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
```

```rust
#[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> {}
```

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

}
```

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

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

}
```

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

}
```

```
Clock,
Bits
String,
Compound { .. }
```

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

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

}
```

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

}
```

Variable metadata

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

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

}
```

Human readable (hierarchical) value

```rust
#[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| {

    })
}
```

```rust
#[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
        }
    })
}
```

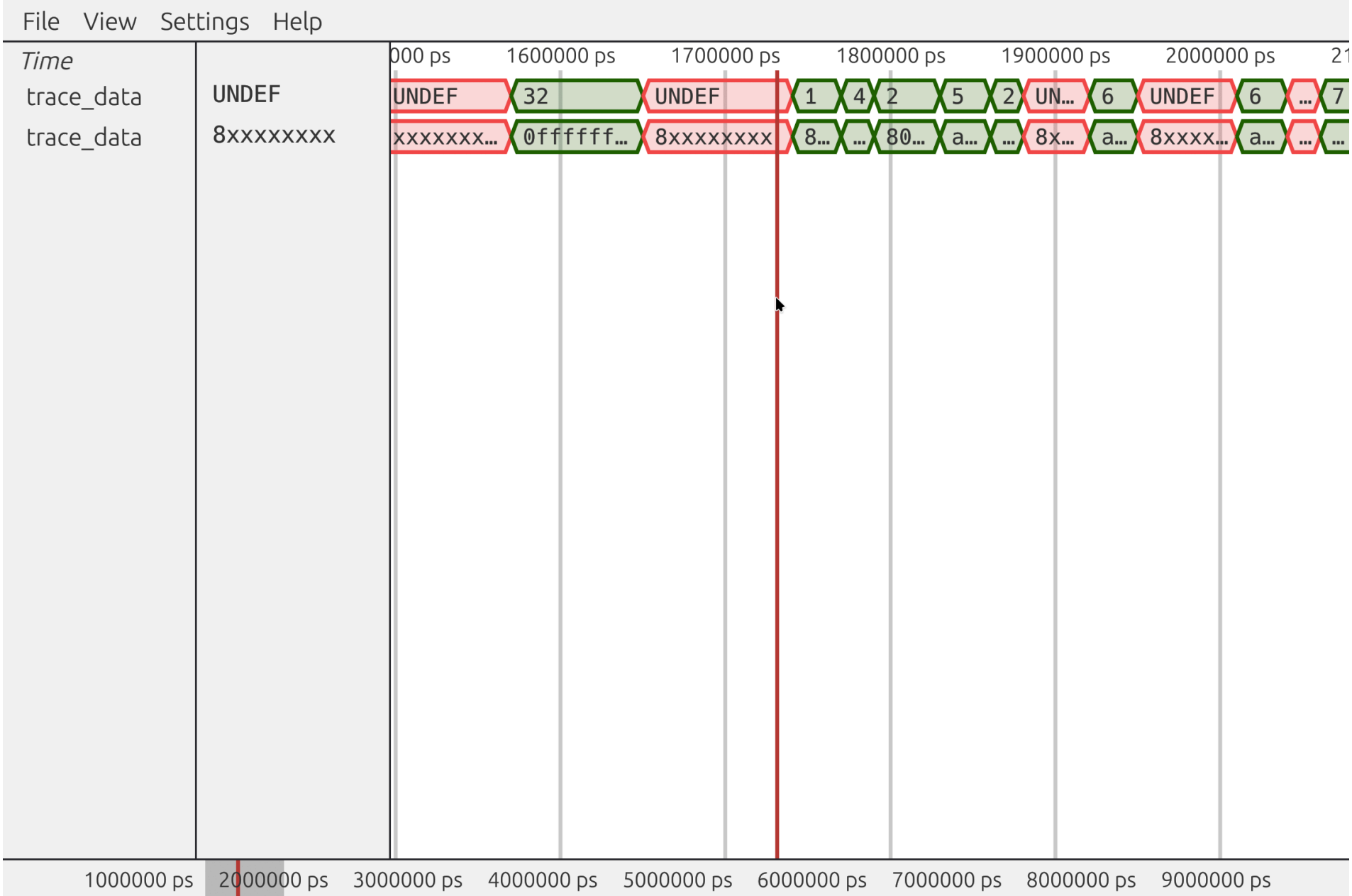
```
#[plugin_fn]
pub fn translate(
    variable,
    value
) -> FnResult<TranslationResult> {
    value.handle_bits(|bits| {
        TranslationResult {
            val: bits.matches(|c| c == '1').count(),
            subfields: vec![],
            kind: ValueKind::Normal
        }
    })
}
```

```rust
#[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

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

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


File View Settings Help
Time
trace_data
UNDEF
trace_data
8xxxxxxxx
000 ps
1600000 ps
1700000 ps
1800000 ps
1900000 ps
2000000 ps
UNDEF
32
UNDEF
1
4
2
5
2
UN…
6
UNDEF
6
…
7
xxxxxxx…
0ffffff…
8xxxxxxxx
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

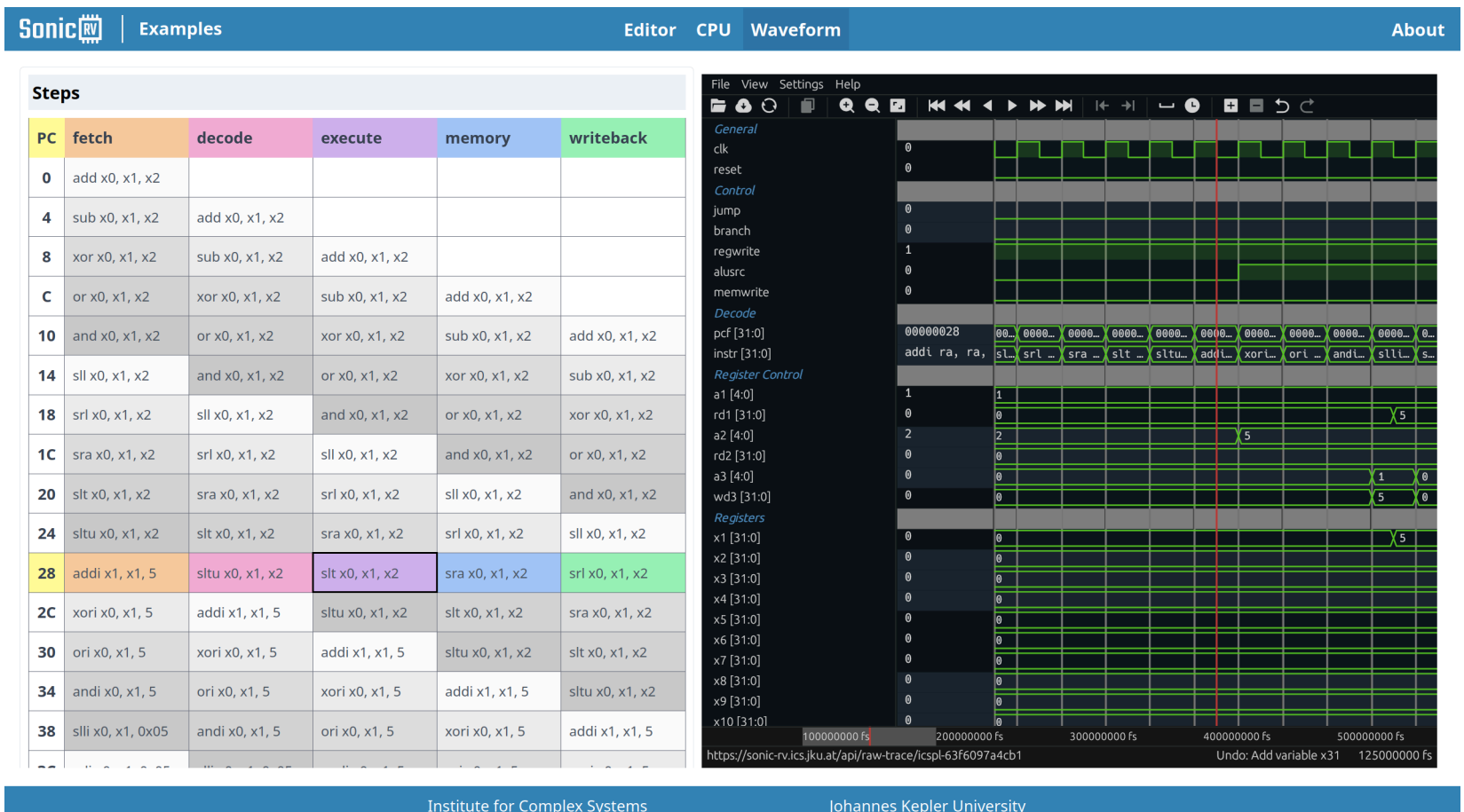
- Add or remove signals
- Zoom or pan around
- Mark parts, draw inside Waveform

Other cool stuff

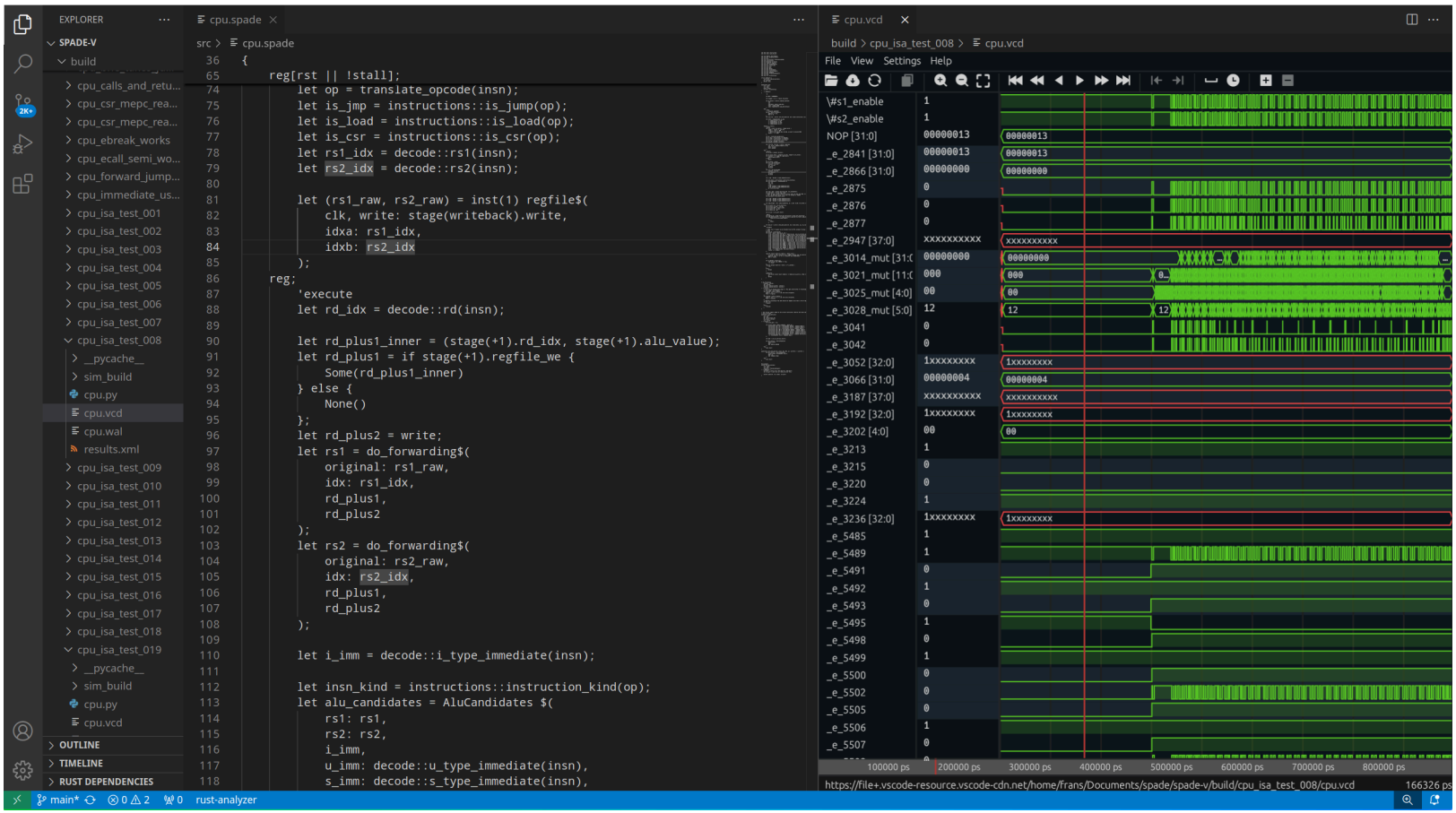
Web Assembly!

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

Embeddable as a teaching tool



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
- Verneri 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!

HM

LINKÖPING UNIVERSITY