Vector Files in Quartus II
Wilfrid Laurier University

Terry Sturtevant

Wilfrid Laurier University

March 25, 2010

Why use vector files?

Why use vector files?

- For a small circuit, creating the vector waveform file graphically is easy

Why use vector files?

- For a small circuit, creating the vector waveform file graphically is easy

  For circuits with lots of inputs, no so much

Why use vector files?

- For a small circuit, creating the vector waveform file graphically is easy

  For circuits with lots of inputs, no so much
- If you already have a truth table for your circuit, creating a vector file is easy

Why use vector files?

- For a small circuit, creating the vector waveform file graphically is easy

  For circuits with lots of inputs, no so much
- If you already have a truth table for your circuit, creating a vector file is easy

  It's a text file based on the truth table

Basic Structure

Basic Structure

- **START** time

Basic Structure

- **START** time (ns)

Basic Structure

- **START** time (ns)
- **STOP** time

Basic Structure

- **START** time (ns)
- **STOP** time (ns)

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time (ns)

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time (ns)

  20 is a good default value

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time (ns)

  20 is a good default value
- **INPUTS** name(s)

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time (ns)

  20 is a good default value
- **INPUTS** name(s)
- **PATTERN** string

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time (ns)

  20 is a good default value
- **INPUTS** name(s)
- **PATTERN** string

  can be *absolute*

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time (ns)

  20 is a good default value
- **INPUTS** name(s)
- **PATTERN** string

  can be *absolute*

  or *relative*

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time (ns)

  20 is a good default value
- **INPUTS** name(s)
- **PATTERN** string

  can be *absolute*

  or *relative*
- **OUTPUTS** name(s)

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time (ns)

  20 is a good default value
- **INPUTS** name(s)
- **PATTERN** string

  can be *absolute*

  or *relative*
- **OUTPUTS** name(s)
- *All statements end with a semicolon.*

Basic Structure

- **START** time (ns)
- **STOP** time (ns)
- **INTERVAL** time (ns)

  20 is a good default value
- **INPUTS** name(s)
- **PATTERN** string

  can be *absolute*

  or *relative*
- **OUTPUTS** name(s)
- *All statements end with a semicolon.*
- *Comments go between percent signs.*

Simple example

Simple example

- Simple circuit

Simple example

- Simple circuit
- 2 inputs, a and b

Simple example

- Simple circuit
- 2 inputs, a and b
- 1 output, c

Example.vec (absolute timing)

Example.vec (absolute timing)

START 0;

Example.vec (absolute timing)

START 0;

STOP 100;

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b;

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0> 0 0;

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0> 0 0;

20> 0 1;

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0> 0 0;

20> 0 1;

40> 1 0;

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0> 0 0;

20> 0 1;

40> 1 0;

60> 1 1;

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0> 0 0;

20> 0 1;

40> 1 0;

60> 1 1;

80> 0 0;

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0> 0 0;

20> 0 1;

40> 1 0;

60> 1 1;

80> 0 0; % Make sure the simulation goes to 80 ns %

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0> 0 0;

20> 0 1;

40> 1 0;

60> 1 1;

80> 0 0; % Make sure the simulation goes to 80 ns %

OUTPUTS c;

Example.vec (absolute timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0> 0 0;

20> 0 1;

40> 1 0;

60> 1 1;

80> 0 0; % Make sure the simulation goes to 80 ns %

OUTPUTS c; % This is the output %

Using absolute timing, you specify what values each variable should have at certain times.

Using absolute timing, you specify what values each variable should have at certain times.

So, in the example, the line

Using absolute timing, you specify what values each variable should have at certain times.

So, in the example, the line

0> 0 0;

Using absolute timing, you specify what values each variable should have at certain times.

So, in the example, the line

0> 0 0;

means *"At time 0ns, set inputs to 0 0 "*

Using absolute timing, you specify what values each variable should have at certain times.

So, in the example, the line

0> 0 0;

means *"At time 0ns, set inputs to 0 0 "*

and line

Using absolute timing, you specify what values each variable should have at certain times.

So, in the example, the line

0> 0 0;

means *"At time 0ns, set inputs to 0 0 "*

and line

20> 0 1;

Using absolute timing, you specify what values each variable should have at certain times.

So, in the example, the line

0> 0 0;

means *"At time 0ns, set inputs to 0 0 "*

and line

20> 0 1;

means *"At time 20ns, set inputs to 0 1 "*

Using absolute timing, you specify what values each variable should have at certain times.

So, in the example, the line

0> 0 0;

means *"At time 0ns, set inputs to 0 0 "*

and line

20> 0 1;

means *"At time 20ns, set inputs to 0 1 "*

Example.vec (relative timing)

Example.vec (relative timing)

START 0;

Example.vec (relative timing)

START 0;

STOP 100;

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b;

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0 0

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0 0

0 1

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0 0

0 1

1 0

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0 0

0 1

1 0

1 1;

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0 0

0 1

1 0

1 1; % Note the single semicolon at the end %

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0 0

0 1

1 0

1 1; % Note the single semicolon at the end %

OUTPUTS c;

Example.vec (relative timing)

START 0;

STOP 100;

INTERVAL 20;

INPUTS a b; % These are the inputs %

PATTERN

0 0

0 1

1 0

1 1; % Note the single semicolon at the end %

OUTPUTS c; % This is the output %

Using relative timing, you specify what values each variable should have at each *interval*.

Using relative timing, you specify what values each variable should have at each *interval*.

So, in the example, the line

Using relative timing, you specify what values each variable should have at each *interval*.

So, in the example, the line

0 0

Using relative timing, you specify what values each variable should have at each *interval*.

So, in the example, the line

0 0

means *"Set inputs to 0 0 at time 0"*

Using relative timing, you specify what values each variable should have at each *interval*.

So, in the example, the line

0 0

means *"Set inputs to 0 0 at time 0"*

and the line

Using relative timing, you specify what values each variable should have at each *interval*.

So, in the example, the line

0 0

means *"Set inputs to 0 0 at time 0"*

and the line

0 1

Using relative timing, you specify what values each variable should have at each *interval*.

So, in the example, the line

0 0

means *"Set inputs to 0 0 at time 0"*

and the line

0 1

means *"Set inputs to 0 1 at the first interval"*

Using relative timing, you specify what values each variable should have at each *interval*.

So, in the example, the line

0 0

means *"Set inputs to 0 0 at time 0"*

and the line

0 1

means *"Set inputs to 0 1 at the first interval"*

Notice that in the relative timing file,

Notice that in the relative timing file,

INPUTS a b; % These are the inputs %

PATTERN

Notice that in the relative timing file,

INPUTS a b; % These are the inputs %

PATTERN

0 0

0 1

1 0

1 1;

Notice that in the relative timing file,

INPUTS a b; % These are the inputs %

PATTERN

0 0

0 1

1 0

1 1;

the highlighted lines are directly from the truth table.

Notice that in the relative timing file,

INPUTS a b; % These are the inputs %

PATTERN

0 0

0 1

1 0

1 1;

the highlighted lines are directly from the truth table. This makes it easy to use the truth table to make a vector file.