Funcalc

A Spreadsheet Research Prototype

Alexander Asp Bock
albo@itu.dk
Computer Science Department
IT UNIVERSITY OF COPENHAGEN

May 10, 2017

Outline

Introduction

Sheet-defined Functions

Functional Paradigms In Spreadsheets

Dataflow Computation

Overview of Funcalc

Overview of Funcalc

- Started in 2014 by Professor Peter Sestoft

Overview of Funcalc

- Started in 2014 by Professor Peter Sestoft
- Part of the **P**opular **P**arallel **P**rogramming (P3) project with Aalborg University (2015-2018)

Overview of Funcalc

- Started in 2014 by Professor Peter Sestoft
- Part of the **P**opular **P**arallel **P**rogramming (P3) project with Aalborg University (2015-2018)
- Currently ~22,000 lines of C# code

Overview of Funcalc

- Started in 2014 by Professor Peter Sestoft
- Part of the **P**opular **P**arallel **P**rogramming (P3) project with Aalborg University (2015-2018)
- Currently ~22,000 lines of C# code
- 2 PhDs, 2 student programmers

P3 Project

P3 Project

- **Long-term vision**:

P3 Project

- **Long-term vision**:
  Exploit abundant shared-memory machines

P3 Project

- **Long-term vision**:

  Exploit abundant shared-memory machines

  A platform understandable by millions

P3 Project

- **Long-term vision**:
  Exploit abundant shared-memory machines
  A platform understandable by millions
- Transparent without user interference

P3 Project

- **Long-term vision**:
  Exploit abundant shared-memory machines
  A platform understandable by millions
- Transparent without user interference
- Less interaction with IT department

P3 Project

- **Long-term vision**:
  Exploit abundant shared-memory machines
  A platform understandable by millions
- Transparent without user interference
- Less interaction with IT department
- Shared-memory multi-core

Why Spreadsheets?

Why Spreadsheets?

- $>$ ~55 million end-user programmers in 2012 [SSM05]

Why Spreadsheets?

- $>$ ~55 million end-user programmers in 2012 [SSM05]
- ~13-25 million spreadsheet users

Why Spreadsheets?

- $>$ ~55 million end-user programmers in 2012 [SSM05]
- ~13-25 million spreadsheet users
- Complex models in biology, physics, economy, finance etc.

Research Opportunities

Research Opportunities

- A tool for *end-user software development* [Bur09]

Research Opportunities

- A tool for *end-user software development* [Bur09]
- Better tools are desirable

Research Opportunities

- A tool for *end-user software development* [Bur09]
- Better tools are desirable
- Spreadsheets are functional and side-effect free

Research Opportunities

- A tool for *end-user software development* [Bur09]
- Better tools are desirable
- Spreadsheets are functional and side-effect free
  - ↳ **Opportunity 1:** Functions, closures and specialization

Research Opportunities

- A tool for *end-user software development* [Bur09]
- Better tools are desirable
- Spreadsheets are functional and side-effect free
  - ↳ **Opportunity 1:** Functions, closures and specialization
  - ↳ **Opportunity 2:** Apply functional programming paradigms to spreadsheets

Research Opportunities

- A tool for *end-user software development* [Bur09]
- Better tools are desirable
- Spreadsheets are functional and side-effect free
  - ↳ **Opportunity 1:** Functions, closures and specialization
  - ↳ **Opportunity 2:** Apply functional programming paradigms to spreadsheets
- Computation order restricted only by dependencies

Research Opportunities

- A tool for *end-user software development* [Bur09]
- Better tools are desirable
- Spreadsheets are functional and side-effect free
  - ↳ **Opportunity 1:** Functions, closures and specialization
  - ↳ **Opportunity 2:** Apply functional programming paradigms to spreadsheets
- Computation order restricted only by dependencies
  - ↳ **Opportunity 3:** Automatic parallelization

Introduction

Sheet-defined Functions

Functional Paradigms In Spreadsheets

Dataflow Computation

The Problem

Opportunity 1

- Task: Compute the area of triangle with sides a, b and c

The Problem

Opportunity 1

- Task: Compute the area of triangle with sides a, b and c

Area of a triangle (Heron's formula)

$\sqrt{s(s-a)(s-b)(s-c)}$, where $s = \frac{a+b+c}{2}$

The Problem

Opportunity 1

- Task: Compute the area of triangle with sides a, b and c

Area of a triangle (Heron's formula)

$\sqrt{s(s-a)(s-b)(s-c)}$, where $s = \frac{a+b+c}{2}$

- Compute s in column `D`

6					
7	a	b	c	s	area
8	3	4	5	=(A8+B8+C8)/2	=SQRT(D8*(D8-A8)*(D8-B8)*(D8-C8))
9	30	40	50	=(A9+B9+C9)/2	=SQRT(D9*(D9-A9)*(D9-B9)*(D9-C9))
10	100	100	100	=(A10+B10+C10)/2	=SQRT(D10*(D10-A10)*(D10-B10)*(D10-C10))
11					

The Problem

Opportunity 1

- Task: Compute the area of triangle with sides a, b and c

Area of a triangle (Heron's formula)

$$\sqrt{s(s-a)(s-b)(s-c)}, \text{ where } s = \frac{a+b+c}{2}$$

- Compute s in column `D`

6					
7	a	b	c	s	area
8	3	4	5	=(A8+B8+C8)/2	=SQRT(D8*(D8-A8)*(D8-B8)*(D8-C8))
9	30	40	50	=(A9+B9+C9)/2	=SQRT(D9*(D9-A9)*(D9-B9)*(D9-C9))
10	100	100	100	=(A10+B10+C10)/2	=SQRT(D10*(D10-A10)*(D10-B10)*(D10-C10))
11					

- Even more verbose if we exclude intermediate computations

6				
7	a	b	c	area
8	3	4	5	=SQRT((A8+B8+C8)/2*((A8+B8+C8)/2-A8)*((A8+B8+C8)/2-B8)*((A8+B8+C8)/2-C8))
9	30	40	50	=SQRT((A9+B9+C9)/2*((A9+B9+C9)/2-A9)*((A9+B9+C9)/2-B9)*((A9+B9+C9)/2-C9))
10	100	100	100	=SQRT((A10+B10+C10)/2*((A10+B10+C10)/2-A10)*((A10+B10+C10)/2-B10)*((A10+B10+C10)/2-C10))
11				

Issues with Formulas

Opportunity 1

Issues with Formulas

Opportunity 1

- Error-prone [Pan98; Pan15; Her+13; Zha+16; AE06]...

Issues with Formulas

Opportunity 1

- Error-prone [Pan98; Pan15; Her+13; Zha+16; AE06]...
- Several tools developed to combat errors [Boc16]

Issues with Formulas

Opportunity 1

- Error-prone [Pan98; Pan15; Her+13; Zha+16; AE06]...
- Several tools developed to combat errors [Boc16]
- Not readily distributable [HPD11]

Issues with Formulas

Opportunity 1

- Error-prone [Pan98; Pan15; Her+13; Zha+16; AE06]...
- Several tools developed to combat errors [Boc16]
- Not readily distributable [HPD11]
- Not particularly readable, no (self-)documentation

Issues with Formulas

Opportunity 1

- Error-prone [Pan98; Pan15; Her+13; Zha+16; AE06]...
- Several tools developed to combat errors [Boc16]
- Not readily distributable [HPD11]
- Not particularly readable, no (self-)documentation
- DRY principle

Issues with Formulas

Opportunity 1

- Error-prone [Pan98; Pan15; Her+13; Zha+16; AE06]...
- Several tools developed to combat errors [Boc16]
- Not readily distributable [HPD11]
- Not particularly readable, no (self-)documentation
- DRY principle
- Potential economic risk (EuSpRiG group [EuS])

Issues with Formulas

Opportunity 1

- Error-prone [Pan98; Pan15; Her+13; Zha+16; AE06]...
- Several tools developed to combat errors [Boc16]
- Not readily distributable [HPD11]
- Not particularly readable, no (self-)documentation
- DRY principle
- Potential economic risk (EuSpRiG group [EuS])
  - ↳ Important decisions based on incorrect data

Issues with Formulas

Opportunity 1

- Error-prone [Pan98; Pan15; Her+13; Zha+16; AE06]...
- Several tools developed to combat errors [Boc16]
- Not readily distributable [HPD11]
- Not particularly readable, no (self-)documentation
- DRY principle
- Potential economic risk (EuSpRiG group [EuS])
  - ↳ Important decisions based on incorrect data
  - ↳ Errors in published works

Issues with Formulas

Opportunity 1

- Error-prone [Pan98; Pan15; Her+13; Zha+16; AE06]...
- Several tools developed to combat errors [Boc16]
- Not readily distributable [HPD11]
- Not particularly readable, no (self-)documentation
- DRY principle
- Potential economic risk (EuSpRiG group [EuS])
  - ↳ Important decisions based on incorrect data
  - ↳ Errors in published works
  - ↳ Errors in million dollar budgets

Proposed Solution: Sheet-Defined Functions

Opportunity 1

Proposed Solution: Sheet-Defined Functions

Opportunity 1

Originally proposed by Simon Peyton-Jones et al. [PBB03; Ben09]

Proposed Solution: Sheet-Defined Functions

Opportunity 1

Originally proposed by Simon Peyton-Jones et al. [PBB03; Ben09]

- Powerful concept for end-user development

Proposed Solution: Sheet-Defined Functions

Opportunity 1

Originally proposed by Simon Peyton-Jones et al. [PBB03; Ben09]

- Powerful concept for end-user development
- Defined using a paradigm they already understand

Proposed Solution: Sheet-Defined Functions

Opportunity 1

Originally proposed by Simon Peyton-Jones et al. [PBB03; Ben09]

- Powerful concept for end-user development
- Defined using a paradigm they already understand
- No external languages

Proposed Solution: Sheet-Defined Functions

Opportunity 1

Originally proposed by Simon Peyton-Jones et al. [PBB03; Ben09]

- Powerful concept for end-user development
- Defined using a paradigm they already understand
- No external languages
- Some languages are slow (like VB.NET)

Proposed Solution: Sheet-Defined Functions

Opportunity 1

Originally proposed by Simon Peyton-Jones et al. [PBB03; Ben09]

- Powerful concept for end-user development
- Defined using a paradigm they already understand
- No external languages
- Some languages are slow (like VB.NET)
- Use runtime compilation for performance

Proposed Solution: Sheet-Defined Functions

Opportunity 1

Originally proposed by Simon Peyton-Jones et al. [PBB03; Ben09]

- Powerful concept for end-user development
- Defined using a paradigm they already understand
- No external languages
- Some languages are slow (like VB.NET)
- Use runtime compilation for performance

"Can you imagine programming in C without procedures, however clever the editor's copy-and-paste technology?" [PBB03]

Sheet-defined Function `TRIAREA`

Opportunity 1

- Task: Compute the area of triangle with sides *a*, *b* and *c*

Sheet-defined Function `TRIAREA`

Opportunity 1

- Task: Compute the area of triangle with sides *a*, *b* and *c*
- Define SDF in a *function sheet*.

Sheet-defined Function `TRIAREA`

Opportunity 1

- Task: Compute the area of triangle with sides *a*, *b* and *c*
- Define SDF in a *function sheet*.

E6	A	B	C	D	E	F
1	'Area of ...					
2	'a	'b	'c	's	'area	
3	3	4	5	=(A3+B3+C3)/2	=SQRT(D3*(D3-A3)*(D3-B3)*(D3-C3))	
4					=DEFINE("triarea", E3, A3, B3, C3)	
5						

Sheet-defined Function `TRIAREA`

Opportunity 1

- Task: Compute the area of triangle with sides *a*, *b* and *c*
- Define SDF in a *function sheet*.

E6	A	B	C	D	E	F
1	'Area of ...					
2	'a	'b	'c	's	'area	
3	3	4	5	=(A3+B3+C3)/2	=SQRT(D3*(D3-A3)*(D3-B3)*(D3-C3))	
4					=DEFINE("triarea", E3, A3, B3, C3)	
5						

- Callable from any sheet

Sheet-defined Function `TRIAREA`

Opportunity 1

- Task: Compute the area of triangle with sides *a*, *b* and *c*
- Define SDF in a *function sheet*.

E6	A	B	C	D	E	F
1	'Area of ...					
2	'a	'b	'c	's	'area	
3	3	4	5	=(A3+B3+C3)/2	=SQRT(D3*(D3-A3)*(D3-B3)*(D3-C3))	
4					=DEFINE("triarea", E3, A3, B3, C3)	

- Callable from any sheet

7	a	b	c	area
8	3	4	5	=TRIAREA(A8;B8;C8)
9	30	40	50	600
10	100	100	100	4330.12701892219

Dual Implementation

Opportunity 1

Dual Implementation

Opportunity 1

- Ordinary sheets

Dual Implementation

Opportunity 1

- Ordinary sheets: Frequently edited, rarely evaluated in full

Dual Implementation

Opportunity 1

- Ordinary sheets: Frequently edited, rarely evaluated in full
- Function sheets

Dual Implementation

Opportunity 1

- Ordinary sheets: Frequently edited, rarely evaluated in full
- Function sheets: Rarely edited, frequently evaluated

Higher-Order Functions

Opportunity 1

Higher-Order Functions

Opportunity 1

- `=CLOSURE("name", `a_1`, ...)` $\Rightarrow$ partially applied SDF

Higher-Order Functions

Opportunity 1

- `=CLOSURE("name", `a_1`, ...)` $\Rightarrow$ partially applied SDF
- Use `=NA()` for late-bound arguments

Higher-Order Functions

Opportunity 1

- `=CLOSURE("name", `a_1`, ...)` $\Rightarrow$ partially applied SDF
- Use `=NA()` for late-bound arguments
- `A1=CLOSURE("TRIAREA", 10, 20, NA())`

Higher-Order Functions

Opportunity 1

- `=CLOSURE("name", `a_1`, ...)` $\Rightarrow$ partially applied SDF
- Use `=NA()` for late-bound arguments
- `A1=CLOSURE("TRIAREA", 10, 20, NA())`
- `=APPLY(f, `b_1`, ...)` applies the function value `f`

Higher-Order Functions

Opportunity 1

- `=CLOSURE("name", `a_1`, ...)` $\Rightarrow$ partially applied SDF
- Use `=NA()` for late-bound arguments
- `A1=CLOSURE("TRIAREA", 10, 20, NA())`
- `=APPLY(f, `b_1`, ...)` applies the function value `f`
- `=APPLY(A1, 30)`

Higher-Order Functions

Opportunity 1

- `=CLOSURE("name", `a_1`, ...)` $\Rightarrow$ partially applied SDF
- Use `=NA()` for late-bound arguments
- `A1=CLOSURE("TRIAREA", 10, 20, NA())`
- `=APPLY(f, `b_1`, ...)` applies the function value `f`
- `=APPLY(A1, 30)`
- `=MAP(f, [`x_1`, `x_2`, ..., `x_n`])`

Example: A General N-sided Die

Opportunity 1

Example: A General N-sided Die

Opportunity 1

30	General n-side die	
31	n =	6
▶ 32	eyes =	=FLOOR(RAND()*B31;1)+1

Example: A General N-sided Die

Opportunity 1

30	General n-side die	
31	n =	6
▶ 32	eyes =	=FLOOR(RAND()*B31;1)+1

28	=CLOSURE("ndie", 6)	=CLOSURE("ndie", 20)
29	=APPLY(A$28)	=APPLY(B$28)
30	=APPLY(A$28)	=APPLY(B$28)
31	=APPLY(A$28)	=APPLY(B$28)
32	=APPLY(A$28)	=APPLY(B$28)
33	=APPLY(A$28)	=APPLY(B$28)

Example: A General N-sided Die

Opportunity 1

30	General n-side die	
31	n =	6
▶ 32	eyes =	=FLOOR(RAND()*B31;1)+1

28	=CLOSURE("ndie", 6)	=CLOSURE("ndie", 20)
29	=APPLY(A$28)	=APPLY(B$28)
30	=APPLY(A$28)	=APPLY(B$28)
31	=APPLY(A$28)	=APPLY(B$28)
32	=APPLY(A$28)	=APPLY(B$28)
33	=APPLY(A$28)	=APPLY(B$28)

28	NDIE(6)	NDIE(20)
29	6	6
30	2	1
31	1	13
32	4	14
33	6	17

Expressiveness of SDFs

Opportunity 1

- What can we express?

Expressiveness of SDFs

Opportunity 1

- What can we express?
  - Reimplemented Excel financial functions in Funcalc [Sør12]

Expressiveness of SDFs

Opportunity 1

- What can we express?
  - Reimplemented Excel financial functions in Funcalc [Sør12]
  - Reimplemented other common Excel functions like `GOALSEEK`, `VLOOKUP`, ... [Ses14]

Expressiveness of SDFs

Opportunity 1

- What can we express?
  - Reimplemented Excel financial functions in Funcalc [Sør12]
  - Reimplemented other common Excel functions like `GOALSEEK`, `VLOOKUP`, ... [Ses14]
  - Translating 16 SISAL programs to Funcalc [Can]

SDF Performance

Opportunity 1

Function	Excel Built-in (ns)	SDF (ns)
PV	1461	804
FV	1445	1138
NPER	1055	472
RATE	2297	44864
PMT	1523	664
FVSCHEDULE	2960	928
IPMT	1593	1732
PPMT	1805	1292
CUMIPMT	3117	3400
CUMPRINC	2742	4072
ISPMT	468	170

TABLE: Performance of Excel Financial Functions vs. SDFs

Advantages and disadvantages

Opportunity 1

Advantages and disadvantages

Opportunity 1

✓ A modular, reusable tool for end-user development

Advantages and disadvantages

Opportunity 1

- ✓ A modular, reusable tool for end-user development
- ✓ More readable than formulas + documentation

Advantages and disadvantages

Opportunity 1

- ✓ A modular, reusable tool for end-user development
- ✓ More readable than formulas + documentation
- ✓ Avoids multiple copies of formulas $\rightarrow$ less error-prone

Advantages and disadvantages

Opportunity 1

- ✓ A modular, reusable tool for end-user development
- ✓ More readable than formulas + documentation
- ✓ Avoids multiple copies of formulas $\rightarrow$ less error-prone
- ✓ Promotes easy sharing as “libraries”

Advantages and disadvantages

Opportunity 1

- ✓ A modular, reusable tool for end-user development
- ✓ More readable than formulas + documentation
- ✓ Avoids multiple copies of formulas $\rightarrow$ less error-prone
- ✓ Promotes easy sharing as “libraries”
- ✓ High level of expressive power

Advantages and disadvantages

Opportunity 1

- ✓ A modular, reusable tool for end-user development
- ✓ More readable than formulas + documentation
- ✓ Avoids multiple copies of formulas $\rightarrow$ less error-prone
- ✓ Promotes easy sharing as “libraries”
- ✓ High level of expressive power
- ✗ Still need to understand complex concepts such as recursion and closures

Advantages and disadvantages

Opportunity 1

- ✓ A modular, reusable tool for end-user development
- ✓ More readable than formulas + documentation
- ✓ Avoids multiple copies of formulas $\rightarrow$ less error-prone
- ✓ Promotes easy sharing as “libraries”
- ✓ High level of expressive power
- ✗ Still need to understand complex concepts such as recursion and closures
- ✗ Currently few debugging tools and general support

★ Demo time! ★

Introduction

Sheet-defined Functions

Functional Paradigms In Spreadsheets

Dataflow Computation

Function Fusion and Transformation

Opportunity 2

Function Fusion and Transformation

Opportunity 2

- What can we borrow from functional programming?

Function Fusion and Transformation

Opportunity 2

- What can we borrow from functional programming?
- `=MAP(F1, MAP(F2, `*array*`))`

Function Fusion and Transformation

Opportunity 2

- What can we borrow from functional programming?
- `=MAP(F1, MAP(F2,` *array*`))`

  `F3 = F1 ∘ F2`

Function Fusion and Transformation

Opportunity 2

- What can we borrow from functional programming?
- `=MAP(F1, MAP(F2, `*array*`))`

  `F3 = F1 ∘ F2`

  Map fusion: Rewrite as `=MAP(F3, `*array*`)`

Function Fusion and Transformation

Opportunity 2

- What can we borrow from functional programming?
- `=MAP(F1, MAP(F2, `*array*`))`

  `F3 = F1 ∘ F2`

  Map fusion: Rewrite as `=MAP(F3, `*array*`)`
- Cannot blindly apply to a spreadsheet

Function Fusion and Transformation

Opportunity 2

- What can we borrow from functional programming?
- `=MAP(F1, MAP(F2, `*array*`))`

  `F3 = F1 ∘ F2`

  Map fusion: Rewrite as `=MAP(F3, `*array*`)`
- Cannot blindly apply to a spreadsheet
  - Only applicable to formulas in the same cell

Function Fusion and Transformation

Opportunity 2

- What can we borrow from functional programming?
- `=MAP(F1, MAP(F2, `*array*`))`

  `F3 = F1 ∘ F2`

  Map fusion: Rewrite as `=MAP(F3, `*array*`)`
- Cannot blindly apply to a spreadsheet
  - Only applicable to formulas in the same cell
  - How to display otherwise?

Anonymous Closures

Opportunity 2

Anonymous Closures

Opportunity 2

- Consider `=MAP(CLOSURE("MULT3_ADD2_COS"), array)`

Anonymous Closures

Opportunity 2

- Consider `=MAP(CLOSURE("MULT3_ADD2_COS"), array)`
- Requires definition of `MULT3_ADD2_COS` (`MULT3`, `ADD2`)

Anonymous Closures

Opportunity 2

- Consider `=MAP(CLOSURE("MULT3_ADD2_COS"), array)`
- Requires definition of `MULT3_ADD2_COS` (`MULT3`, `ADD2`)
- Simple expression, but quite verbose

Anonymous Closures

Opportunity 2

- Consider `=MAP(CLOSURE("MULT3_ADD2_COS"), array)`
- Requires definition of `MULT3_ADD2_COS` (`MULT3`, `ADD2`)
- Simple expression, but quite verbose
- `=MAP(COS(@1 * 3 + 2), array)`

Anonymous Closures

Opportunity 2

- Consider `=MAP(CLOSURE("MULT3_ADD2_COS"), array)`
- Requires definition of `MULT3_ADD2_COS` (`MULT3`, `ADD2`)
- Simple expression, but quite verbose
- `=MAP(COS(@1 * 3 + 2), array)`
- `@1` refers to first argument

Anonymous Closures

Opportunity 2

- Consider `=MAP(CLOSURE("MULT3_ADD2_COS"), array)`
- Requires definition of `MULT3_ADD2_COS` (`MULT3`, `ADD2`)
- Simple expression, but quite verbose
- `=MAP(COS(@1 * 3 + 2), array)`
- `@1` refers to first argument
- `@1`, `@2`, `...`, `@N`

Anonymous Closures

Opportunity 2

- Consider `=MAP(CLOSURE("MULT3_ADD2_COS"), array)`
- Requires definition of `MULT3_ADD2_COS` (`MULT3`, `ADD2`)
- Simple expression, but quite verbose
- `=MAP(COS(@1 * 3 + 2), array)`
- `@1` refers to first argument
- `@1`, `@2`, ..., `@N`
- `@* = [@1, @2, ..., @N]`

Anonymous Closures

Opportunity 2

- Consider `=MAP(CLOSURE("MULT3_ADD2_COS"), array)`
- Requires definition of `MULT3_ADD2_COS` (`MULT3`, `ADD2`)
- Simple expression, but quite verbose
- `=MAP(COS(@1 * 3 + 2), array)`
- `@1` refers to first argument
- `@1`, `@2`, `...`, `@N`
- `@* = [@1, @2, ..., @N]`
- Call to `CLOSURE` removed

Anonymous Closures

Opportunity 2

- Consider `=MAP(CLOSURE("MULT3_ADD2_COS"), array)`
- Requires definition of `MULT3_ADD2_COS` (`MULT3`, `ADD2`)
- Simple expression, but quite verbose
- `=MAP(COS(@1 * 3 + 2), array)`
- `@1` refers to first argument
- `@1`, `@2`, `...`, `@N`
- `@* = [@1, @2, ..., @N]`
- Call to `CLOSURE` removed
- Regenerate at load-time

Introduction

Sheet-defined Functions

Functional Paradigms In Spreadsheets

Dataflow Computation

Dataflow

Opportunity 3

Dataflow

Opportunity 3

- **Motivation:**

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:
  - SISAL

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:
  - SISAL
    ⇒ **S**treams and **I**terations In A **S**ingle **A**ssignment **L**anguage [Can; McG+85]

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:
  - SISAL
    ⇒ **S**treams and **I**terations In A **S**ingle **A**ssignment **L**anguage [Can; McG+85]
    - Functional (first-order) replacement for Fortran in scientific computing

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:
  - SISAL
    ⇒ **S**treams and **I**terations In A **S**ingle **A**ssignment **L**anguage [Can; McG+85]
  - Functional (first-order) replacement for Fortran in scientific computing
  - Optimising compiler for automatically extracting implicit parallelism [Sar89]

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:
  - SISAL
    ⇒ **S**treams and **I**terations In A **S**ingle **A**ssignment **L**anguage [Can; McG+85]
  - Functional (first-order) replacement for Fortran in scientific computing
  - Optimising compiler for automatically extracting implicit parallelism [Sar89]
    - *Compile-time* partitioning

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:
  - SISAL
    ⇒ **S**treams and **I**terations In A **S**ingle **A**ssignment **L**anguage [Can; McG+85]
  - Functional (first-order) replacement for Fortran in scientific computing
  - Optimising compiler for automatically extracting implicit parallelism [Sar89]
    - *Compile-time* partitioning
    - *Runtime* scheduling

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:
  - SISAL
    ⇒ **S**treams and **I**terations In A **S**ingle **A**ssignment **L**anguage [Can; McG+85]
  - Functional (first-order) replacement for Fortran in scientific computing
  - Optimising compiler for automatically extracting implicit parallelism [Sar89]
    - *Compile-time* partitioning
    - *Runtime* scheduling
  - Ran on par with Fortran on Cray-Y-MP/864 shared-memory multi-core machine

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:
  - SISAL
    ⇒ **S**treams and **I**terations In A **S**ingle **A**ssignment **L**anguage [Can; McG+85]
  - Functional (first-order) replacement for Fortran in scientific computing
  - Optimising compiler for automatically extracting implicit parallelism [Sar89]
    - *Compile-time* partitioning
    - *Runtime* scheduling
  - Ran on par with Fortran on Cray-Y-MP/864 shared-memory multi-core machine
  - Spreadsheets (dataflow) + (higher-order) SDFs ⇒ SISAL

Dataflow

Opportunity 3

- **Motivation:** Parallel Recalculation
- Background:
  - SISAL
    $\Rightarrow$ **S**treams and **I**terations In A **S**ingle **A**ssignment **L**anguage [Can; McG+85]
  - Functional (first-order) replacement for Fortran in scientific computing
  - Optimising compiler for automatically extracting implicit parallelism [Sar89]
    - *Compile-time* partitioning
    - *Runtime* scheduling
  - Ran on par with Fortran on Cray-Y-MP/864 shared-memory multi-core machine
  - Spreadsheets (dataflow) + (higher-order) SDFs $\Rightarrow$ SISAL
- **Project idea**: Revisit and modernize Sarkar's work for spreadsheets

Target Audience

Opportunity 3

Target Audience

Opportunity 3

- Not your everyday spreadsheet user

Target Audience

Opportunity 3

- Not your everyday spreadsheet user
- Not HPC communities

Target Audience

Opportunity 3

- Not your everyday spreadsheet user
- Not HPC communities
- Spreadsheet users with large datasets:

Target Audience

Opportunity 3

- Not your everyday spreadsheet user
- Not HPC communities
- Spreadsheet users with large datasets:
  - Their primary computational model

Target Audience

Opportunity 3

- Not your everyday spreadsheet user
- Not HPC communities
- Spreadsheet users with large datasets:
  - Their primary computational model
  - No formal training in IT or programming

Algorithm Outline

Opportunity 3

1. **GR Graph Construction**
2. Cost Assignment
3. Partitioning
4. Task Scheduling

Algorithm Outline

Opportunity 3

1. **GR Graph Construction**
2. Cost Assignment
3. Partitioning
4. Task Scheduling

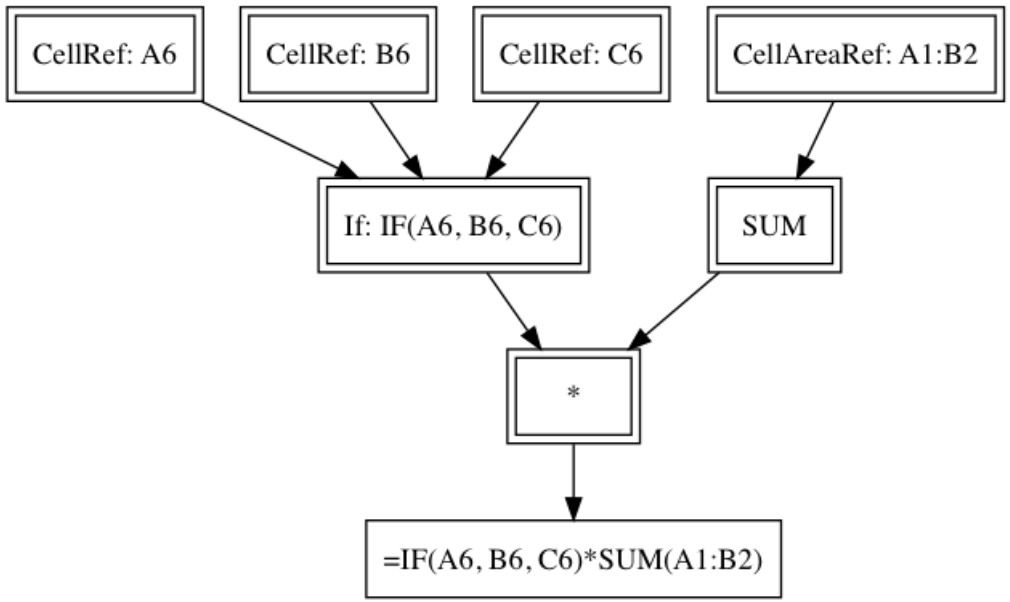
```
=IF(A6, B6, C6)*SUM(A1:B2)
```

Algorithm Outline

Opportunity 3

```
=IF(A6, B6, C6)*SUM(A1:B2)
```

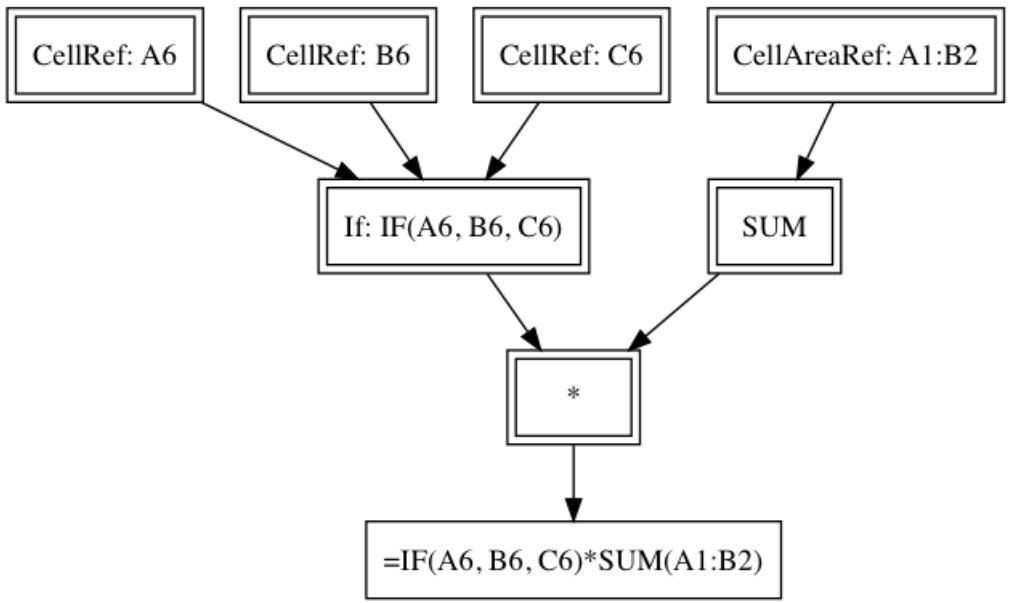
1. **GR Graph Construction**
2. Cost Assignment
3. Partitioning
4. Task Scheduling



Algorithm Outline

Opportunity 3

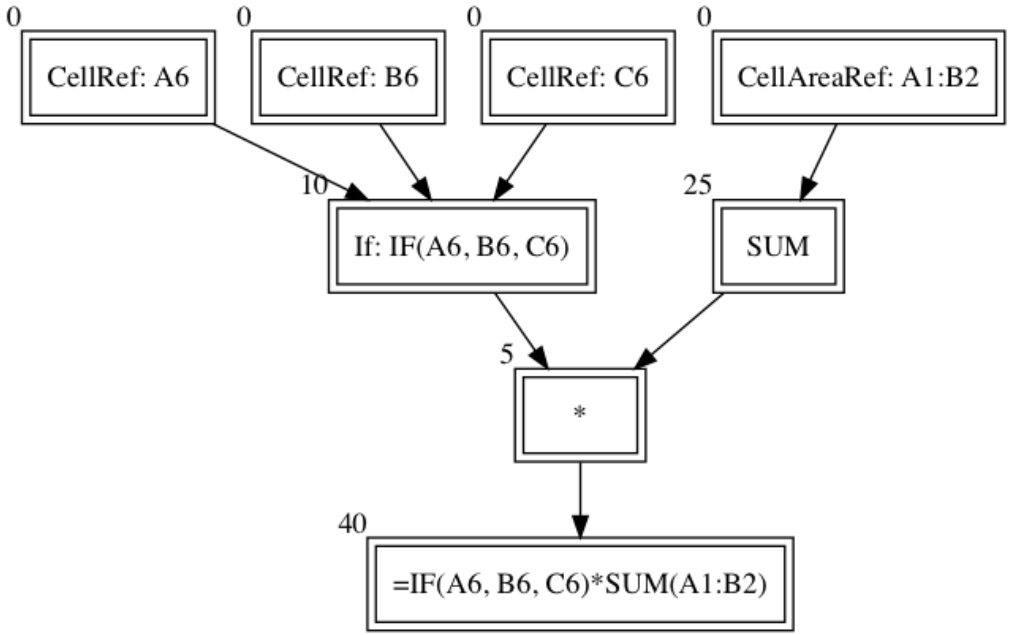
1. GR Graph Construction
2. **Cost Assignment**
3. Partitioning
4. Task Scheduling



Algorithm Outline

Opportunity 3

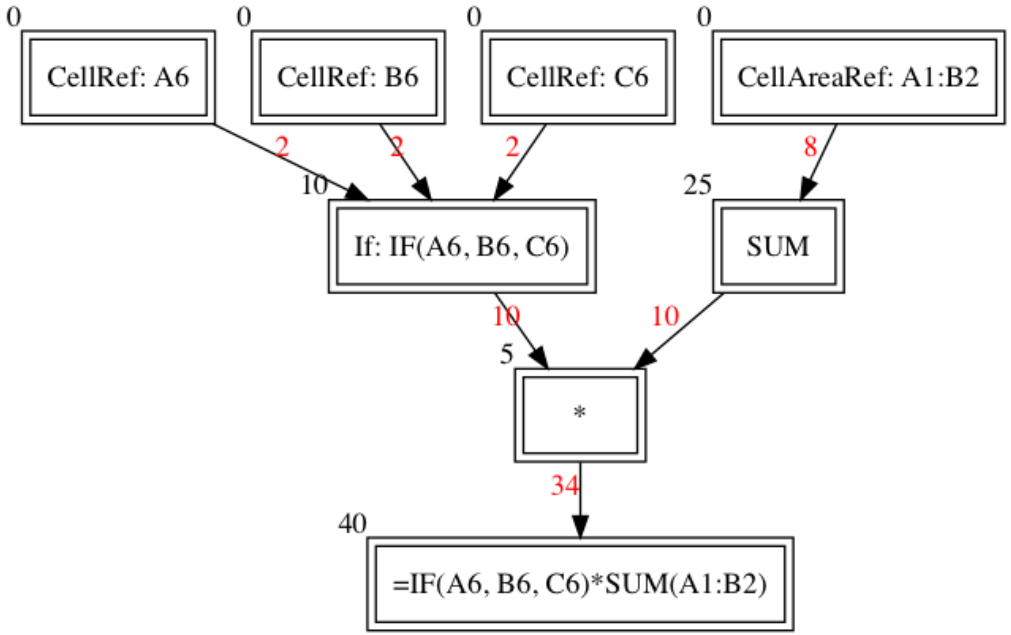
1. GR Graph Construction
2. **Cost Assignment**
3. Partitioning
4. Task Scheduling



Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. **Cost Assignment**
3. Partitioning
4. Task Scheduling



Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (I)**
4. Task Scheduling

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (I)**
4. Task Scheduling

- Objective function `F` balances:

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (I)**
4. Task Scheduling

- Objective function `F` balances:
  - Communication overhead

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (I)**
4. Task Scheduling

- Objective function `F` balances:
  - Communication overhead
  - Critical path cost

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (I)**
4. Task Scheduling

- Objective function `F` balances:
  - Communication overhead
  - Critical path cost
- *Fine* partition: Overhead term will dominate

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (I)**
4. Task Scheduling

- Objective function `F` balances:
  - Communication overhead
  - Critical path cost
- *Fine* partition: Overhead term will dominate
- *Coarse* partition: Critical path term will dominate

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (II)**
4. Task Scheduling

1. Partition the graph into task partitions

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (II)**
4. Task Scheduling

1. Partition the graph into task partitions
2. Put all nodes in a task by themselves

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (II)**
4. Task Scheduling

1. Partition the graph into task partitions
2. Put all nodes in a task by themselves
3. Iteratively merge pairs of tasks

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (II)**
4. Task Scheduling

1. Partition the graph into task partitions
2. Put all nodes in a task by themselves
3. Iteratively merge pairs of tasks

   Merge task with largest overhead

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (II)**
4. Task Scheduling

1. Partition the graph into task partitions
2. Put all nodes in a task by themselves
3. Iteratively merge pairs of tasks

   Merge task with largest overhead
4. Repeat until all nodes in a single task

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (II)**
4. Task Scheduling

1. Partition the graph into task partitions
2. Put all nodes in a task by themselves
3. Iteratively merge pairs of tasks

   Merge task with largest overhead
4. Repeat until all nodes in a single task
5. Record iteration `i` that minimised an objective function `F`

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. **Partitioning (II)**
4. Task Scheduling

1. Partition the graph into task partitions
2. Put all nodes in a task by themselves
3. Iteratively merge pairs of tasks

   Merge task with largest overhead
4. Repeat until all nodes in a single task
5. Record iteration `i` that minimised an objective function `F`
6. Reconstruct the $\texttt{i}^{th}$ task partition

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. Partitioning
4. **Task Scheduling**

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. Partitioning
4. **Task Scheduling**

- Ensure that task partitions are acyclic

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. Partitioning
4. **Task Scheduling**

- Ensure that task partitions are acyclic
- Once a task has all inputs

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. Partitioning
4. **Task Scheduling**

- Ensure that task partitions are acyclic
- Once a task has all inputs

  $\Rightarrow$ Run to completion

Algorithm Outline

Opportunity 3

1. GR Graph Construction
2. Cost Assignment
3. Partitioning
4. **Task Scheduling**

- Ensure that task partitions are acyclic
- Once a task has all inputs

  $\Rightarrow$ Run to completion
- Will use the Task Parallel Library [Mic; LSB09]

Spreadsheets Are A Different Paradigm

Opportunity 3

Spreadsheets Are A Different Paradigm

Opportunity 3

1. Cyclic cell dependencies

Spreadsheets Are A Different Paradigm

Opportunity 3

1. Cyclic cell dependencies
2. SDFs evaluate as compiled bytecode

Spreadsheets Are A Different Paradigm

Opportunity 3

1. Cyclic cell dependencies
2. SDFs evaluate as compiled bytecode
3. No “main” function + volatile cells

Spreadsheets Are A Different Paradigm

Opportunity 3

1. Cyclic cell dependencies
2. SDFs evaluate as compiled bytecode
3. No “main” function + volatile cells
4. Sarkar uses *compile-time partitioning*

Spreadsheets Are A Different Paradigm

Opportunity 3

1. Cyclic cell dependencies
2. SDFs evaluate as compiled bytecode
3. No “main” function + volatile cells
4. Sarkar uses *compile-time partitioning*
    - Spreadsheets have no concept of compile-time

Spreadsheets Are A Different Paradigm

Opportunity 3

1. Cyclic cell dependencies
2. SDFs evaluate as compiled bytecode
3. No “main” function + volatile cells
4. Sarkar uses *compile-time partitioning*
   - ▸ Spreadsheets have no concept of compile-time
   - ✗ Incurred runtime overhead

Spreadsheets Are A Different Paradigm

Opportunity 3

1. Cyclic cell dependencies
2. SDFs evaluate as compiled bytecode
3. No “main” function + volatile cells
4. Sarkar uses *compile-time partitioning*
   - ▸ Spreadsheets have no concept of compile-time
   - ✗ Incurred runtime overhead
   - ✓ Take advantage of runtime information

Other Ongoing Projects

Other Ongoing Projects

- Array Programming In Spreadsheets [Bie16]

Other Ongoing Projects

- Array Programming In Spreadsheets [Bie16]
- Excel add-in for Funcalc (student programmers)

Other Ongoing Projects

- Array Programming In Spreadsheets [Bie16]
- Excel add-in for Funcalc (student programmers)
- Expression rewriting

Other Ongoing Projects

- Array Programming In Spreadsheets [Bie16]
- Excel add-in for Funcalc (student programmers)
- Expression rewriting
- Transform copy-equivalent formulae into function calls

Other Ongoing Projects

- Array Programming In Spreadsheets [Bie16]
- Excel add-in for Funcalc (student programmers)
- Expression rewriting
- Transform copy-equivalent formulae into function calls
- Model-checking spreadsheet computations with UPPAAL [Uni15] (Aalborg university)

Other Ongoing Projects

- Array Programming In Spreadsheets [Bie16]
- Excel add-in for Funcalc (student programmers)
- Expression rewriting
- Transform copy-equivalent formulae into function calls
- Model-checking spreadsheet computations with UPPAAL [Uni15] (Aalborg university)
- Function fusion + anonymous closures

Additional Resources

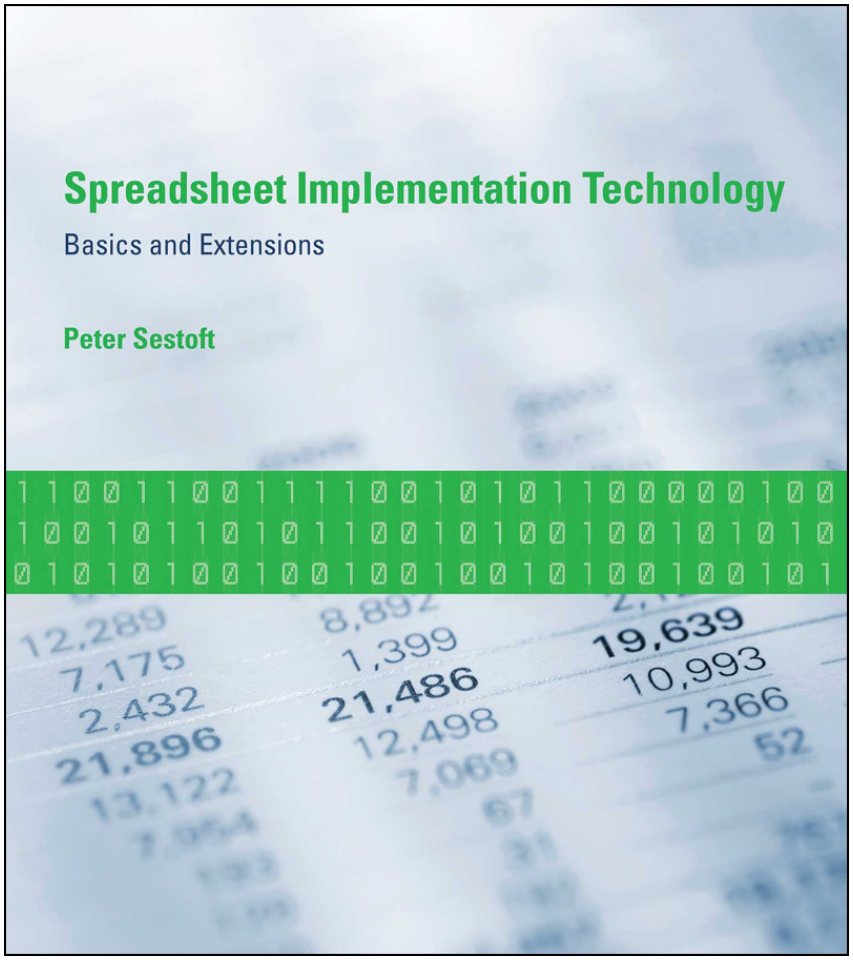
- Homepage: `http://www.itu.dk/people/sestoft/funcalc/`

Additional Resources

- Homepage: `http://www.itu.dk/people/sestoft/funcalc/`
- P3: `http://www.itu.dk/people/sestoft/p3/`

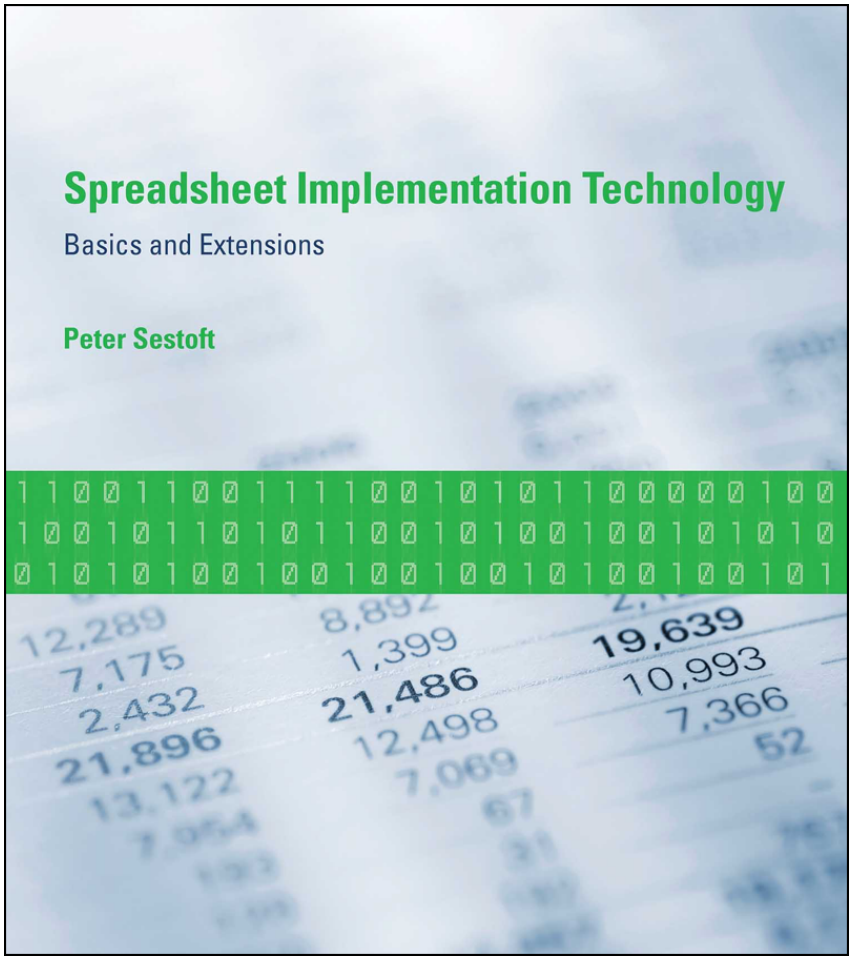
Additional Resources

- Homepage: `http://www.itu.dk/people/sestoft/funcalc/`
- P3: `http://www.itu.dk/people/sestoft/p3/`
- Book on Spreadsheet Technology, MIT Press [Ses14]



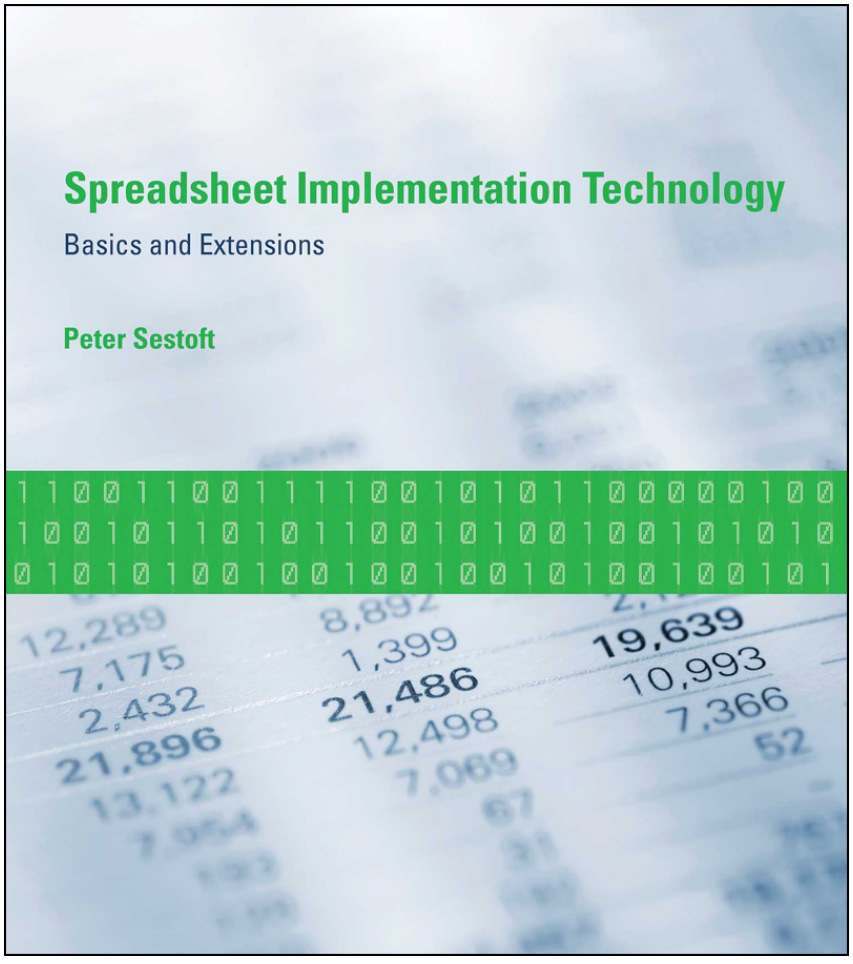
Additional Resources

- Homepage: `http://www.itu.dk/people/sestoft/funcalc/`
- P3: `http://www.itu.dk/people/sestoft/p3/`
- Book on Spreadsheet Technology, MIT Press [Ses14] Disclaimer!



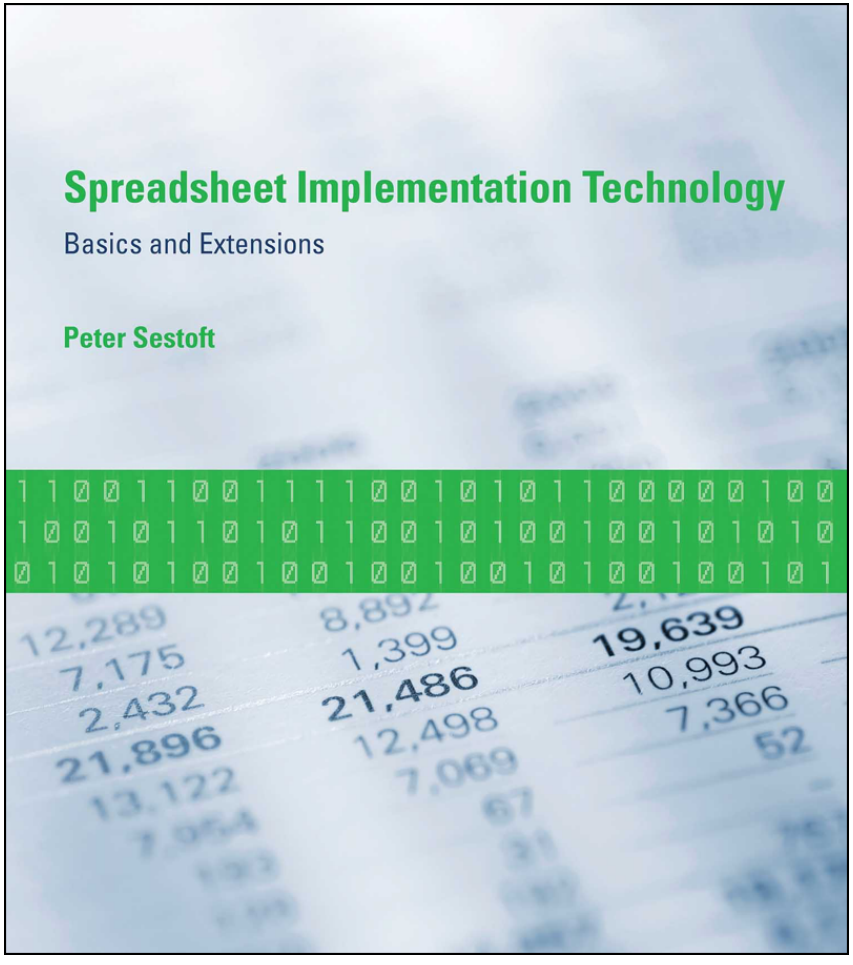
Additional Resources

- Homepage: `http://www.itu.dk/people/sestoft/funcalc/`
- P3: `http://www.itu.dk/people/sestoft/p3/`
- Book on Spreadsheet Technology, MIT Press [Ses14] Disclaimer!
- Funcalc is closed-source at the moment



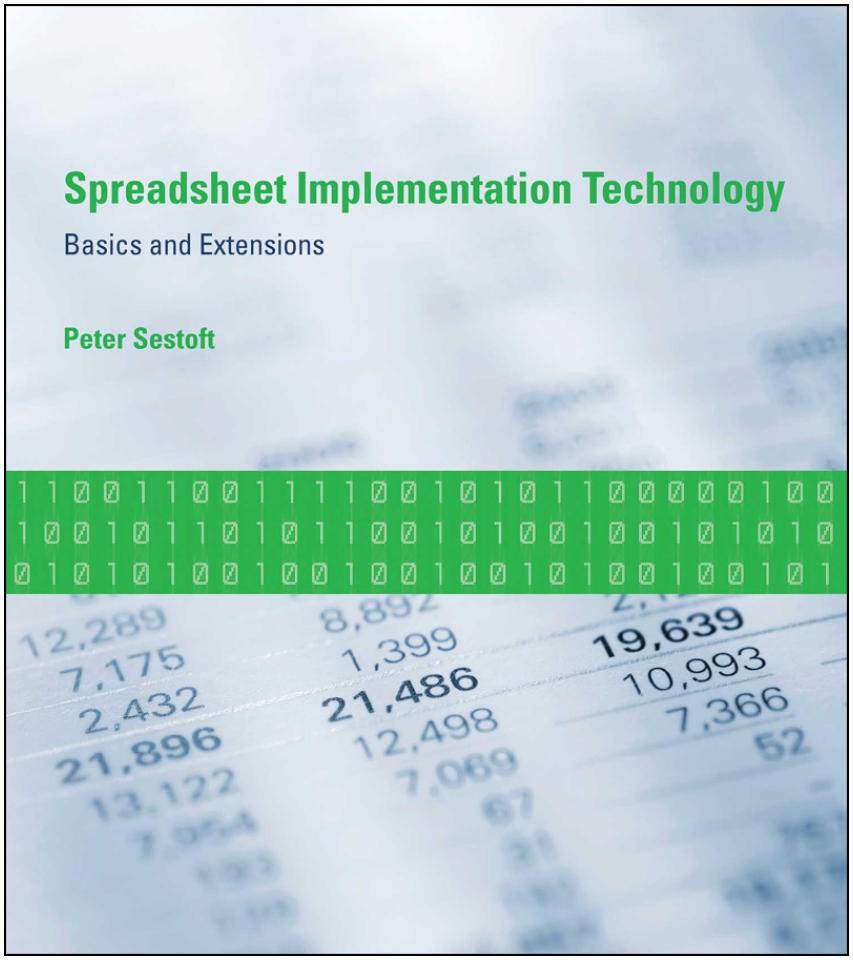
Additional Resources

- Homepage: `http://www.itu.dk/people/sestoft/funcalc/`
- P3: `http://www.itu.dk/people/sestoft/p3/`
- Book on Spreadsheet Technology, MIT Press [Ses14] Disclaimer!
- Funcalc is closed-source at the moment
- Funcalc project site: `http://www.itu.dk/people/sestoft/funcalc/`

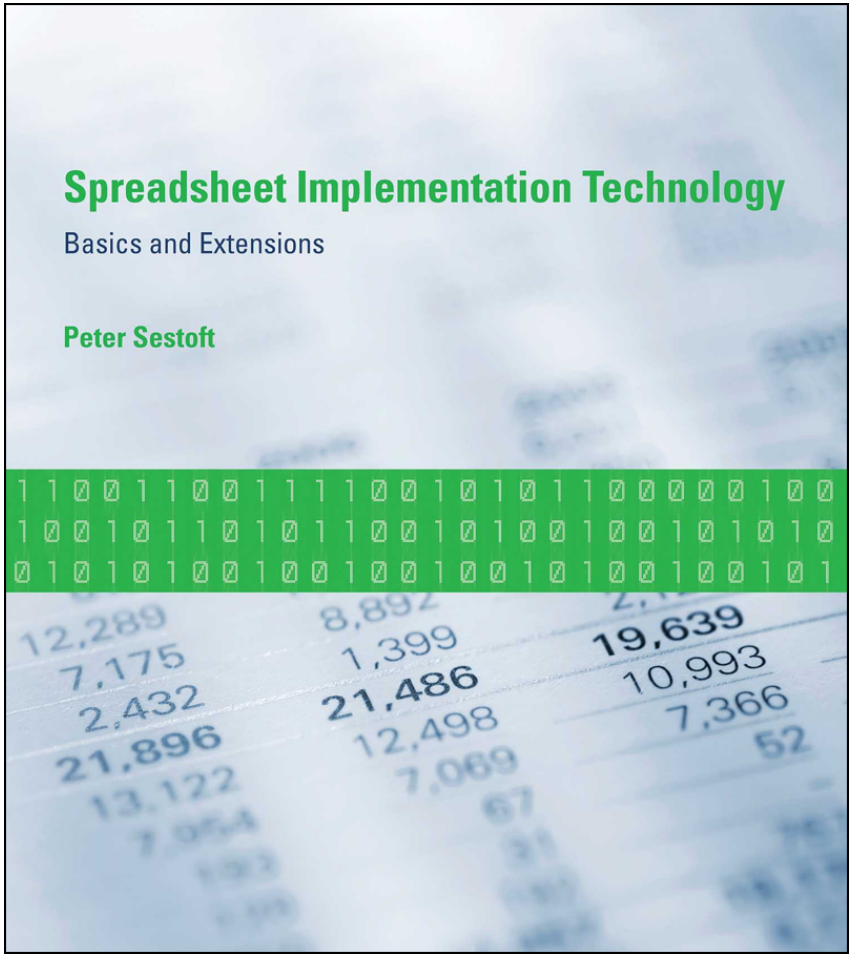


Additional Resources

- Homepage: `http://www.itu.dk/people/sestoft/funcalc/`
- P3: `http://www.itu.dk/people/sestoft/p3/`
- Book on Spreadsheet Technology, MIT Press [Ses14] Disclaimer!
- Funcalc is closed-source at the moment
- Funcalc project site: `http://www.itu.dk/people/sestoft/funcalc/`
- UPPAAL site: `http://www.uppaal.org/`

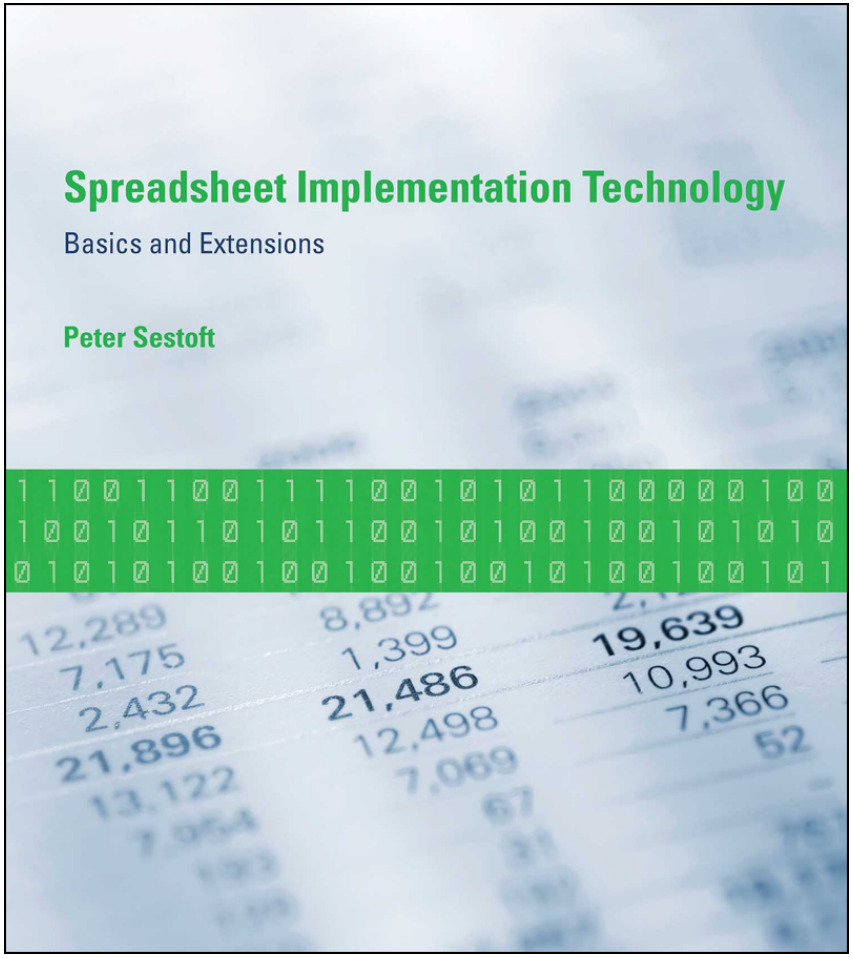


Additional Resources



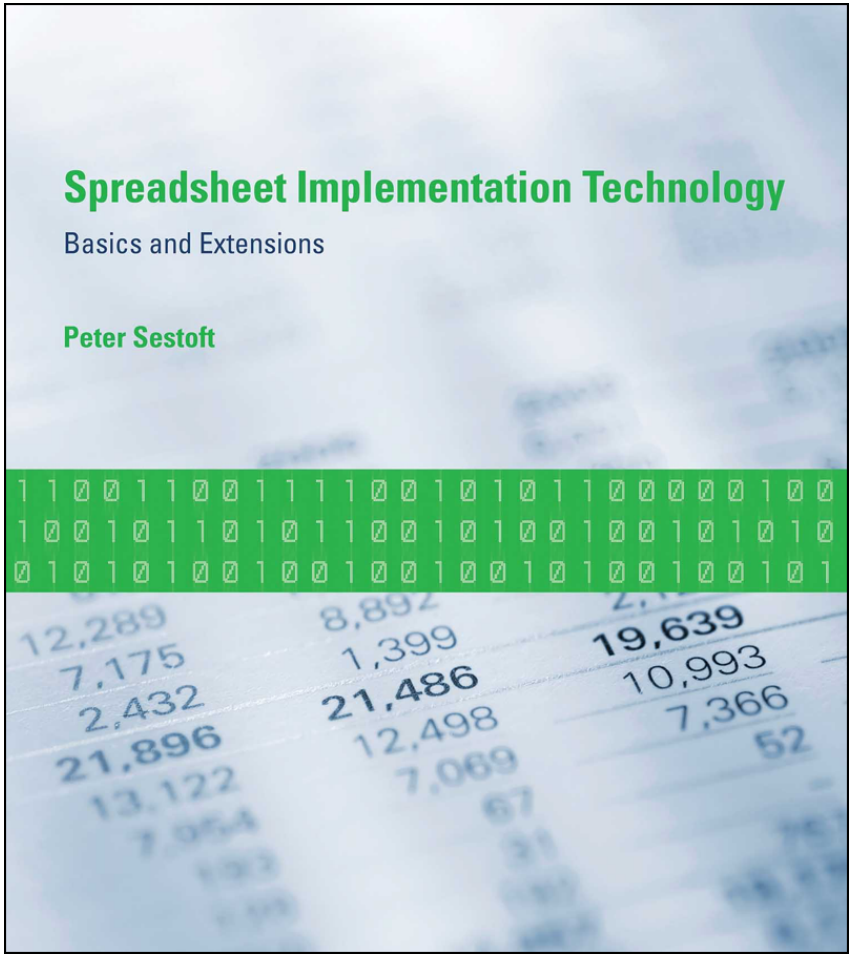
Additional Resources

- "*A Literature Review On Spreadsheet Technology*" [Boc16]



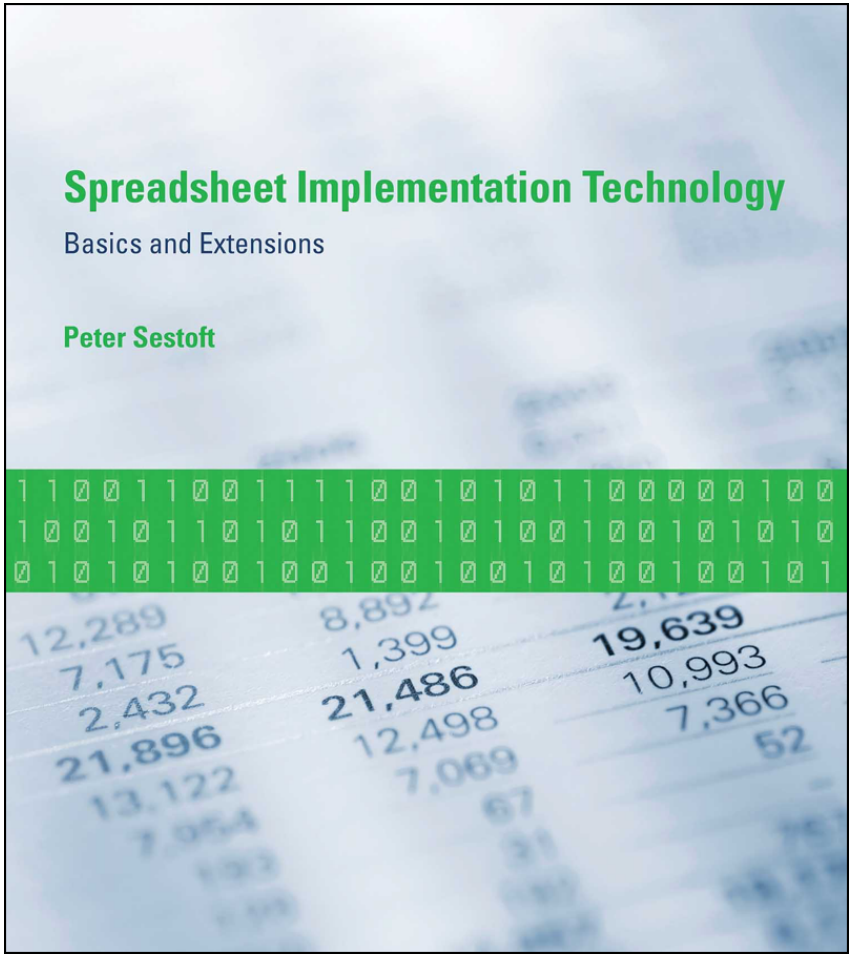
Additional Resources

- "*A Literature Review On Spreadsheet Technology*" [Boc16] Disclaimer!



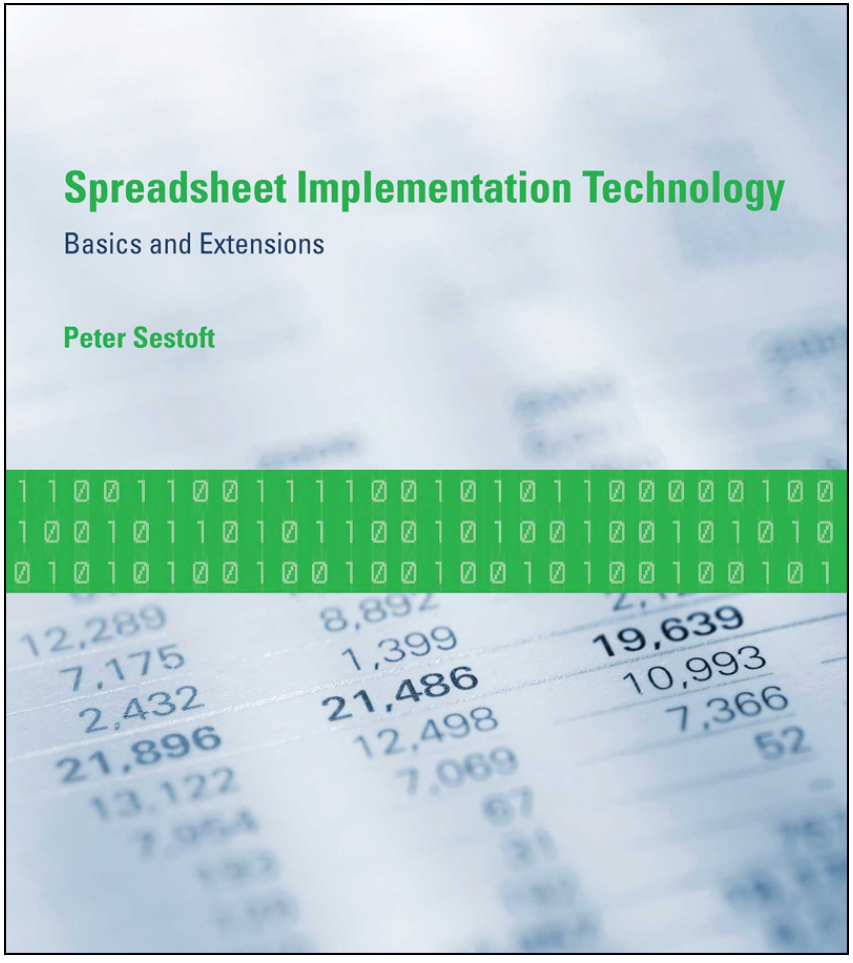
Additional Resources

- "*A Literature Review On Spreadsheet Technology*" [Boc16] Disclaimer!
- "*Declarative Parallel Programming in Spreadsheet End-User Development*" [Bie16]



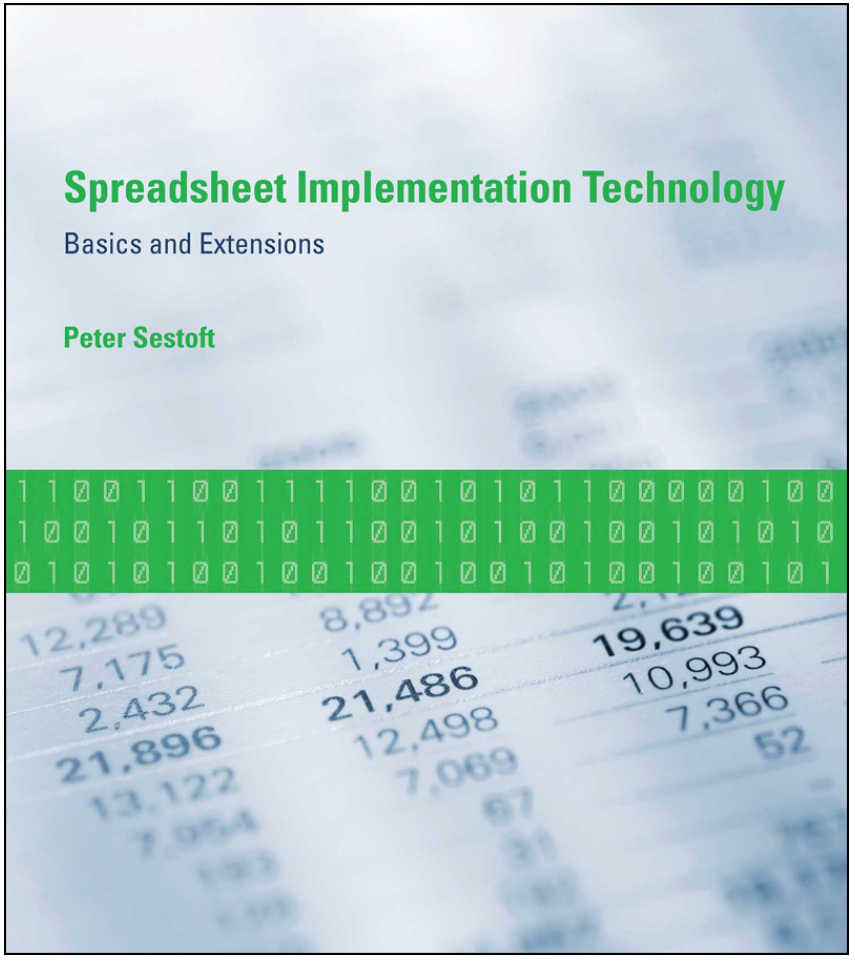
Additional Resources

- *"A Literature Review On Spreadsheet Technology"* [Boc16] Disclaimer!
- *"Declarative Parallel Programming in Spreadsheet End-User Development"* [Bie16] Disclaimer!



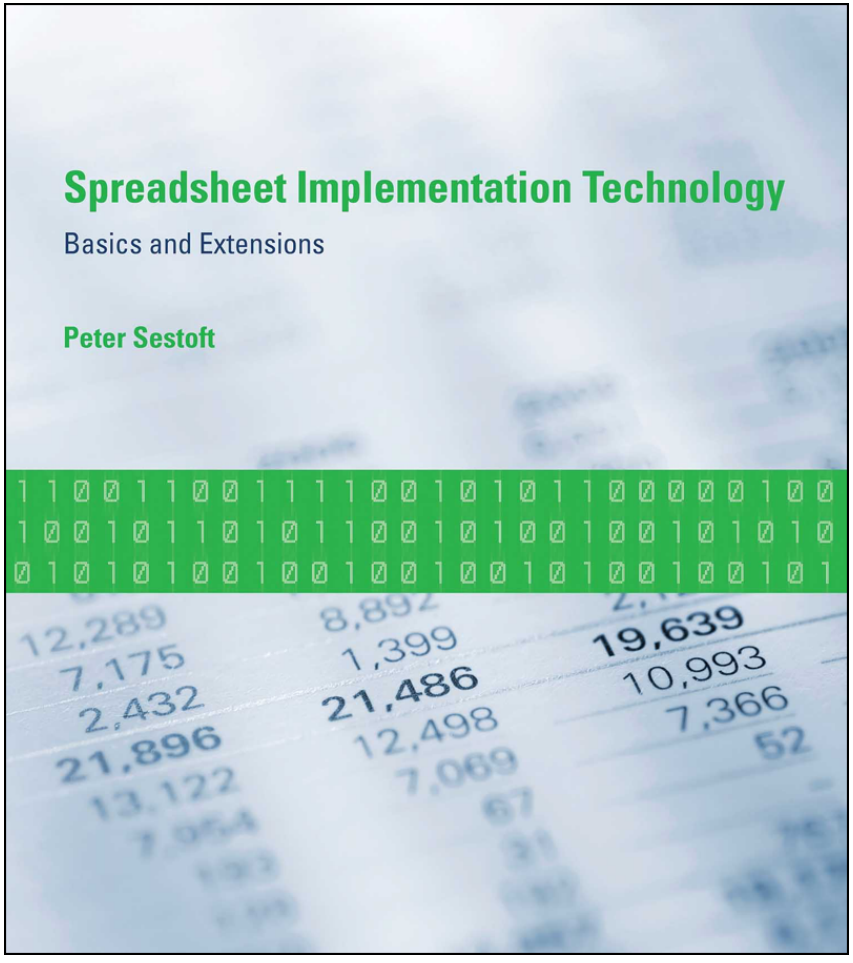
Additional Resources

- "*A Literature Review On Spreadsheet Technology*" [Boc16] Disclaimer!
- "*Declarative Parallel Programming in Spreadsheet End-User Development*" [Bie16] Disclaimer!
- SISAL language reference [McG+85]



Additional Resources

- "*A Literature Review On Spreadsheet Technology*" [Boc16] Disclaimer!
- "*Declarative Parallel Programming in Spreadsheet End-User Development*" [Bie16] Disclaimer!
- SISAL language reference [McG+85]
- SISAL tutorial + 16 example programs [Can]



Thank You For Your Attention!

Thank You For Your Attention!

Questions?

References I

Christopher Scaffidi, Mary Shaw, and Brad Myers. “Estimating the numbers of end users and end user programmers”. In: *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*. IEEE. 2005, pp. 207–214.

Margaret Burnett. “What Is End-User Software Engineering and Why Does It Matter?” In: *End-User Development: 2nd International Symposium, IS-EUD 2009, Siegen, Germany, March 2-4, 2009. Proceedings*. Ed. by Volkmar Pipek et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 15–28. ISBN: 978-3-642-00427-8. DOI: 10.1007/978-3-642-00427-8_2. URL: http://dx.doi.org/10.1007/978-3-642-00427-8_2.

References II

Raymond R. Panko. “What we know about spreadsheet errors”. In: *Journal of Organizational and End User Computing (JOEUC)* 10.2 (1998), pp. 15–21.

Raymond R. Panko. *What we dont know about spreadsheet errors today*. 2015.

Felienne Hermans et al. “Data Clone Detection and Visualization in Spreadsheets”. In: *Proceedings of the 2013 International Conference on Software Engineering*. ICSE ’13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 292–301. ISBN: 978-1-4673-3076-3. URL: http://dl.acm.org/citation.cfm?id=2486788.2486827.

References III

Ruiqing Zhang et al. “How effectively can spreadsheet anomalies be detected: An empirical study”. In: *Journal of Systems and Software* (2016). ISSN: 0164-1212. DOI: http://dx.doi.org/10.1016/j.jss.2016.03.061. URL: http://www.sciencedirect.com/science/article/pii/S0164121216300103.

Robin Abraham and Martin Erwig. “Type Inference for Spreadsheets”. In: *Proceedings of the 8th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*. PPDP ’06. New York, NY, USA: ACM, 2006, pp. 73–84. ISBN: 1-59593-388-3. DOI: 10.1145/1140335.1140346. URL: http://doi.acm.org/10.1145/1140335.1140346.

References IV


Alexander Asp Bock. *A Literature Review of Spreadsheet Technology*. technical report 199. IT University of Copenhagen, Nov. 30, 2016. 33 pp.


Felienne Hermans, Martin Pinzger, and Arie van Deursen. “Breviz: Visualizing spreadsheets using dataflow diagrams”. In: *arXiv preprint arXiv:1111.6895* (2011).


EuSpRiG. *EuSpRiG Horror Stories*. URL: `http://eusprig.org/horror-stories.htm` (visited on 06/14/2016).

References V

Simon Peyton-Jones, Alan Blackwell, and Margaret Burnett. “A User-centred Approach to Functions in Excel”. In: *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming*. ICFP ’03. New York, NY, USA: ACM, 2003, pp. 165–176. ISBN: 1-58113-756-7. DOI: 10.1145/944705.944721. URL: http://doi.acm.org/10.1145/944705.944721.

Lee Benfield. “FMD: Functional Development in Excel”. In: *Proceedings of the 2009 Video Workshop on Commercial Users of Functional Programming: Functional Programming As a Means, Not an End*. CUFP ’09. New York, NY, USA: ACM, 2009. ISBN: 978-1-60558-943-5. DOI: 10.1145/1668113.1668121. URL: http://doi.acm.org/10.1145/1668113.1668121.

References VI


Jens Zeilund Sørensen. “An Evaluation of Sheet-Defined Financial Functions in Funcalc”. Master Thesis. IT University of Copenhagen, 2012.


Peter Sestoft. *Spreadsheet Implementation Technology*. The MIT Press, 2014. ISBN: 9780262526647.


David C. Cann. *SISAL 1.2: A Brief Introduction and Tutorial*. Lawrence Livermore National Laboratory.


James McGraw et al. *SISAL: Streams and Iteration in a Single Assignment Language, Language Referece Manual*. Tech. rep. Version 1.2. Lawrence Livermore National Laboratory, Mar. 1, 1985.

References VII


Vivek Sarkar. *Partitioning and Scheduling Parallel Programs for Multiprocessors*. Research Monographs In Parallel and Distributed Computing. Cambridge, Massachusetts: MIT Press, 1989. ISBN: 0262691302.


Microsoft. *Task Parallel Library*. URL: `https://msdn.microsoft.com/da-dk/library/dd460717.aspx` (visited on 08/12/2016).


Daan Leijen, Wolfram Schulte, and Sebastian Burckhardt. "The Design of a Task Parallel Library". In: *SIGPLAN Not.* 44.10 (Oct. 2009), pp. 227–242. ISSN: 0362-1340. DOI: 10.1145/1639949.1640106. URL: `http://doi.acm.org/10.1145/1639949.1640106`.


Florian Biermann. *Declarative Parallel Programming in Spreadsheet End-User Development: A Literature Review*. 2016.

References VIII

Aalborg University. *UPPAAL homepage*. June 2, 2015. URL: http://uppaal.org/ (visited on 04/19/2017).