

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 { .. }
```

```
#[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 {onResult> {
}
}
```

```
#[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> {}
```

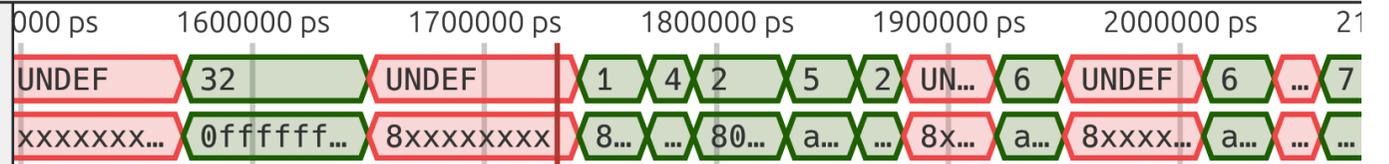
```
cargo build
```

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

Time

trace_data

UNDEF



trace_data

8xxxxxxxxx

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

SonicRV
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

Institute for Complex Systems
Johannes Kepler University

Embeddable as a VSCode plugin

The image shows a VSCode editor window with two main panes. The left pane displays a Rust source file named `cpu.spade` with the following code:

```
36 {
65   reg[rst || !stall];
74   let op = translate_opcode(insn);
75   let is_jmp = instructions::is_jump(op);
76   let is_load = instructions::is_load(op);
77   let is_csr = instructions::is_csr(op);
78   let rs1_idx = decode::rs1(insn);
79   let rs2_idx = decode::rs2(insn);
80
81   let (rs1_raw, rs2_raw) = inst(1) regfile$(
82     clk, write: stage(writeback).write,
83     idxa: rs1_idx,
84     idxb: rs2_idx
85   );
86   reg;
87   *execute
88   let rd_idx = decode::rd(insn);
89
90   let rd_plus1_inner = (stage(+1).rd_idx, stage(+1).alu_value);
91   let rd_plus1 = if stage(+1).regfile_we {
92     Some(rd_plus1_inner)
93   } else {
94     None()
95   };
96   let rd_plus2 = write;
97   let rs1 = do_forwarding$(
98     original: rs1_raw,
99     idx: rs1_idx,
100    rd_plus1,
101    rd_plus2
102  );
103  let rs2 = do_forwarding$(
104    original: rs2_raw,
105    idx: rs2_idx,
106    rd_plus1,
107    rd_plus2
108  );
109
110  let i_imm = decode::i_type_immediate(insn);
111  let insn_kind = instructions::instruction_kind(op);
112  let alu_candidates = AluCandidates $(
113    rs1: rs1,
114    rs2: rs2,
115    i_imm,
116    u_imm: decode::u_type_immediate(insn),
117    s_imm: decode::s_type_immediate(insn),
```

The right pane shows a hardware simulator interface for `cpu.vcd`. It displays a list of signals and their corresponding values over time. The signals include `Vw1_enable`, `Vw2_enable`, `NOP [31:0]`, `_e_2841 [31:0]`, `_e_2866 [31:0]`, `_e_2875`, `_e_2876`, `_e_2877`, `_e_2947 [37:0]`, `_e_3014_mut [31:0]`, `_e_3021_mut [11:0]`, `_e_3025_mut [4:0]`, `_e_3028_mut [5:0]`, `_e_3041`, `_e_3042`, `_e_3052 [32:0]`, `_e_3066 [31:0]`, `_e_3187 [37:0]`, `_e_3192 [32:0]`, `_e_3202 [4:0]`, `_e_3213`, `_e_3215`, `_e_3220`, `_e_3224`, `_e_3236 [32:0]`, `_e_5485`, `_e_5489`, `_e_5491`, `_e_5492`, `_e_5493`, `_e_5495`, `_e_5498`, `_e_5499`, `_e_5500`, `_e_5502`, `_e_5505`, `_e_5506`, and `_e_5507`. The simulator shows a sequence of values for each signal over time, with a time axis ranging from 0 to 800,000 ps.

Surver

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

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
- Vernerir 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!

