

Forth im FPGA

Andrew Read's N.I.G.E.-Machine

Ulrich Hoffmann <uho@forth-ev.de>

Forth im FPGA

Andrew Read's N.I.G.E.-Machine

- Offene Systeme
- N.I.G.E.-Machine
- Nexys4-Board (Digilent)
- Demo
- Projekt
- Fazit

Offene Systeme

- Quellcode liegt vollständig vor.
- Jeder Aspekt ist zu verstehen (bis zum Metall)
- Kann vollständig neu generiert werden.
- Forth - Lisp - Oberon - Linux

- plus Selbstbezüglichkeit
 - System kann sich selbst neu generieren.
 - Quellcode ist in der/den Systemsprachen
 - Meta-Compiler

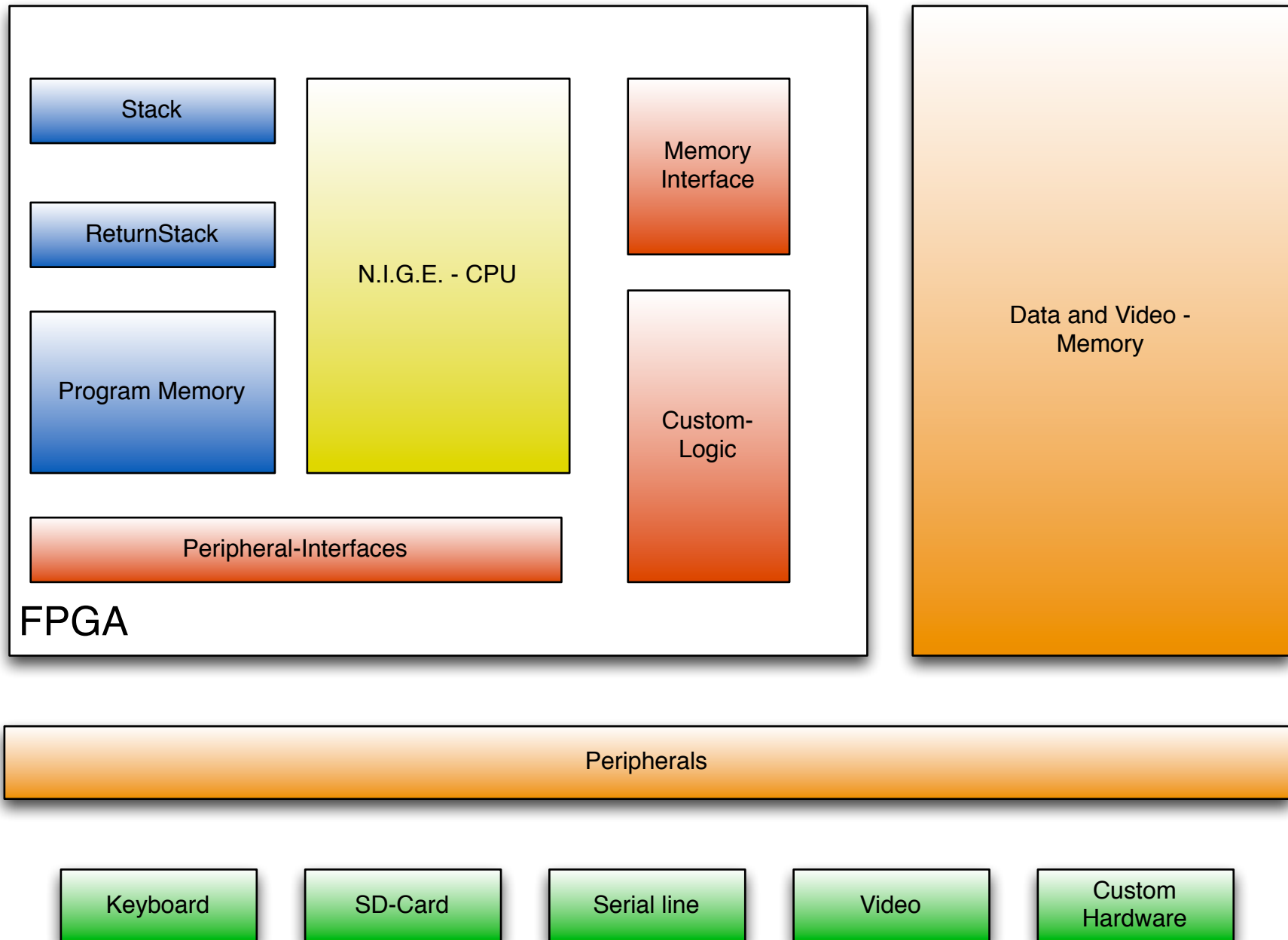
N.I.G.E. -Machine

- Forth Computer im FPGA
- von Andrew Read auf der EuroForth 2012 in Oxford vorgestellt.
- komplett mit Tastatur/Video/Massenspeicher
- 32-Bit Forth-Softcore-CPU in VHDL
- 8-Bit-Instruktionen (Microcode)
- deterministische Ausführungszeiten trotz Pipeline
- kurze Interruptlatenz
- 50 Seiten Handbuch
- GPL / commercial License

Wieso N.I.G.E.?

- **Prototyping von neuer Hardware**
 - Anschluss neuer Peripherie (z.B. über PMods)
 - Entwurf der Custom-Logic in VHDL
 - Anschluss an N.I.G.E.-CPU (einfach)
 - Treiber interaktiv direkt auf N.I.G.E. entwickeln.
 - Nur in Forth - no assembly required
- **N.I.G.E. ist ein offenes System**
 - lernen und verstehen bis zum Metall

N.I.G.E.-Machine



Befehlssatz 63 Instruktionen

- **Stack-Manipulation** 15 Instruktionen
NOP (no operation),
DROP DUP ?DUP SWAP OVER NIP ROT >R R@ R> SP@ SP! RP@ RP!
- **Mathematische Operationen** 12 Instruktionen
+ - NEGATE I+ I- ASL ASR U* * +c -c / U/
- **Vergleichsoperationen** 11 Instruktionen
= and <> < > U< U> 0= 0<> 0< 0> FALSE
- **Bitweise Operationen** 7 Instruktionen
AND OR INVERT XOR LSL LSR byte and word sign extension to 32 bits
- **Speicheroperationen** 6 Instruktionen
C@ C! W@ W! @ !
- **Literale Lade-Operationen** 3 Instruktionen
LOAD longword, word oder byte-Werte aus dem Programmspeicher
- **Kontrollstruktur-Operationen** 6 Instruktionen
JMP (jump to the address on the parameter stack), BSR and JSR (branch/jump to subroutine), RTS (return from subroutine) BEQ, BRA (conditional and unconditional)
- **Ausnahmebehandlung** 3 Instruktionen
TRAP (software vector) RTS_TRAP (single- step), and RTI (return from interrupt)

N.I.G.E.-Machine

- Forth-System
- ANS-Forth kompatibel
- Kern in N.I.G.E.-Assembler geschrieben
(Cross-Assembler auf gforth, vfx, swift)
- Alles andere in Forth auf N.I.G.E.
- FAT32-Filesystem für SD-Karte
- Full-Screen-Editor

In N.I.G.E.-Assembler

OVER

```
OVER.LF dc.l    SWAP.NF
OVER.NF dc.b    4 128 +
           dc.b  char R char E char V char 0
OVER.SF dc.w    1
OVER.CF over, rts
;
```

In N.I.G.E.-Assembler

COUNT

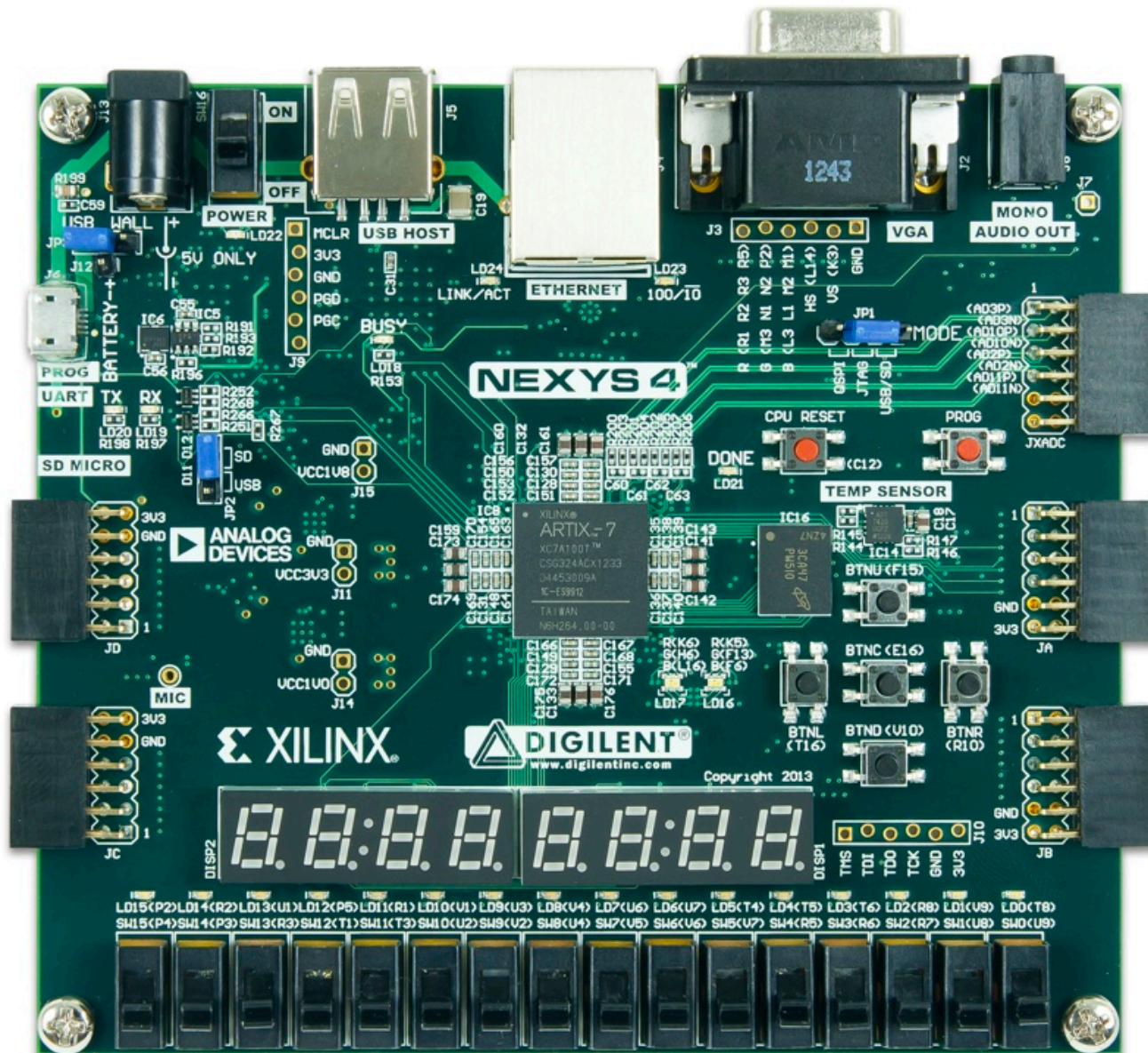
```
;
; COUNT ( addr -- c-addr n)
;
COUNT.LF  dc.l    $=.NF
COUNT.NF  dc.b    5 128 +
           dc.b    char T char N char U char 0 char C
COUNT.SF  dc.w    COUNT.Z COUNT.CF del
COUNT.CF  dup     ( addr addr)
           1+     ( addr c-addr)
           swap   ( c-addr addr)
COUNT.Z   fetch.b,rts ( c-addr n)
```

In N.I.G.E.-Assembler

VARIABLE

```
;
; VARIABLE ( -- create a variable)
VARIABLE.LF dc.l      ALLOT.NF
VARIABLE.NF dc.b      8 128 +
                dc.b   char E char L char B char A char I char R char A char V
VARIABLE.SF dc.w      VARIABLE.Z VARIABLE.CF del
VARIABLE.CF js1 COLON.CF                ; initiate the word
                #.w HERE_
                fetch.l
                #.b 6                ; allow enough space for a 4 byte PFA address, Load and RTS
                +
                dup                    (PFA PFA)
                js1 LITERAL.CF          ; compile the PFA
                js1 SEMICOLON.CF
                #.b 4                    ( PFA 4) ; allocate space for the the PFA
                +                        ( HERE ' )
                #.w      HERE_            ( HERE ' &HERE)
VARIABLE.Z store.l,rts
```

Nexys4 FPGA-Board Digilent



Quelle: trenz-electronic.de

Demo

Projekt



Quelle: github.com

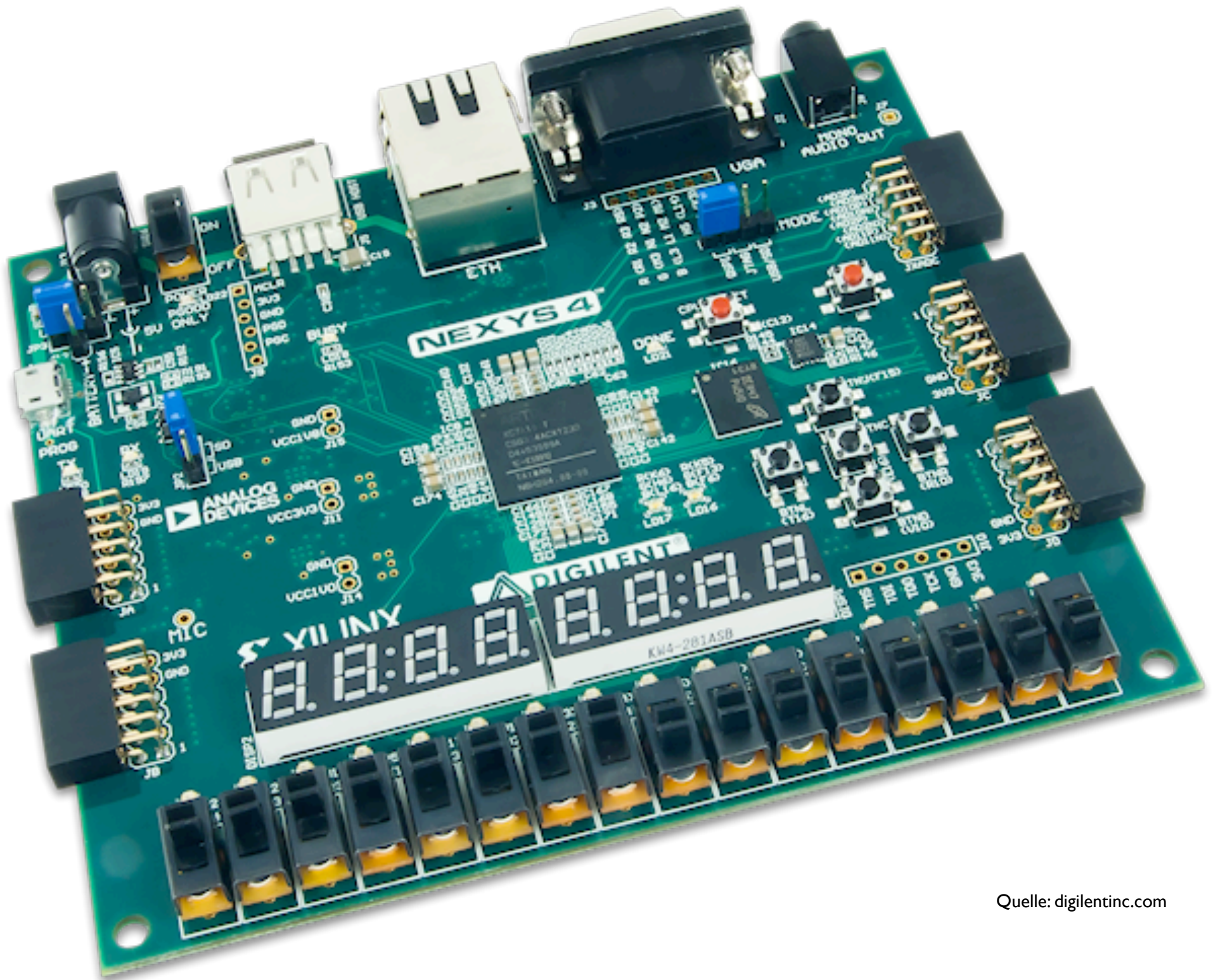
- N.I.G.E.-Projekt gehostet auf GitHub

<https://github.com/Anding/N.I.G.E.-Machine>

- Mitstreiter gesucht

Fazit

- N.I.G.E. ist benutzbar.
- N.I.G.E. ist ein offenes System
 - nicht selbstbezüglich
 - System in N.I.G.E.-Assembler (nicht in Forth)
 - CPU in VHDL
- Prototyping neuer Hardware
 - Systematisches Vorgehen
 - kurze Entwicklungsgeschwindigkeit



Quelle: digilentinc.com