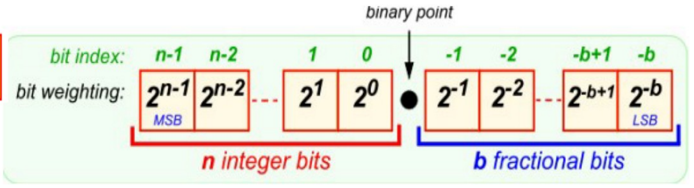
# 04_Basic_XDC.pdf

**FIX and UFIX Types**

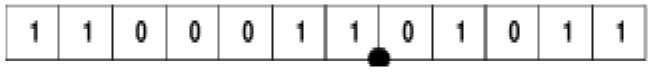
```
FIX:  signed 2s complement
UFIX: unsigned
```

Zweierkomplement ->
negatives MSB-Gewicht



**Knowledge Check**

VA_saturation2.circ VA_saturation.circ FPGAArithmetic_1.pdf



Format =

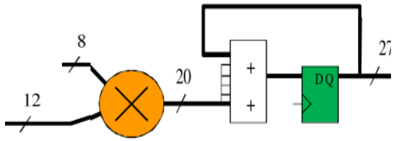
Value =

12-bit, Max:+1, Min:-1, Format: < FIX_12_10 > 10.0000000000 .. 01.1111111111= $-2..(2-2^{-10})$

< UFIX_12_10 > 00.0000000000 .. 11.1111111111= $0..(4-2^{-10})$

10-bit, Max:0.8 ,Min: 0.2, Format: < UFIX_10_10 > .0000000000 .. .1111111111

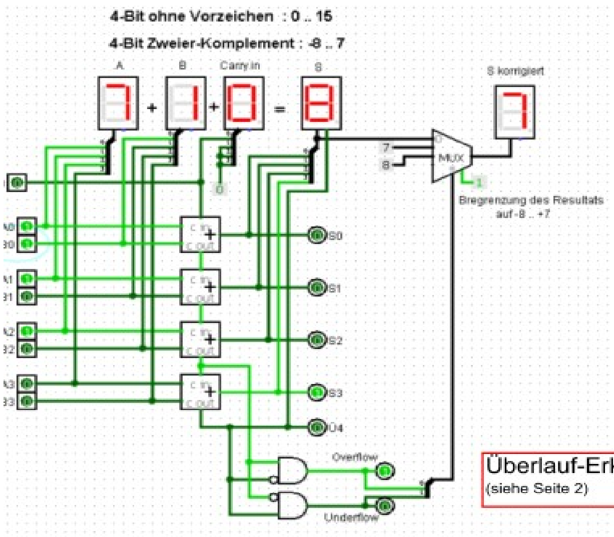
11-bit, Max:278,Min: -138, Format: < FIX_11_1 > 1000000000.0 .. 0111111111.1= -512 .. 511.5

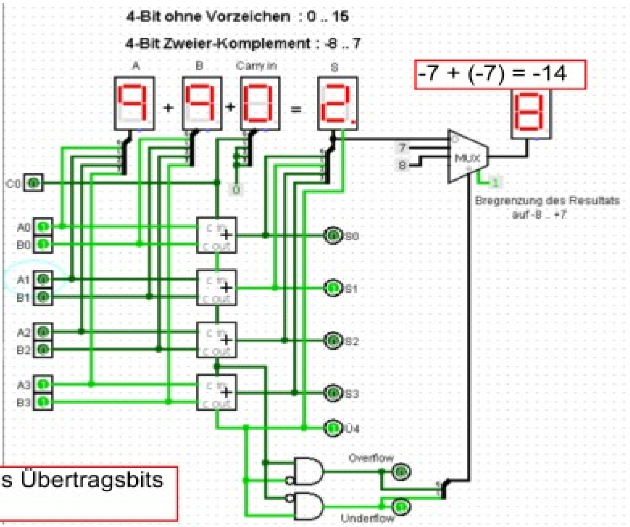


<Fix_12_9> + <Fix_8_3> = <Fix_15_9>

<Fix_8_7> x <Ufix_8_6> = <Fix_16_13>

## Saturation -- Sättigungs-Logik

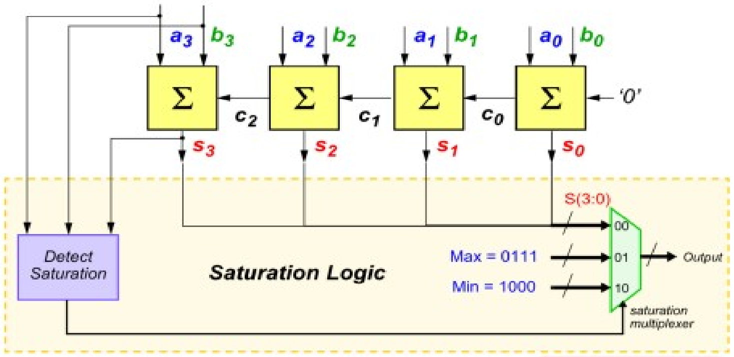

4-Bit ohne Vorzeichen : 0 .. 15
4-Bit Zweier-Komplement : -8 .. 7
A
B
Carry in
S
S korrigiert
MUX
Bregrenzung des Resultats auf -8 .. +7
S0
S1
S2
S3
Ü4
Overflow
Underflow


4-Bit ohne Vorzeichen : 0 .. 15
4-Bit Zweier-Komplement : -8 .. 7
A
B
Carry in
S
MUX
Bregrenzung des Resultats auf -8 .. +7
A0
B0
A1
B1
A2
B2
A3
B3
S0
S1
S2
S3
Ü4
Overflow
Underflow

-7 + (-7) = -14

Überlauf-Erkennung aus Übertragsbits
(siehe Seite 2)

sign extension



Überlauf-Erkennung aus den Vorzeichen der Summanden und der Summe

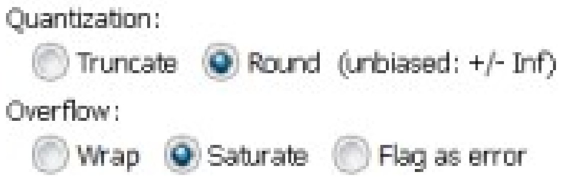
(FPGAArithmetic_2.pdf)

# 05_Signal_Routing.pdf

## Quantization

– Truncate: Discard bits to the right of the least significant bit

– Round: Round to the nearest representable value or to the value farthest from zero if there are two equidistant nearest representable values



Quantization:
( ) Truncate [✓ Round] (unbiased: +/- Inf)

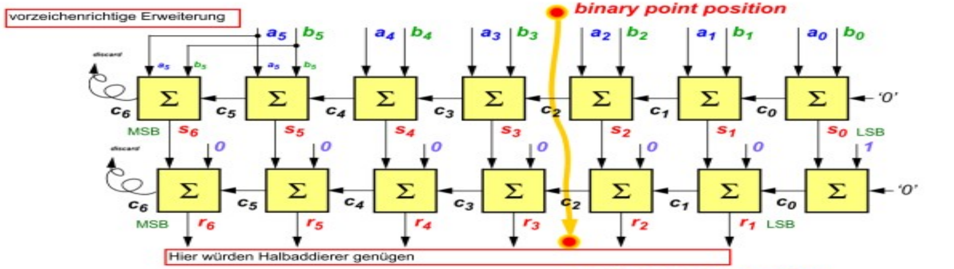
Overflow:
( ) Wrap [✓ Saturate] ( ) Flag as error

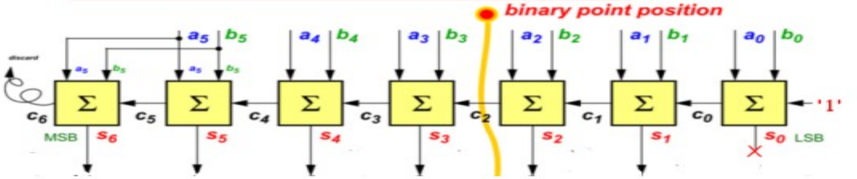
**Runden**

*In this example,* (FPGAArithmetic_2.pdf) *two 6-bit numbers are added (each with 3 integer and 3 fractional bits, [3:3]), then the answer is rounded to [4:2] …*

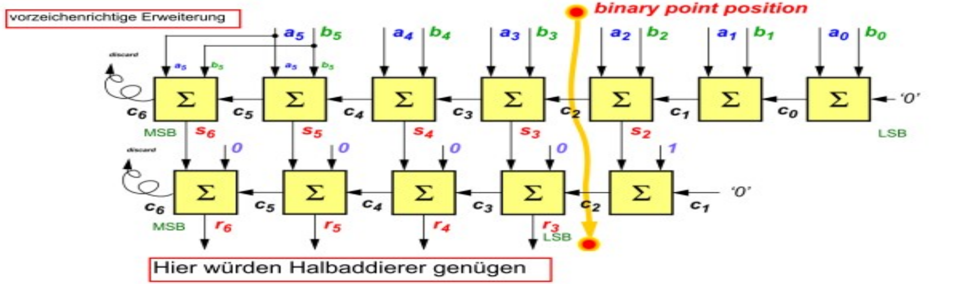
*Observe that incorporating rounding doubles the cost in this case! The bottom row of adders is entirely due to rounding.*

Optimierung der Schaltung:

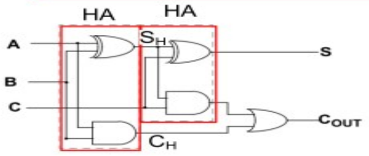

vorzeichenrichtige Erweiterung
binary point position
$a_5$ $b_5$ $a_4$ $b_4$ $a_3$ $b_3$ $a_2$ $b_2$ $a_1$ $b_1$ $a_0$ $b_0$
discard
Σ
$c_6$ $c_5$ $c_4$ $c_3$ $c_2$ $c_1$ $c_0$
'0'
MSB $s_6$ $s_5$ $s_4$ $s_3$ $s_2$ $s_1$ $s_0$ LSB
0 0 0 0 0 0 1
'0'
MSB $r_6$ $r_5$ $r_4$ $r_3$ $r_2$ $r_1$ LSB
Hier würden Halbaddierer genügen

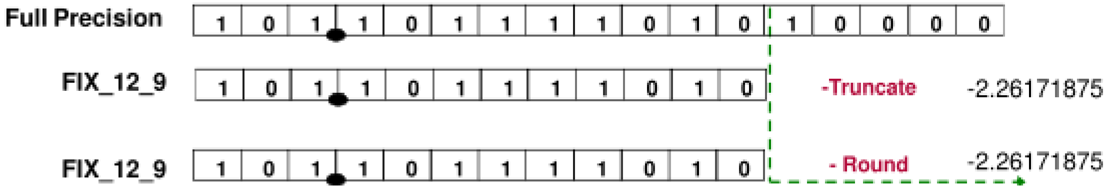

binary point position
$a_5$ $b_5$ $a_4$ $b_4$ $a_3$ $b_3$ $a_2$ $b_2$ $a_1$ $b_1$ $a_0$ $b_0$
discard
Σ
$c_6$ $c_5$ $c_4$ $c_3$ $c_2$ $c_1$ $c_0$
'1'
MSB $s_6$ $s_5$ $s_4$ $s_3$ $s_2$ $s_1$ $s_0$ LSB

*In this example,* (FPGAArithmetic_2.pdf) *two 6-bit numbers are added (each with 3 integer and 3 fractional bits, [3:3]), then the answer is rounded to [4:0] …*


vorzeichenrichtige Erweiterung
binary point position
$a_5$ $b_5$ $a_4$ $b_4$ $a_3$ $b_3$ $a_2$ $b_2$ $a_1$ $b_1$ $a_0$ $b_0$
discard
Σ
$c_6$ $c_5$ $c_4$ $c_3$ $c_2$ $c_1$ $c_0$
'0'
MSB $s_6$ $s_5$ $s_4$ $s_3$ $s_2$ LSB
0 0 0 0 1
'0'
MSB $r_6$ $r_5$ $r_4$ $r_3$ LSB
Hier würden Halbaddierer genügen

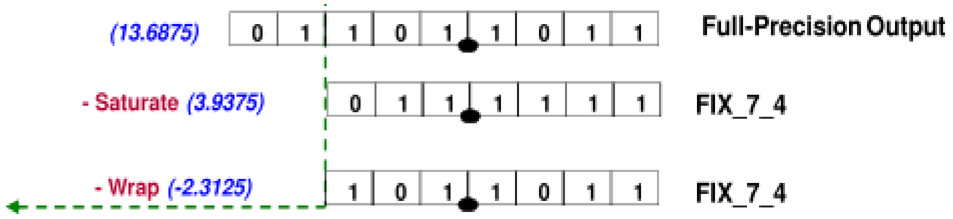
Volladdierer aus zwei Halbaddierern



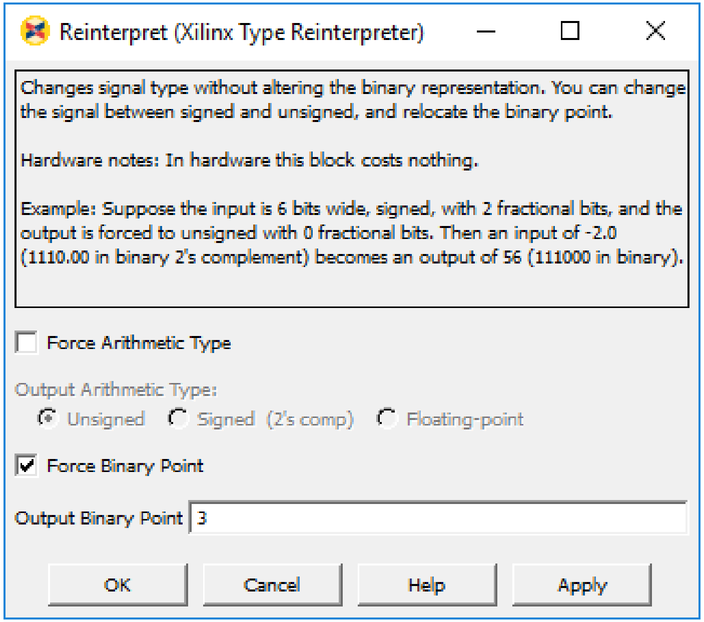


## Overflow

– Saturate to the largest positive (or maximum negative) value
– Wrap the value (that is, discard any significant bits beyond the most significant bit in the fixed-point number)
– Flag an overflow as a Simulink error during simulation



force
Reinterpret



Changes signal type without altering the binary representation. You can change the signal between signed and unsigned, and relocate the binary point.

Hardware notes: In hardware this block costs nothing.

Example: Suppose the input is 6 bits wide, signed, with 2 fractional bits, and the output is forced to unsigned with 0 fractional bits. Then an input of -2.0 (1110.00 in binary 2's complement) becomes an output of 56 (111000 in binary).

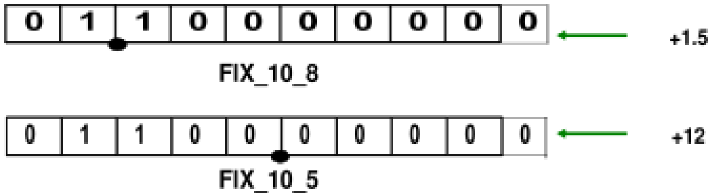
☐ Force Arithmetic Type

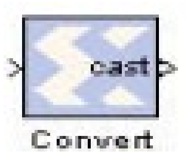
Output Arithmetic Type: [✓ Unsigned] Signed (2's comp) Floating-point

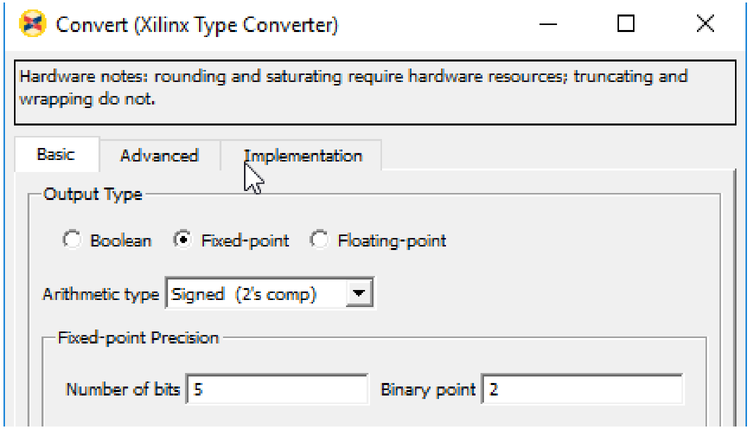
☑ Force Binary Point

Output Binary Point: 3

Der Bitvektor bleibt unverändert.
Der Wert ändert sich durch die
neue Interpretion.







Hardware notes: rounding and saturating require hardware resources; truncating and wrapping do not.

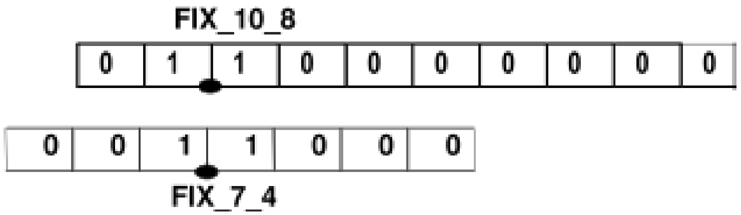
Output Type: Boolean [✓ Fixed-point] Floating-point

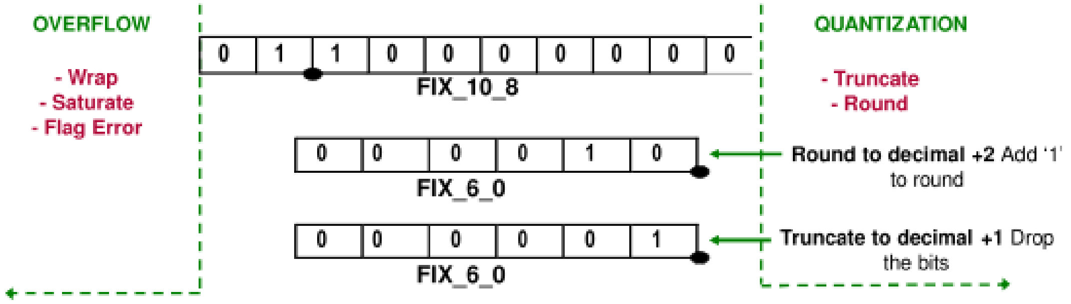
Arithmetic type: Signed (2's comp)

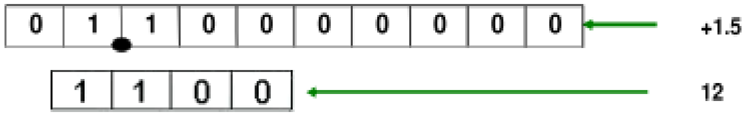
Fixed-point Precision

Number of bits: 5 Binary point: 2

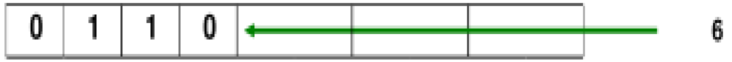
Format ändern, Wert erhalten.
(soweit dies möglich ist,
sonst wird gerundet oder abgeschnitten)



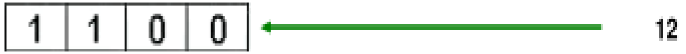




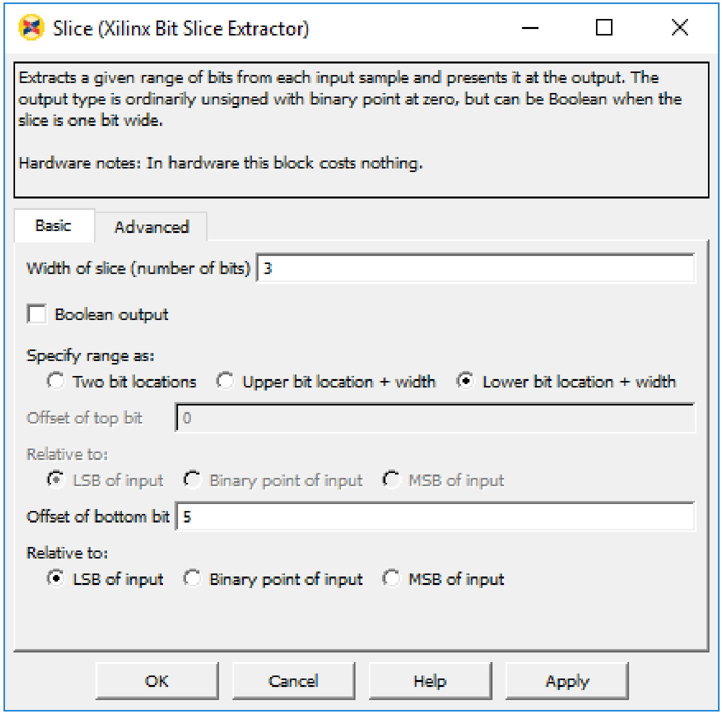
FIX_10_8 → four-bit slice offset bottom bit of the slice by five bits



Upper-bit location + width: Offset of top bit from MSB = 0 and width = 4



Two-bit locations: Offset of top bit from MSB of input = -1 and offset of bottom bit from LSB of input = 5



Slice (Xilinx Bit Slice Extractor)

Extracts a given range of bits from each input sample and presents it at the output. The output type is ordinarily unsigned with binary point at zero, but can be Boolean when the slice is one bit wide.

Hardware notes: In hardware this block costs nothing.

Basic | Advanced

Width of slice (number of bits): 3

[ ] Boolean output

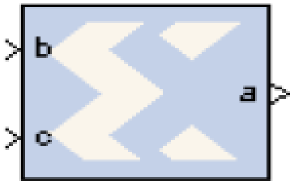
Specify range as: ( ) Two bit locations ( ) Upper bit location + width [✓ Lower bit location + width]

Offset of top bit: 0

Relative to: [✓ LSB of input] ( ) Binary point of input ( ) MSB of input

Offset of bottom bit: 5

Relative to: [✓ LSB of input] ( ) Binary point of input ( ) MSB of input



BitBasher6

BitBasher
Bit manipulation and augmentation through textual specification
Based on Verilog syntax

- Concat inputs b,d,e and f :
  a = {b,d,e,f}
  Input b forms the msb’s of output a and f forms the lsb’s of output a

- Slicing
  a = {b[17:7]}
  Bits are numbered from 0(lsb) – bit_width-1(msb)

- Using constants
  – a= {4’o14, 4’hf, 3’b110, 5’d22,b}
  – 4’o14 represents an octal constant of bit-width 4 and octal value 14
  – b,d,h represent binary, decimal and hex respectively
  – Must have at least one variable

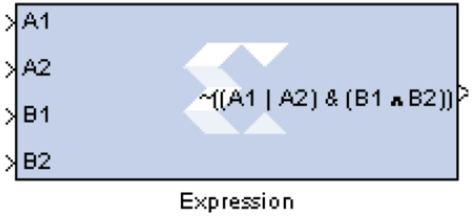
- Bit reversal
  a = {b[0:7]}
  B input is assumed to be 8 bits wide

- Repeating
  a = {4{b,d}}
  4 represents the repeat factor for the enclosed expression {b,d} and is equivalent to {b,d,b,d,b,d,b,d}

- Multiple output
  – New-line is used as a separator of each output expression

## Expression Block (bitwise logical expression)

– And - & (highest precedence)
– Or - |
– Not - ~
– Xor - ^ (lowest precedence)
– Precedence can be changed by using parenthesis



~((A1 | A2) & (B1 ^ B2))

## Knowledge Check

What is quantization? Why does it occur? State the two options available to handle it

– Quantization is a process of handling higher-precision number representation with a lower-precision number representation
– In the Simulink tool, the numbers are represented in double-precision; whereas in the Xilinx blockset, the numbers are represented in fixed point
– Truncate and Rounding are the two available options to handle it

What is an overflow? Why does it occur? State the three options available to handle

– An overflow occurs when a large number is represented in a smaller range representation
– In the Simulink tool, the numbers are represented in double-precision; whereas in the Xilinx blockset, the numbers are represented in fixed point
– Saturate, Wrap the value, and Flag an Error are the three available options

What is the purpose of HDL wrapper? ( *to wrap* = ‚wickeln‘, ‚einhüllen‘;wrapper = ‚Verpackung', ‚Umschlag'; in der Informationstechnik ein Stück Software, welches ein anderes Stück Software umgibt )

– Extend the IP core/block functionality, allowing various data types

– Simplify the IP core interface, providing only necessary ports on a block

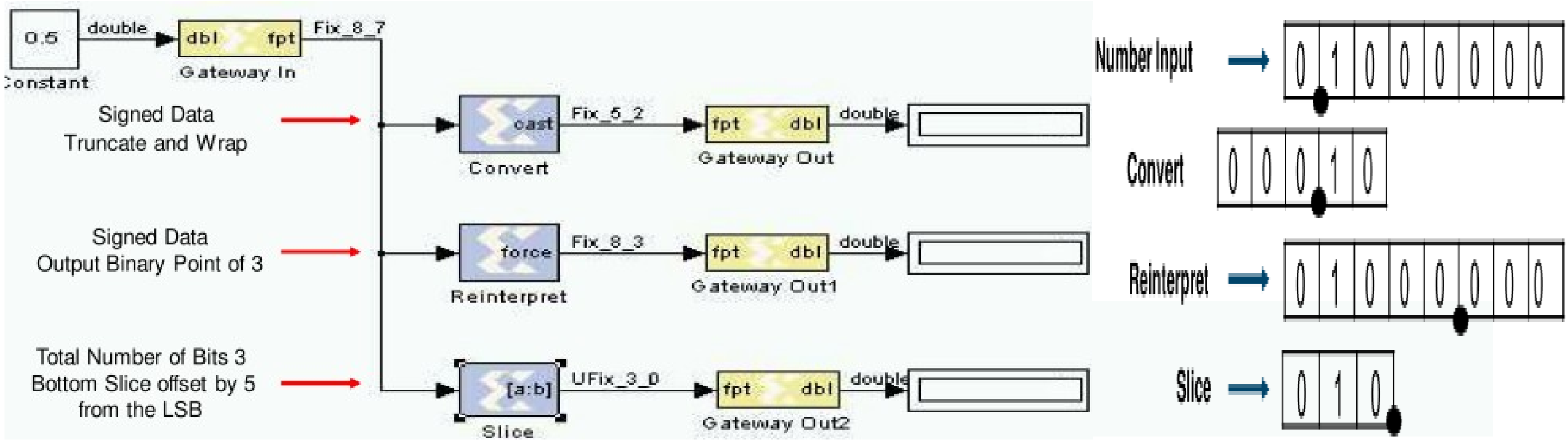
  • For example, DSP48 macro

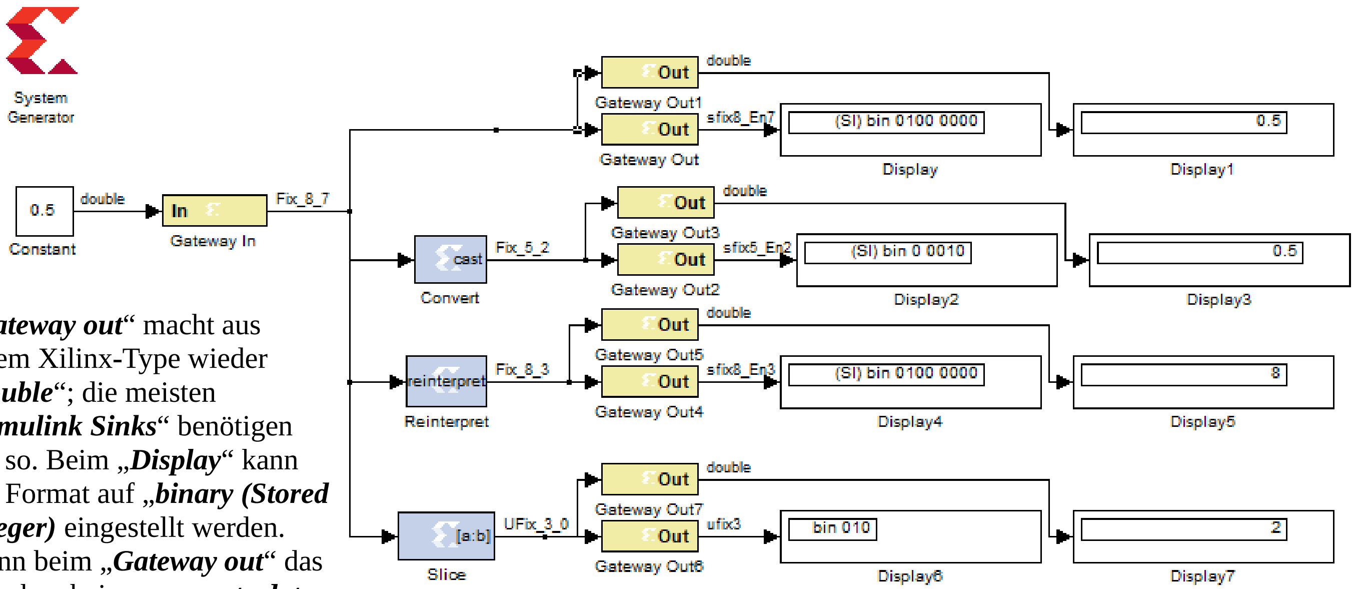
Why do you need bit picking? State the four blocks available for this purpose

– There may be a need to:

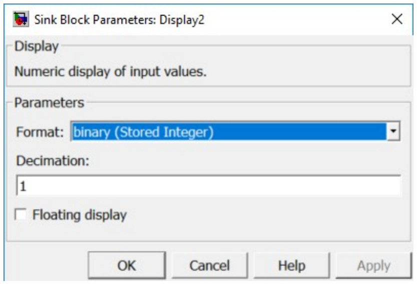
- Combine two data buses together to form a new bus
- Force a conversion of data type, including the number of bits and binary bits
- Reinterpret unsigned data as signed or the converse
- Extract certain bits of data, especially when there is bit growth

– The four blocks available are: Concat, Convert, Reinterpret, Slice





„***Gateway out***“ macht aus einem Xilinx-Type wieder „***double***“; die meisten „***Simulink Sinks***“ benötigen das so. Beim „***Display***“ kann das Format auf „***binary (Stored Integer)***“ eingestellt werden. Wenn beim „***Gateway out***“ das Häkchen bei: „***propagate data type to output***“ gesetzt wird, ergibt dies eine Binär-Darstellung (Nachkomma-stellen können mit „***Display*** → ***Post Data Types***“ ermittelt werden; ***sfix8_En3*** bedeutet ***8*** Bit insgesamt, davon ***3*** nach dem Komma).


Sink Block Parameters: Display2

Display

Numeric display of input values.

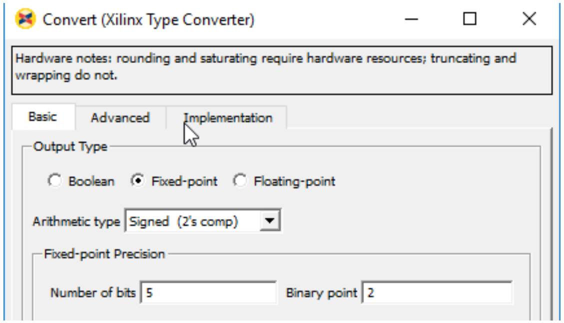
Parameters

Format: binary (Stored Integer)

Decimation:

1

☐ Floating display


Convert (Xilinx Type Converter)

Hardware notes: rounding and saturating require hardware resources; truncating and wrapping do not.

Output Type

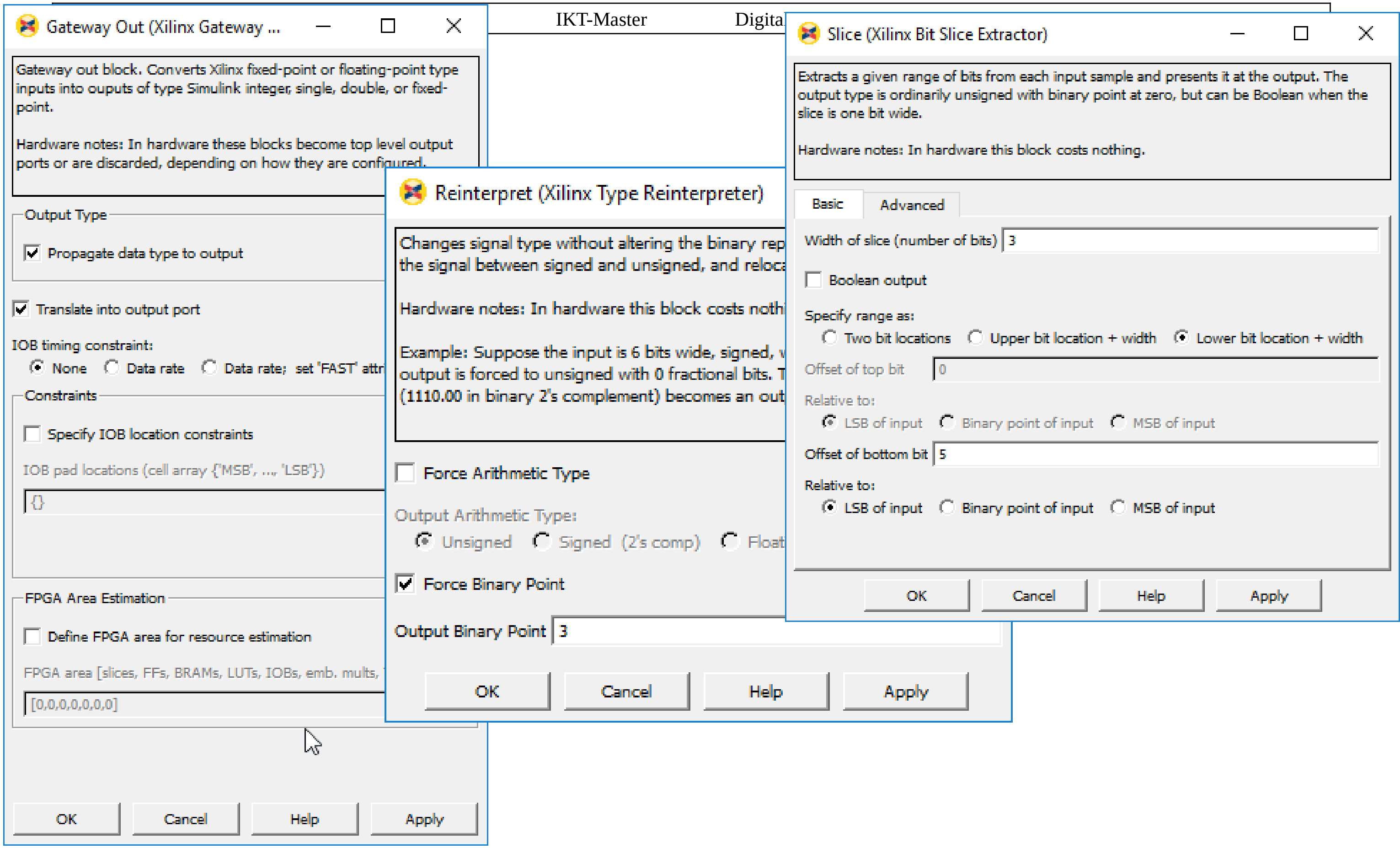
Boolean [✓ Fixed-point] Floating-point

Arithmetic type: Signed (2's comp)

Fixed-point Precision

Number of bits: 5

Binary point: 2



Gateway Out (Xilinx Gateway ...

Gateway out block. Converts Xilinx fixed-point or floating-point type inputs into ouputs of type Simulink integer, single, double, or fixed-point.

Hardware notes: In hardware these blocks become top level output ports or are discarded, depending on how they are configured.

Output Type

[✓] Propagate data type to output

[✓] Translate into output port

IOB timing constraint: [✓ None] Data rate Data rate; set 'FAST' attri

Constraints

[ ] Specify IOB location constraints

IOB pad locations (cell array {'MSB', ..., 'LSB'}) {}

FPGA Area Estimation

[ ] Define FPGA area for resource estimation

FPGA area [slices, FFs, BRAMs, LUTs, IOBs, emb. mults, [0,0,0,0,0,0,0]

Reinterpret (Xilinx Type Reinterpreter)

Changes signal type without altering the binary rep the signal between signed and unsigned, and reloca

Hardware notes: In hardware this block costs nothi

Example: Suppose the input is 6 bits wide, signed, w output is forced to unsigned with 0 fractional bits. T (1110.00 in binary 2's complement) becomes an out

[ ] Force Arithmetic Type

Output Arithmetic Type: [✓ Unsigned] Signed (2's comp) Float

[✓] Force Binary Point

Output Binary Point 3

Slice (Xilinx Bit Slice Extractor)

Extracts a given range of bits from each input sample and presents it at the output. The output type is ordinarily unsigned with binary point at zero, but can be Boolean when the slice is one bit wide.

Hardware notes: In hardware this block costs nothing.

Basic | Advanced

Width of slice (number of bits) 3

[ ] Boolean output

Specify range as: Two bit locations Upper bit location + width [✓ Lower bit location + width]

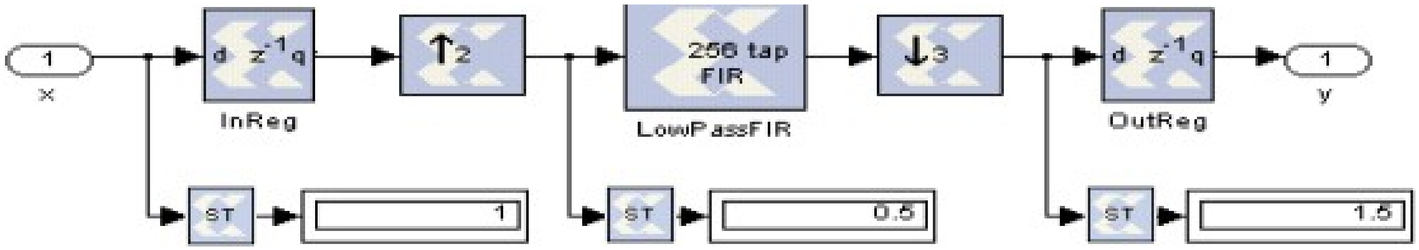
Offset of top bit 0

Relative to: [✓ LSB of input] Binary point of input MSB of input

Offset of bottom bit 5

Relative to: [✓ LSB of input] Binary point of input MSB of input

## 07_Multirate_Systems.pdf



| Block Output | x | Up Sample | Down Sample |
|---|---|---|---|
| Sample Period | | | |
| Sample Period (GCD) | | | |

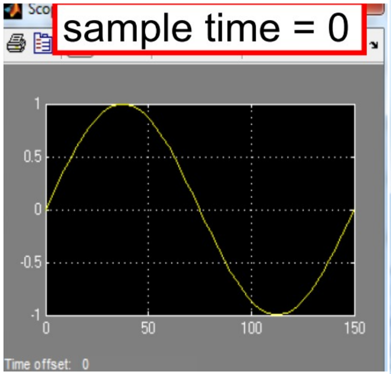
**Simulink System Period:**

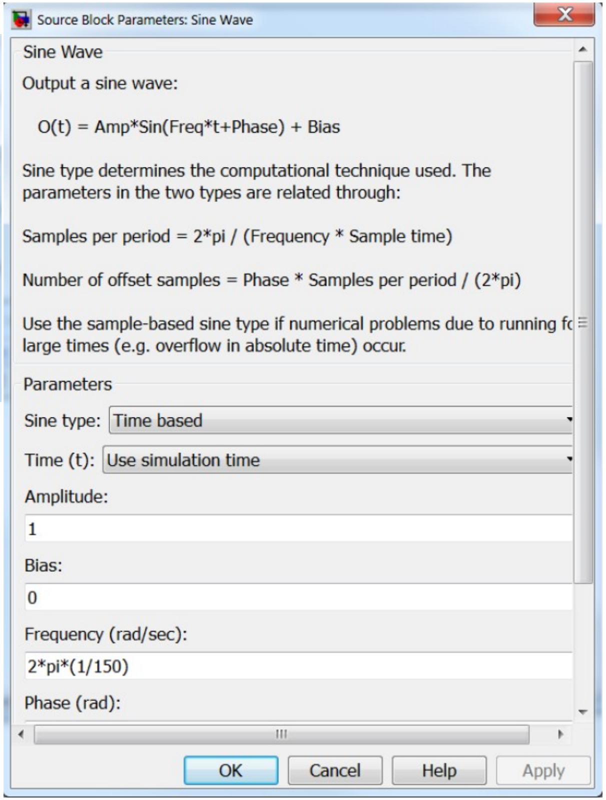
# LAB 1 Simulink

**Step 1**: Introduction to Simulink

→ Start MATLAB via System Generator (menu or icon)

Create a new model, Simulink Library → Sources : Sine Wave, → Sinks : Scope
Sine Wave: frequency of 2*pi*(1/150);
Enable port data display;
simulation stop time = 150





Source Block Parameters: Sine Wave

Sine Wave

Output a sine wave:

O(t) = Amp*Sin(Freq*t+Phase) + Bias

Sine type determines the computational technique used. The parameters in the two types are related through:

Samples per period = 2*pi / (Frequency * Sample time)

Number of offset samples = Phase * Samples per period / (2*pi)

Use the sample-based sine type if numerical problems due to running fo large times (e.g. overflow in absolute time) occur.

Parameters

Sine type: Time based

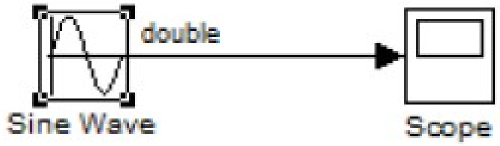
Time (t): Use simulation time

Amplitude: 1

Bias: 0

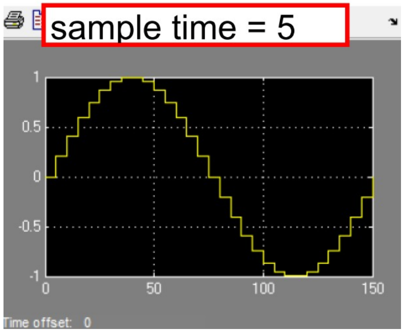
Frequency (rad/sec): 2*pi*(1/150)

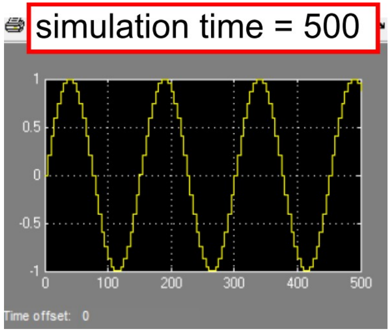
Phase (rad):



C:\all_CD_content_files\Digitale_SystemeSS19\labs\lab1

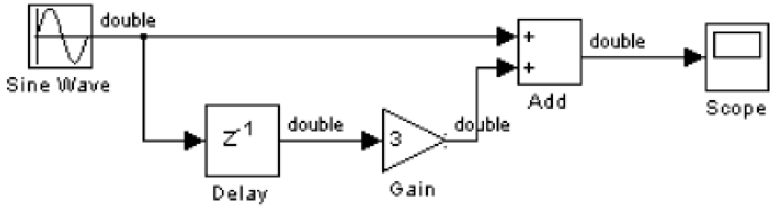
**Step 2:** Analyze the effect of the sampling time
Sine wave: sample time = 5;
simulation time = 500;
simulation time = 150;

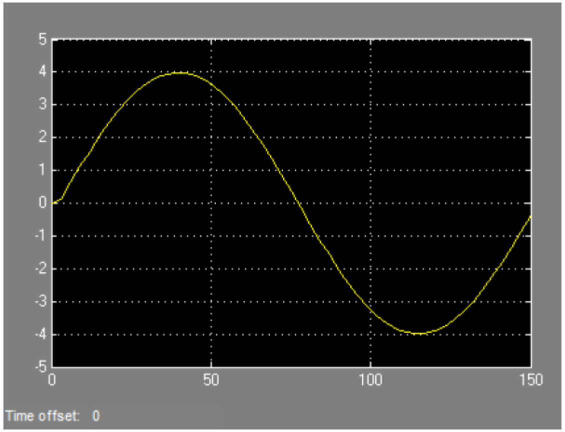




**Step 3:** Create a simple filter design using Simulink Blocks
**Sine wave: sample time = 0; simulation time = 150;**

**$y(n)= x(n) + 3 * x(n-1)$ $Y(z)= X(z) + 3*z^{-1} *X(z) = X(z)*(1 + 3*z^{-1})$**

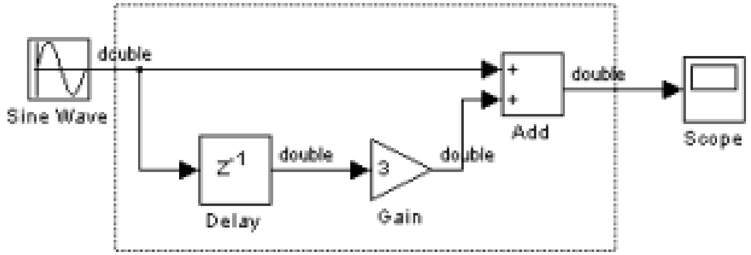


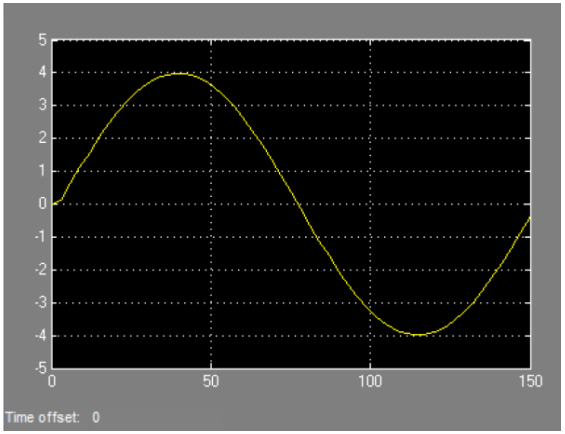


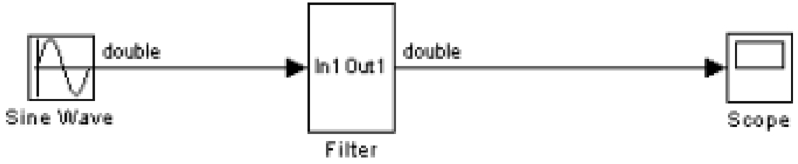
**Step 4:** Create a subsystem

Edit > Create Subsystem

Alternately: Ctlr+G

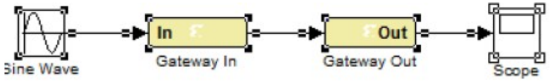






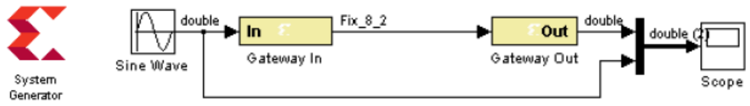
# LAB 2 12 x 8 MAC Using the Xilinx System Generator Lab

Multiplier input data widths of 12 bits and 8 bits of signed data, output width of 20 bits;
Accumulator output width of 27-bits



**Step 1:** Introduce the Xilinx Blockset
Sine Wave (set frequency to 2*pi*1/150)
Gateway In ( Number of bits = 8, Binary Point = 2, Quantization =round, overflow = saturate)

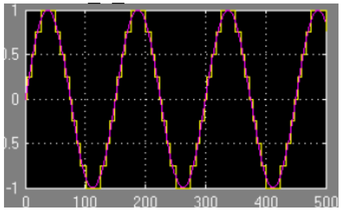
MUX between the Gateway Out and the Scope
system generator token,
select Wide Nonscalar Lines, Signal dimensions,
Port data types under the Format > Port/Signal Displays
Edit > Update Diagram



D:\all_CD_content\Xilinx_DSP_sysgen14\labsource\labs\labs\lab2\lab2.mdl

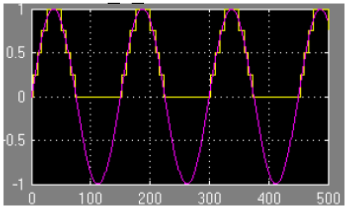
**Step 2:** Evaluate the precision and analyze the effect on output (fix_8_2)
Simulation > Configuration Parameters (stop time =500); run

Fix_8_2

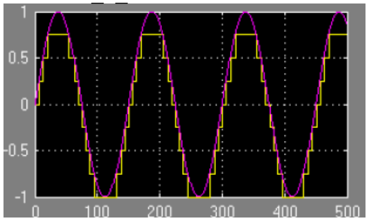


Gateway In,
Output Data
Type =
Unsigned
Quantization =
round

UFix_8_2



Fix_8_2



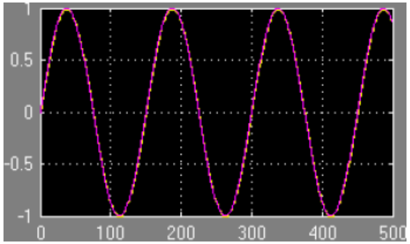
Gateway In,
Output Data
Type = Signed
Quantization =
**truncate**

| Gewichte | -32 | 16 | 8 | 4 | 2 | 1 | . | ½ | ¼ |
|---|---|---|---|---|---|---|---|---|---|
| Fix_8_2: | 0 | 0 | 0 | 0 | 0 | 0 | . | 0 | 0 |
| Min: | 1 | 0 | 0 | 0 | 0 | 0 | . | 0 | 0 |
| Max: | 0 | 1 | 1 | 1 | 1 | 1 | . | 1 | 1 |

| Gewichte | 32 | 16 | 8 | 4 | 2 | 1 | . | ½ | ¼ |
|---|---|---|---|---|---|---|---|---|---|
| UFix_8_2: | 0 | 0 | 0 | 0 | 0 | 0 | . | 0 | 0 |
| Min: | 0 | 0 | 0 | 0 | 0 | 0 | . | 0 | 0 |
| Max: | 1 | 1 | 1 | 1 | 1 | 1 | . | 1 | 1 |

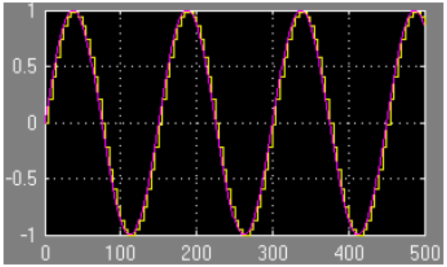
Fix_8_6

Gateway In,
Output Data
binary point = 6
fix_8_6
Reduce
quantization
error

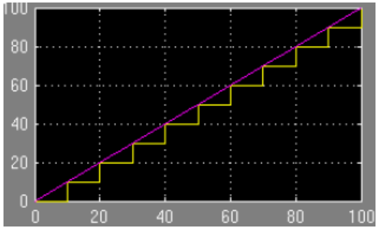


Fix_8_6

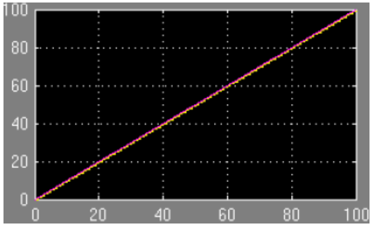
Gateway In,
sample period = 5
fix_8_6



Replace the sine wave source with the ramp
Stop Time to 100
Gateway In: Binary Point = 0 ,
(fix_8_0)
Sample Period=10



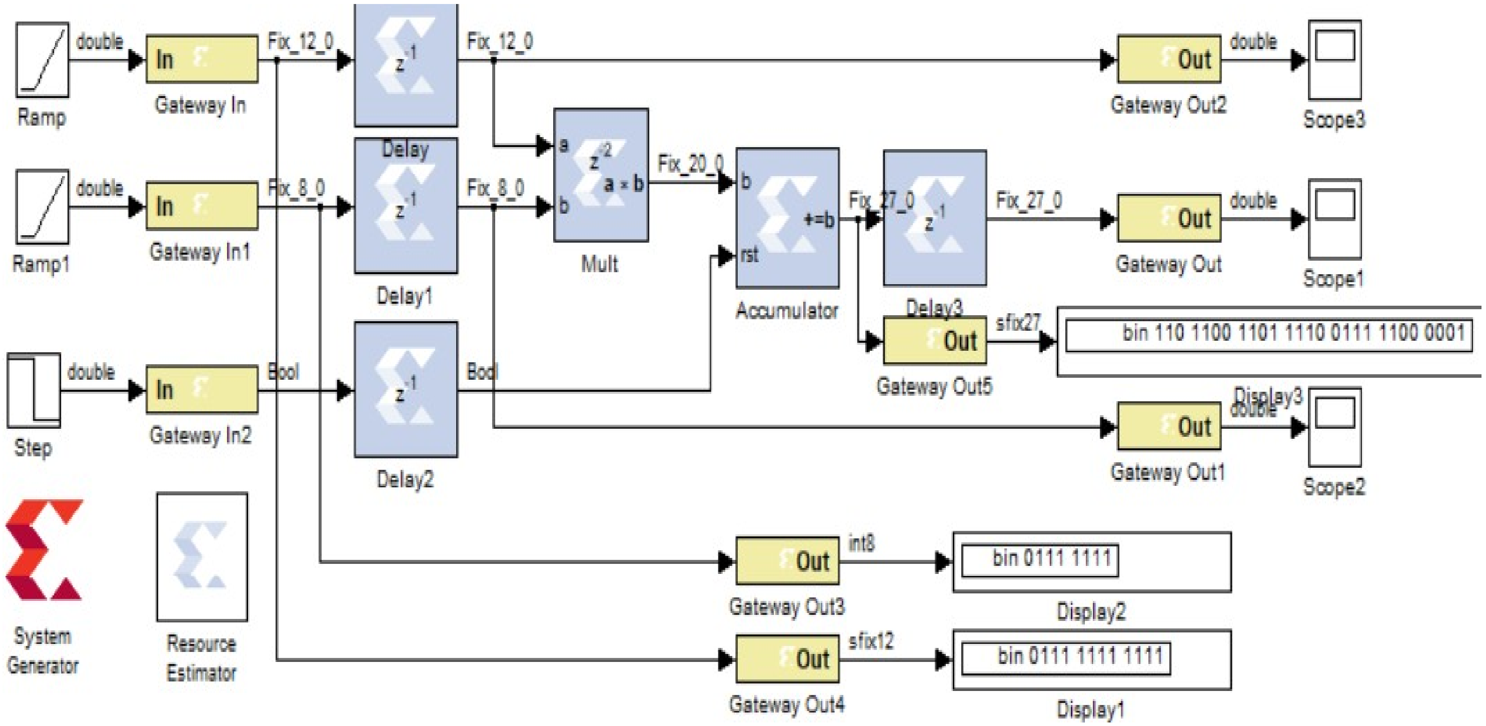
Gateway In,
sample period = 1



## LAB 2 Step 3: Designing a 12 x 8 MAC

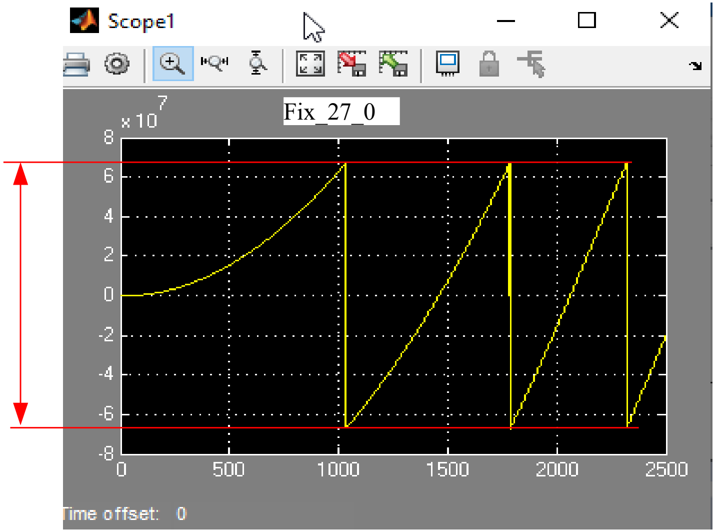
C:\all_CD_content\Xilinx_DSP_sysgen14.7\labsource\labs\labs\lab2\lab2_2.mdl

„Gateway Out Option“ zur Darstellung der Binärformate, siehe S. 11



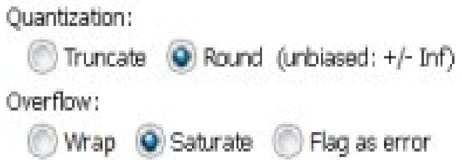
## LAB 2 Step 4: Configure and Simulate the 12 x 8 MAC

- o 12-bit input: signed data, binary point 0, sampling period 1
- o 8-bit input: signed data, binary point 0, sampling period 1
- o 3 rd input: Boolean type
- o Gateway Out: "Translate in to output port" box checked
- o Multiplier: Latency set to 3
- o Accumulator: output width of 27 bits, overflow method to wrap
- o Simulation parameters: stop time to 2500



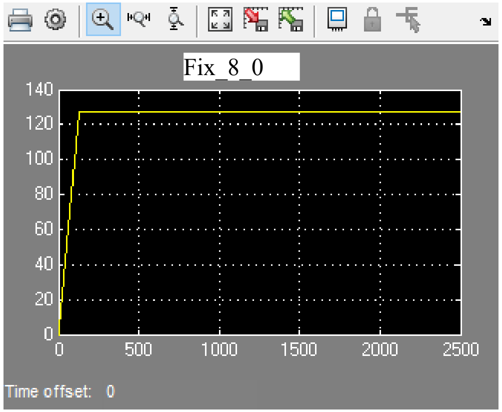
Q1,Q2 (L=2):
At what time the first transition (sharp) occurs?
Antwort: 1037,
Q: Min-Wert ? Max-Wert ?
Begründung ?
2^26=67.108.864

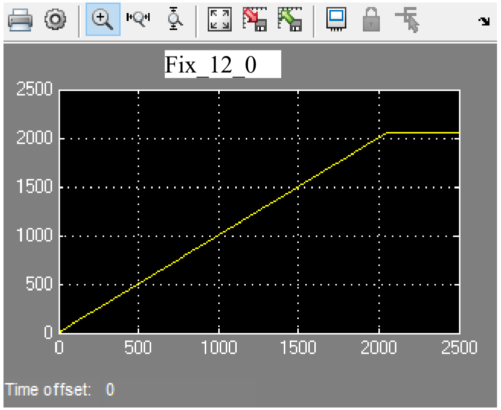
Gateway In Options



Quantization:
( ) Truncate (●) Round (unbiased: +/- Inf)

Overflow:
( ) Wrap (●) Saturate ( ) Flag as error





## LAB 2 **Step 5:** Estimate the Resources

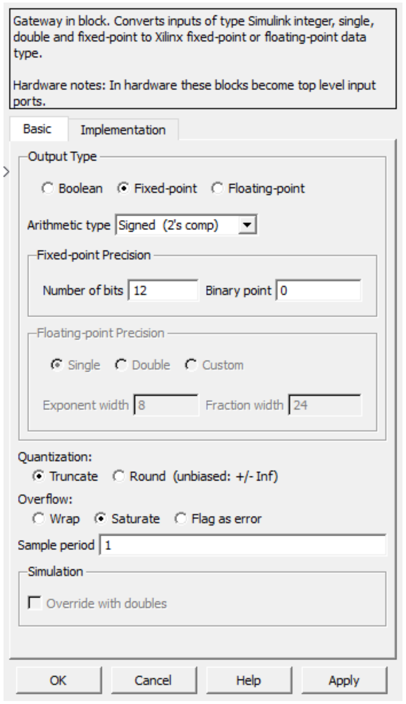
compilation target = HDL Netlist;
compilation target = HDL Netlist
Part: Spartan 6 xc 6 slx45-2csg324
Add the Resource Estimator
(from Xilinx Tools library)

Q3
multiplier latency = 2
Number of Slices:
Number of FFs :
Number of LUTS:
Number of IOBs :

Q4
multiplier latency = 3
Number of Slices:
Number of FFs :
Number of LUTS:
Number of IOBs :

Q5
multiplier latency = 2
Use Embedded Multipliers
Estimate Area
Number of Slices:
Number of FFs :
Number of LUTS:
Number of Mults/DSP48s:
Number of IOBs :

Q5
multiplier latency = 3
Use Embedded Multipliers
Estimate Area
Number of Slices:
Number of FFs :
Number of LUTS:
Number of Mults/DSP48s:
Number of IOBs :

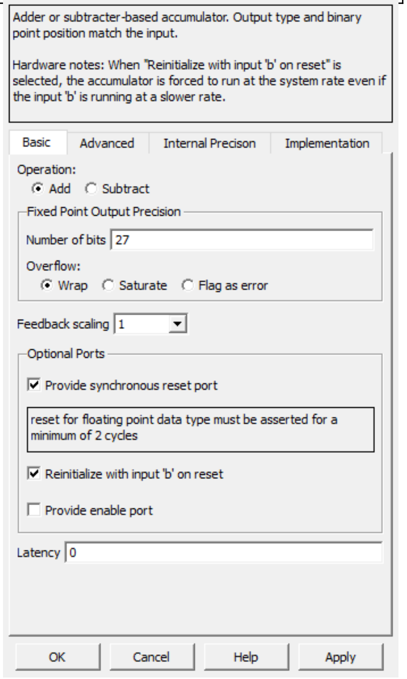


Gateway in block. Converts inputs of type Simulink integer, single, double and fixed-point to Xilinx fixed-point or floating-point data type.

Hardware notes: In hardware these blocks become top level input ports.

Basic | Implementation

Output Type: Boolean [✓ Fixed-point] Floating-point
Arithmetic type: Signed (2's comp)
Fixed-point Precision: Number of bits 12, Binary point 0
Floating-point Precision: [✓ Single] Double Custom
Exponent width 8, Fraction width 24
Quantization: [✓ Truncate] Round (unbiased: +/- Inf)
Overflow: Wrap [✓ Saturate] Flag as error
Sample period 1
Simulation: ☐ Override with doubles



Adder or subtracter-based accumulator. Output type and binary point position match the input.

Hardware notes: When "Reinitialize with input 'b' on reset" is selected, the accumulator is forced to run at the system rate even if the input 'b' is running at a slower rate.

Basic | Advanced | Internal Precison | Implementation

Operation: [✓ Add] Subtract
Fixed Point Output Precision: Number of bits 27
Overflow: [✓ Wrap] Saturate Flag as error
Feedback scaling 1
Optional Ports:
☑ Provide synchronous reset port
reset for floating point data type must be asserted for a minimum of 2 cycles
☑ Reinitialize with input 'b' on reset
☐ Provide enable port
Latency 0

## LAB 2 Step 6: Generate the VHDL Core

latency of the multiplier = 2
uncheck the embedded multiplier usage
System Generator icon
Click Generate

ISE Design Suite 14.7
File > Open Project and select mac_cw.xise
Highlight the top-level file, called mac_cw.vhd,
and double-click on Implement Design

- Compilation: HDL Netlist
- Part: Spartan 6 xc 6 slx45-2csg324
- Synthesis Tool: XST
- Hardware Description Language: VHDL
- Target Directory: ./ise
- Create Testbench: unchecked
- FPGA System Clock Period (ns): 10

Question 7
Open the Place and Route report file and fill in the following information
Number of Slice Registers:
Number of Slice LUTs:
**Question 8**
Open the post-PAR Static Timing report and
fill in the following information
Maximum clock frequency:

## **LAB 2** Step 6 Generate the HDL Code

6-1-1. Set the latency of the multiplier block to 2, and uncheck the embedded multiplier usage option.
6-1-2. Select Area as the goal and check Use behavioral HDL option.
6-1-3. Set the Fabric as the target and check Use behavioral HDL option of the accumulator.
6-1-4. Double-click the System Generator icon and specify the following settings.

6-1-5. Click Generate to generate the HDL code and ISE Project files.
6-1-6. Select Start > All Programs > Xilinx Design Tools > ISE Design Suite 14.3 > ISE Design Tools > Project Navigator.
6-1-7. Open the generated project by selecting File > Open Project and select mac_cw.xise in the ise project directory.
6-1-8. Highlight the top-level file, called mac_cw.vhd, and double-click on Implement Design.

Question 7

Open the Place and Route report file and fill in the following information
Number of Slice Registers: ________________________ 55
Number of Slice LUTs: ________________________ 37
Question 8
Open the post-PAR Static Timing report and fill in the following information

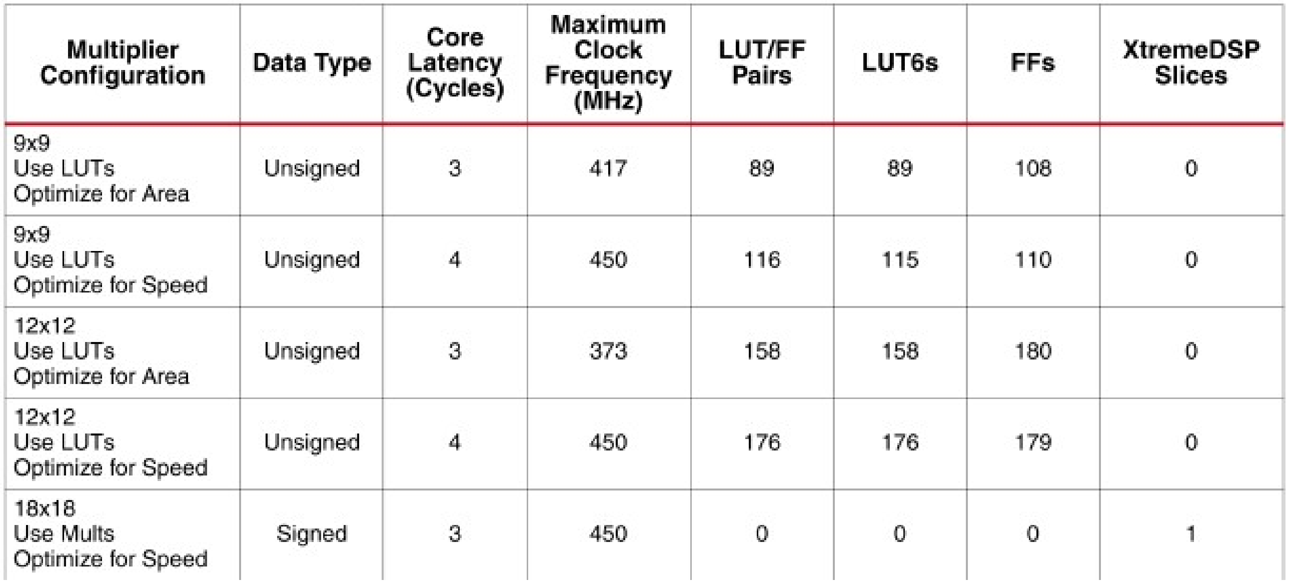
**Maximum clock frequency: ~ 255.62 MHz (3.912 ns) check embedded multiplier usage : ~ 163 MHz ??**

## mult_gen_ds255.pdf Xilinx DS255 Multiplier v11.2, Data Sheet

Features

- Drop-in module for Virtex®-7 and Kintex™-7, Virtex-6, Virtex-5, Virtex-4, Spartan®-6, **Spartan-3/XA, Spartan-3E/XA, Spartan-3A/3AN/3A DSP/XA FPGAs**

*Table 3:* **Virtex-6 FPGA Family Performance and Resource Utilization**



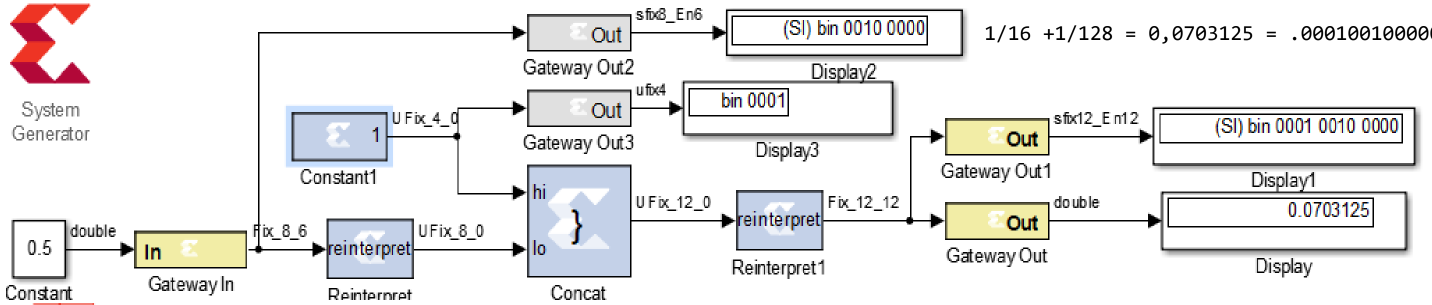
| Multiplier Configuration | Data Type | Core Latency (Cycles) | Maximum Clock Frequency (MHz) | LUT/FF Pairs | LUT6s | FFs | XtremeDSP Slices |
|---|---|---|---|---|---|---|---|
| 9x9<br>Use LUTs<br>Optimize for Area | Unsigned | 3 | 417 | 89 | 89 | 108 | 0 |
| 9x9<br>Use LUTs<br>Optimize for Speed | Unsigned | 4 | 450 | 116 | 115 | 110 | 0 |
| 12x12<br>Use LUTs<br>Optimize for Area | Unsigned | 3 | 373 | 158 | 158 | 180 | 0 |
| 12x12<br>Use LUTs<br>Optimize for Speed | Unsigned | 4 | 450 | 176 | 176 | 179 | 0 |
| 18x18<br>Use Mults<br>Optimize for Speed | Signed | 3 | 450 | 0 | 0 | 0 | 1 |

05_Signal_Routing.pdf; Signal Conversion; Bit Picking(Reinterpret, Convert, Concat, Slice); Overflow(Saturate, Wrap)
05a_Lab3_Intro.pdf ( pad the dual-port RAM data). Convert FIX_12_12 to FIX_8_6 (Slice.,Reinterprate)

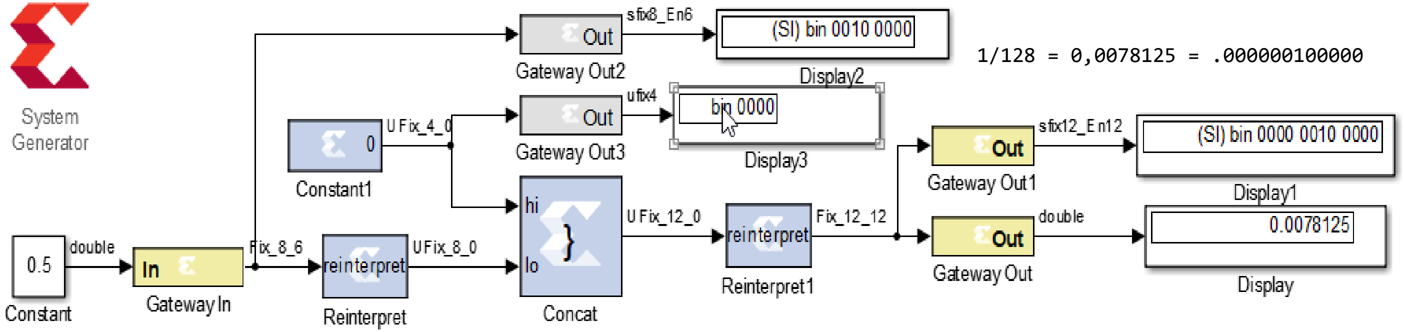
„Gateway Out Option“ zur Darstellung der Binärformate, siehe S. 11

# LAB 3 Signal Routing

## Step 1 Create the Padding Logic .._content\Xilinx_DSP_sysgen14\labsource\labs\labs\padding.mdl



1/16 +1/128 = 0,0703125 = .000100100000

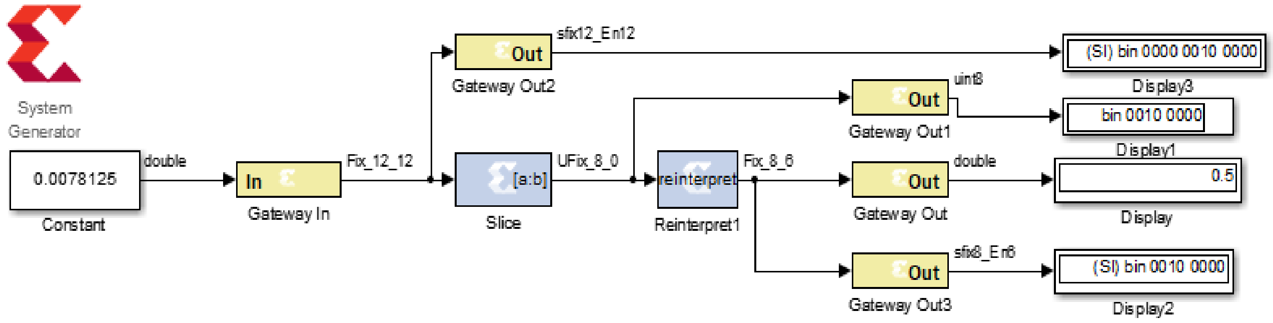


1/128 = 0,0078125 = .000000100000

Q1 Which blocks will be necessary to convert FIX_8_6 to UFIX_8_0, then to UFIX_12_0, and finally to FIX_12_12?

**Fix_8_6 → UFix_8_0** „Reinterpret“ → UFIX_12_0 Constant and Concat → FIX_12_12 Reinterpret

## Lab 3 Step 2 Create the Unpadding Logic .._content\Xilinx_DSP_sysgen14\labsource\labs\labs\unpadding.mdl

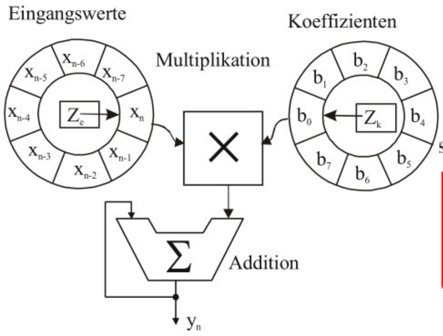


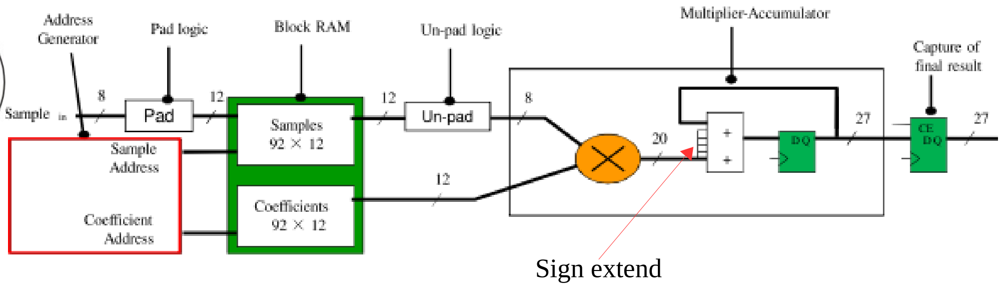
Q2:

Which blocks will be necessary to convert FIX_12_12 to UFIX_8_0 and then to FIX_8_6?

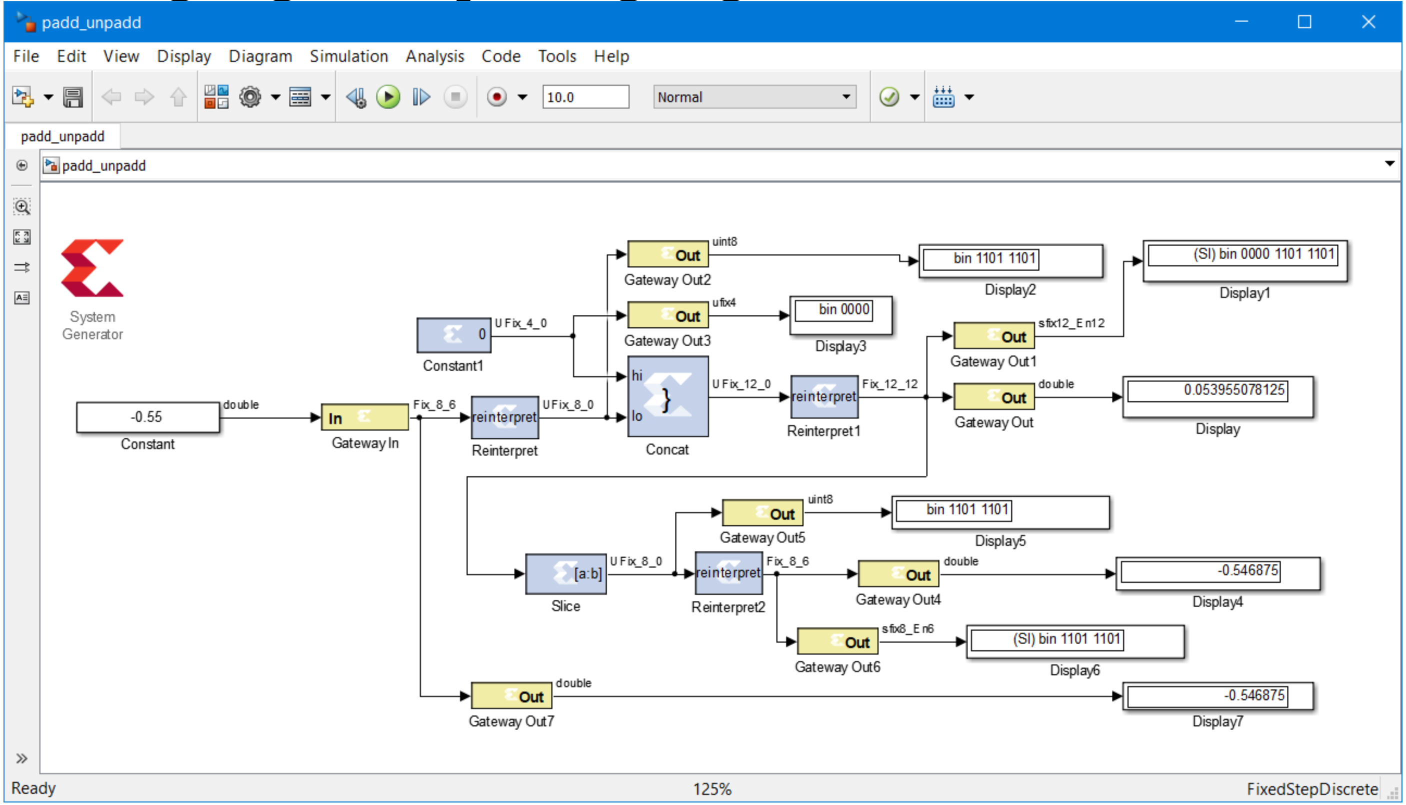
FIX_12_12 → Fix_8_0 Slice → Fix_8_6 Reinterpret

06_System_Control.pdf Control Mechanisms(Enable and Reset Ports, Valid and Invalid Ports), MCode Block (State Machines, )



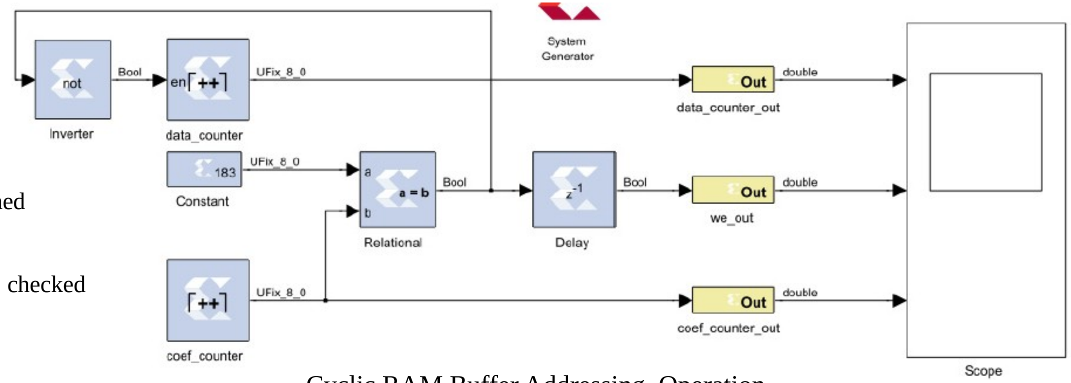


# Padding Logic – Unpadding Logic – Test with other Values



# LAB 4 Adressgenerator (Implementing System Control Lab )

Step 1 Creating the Design using Blocks .._content\Xilinx_DSP_sysgen14\labsource\labs\labs\counter_enabled.mdl



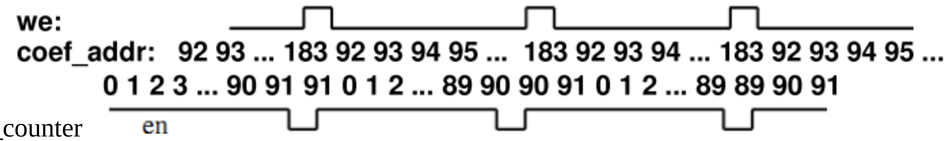
data_counter block

- Number of bits: 8
- Output type: Unsigned
- Initial value: 0
- Count to value: 91
- Provide enable port: checked

coef_counter block

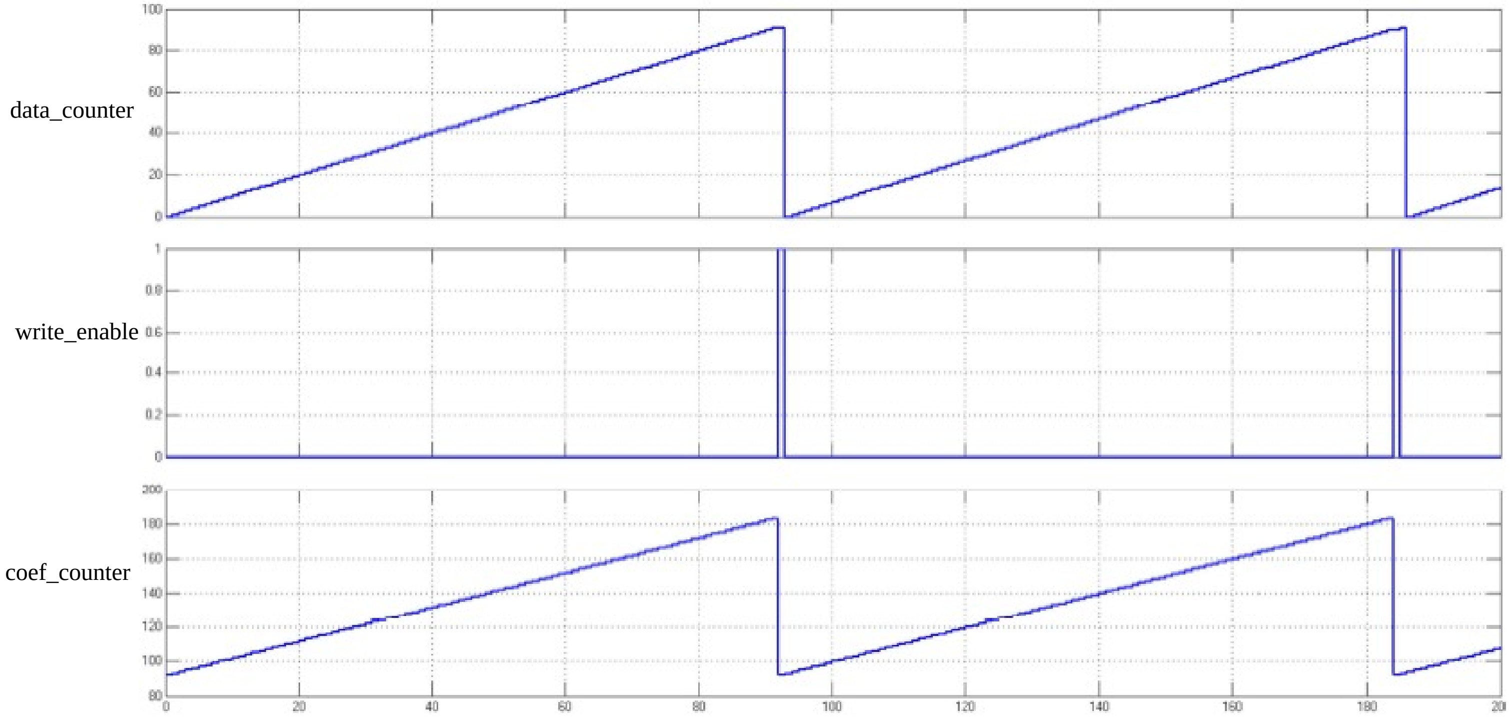
- Number of bits: 8
- Output type: Unsigned
- Initial value: 92
- Count to value: 183

Cyclic RAM Buffer Addressing, Operation
Create control logic using basic elements and M-Code

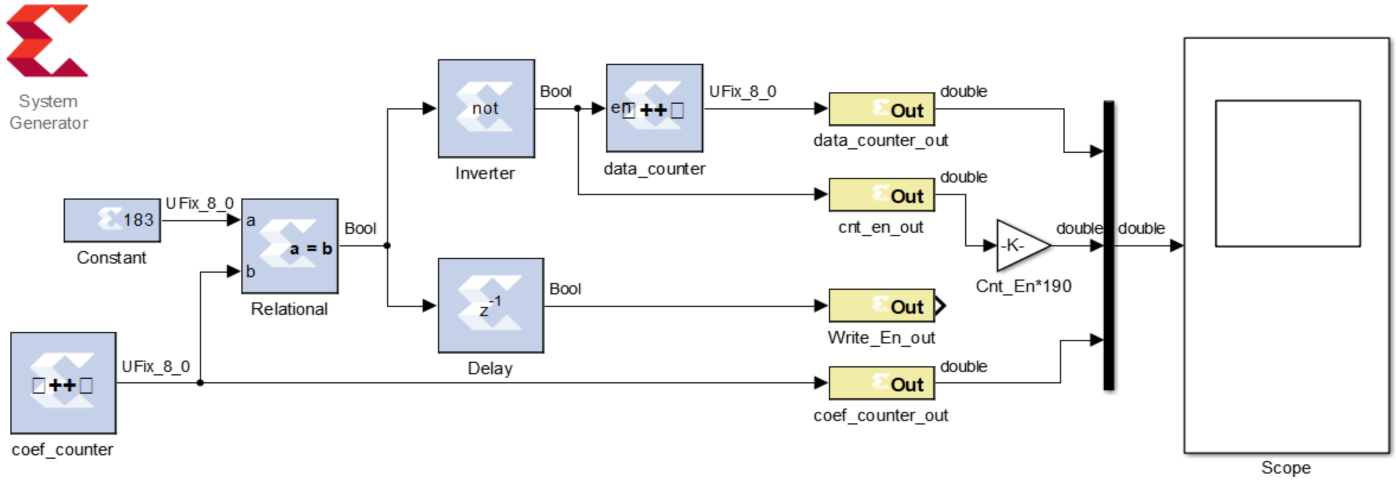

we:
coef_addr: 92 93 ... 183 92 93 94 95 ... 183 92 93 94 ... 183 92 93 94 95 ...
0 1 2 3 ... 90 91 91 0 1 2 ... 89 90 90 91 0 1 2 ... 89 89 90 91
en

en port of the data_counter

**Simulate the design for 200 and verify that the output is similar to the figure shown below:**

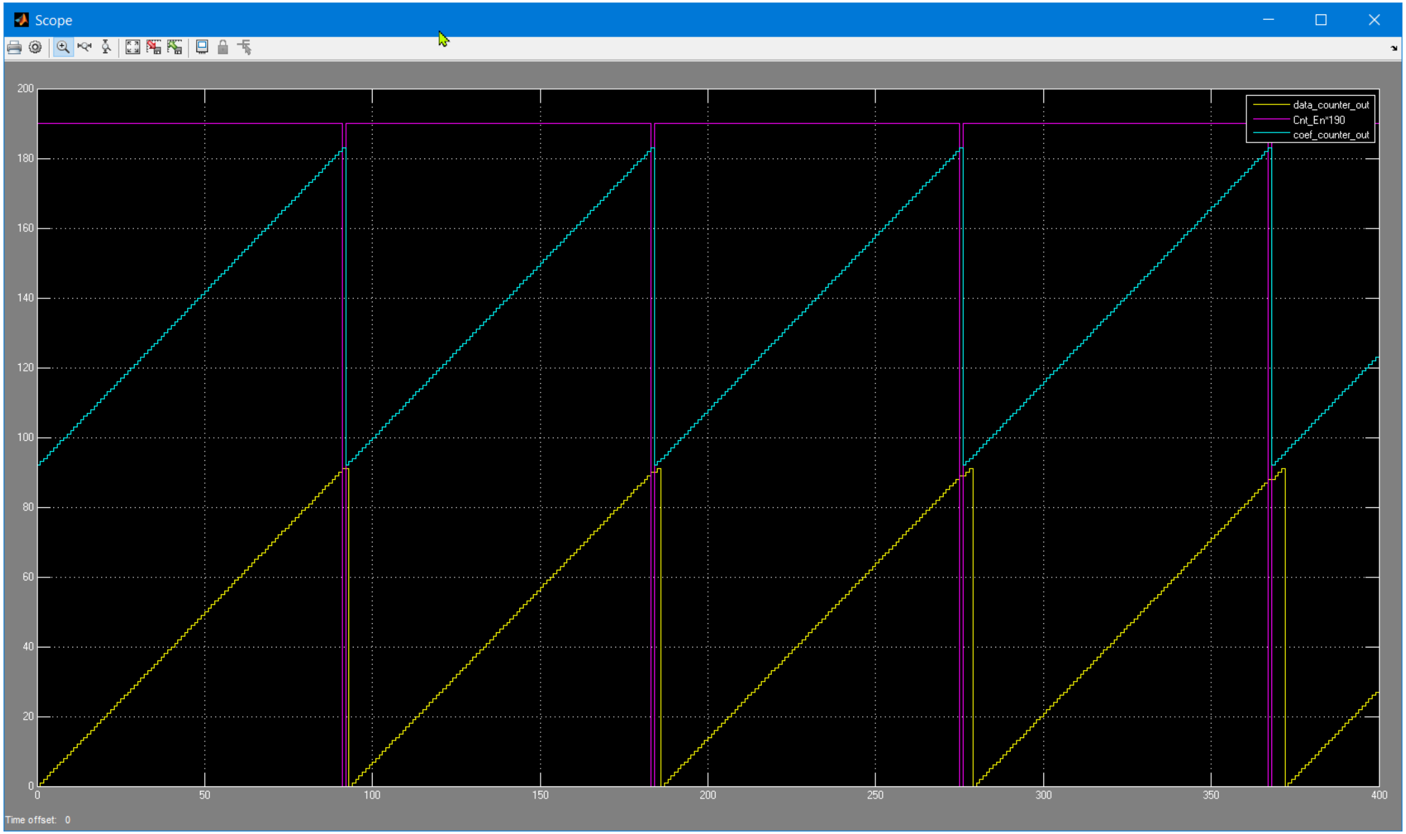

data_counter
write_enable
coef_counter
100
80
60
40
20
0
1
0.8
0.6
0.4
0.2
0
200
180
160
140
120
100
80
0
20
40
60
80
100
120
140
160
180
200

# LAB 4 Adressgenerator (zur besseren Anschaulichkeit geändertes Blockschaltbild)



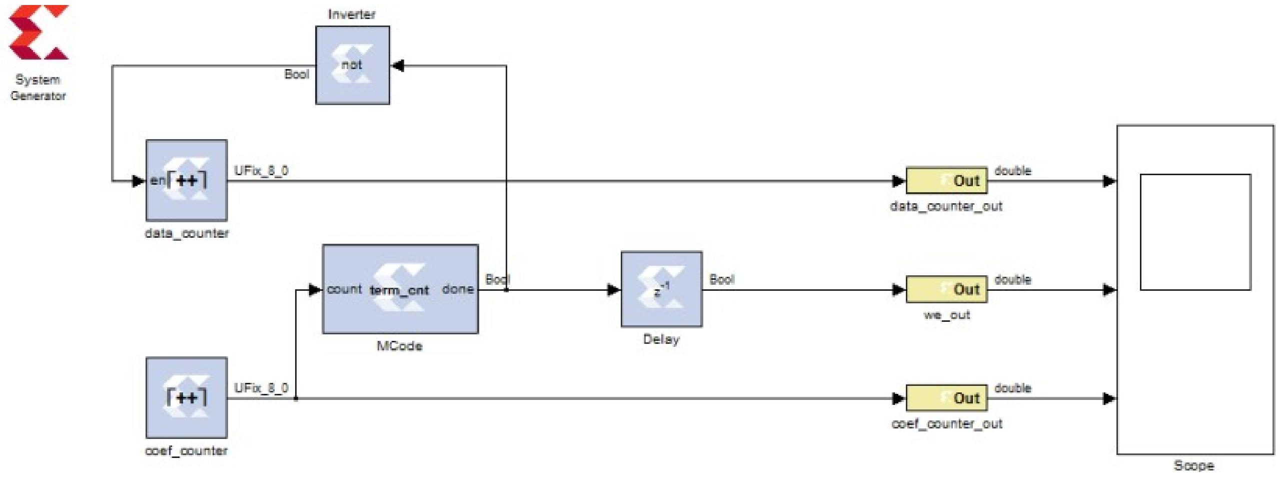
Beim geänderten Blockschaltbild wurde das *Scope* über einen Multiplexer angesteuert, so dass die drei Signale in einem Bild dargestellt werden. Dadurch wird deutlich, dass sich die *Counter-Signale* nicht überlappen. Dies ist wichtig, da es sich um einen Adressgenerator für getrennte Adress-Bereiche im gemeinsamen *Block-Memory* handelt. Das *Count-Enable-Signal* ***cnt_en*** wurde herausgeführt und mit *190* multipliziert um die Sichtbarkeit des ***Booleschen*** Signals mit (***0*** oder ***1***) zu verbessern. Es ist so deutlich erkennbar, dass ***cnt_en*** gleich ***0*** ist, wenn der ***coef_counter*** den Wert ***183*** hat und dass der ***data_counter*** dann für einen Zeitschritt angehalten wird (das Signal ***cnt_en_out*** vor der Verwendung in *Lab5* wieder entfernt werden).

# Lab 4 Scope-Bild des geänderten Blockschaltbilds



## Lab 4 Step 2 Develop an MCode Model

.._content\Xilinx_DSP_sysgen14\labsource\labs\labs\counter_enabled_mcode.mdl



Add the MCode block in the design from ***Xilinx*** Blockset > Index
Select File > New > Function from the ***MATLAB*** main window

Write an ***MCode*** function that will look at the count input and set done to true when it is equal to 183, or *else* it will set done to *false*.

Save the file as ***term_cnt.m***

Double-click the ***MCode*** block, and enter ***term_cnt*** as the function name in the Block Parameter field.

Complete the rest of the design, making sure to add a register at the output of the ***MCode*** block.

```
MCode solution
function done = term_cnt(count)
if count == 183
   done = true;
else
   done = false;
end
```