Digital and SoC Design

Course Project

Jaideep Singh Heer (M20CS056)

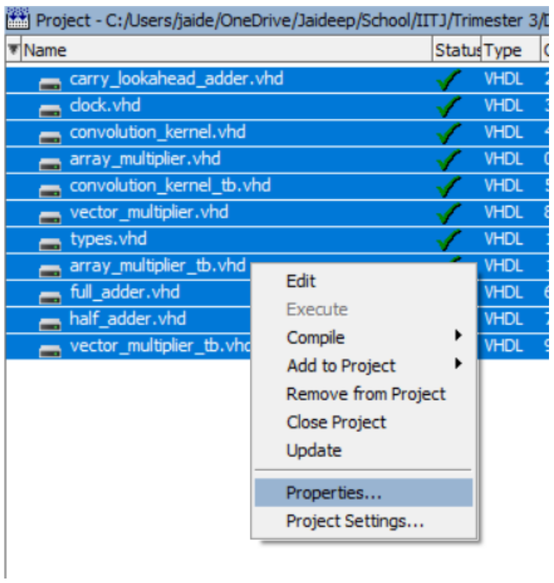
System design

The convolution kernel is designed as a combinational circuit to compute a convolution of $2^T \times 2^T$ Kernel on any input matrix $M \times M$. This is achieved using Carry Look-Ahead Adders and Array Multipliers. First, Vector Dot Product component is created using the Carry Look-Ahead Adders and Array Multipliers, then these Vector Dot Product components are used to convolve the given kernel over the input matrix by treating both the kernel and the convolution area as one long flat vector and performing a dot product of both.

Compilation Requirements

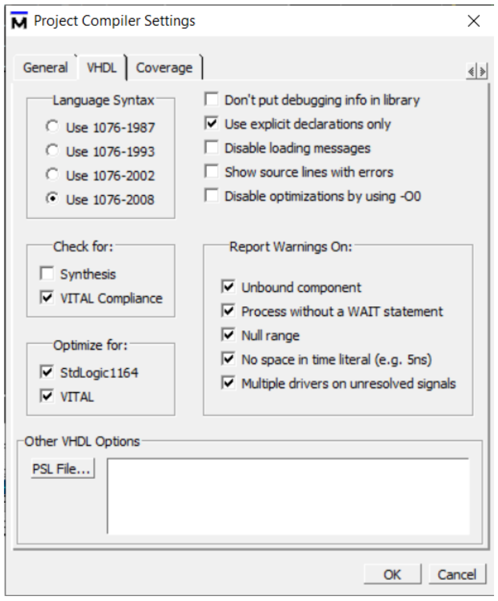
Compiling the code on ModelSim requires setting the VHDL language version to 2008 for all code files. This is due to the definition of vector_T and matrix_T data types requiring some newer features. To set the VHDL version,

- Select all files in the project.
- Right-click and click on properties.
- Under the VHDL tab, select 'Use 1076-2008' radio button for Language Syntax.
- Click OK and compile all files.



Project - C:/Users/jaide/OneDrive/Jaideep/School/IITJ/Trimester 3/D

Name	Status	Type
carry_lookahead_adder.vhd	✓	VHDL
clock.vhd	✓	VHDL
convolution_kernel.vhd	✓	VHDL
array_multiplier.vhd	✓	VHDL
convolution_kernel_tb.vhd	✓	VHDL
vector_multiplier.vhd	✓	VHDL
types.vhd	✓	VHDL
array_multiplier_tb.vhd		VHDL
full_adder.vhd		VHDL
half_adder.vhd		VHDL
vector_multiplier_tb.vhd		VHDL



Project Compiler Settings

General | VHDL | Coverage

Language Syntax
- ( ) Use 1076-1987
- ( ) Use 1076-1993
- ( ) Use 1076-2002
- (●) Use 1076-2008

- [ ] Don't put debugging info in library
- [✓] Use explicit declarations only
- [ ] Disable loading messages
- [ ] Show source lines with errors
- [ ] Disable optimizations by using -O0

Check for:
- [ ] Synthesis
- [✓] VITAL Compliance

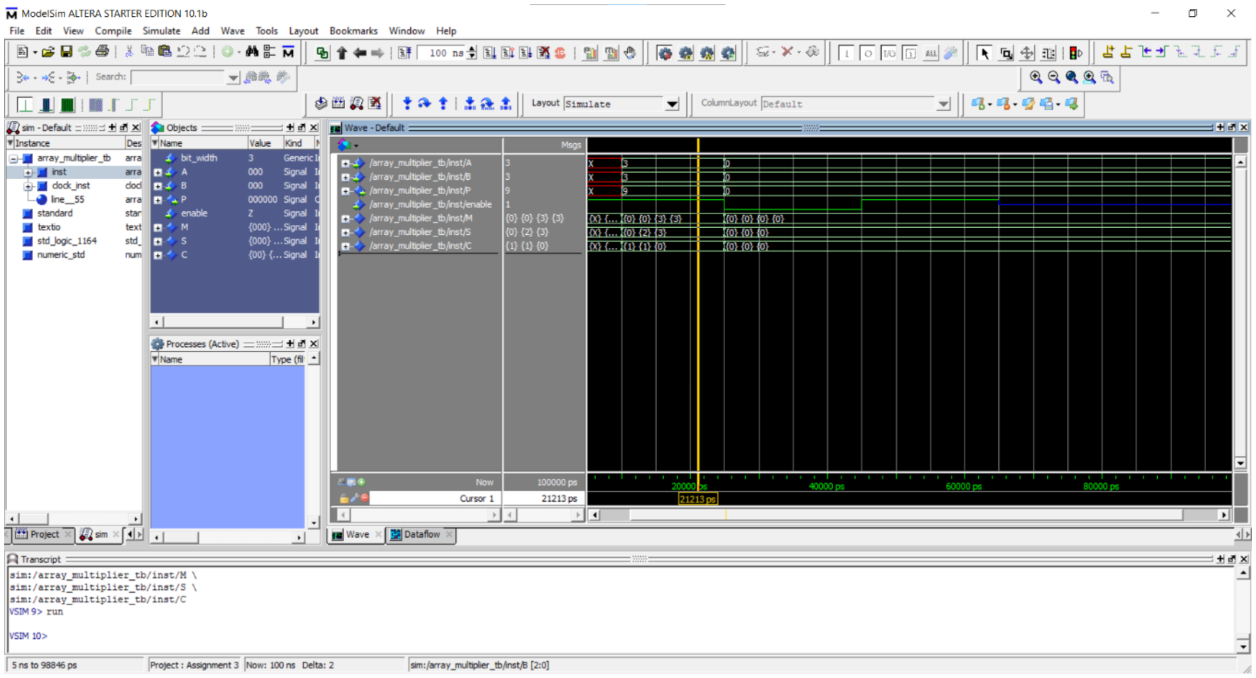
Report Warnings On:
- [✓] Unbound component
- [✓] Process without a WAIT statement
- [✓] Null range
- [✓] No space in time literal (e.g. 5ns)
- [✓] Multiple drivers on unresolved signals

Optimize for:
- [✓] StdLogic1164
- [✓] VITAL

Other VHDL Options

Results

Array Multiplier

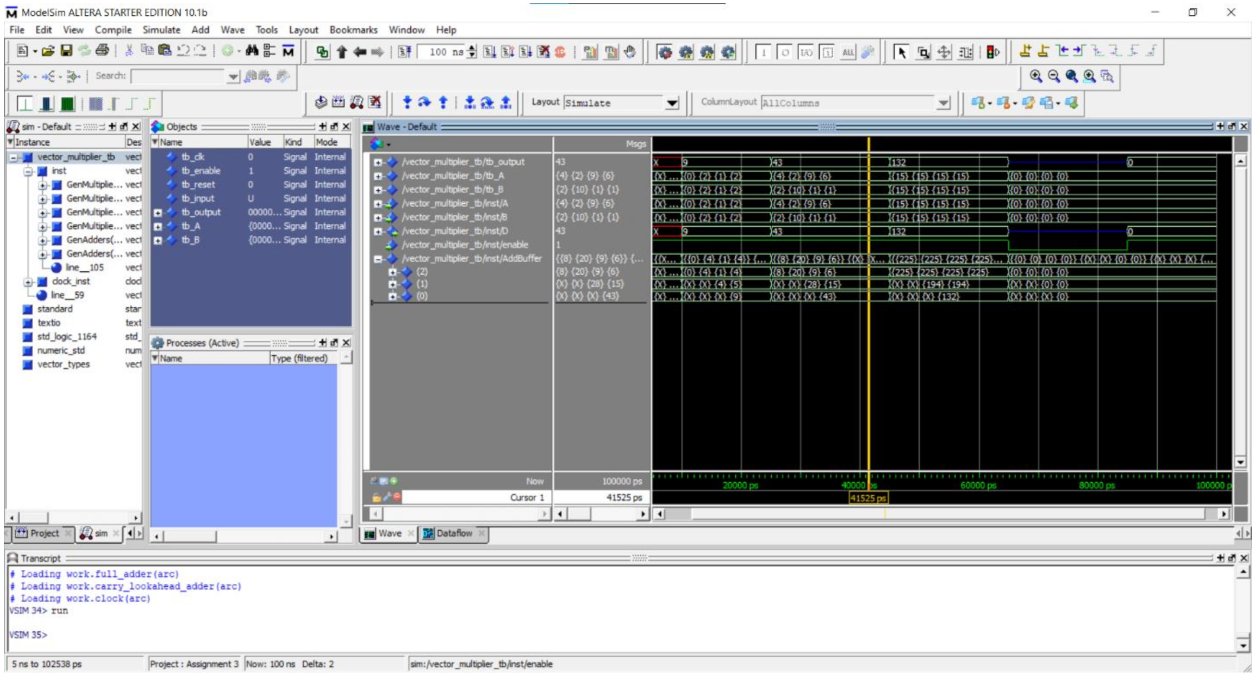


```
/array_multiplier_tb/inst/A        3
/array_multiplier_tb/inst/B        3
/array_multiplier_tb/inst/P        9
/array_multiplier_tb/inst/enable   1
/array_multiplier_tb/inst/M        {0} {0} {3} {3}
/array_multiplier_tb/inst/S        {0} {2} {3}
/array_multiplier_tb/inst/C        {1} {1} {0}

Now       100000 ps
Cursor 1  21213 ps
```

```
sim:/array_multiplier_tb/inst/M \
sim:/array_multiplier_tb/inst/S \
sim:/array_multiplier_tb/inst/C
VSIM 9> run

VSIM 10>
```

```
5 ns to 98846 ps    Project : Assignment 3    Now: 100 ns  Delta: 2    sim:/array_multiplier_tb/inst/B [2:0]
```

Vector Multiplier

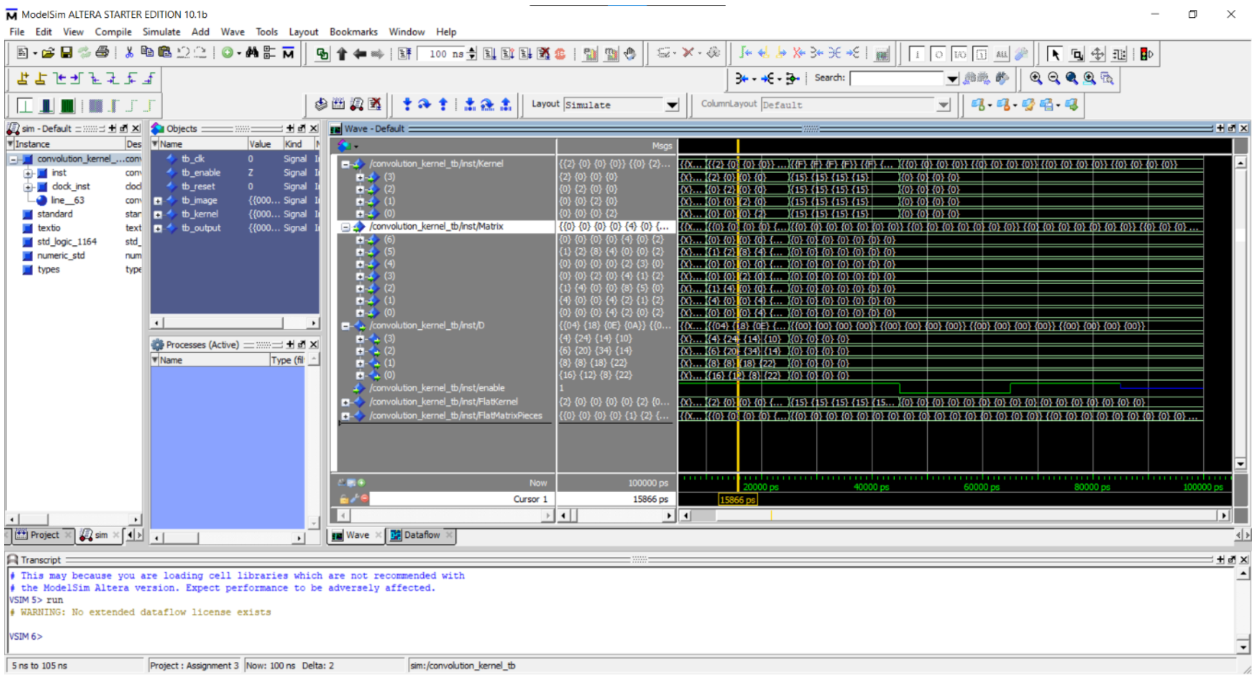


```
/vector_multiplier_tb/tb_output      43
/vector_multiplier_tb/tb_A           {4} {2} {9} {6}
/vector_multiplier_tb/tb_B           {2} {10} {1} {1}
/vector_multiplier_tb/inst/A         {4} {2} {9} {6}
/vector_multiplier_tb/inst/B         {2} {10} {1} {1}
/vector_multiplier_tb/inst/D         43
/vector_multiplier_tb/inst/enable    1
/vector_multiplier_tb/inst/AddBuffer {{8} {20} {9} {6}} {...
    (2)                              {8} {20} {9} {6}
    (1)                              {X} {X} {28} {15}
    (0)                              {X} {X} {X} {43}

Now       100000 ps
Cursor 1  41525 ps
```

```
# Loading work.full_adder(arc)
# Loading work.carry_lookahead_adder(arc)
# Loading work.clock(arc)
VSIM 34> run

VSIM 35>
```

```
5 ns to 102538 ps    Project : Assignment 3    Now: 100 ns  Delta: 2    sim:/vector_multiplier_tb/inst/enable
```

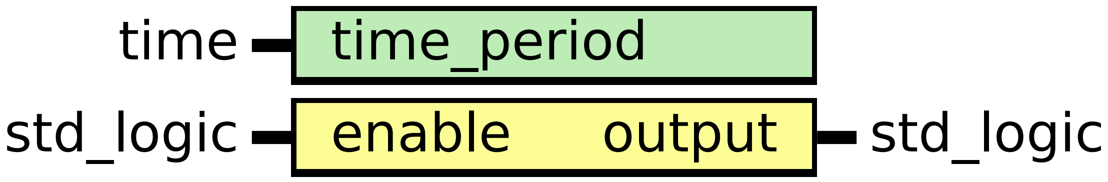
Convolution Kernel


```
# This may because you are loading cell libraries which are not recommended with
# the ModelSim Altera version. Expect performance to be adversely affected.
VSIM 5> run
# WARNING: No extended dataflow license exists

VSIM 6>
```

```
5 ns to 105 ns    Project : Assignment 3    Now: 100 ns  Delta: 2    sim:/convolution_kernel_tb
```

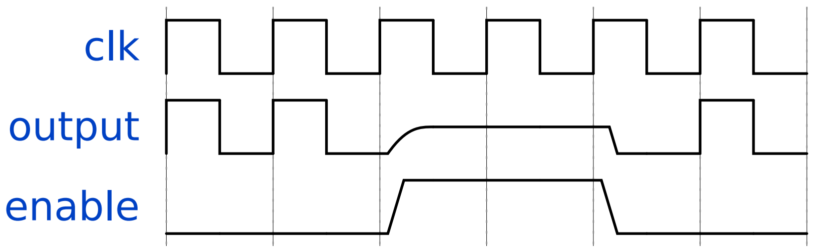
Clock

Diagram



Description

This clock simply switches between '1' and '0' with the given time period.
Internally, it uses the transport delay feature of VHDL.
The default time period is set to 10 ns.
If enable != '1' then the output is set to high impedence.
Note: Even when enable != '1' the clock continues to tick.



Generics and ports

Table 1.1 Generics

Generic name	Type	Value	Description
time_period	time	10 ns	Time period of the clock.

Table 1.2 Ports

Port name	Direction	Type	Description
enable	in	std_logic	If set to anything other than '1', then O/P is high impedence.
output	out	std_logic	Either '1' or '0' depending on the clock signal.

Signals, constants and types

Signals

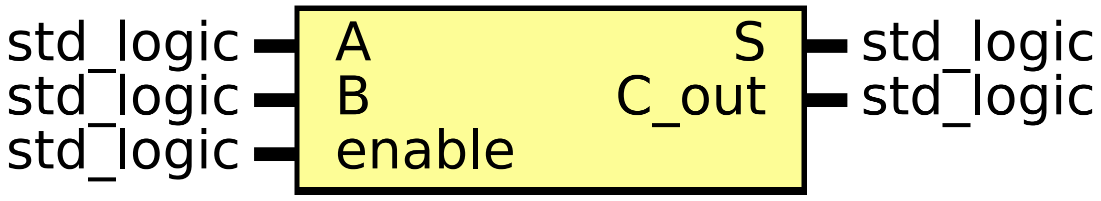
Name	Type	Description
internal_clock	std_logic	

Processes

- **clock_tick**: ***(enable, internal_clock)***

Half Adder

Diagram



Description

This is half adder combinational circuit.
If enable != '1' then the output is set to high impedence.

Generics and ports

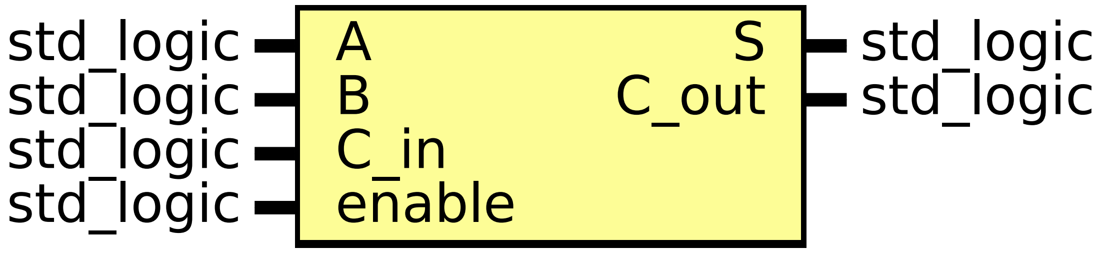
Table 1.1 Generics

Table 1.2 Ports

Port name	Direction	Type	Description
A	in	std_logic	Input bit A.
B	in	std_logic	Input bit B.
S	out	std_logic	Output bit Sum.
C_out	out	std_logic	Output bit Carry.
enable	in	std_logic	If not = '1', output is high impedence and internal state is frozen.

Full Adder

Diagram



Description

This is full adder combinational circuit.
If enable != '1' then the output is set to high impedence.

Generics and ports

Table 1.1 Generics

Table 1.2 Ports

Port name	Direction	Type	Description
A	in	std_logic	Input bit A.
B	in	std_logic	Input bit B.
C_in	in	std_logic	Input bit Carry.
S	out	std_logic	Output bit Sum.
C_out	out	std_logic	Output bit Carry.
enable	in	std_logic	If not = '1', output is high impedence and internal state is frozen.

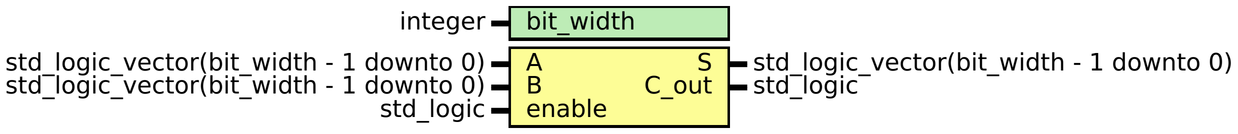
Signals, constants and types

Signals

Name	Type	Description
I	std_logic	Common signal A xor B.

Carry Look-ahead Adder

Diagram


integer
bit_width
std_logic_vector(bit_width - 1 downto 0)
std_logic_vector(bit_width - 1 downto 0)
std_logic
A
B
enable
S
C_out
std_logic_vector(bit_width - 1 downto 0)
std_logic

Description

A N-bit adder using carry look-ahead.
This is combinational circuit.
If enable != '1' then the output is set to high impedence.

Generics and ports

Table 1.1 Generics

Generic name	Type	Value	Description
bit_width	integer	4	The bit wifth of input integers.

Table 1.2 Ports

Port name	Direction	Type	Description
A	in	std_logic_vector(bit_width - 1 downto 0)	Input integer A.
B	in	std_logic_vector(bit_width - 1 downto 0)	Input integer B.
S	out	std_logic_vector(bit_width - 1 downto 0)	Output Sum=A+B.
C_out	out	std_logic	Output Carry.
enable	in	std_logic	If not = '1', output is high impedence.

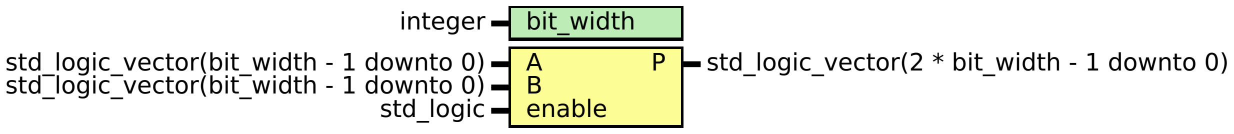
Signals, constants and types

Signals

Name	Type	Description
G	std_logic_vector(bit_width - 1 downto 0)	Generate signals
P	std_logic_vector(bit_width - 1 downto 0)	Propagate signals
C	std_logic_vector(bit_width downto 0)	Carry buffer

Array Multiplier

Diagram


integer
bit_width
std_logic_vector(bit_width - 1 downto 0)
std_logic_vector(bit_width - 1 downto 0)
std_logic
A
B
enable
P
std_logic_vector(2 * bit_width - 1 downto 0)

Description

A N-bit multiplier using add-shift.
Note: The output has double the bit width of the input.
This is combinational circuit.
If enable != '1' then the output is set to high impedence.
Reference: https://faculty.weber.edu/fonbrown/ee3610/arraymult.txt

Generics and ports

Table 1.1 Generics

Generic name	Type	Value	Description
bit_width	integer	8	The bit wifth of input integers.

Table 1.2 Ports

Port name	Direction	Type	Description
A	in	std_logic_vector(bit_width - 1 downto 0)	Input integer A.
B	in	std_logic_vector(bit_width - 1 downto 0)	Input integer B.
P	out	std_logic_vector(2 * bit_width - 1 downto 0)	Output result A*B.
enable	in	std_logic	If not = '1', output is high impedence.

Signals, constants and types

Signals

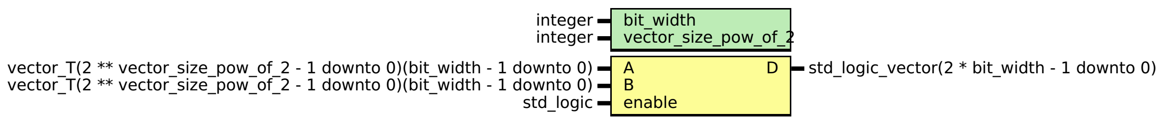
Name	Type	Description
M	Array_T(bit_width downto 0, bit_width - 1 downto 0)	Used to store bit multiplications A(i) and B(i).
S	Array_T(bit_width - 1 downto 0, bit_width - 1 downto 0)	Used to store intermediate sums.
C	Array_T(bit_width - 1 downto 0, bit_width - 2 downto 0)	Used to store intermediate carry.

Types

Name	Type	Description
Array_T		Type for 2D array of std_logic.

Vector Dot Product Unit

Diagram


integer
integer
bit_width
vector_size_pow_of_2
vector_T(2 ** vector_size_pow_of_2 - 1 downto 0)(bit_width - 1 downto 0)
vector_T(2 ** vector_size_pow_of_2 - 1 downto 0)(bit_width - 1 downto 0)
std_logic
A
B
enable
D
std_logic_vector(2 * bit_width - 1 downto 0)

Description

A N-bit 2^T length vector multiplier (dot product) using array multipliers and carry lookahead adders.
It requires 2^T as it uses logrithmic parallel addition by dividing the addition into T stages of carry look-ahead adders.
Note: The output has double the bit width of the input.
This ignores (discards carry) any integer overflows during accumulation of the dot product.
NOTE: To compile this file please set modelsim to use VHDL 2008 for all files. See:
https://forums.xilinx.com/t5/Synthesis/Array-of-Unconstrained-Array/td-p/681253
This is combinational circuit.
If enable != '1' then the output is set to high impedence.

Generics and ports

Table 1.1 Generics

Generic name	Type	Value	Description
bit_width	integer	8	Bit width for input integers.
vector_size_pow_of_2	integer	2	The length 2^T of input vectors.

Table 1.2 Ports

Port name	Direction	Type	Description
A	in	vector_T(2 ** vector_size_pow_of_2 - 1 downto 0)(bit_width - 1 downto 0)	Input vector A.
B	in	vector_T(2 ** vector_size_pow_of_2 - 1 downto 0)(bit_width - 1 downto 0)	Input vector B.
D	out	std_logic_vector(2 * bit_width - 1 downto 0)	Output number A.B (dot product). Has integrs with 2*bit_width bits.
enable	in	std_logic	If not = '1', output is high impedence and internal state is frozen.

Signals, constants and types

Signals

Name	Type	Description
AddBuffer	Array_T(vector_size_pow_of_2 downto 0, 2 ** vector_size_pow_of_2 - 1 downto 0)(2 * bit_width - 1 downto 0)	Stores addition stages

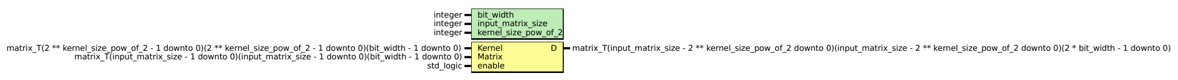
Types

Name	Type	Description

Array_T		Type for 2D array of std_logic.

Convolution Kernel

Diagram


integer
integer
integer
bit_width
input_matrix_size
kernel_size_pow_of_2
matrix_T(2 ** kernel_size_pow_of_2 - 1 downto 0)(2 ** kernel_size_pow_of_2 - 1 downto 0)(bit_width - 1 downto 0)
matrix_T(input_matrix_size - 1 downto 0)(input_matrix_size - 1 downto 0)(bit_width - 1 downto 0)
std_logic
Kernel
Matrix
enable
D
matrix_T(input_matrix_size - 2 ** kernel_size_pow_of_2 downto 0)(input_matrix_size - 2 ** kernel_size_pow_of_2 downto 0)(2 * bit_width - 1 downto 0)

Description

A N-bit K x K convolution kernel processor for a M x M input matrix.
Uses a stride of 1 since with 1-stride output any other stride output can be extracted.
Uses array multipliers and carry lookahead adders via Vector Dot Product Units.
Note: The output has double the bit width of the input.
NOTE: To compile this file please set modelsim to use VHDL 2008 for all files. See:
https://forums.xilinx.com/t5/Synthesis/Array-of-Unconstrained-Array/td-p/681253
This is combinational circuit.
If enable != '1' then the output is set to high impedence.

Generics and ports

Table 1.1 Generics

Generic name	Type	Value	Description
bit_width	integer	4	Bit width for every integer.
input_matrix_size	integer	7	The matrix dimension M x M.
kernel_size_pow_of_2	integer	2	The kernel dimension 2^K x 2^K. This is powered by 2 to use logrithmic addition in Vector Dot Product Unit.

Table 1.2 Ports

Port name	Direction	Type	Description
Kernel	in	matrix_T(2 ** kernel_size_pow_of_2 - 1 downto 0)(2 ** kernel_size_pow_of_2 - 1 downto 0)(bit_width - 1 downto 0)	Input Kernel to use. Must be of dimension 2^K x 2^K.
Matrix	in	matrix_T(input_matrix_size - 1 downto 0)(input_matrix_size - 1 downto 0)(bit_width - 1 downto 0)	Input matrix to apply kernel on. Must have dimension of M x M.
D	out	matrix_T(input_matrix_size - 2 ** kernel_size_pow_of_2 downto 0)(input_matrix_size - 2 ** kernel_size_pow_of_2 downto 0)(2 * bit_width - 1 downto 0)	Output result A.B (dot product). Dimension is square matrix of size M - 2^K + 1. Output also has 2*bit_width for all integers.
enable	in	std_logic	If not = '1', output is high impedence.

Signals, constants and types

Signals

Name	Type	Description
FlatKernel	vector_T(2 ** (2 * kernel_size_pow_of_2) - 1 downto 0)(bit_width - 1 downto 0)	Stores flattned kernel
FlatMatrixPieces	matrix_T((input_matrix_size - 2 ** kernel_size_pow_of_2 + 1) ** 2 - 1 downto 0)(2 ** (2 * kernel_size_pow_of_2) - 1 downto 0)(bit_width - 1 downto 0)	Stores flattned matrix peices

Processes

- **FlattenKernel**: ***(Kernel)***