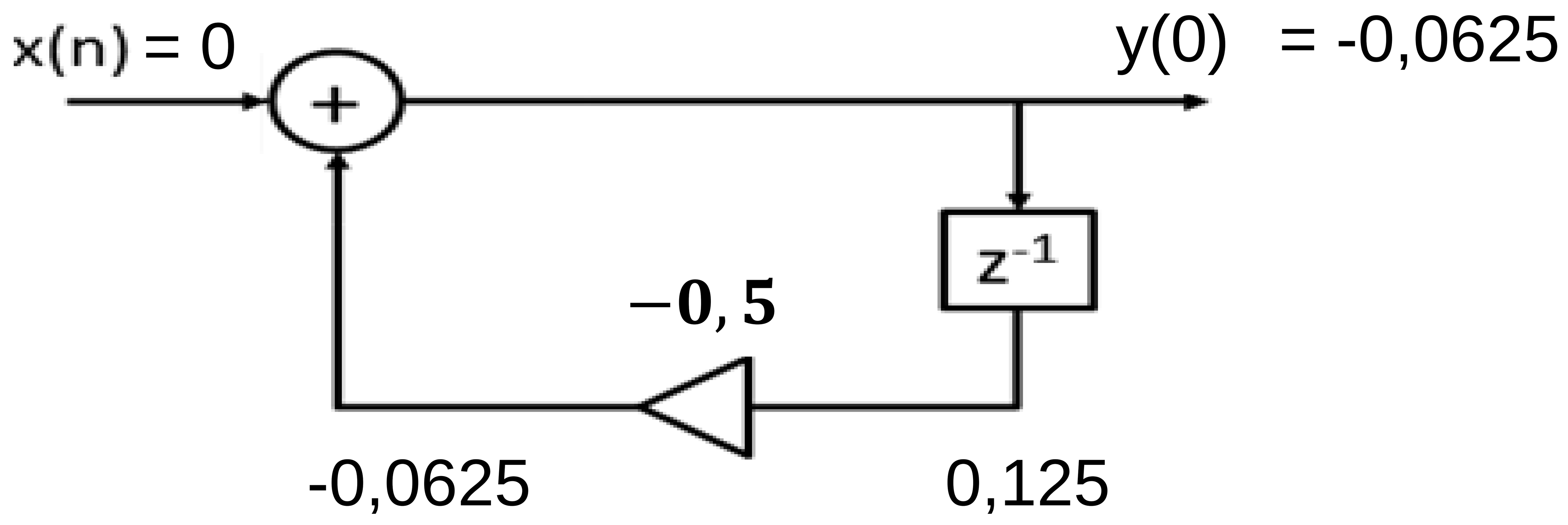
- We use a null input and non-deterministic values to previous outputs
- An *assert* detects a failure if the set of previous states of the outputs repeats

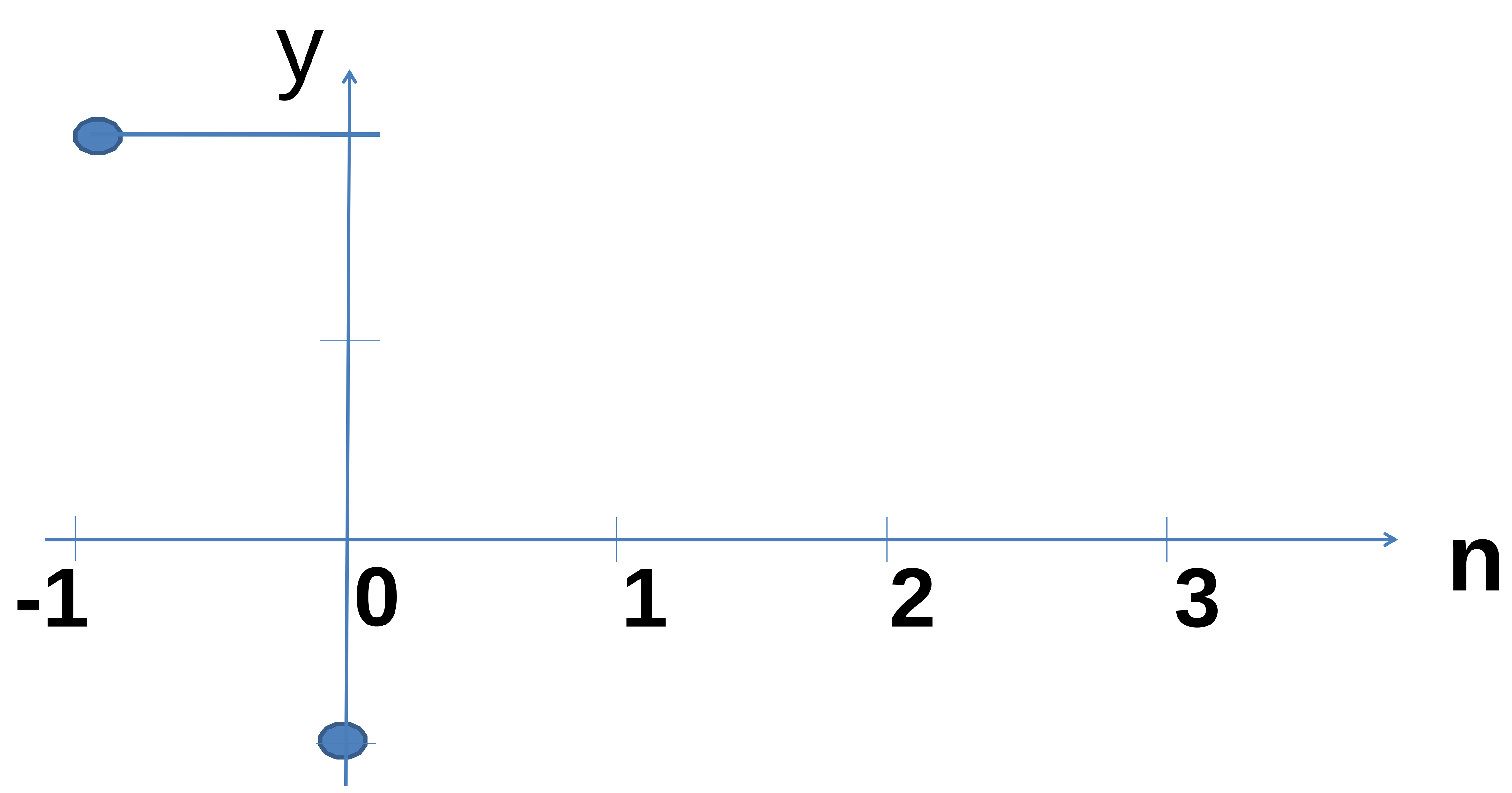




Example 2: Limit Cycle Verification

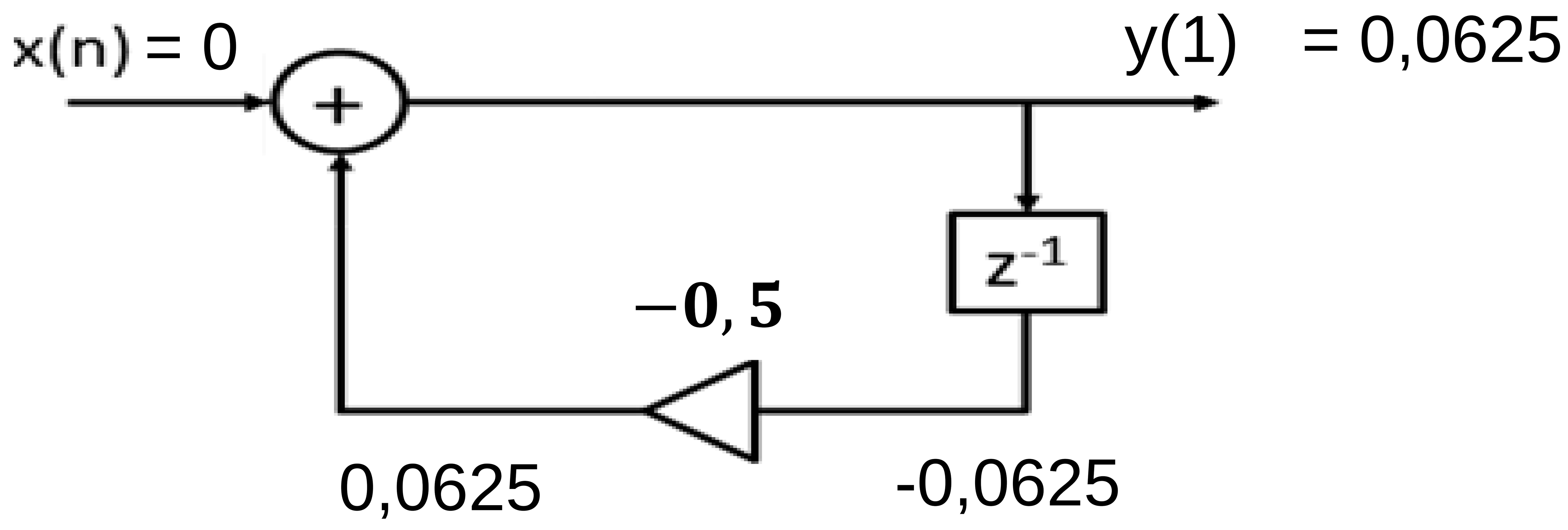
- We use a null input and non-deterministic values to previous outputs
- An *assert* detects a failure if the set of previous states of the outputs repeats

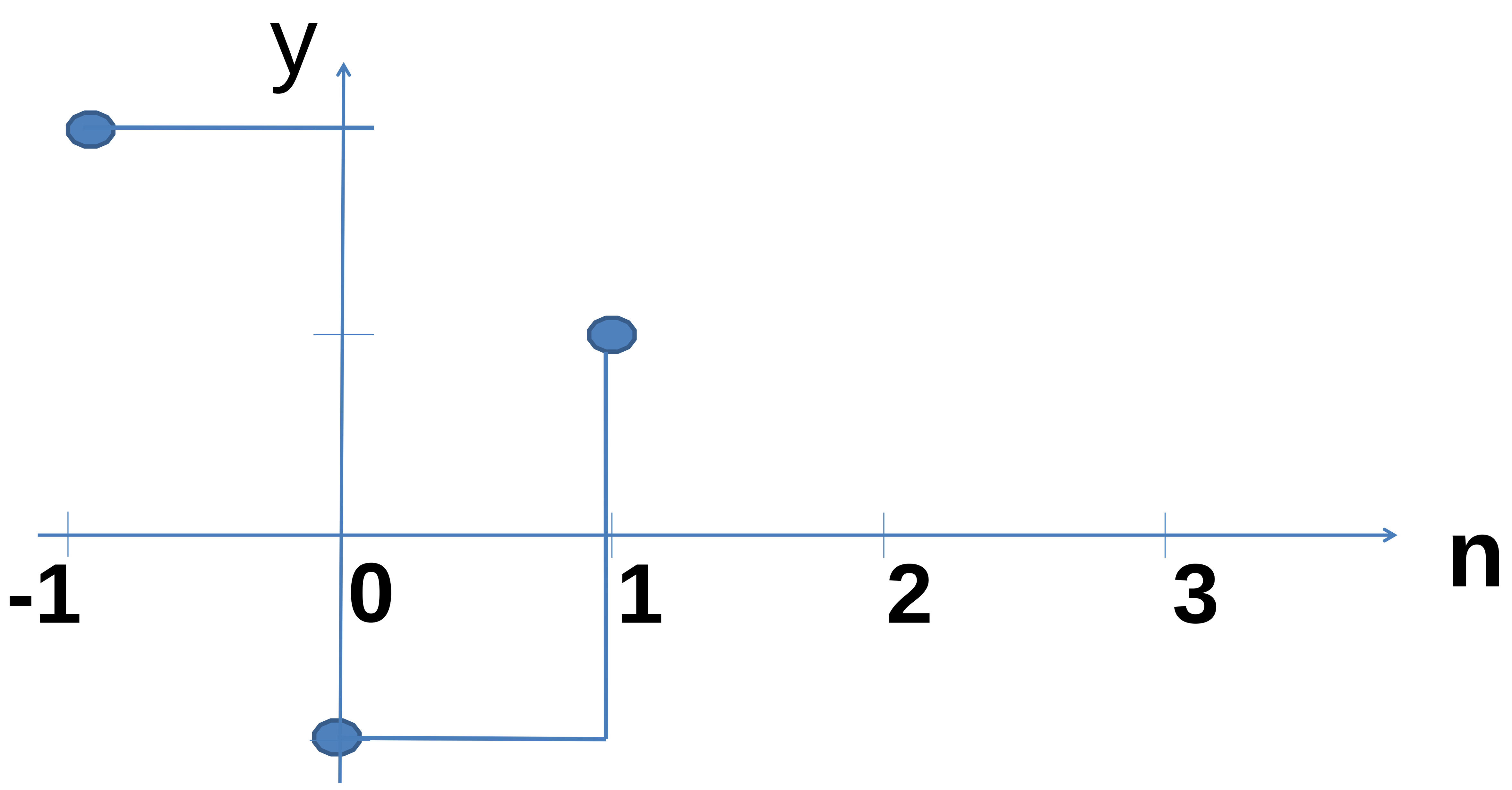




Example 2: Limit Cycle Verification

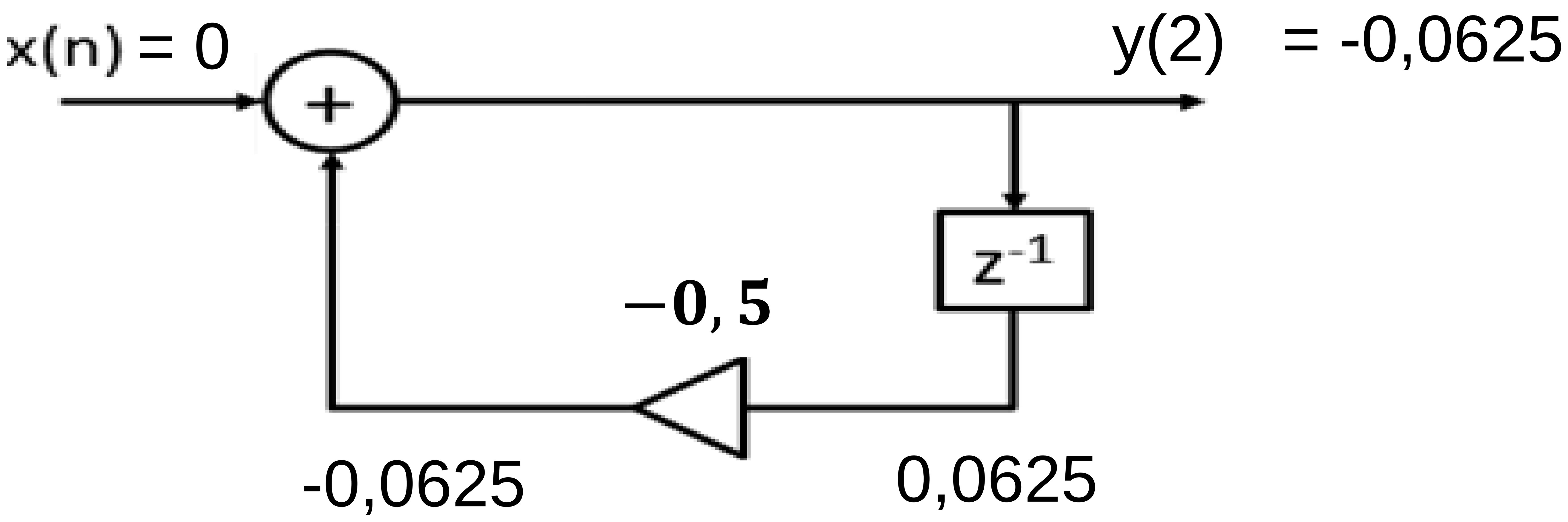
- We use a null input and non-deterministic values to previous outputs
- An *assert* detects a failure if the set of previous states of the outputs repeats

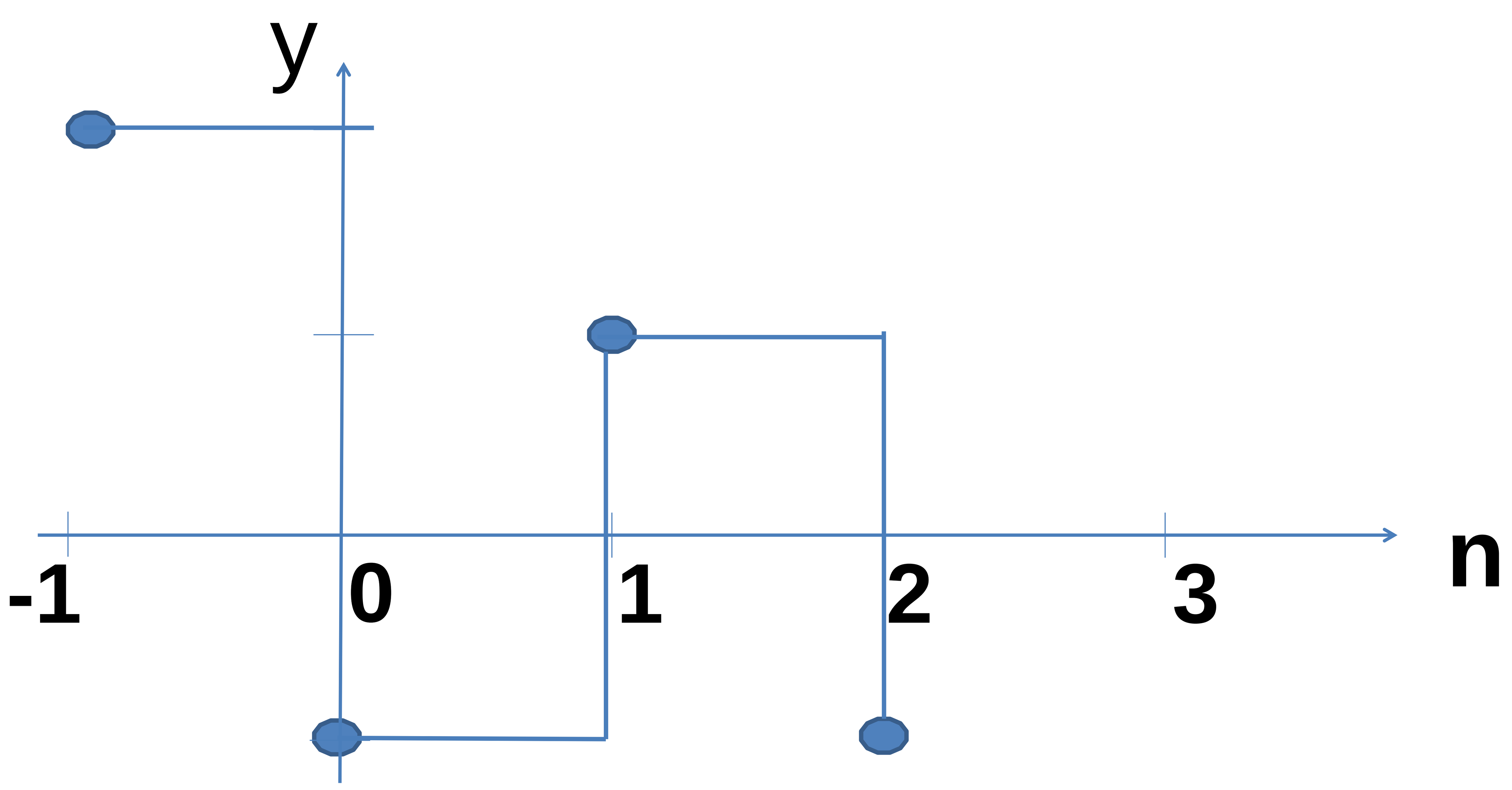




Example 2: Limit Cycle Verification

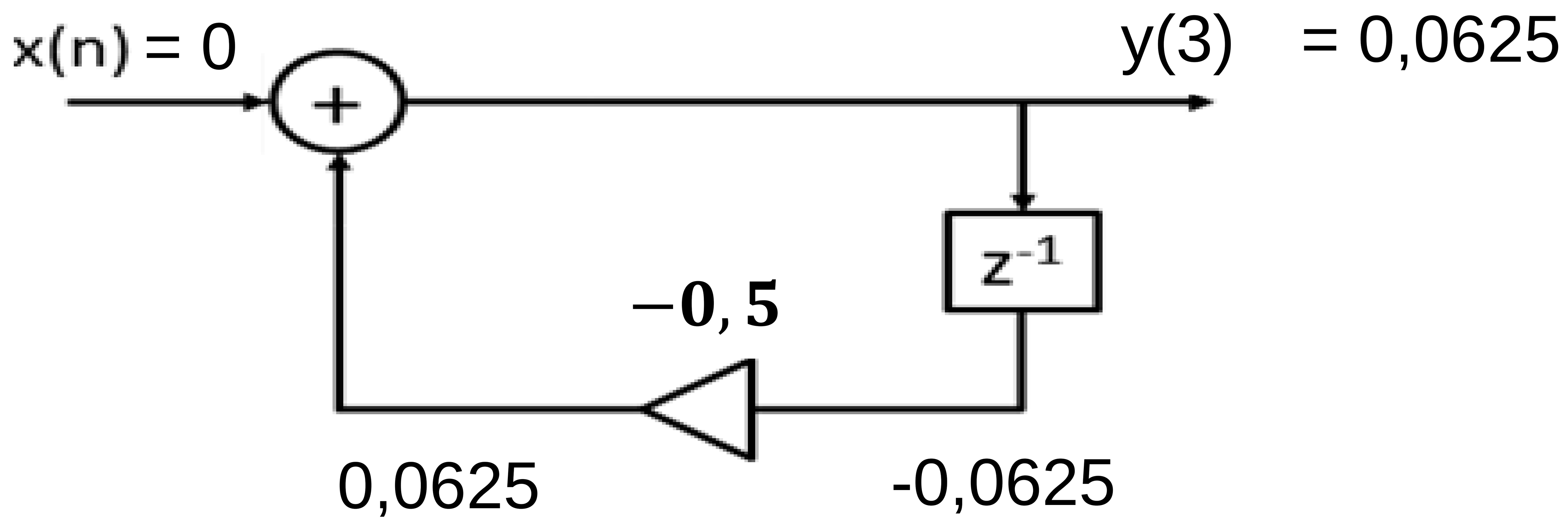
- We use a null input and non-deterministic values to previous outputs
- An *assert* detects a failure if the set of previous states of the outputs repeats

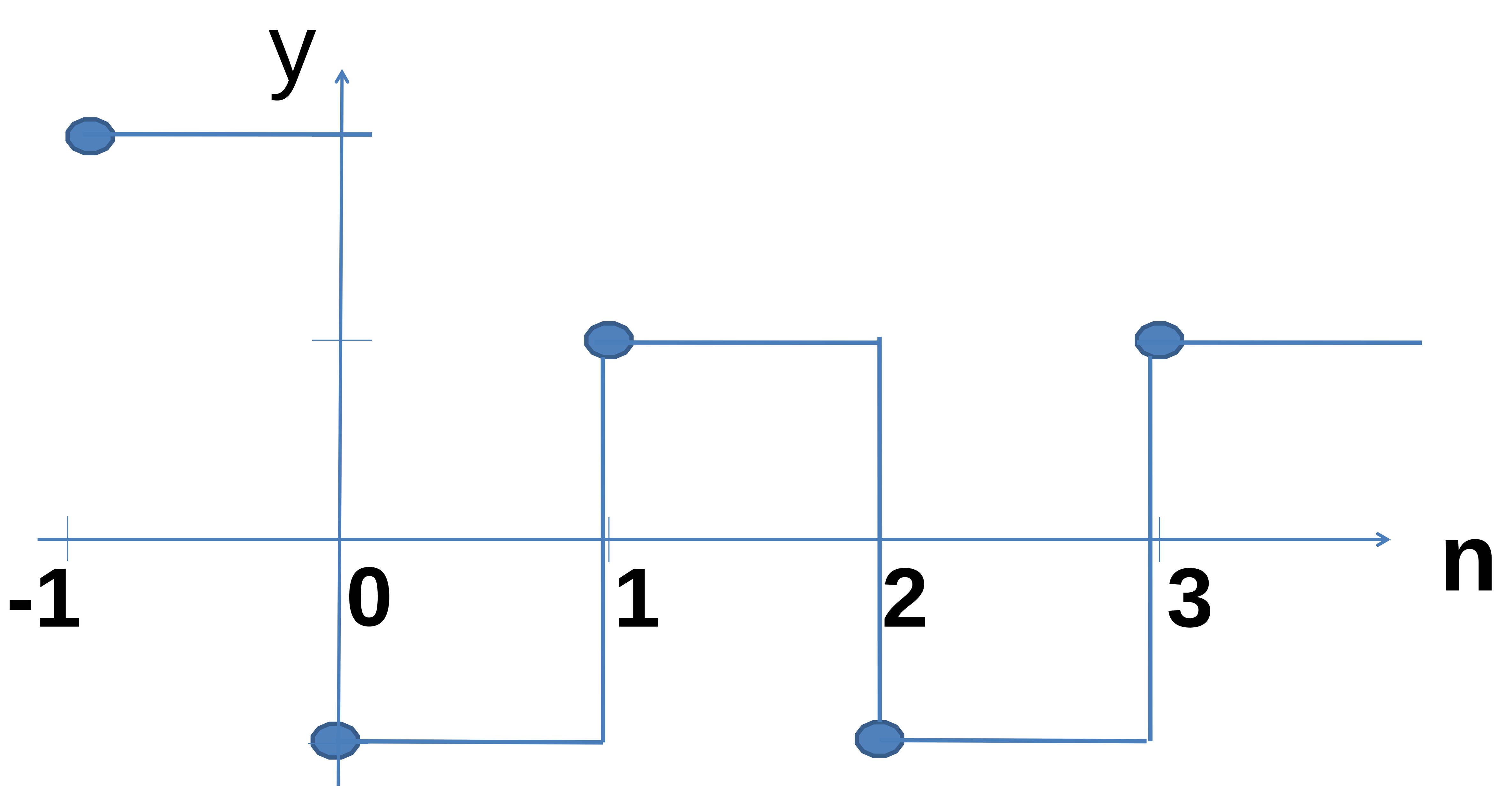




Example 2: Limit Cycle Verification

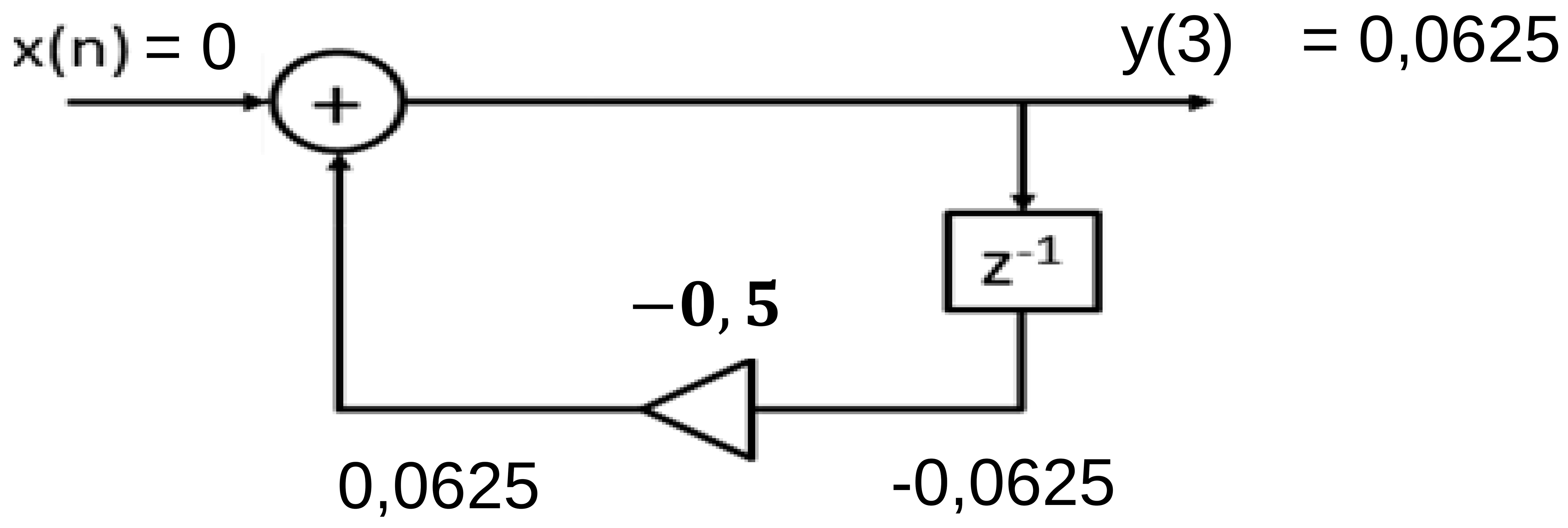
- We use a null input and non-deterministic values to previous outputs
- An *assert* detects a failure if the set of previous states of the outputs repeats

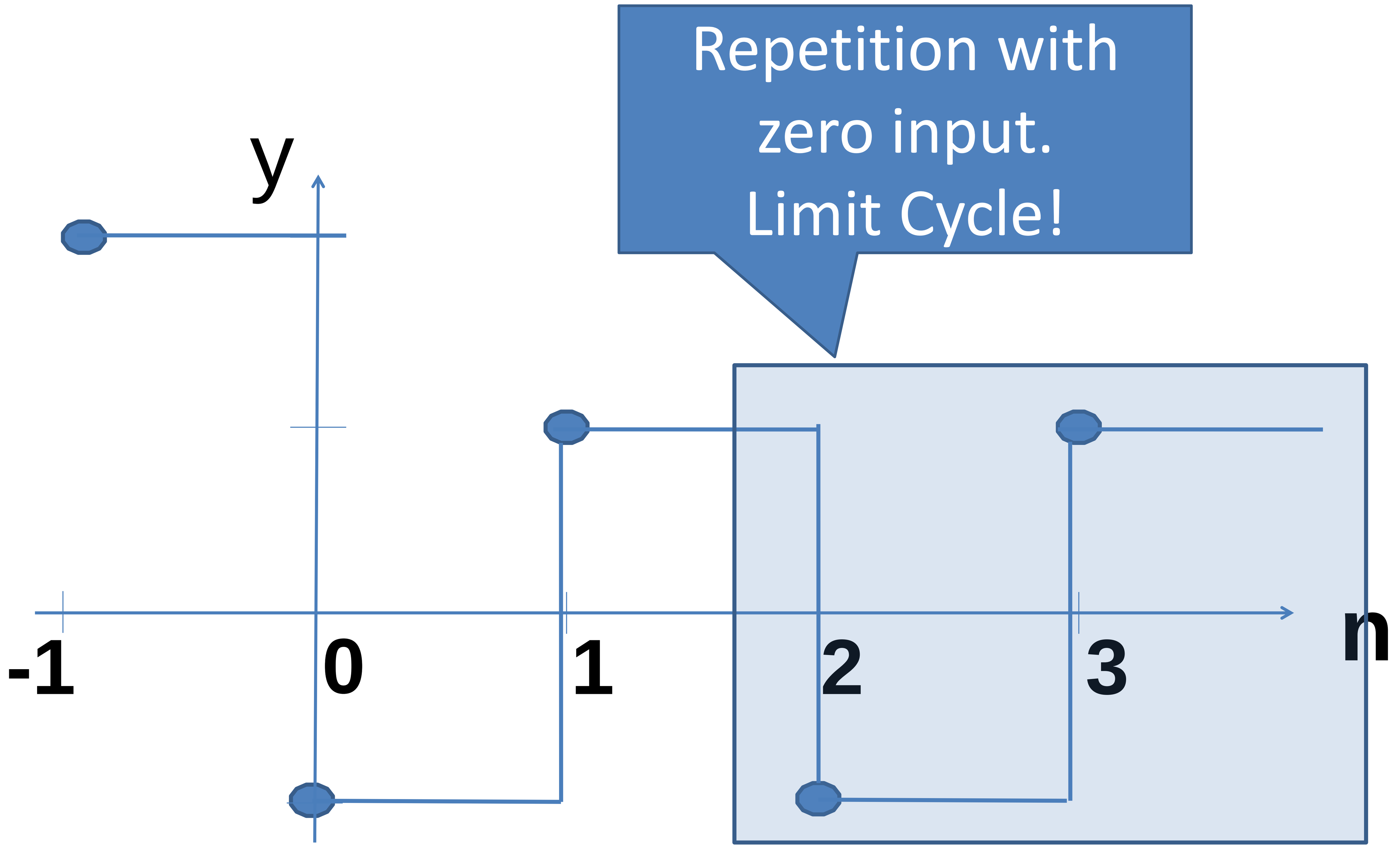




Example 2: Limit Cycle Verification

- We use a null input and non-deterministic values to previous outputs
- An *assert* detects a failure if the set of previous states of the outputs repeats





Example 3: Time Constraints Verification

Example 3: Time Constraints Verification

- Based on worst case execution time (WCET)

Example 3: Time Constraints Verification

- Based on worst case execution time (WCET)
- Example of operation on a code for MSP430:

Example 3: Time Constraints Verification

- Based on worst case execution time (WCET)
- Example of operation on a code for MSP430:

```
float iirFilterI() {
      float yn = 0;
      for (int k = 0; k < M; k++){
            yn += *b++ * *x--;
      }
      for (int k = 1; k < N; k++){
            yn -= *a++ * *y--;
      }
      return yn;
}
```

Example 3: Time Constraints Verification

- Based on worst case execution time (WCET)
- Example of operation on a code for MSP430:

```
float iirFilterI() {
     float yn = 0;
     for (int k = 0; k < M; k++){
          yn += *b++ * *x--;
     }
     for (int k = 1; k < N; k++){
          yn -= *a++ * *y--;
     }
     return yn;
}
```

MOV.W	@r9+,r12	5 cycles
MOV.W	@r9+,r13	5 cycles
SUB.W	#4,r10	5 cycles
MOV.W	4(r10),r14	3 cycles
MOV.W	6(r10),r15	3 cycles
CALL	#__fs_mpy	5 cycles
MOV.W	r7,r14	1 cycle
MOV.W	r8,r15	1 cycle
CALL	#__fs_add	5 cycles
MOV.W	r12,r7	1 cycle
MOV.W	r13,r8	1 cycle

Example 3: Time Constraints Verification

- Based on worst case execution time (WCET)
- Example of operation on a code for MSP430:

```
float iirFilterI() {
      float yn = 0;
      for (int k = 0; k < M; k++){
            yn += *b++ * *x--;
      }
      for (int k = 1; k < N; k++){
            yn -= *a++ * *y--;
      }
      return yn;
}
```

MOV.W	@r9+,r12	5 cycles
MOV.W	@r9+,r13	5 cycles
SUB.W	#4,r10	5 cycles
MOV.W	4(r10),r14	3 cycles
MOV.W	6(r10),r15	3 cycles
CALL	#__fs_mpy	5 cycles
MOV.W	r7,r14	1 cycle
MOV.W	r8,r15	1 cycle
CALL	#__fs_add	5 cycles
MOV.W	r12,r7	1 cycle
MOV.W	r13,r8	1 cycle

- Number of instructions depends of filter order
- Execution time will depends of plataform and word length

Example 3: Time Constraints Verification

- Based on worst case execution time (WCET)
- Example of operation on a code for MSP430:

```
float iirFilterI() {
     float yn = 0;
     for (int k = 0; k < M; k++){
          yn += *b++ * *x--;
     }
     for (int k = 1; k < N; k++){
          yn -= *a++ * *y--;
     }
     return yn;
}
```

MOV.W	@r9+,r12	5 cycles
MOV.W	@r9+,r13	5 cycles
SUB.W	#4,r10	5 cycles
MOV.W	4(r10),r14	3 cycles
MOV.W	6(r10),r15	3 cycles
CALL	#__fs_mpy	5 cycles
MOV.W	r7,r14	1 cycle
MOV.W	r8,r15	1 cycle
CALL	#__fs_add	5 cycles
MOV.W	r12,r7	1 cycle
MOV.W	r13,r8	1 cycle

Example 3: Time Constraints Verification

- Based on worst case execution time (WCET)
- Example of operation on a code for MSP430:

```
float iirFilterI() {
    float yn = 0;
    for (int k = 0; k < M; k++){
        yn += *b++ * *x--;
    }
    for (int k = 1; k < N; k++){
        yn -= *a++ * *y--;
    }
    return yn;
}
```

MOV.W	@r9+,r12	5 cycles
MOV.W	@r9+,r13	5 cycles
SUB.W	#4,r10	5 cycles
MOV.W	4(r10),r14	3 cycles
MOV.W	6(r10),r15	3 cycles
CALL	#__fs_mpy	5 cycles
MOV.W	r7,r14	1 cycle
MOV.W	r8,r15	1 cycle
CALL	#__fs_add	5 cycles
MOV.W	r12,r7	1 cycle
MOV.W	r13,r8	1 cycle

- The model checker aplies and arbitrary input searching for the negation of
  - $l_{timing} \Leftrightarrow ((N \times T) \leq D)$

Example 3: Time Constraints Verification

- Based on worst case execution time (WCET)
- Example of operation on a code for MSP430:

```
float iirFilterI() {
     float yn = 0;
     for (int k = 0; k < M; k++){
          yn += *b++ * *x--;
     }
     for (int k = 1; k < N; k++){
          yn -= *a++ * *y--;
     }
     return yn;
}
```

MOV.W	@r9+,r12	5 cycles
MOV.W	@r9+,r13	5 cycles
SUB.W	#4,r10	5 cycles
MOV.W	4(r10),r14	3 cycles
MOV.W	6(r10),r15	3 cycles
CALL	#__fs_mpy	5 cycles
MOV.W	r7,r14	1 cycle
MOV.W	r8,r15	1 cycle
CALL	#__fs_add	5 cycles
MOV.W	r12,r7	1 cycle
MOV.W	r13,r8	1 cycle

- The model checker aplies and arbitrary input searching for the negation of
  - $l_{timing} \Leftrightarrow ((N \times T) \leq D)$

Deadline

Number of cycles

Cycle time

Experimental Results

#	Filter	N	M	$\sum\lvert h_k\rvert$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

UFAM

Experimental Results

#	Filter	N	M	$\sum\|h_k\|$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	[illegible]	[illegible]	[illegible]	6	OF	579	634	<1
3	LP-IIR	3	1	[illegible]	[illegible]	[illegible]	6	OF, LC	210	29	<1
4	LP-IIR	3	1	[illegible]	[illegible]	[illegible]	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

Different types of filters with different size were verified

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

Experimental Results

#	Filter	N	M	$\sum\|h_k\|$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

Experimental Results

#	Filter	N	M	$\sum\|h_k\|$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	[illegible]	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	[illegible]	[illegible]	[illegible]	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	[illegible]	[illegible]	[illegible]	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	[illegible]	[illegible]	[illegible]	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

Faster verification of low order filters and with smaller word-length

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

UFAM

Experimental Results

#	Filter	N	M	$\sum\lvert h_k\rvert$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

Experimental Results

#	Filter	N	M	$\sum\|h_k\|$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	[illegible]	[illegible]	[illegible]	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	[illegible]	[illegible]	[illegible]	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	[illegible]	[illegible]	[illegible]	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	[illegible]	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

Fast verification of time constraints (sequential code)

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

UFAM

Experimental Results

#	Filter	N	M	$\sum\|h_k\|$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

UFAM

Experimental Results

#	Filter	N	M	∑|h$_k$|	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	[illegible]	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	[illegible]	[illegible]	[illegible]	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	[illegible]	[illegible]	[illegible]	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

Time constraint failure for high order filters

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

UFAM

Experimental Results

#	Filter	N	M	$\sum\|h_k\|$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

Experimental Results

#	Filter	N	M	$\sum\|h_k\|$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,			C	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,			-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,			C	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

Overflow detected even for conservative cases

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

Experimental Results

#	Filter	N	M	$\sum\|h_k\|$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

Experimental Results

#	Filter	N	M	∑|h$_k$|	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	[illegible]	[illegible]	[illegible]	[illegible]
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	[illegible]	[illegible]	[illegible]	[illegible]
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	[illegible]	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

Higher verification time for high order filters

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out

Experimental Results

#	Filter	N	M	$\sum\lvert h_k\rvert$	Input range	Word-length	Input size	Failures	Verification time (s)		
									OF	LC	TC
1	LP-IIR	2	1	2	[-1,1]	<2,4>	6	OF, LC	39	4	<1
2	LP-Butterworth-IIR	3	3	1.2	[-1.6,1.6]	<2,5>	6	OF	579	634	<1
3	LP-IIR	3	1	4	[-1,1]	<3,4>	6	OF, LC	210	29	<1
4	LP-IIR	3	1	1.56	[-1,1]	<2,4>	6	-	110	51	<1
5	LP-FIR	1	31	1.93	[-1,1]	<2,6>	31	TC	TO	98	1
6	HP-ChebyshevI-IIR	3	3	1.33	[-1,1]	<2,10>	6	-	853	2058	<1
7	BP-Elliptic-IIR	3	3	1.24	[-1.0,1.0]	<2,10>	6	LC	546	474	<1
8	BS-Butterworth-IIR	3	3	1.81	[-1.0,1.0]	<2,8>	6	OF	106	1299	<1
9	BP-Elliptic-IIR	5	5	0.91	[-1.1,1.1]	<1,8>	10	OF, LC	7	20	<1
10	HP-Butterworth-IIR	5	5	1.58	[-1.27, 1.27]	<2,6>	10	LC	2468	1508	<1
11	BP-ChebyshevI-IIR	5	5	1.51	[-1.33, 1.33]	<2,6>	10	-	TO	TO	<1
12	HP-Elliptic-IIR	7	7	5.39	[-1,1]	<3,13>	14	TC	73	TO	<1

OF – Overflow, LC – Limit Cycle, TC – Time constraint, TO - Time out