

New in Surfer 0.4.0

New in Surfer 0.4.0

- **Translator plugins**

New in Surfer 0.4.0

- **Translator plugins**
- **Signal grouping**

New in Surfer 0.4.0

- **Translator plugins**
- **Signal grouping**
- **Waveform Control Protocol** improvements

New in Surfer 0.4.0

- **Translator plugins**
- **Signal grouping**
- **Waveform Control Protocol** improvements
- **130** new MRs!

Translator Plugins

Translators **were semi-easy** to add

```
pub trait Translator {  
    fn variable_info(&self, variable: &VariableMeta)  
        -> Result<VariableInfo>;  
    fn translate(&self, variable, value)  
        -> Result<TranslationResult>;  
    fn translates(&self, variable: &VariableMeta)  
        -> Result<TranslationPreference>;  
}
```

Translator Plugins



Demo Time!

```
cargo generate \  
  --template gl:surfer-project/translator-template
```

```
#[plugin_fn]  
pub fn name() -> FnResult<&'static str> {}
```

```
#[plugin_fn]  
pub fn variable_info(variable: VariableMeta<(), ()>)  
    -> FnResult<VariableInfo> {}
```

```
#[plugin_fn]  
pub fn translate(  
    TranslateParams { variable, value }: TranslateParams,  
) -> FnResult<TranslationResult> {}
```

```
#[plugin_fn]  
pub fn translates(_variable: VariableMeta<(), ()>)  
    -> FnResult<TranslationPreference> {}
```

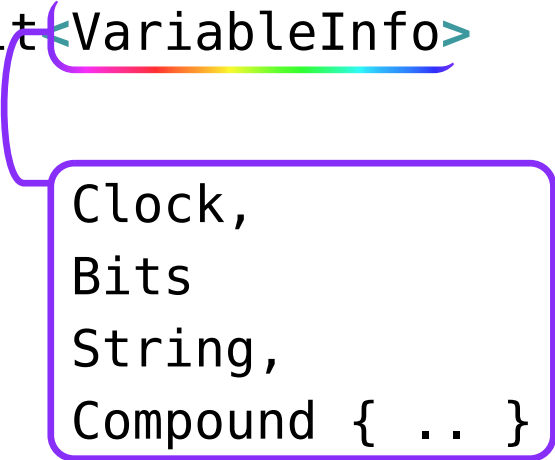
```
#[plugin_fn]  
pub fn name() -> FnResult<&'static str> {  
  
}
```

```
#[plugin_fn]
pub fn name() -> FnResult<&'static str> {
    Ok("FSiC")
}
```

```
#[plugin_fn]
pub fn variable_info(variable: VariableMeta<(), ()>)
    -> FnResult<VariableInfo>
{

}
```

```
#[plugin_fn]
pub fn variable_info(variable: VariableMeta<(), ()>)
    -> FnResult<VariableInfo>
{
    Clock,
    Bits
    String,
    Compound { .. }
}
```



A diagram illustrating the return type `VariableInfo`. A purple line connects the `VariableInfo` type in the function signature to a purple-bordered box. Inside the box, the fields of `VariableInfo` are listed: `Clock`, `Bits`, `String`, and `Compound { .. }`. A rainbow-colored underline is present under the `VariableInfo` text in the signature.

```
#[plugin_fn]
pub fn variable_info(variable: VariableMeta<(), ()>)
    -> FnResult<VariableInfo>
{
    VariableInfo::String
}
```



```
#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {

}
```

```
#[plugin_fn]
pub fn translate(
    variable,
    value
) -> Variable metadata ActionResult> {

}
```

```
#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
    Variable value
}
```

```
#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
}

```

Human readable (hierarchical) value

```
#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
    value.handle_bits(|bits| {

    })
}
```

```
#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
    value.handle_bits(|bits| {
    })
}
```

Take care of X's and friends

```
#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
    value.handle_bits(|bits| {

    })
}
```

```

#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
    value.handle_bits(|bits| {
        TranslationResult {
            val: bits.matches(|c| c == '1').count(),
            subfields: vec![],
            kind: ValueKind::Normal
        }
    })
}

```



```

#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
    value.handle_bits(|bits| {
        TranslationResult {
            val: bits.matches(|c| c == '1').count(),
            subfields: vec![],
            kind: ValueKind::Normal
        }
    })
}

```

```

#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
    value.handle_bits(|bits| {
        TranslationResult {
            val: bits.matches(|c| c == '1').count(),
            subfields: vec![],
            kind: ValueKind::Normal
        }
    })
}

```

```

#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
    value.handle_bits(|bits| {
        TranslationResult {
            val: bits.matches(|c| c == '1').count(),
            subfields: vec![],
            kind: ValueKind::Normal
        }
    })
}

```

```
#[plugin_fn]
pub fn translate(
  variable,
  value
) -> FnResult<TranslationResult> {
  value.handle_bits(|bits| {
    TranslationResult {
      val: bits.matches(|c| c == '1').count(),
      subfields: vec![],
      kind: Count number of 1s
    }
  })
}
```

```
#[plugin_fn]  
pub fn translates(_variable: VariableMeta<(), ()>)  
    -> FnResult<TranslationPreference> {}
```

```
cargo build
```

```
cp target/fsic/fsic.wasm ~/.local/share/swim/translators
```

Time

trace_data

trace_data

UNDEF

8xxxxxxxx

000 ps 1600000 ps 1700000 ps 1800000 ps 1900000 ps 2000000 ps 2100000 ps

UNDEF	32	UNDEF	1	4	2	5	2	UN...	6	UNDEF	6	...	7
xxxxxxx...	0ffffff...	8xxxxxxxxx	8...	...	80...	a...	...	8x...	a...	8xxxx...	a...	...	...

1000000 ps

2000000 ps

3000000 ps

4000000 ps

5000000 ps

6000000 ps

7000000 ps

8000000 ps

9000000 ps

Remote Control via WCP

Remote Control via WCP

Remote Control Protocol via TCP, (Web)Sockets, ...

Remote Control via WCP

Remote Control Protocol via TCP, (Web)Sockets, ...

Full control over the waveform viewer

- Add or remove signals
- Zoom or pan around
- Mark parts, draw inside Waveform

Other cool stuff



Web Assembly!

- No installation
 - Inspect waves in continuous integration
- **Embeddable**

Embeddable as a teaching tool


Examples

Editor
CPU
Waveform

About

Steps					
PC	fetch	decode	execute	memory	writeback
0	add x0, x1, x2				
4	sub x0, x1, x2	add x0, x1, x2			
8	xor x0, x1, x2	sub x0, x1, x2	add x0, x1, x2		
C	or x0, x1, x2	xor x0, x1, x2	sub x0, x1, x2	add x0, x1, x2	
10	and x0, x1, x2	or x0, x1, x2	xor x0, x1, x2	sub x0, x1, x2	add x0, x1, x2
14	sll x0, x1, x2	and x0, x1, x2	or x0, x1, x2	xor x0, x1, x2	sub x0, x1, x2
18	srl x0, x1, x2	sll x0, x1, x2	and x0, x1, x2	or x0, x1, x2	xor x0, x1, x2
1C	sra x0, x1, x2	srl x0, x1, x2	sll x0, x1, x2	and x0, x1, x2	or x0, x1, x2
20	slt x0, x1, x2	sra x0, x1, x2	srl x0, x1, x2	sll x0, x1, x2	and x0, x1, x2
24	sltu x0, x1, x2	slt x0, x1, x2	sra x0, x1, x2	srl x0, x1, x2	sll x0, x1, x2
28	addi x1, x1, 5	sltu x0, x1, x2	slt x0, x1, x2	sra x0, x1, x2	srl x0, x1, x2
2C	xori x0, x1, 5	addi x1, x1, 5	sltu x0, x1, x2	slt x0, x1, x2	sra x0, x1, x2
30	ori x0, x1, 5	xori x0, x1, 5	addi x1, x1, 5	sltu x0, x1, x2	slt x0, x1, x2
34	andi x0, x1, 5	ori x0, x1, 5	xori x0, x1, 5	addi x1, x1, 5	sltu x0, x1, x2
38	slli x0, x1, 0x05	andi x0, x1, 5	ori x0, x1, 5	xori x0, x1, 5	addi x1, x1, 5

File View Settings Help

General

clk

reset

Control

jump

branch

regwrite

aluscrc

memwrite

Decode

pcf [31:0]

instr [31:0]

Register Control

a1 [4:0]

rd1 [31:0]

a2 [4:0]

rd2 [31:0]

a3 [4:0]

wd3 [31:0]

Registers

x1 [31:0]

x2 [31:0]

x3 [31:0]

x4 [31:0]

x5 [31:0]

x6 [31:0]

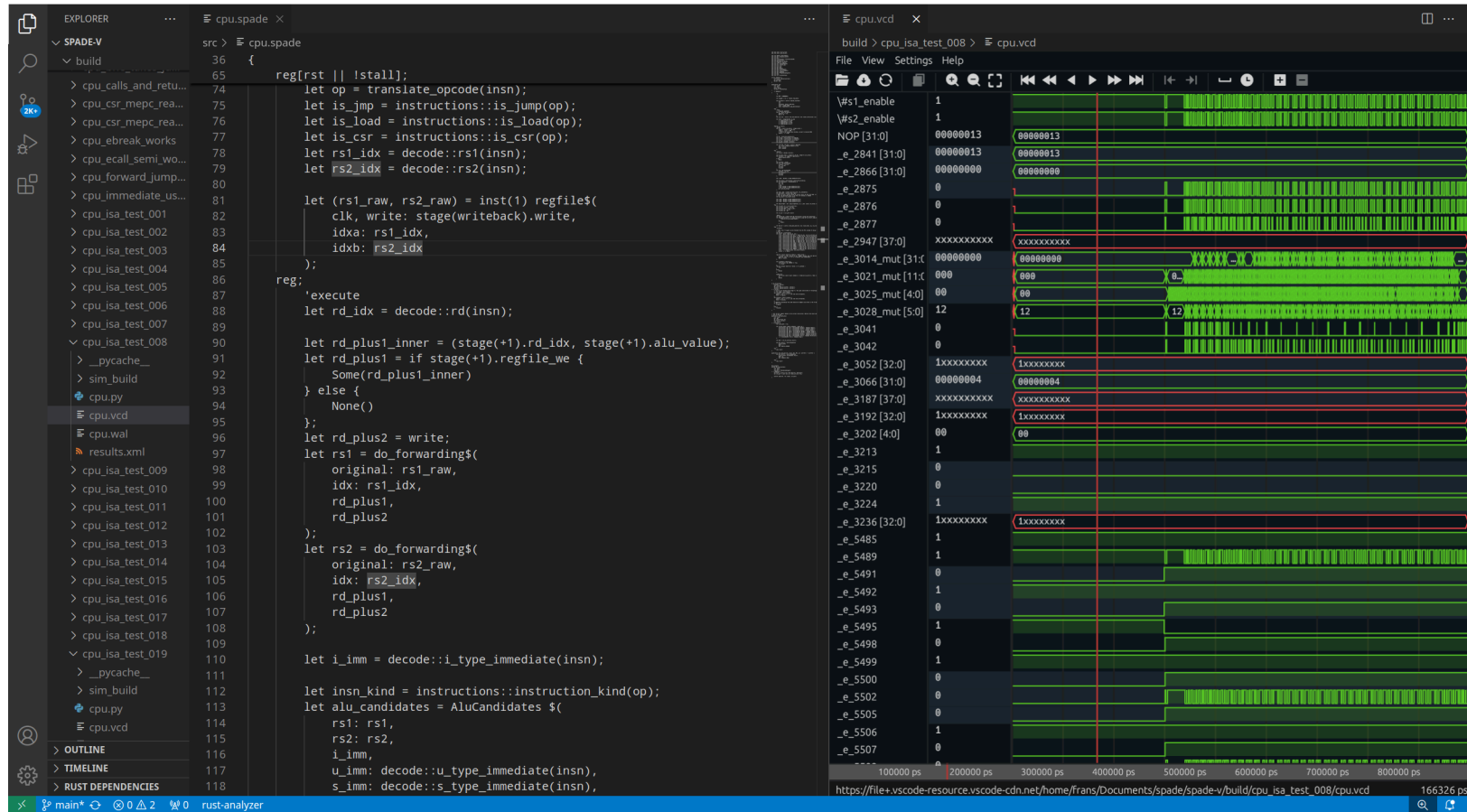
x7 [31:0]

x8 [31:0]

x9 [31:0]

x10 [31:0]

Embeddable as a VSCode plugin



Surver

- Host waveforms on your simulator server
- Look at them on your dev-machine
 - Only signals you view are donwloaded

Wait, this is the analog session!?

Thanks!

Contributors

- Alejandro Tafalla
- Andreas Wallner
- Ben Mattes Krusekamp
- Yehowshua Immanuel
- Christian
- Christian Dattinger
- Daniel Grosse
- ecstrema
- Felix Roithmayr
- Francesco Urbani
- Greg Chadwick
- Gustav Sörnäs
- Gökçe Aydos
- Hugo Lundin
- Jakub Urbanczyk
- James Connolly
- John Schulz
- Kacper Uminski
- Kaleb Barrett
- Kevin Läufer
- Lars Kadel
- Lucas Klemmer
- Lukas Scheller
- Matt Taylor
- Ondrej Ille
- Oscar Gustafsson
- Remi Marche
- Riley
- Robin Ole Heinemann
- Rong “Mantle” Bao
- schilkp
- Theodor Lindberg
- Todd Strader
- Tom Verbeure
- TopTortoise
- Verner Hirvonen
- Øystein Hovind

Everyone who has provided feedback

Conclusions

Surfer is a **snappy** and **extensible** waveform viewer that **runs everywhere**

Conclusions

Surfer is a **snappy** and **extensible** waveform viewer that **runs everywhere**

What do **you** want from a waveform viewer?

Conclusions

Surfer is a **snappy** and **extensible** waveform viewer that **runs everywhere**

What do **you** want from a waveform viewer?

Try it on your phone!

