Innovative Schaltungstechnik statt Software - SHUFFLE SORT: VLSI-Beispiel eines Sortierers

K. P. Bastian, R. W. Hartenstein, W. Nebel,
May 1985, report 133/85, University of Kaiserslautern

nach einem Vortrag, gehalten auf der Tagung "Mikroelektronik in der Automatisierungstechnik", veranst. in Baden-Baden, April 1985, von der VDI/VDE-Gesellschaft für Meß und Regelunstechnik (GMR).

Die Arbeiten wurden teilw. unterstützt durch das Bundesministerium für Forschung und Technologie in Rahmen des Verbundprojekt E.I.S.

Zusammenfassung

Das Papier gibt eine Praxis-orientierte Einführung in die Methode des "strukturierten Entwurfes" hochintegrierter Bausteine. Es wird veranschaulicht, daß Ingenieure mit Erfahrung in Digitaltechnik und Informatiker die Grundzüge des Entwurfes hochintegrierter MOS-Schaltungen rasch erlernen können. Dies wird an einem praktischen Entwurfs-Beispiel gezeigt: ein Sortier-Algorithmus wird in Form einer spezifischen integrierten Schaltung realisiert, die außer dem Taktgeber keinerlei Steuerwerk mehr benötigt, weder in programmierter Form, noch sonstwie realisiert. Deshalb kann man von einer direkten Silizium-Realisierung des Sortier-Algorithmus sprechen. Das Papier gibt auch eine Darstellung der mit dem "strukturierten Entwurf" sich rasch ausbreitenden stark innovativen neuen Mikroelektronik-Szene nebst den dabei typischen Infrastrukturen. Das vorliegende Papier behandelt nur den Entwurf voll Kunden-spezifischer integrierter Schaltungen, nicht jedoch die Anwendung von Gate-Arrays und deren Varianten.

Einleitung

Durch eine seit Ende der 70er-Jahre von den Universitäten in den USA ausgehende stark innovative Entwicklung, die auch rasch zu einer größeren Zahl von "spin-offs" führte, ist eine neuartige Mikroelektronik-Szene entstanden. Deren ungeheure Stoßkraft ist auf eine Reihe von zumindest unterstützenden parallel dazu verlaufenden Entwicklungen zurückzuführen:

- lohnende Anwendung oft bei geringen Stückzahlen
- Trennung zwischen Technologie und Entwurf
- Entmystifizierung der Mikroelektronik (Lehre)
- Entwertung der klassischen Hardware-Theorie
- Vordringen besserer CAD-Werkzeuge
- drastische Erhöhung der Designer-Produktivität
- auch künftig weiter sinkende Preise für CAD-Ausstattung

Die Trennung zwischen Entwurf und Technologie macht es möglich, daß eine Entwurfs-Abteilung für integrierte Schaltungen fernab einer Fabrikations-Stätte für diese erfolgreich operieren kann, ja sogar mit mehreren Herstellern arbeiten kann ("second sourcing": zur Wahrung der Unabhängigkeit). Diese Trennung wurde auch in der Lehre vollzogen, was zu einer erheblichen "Entrümpelung" des Lehrstoff geführt hat: für den Entwickler (sofern er sich auf den Bereich des Entwurfes von Digital-Schaltungen beschränkt) sind nur noch wenige "naive" Technologie-Kenntnisse notwendig. Einem Textbuch /1/ mit Schrittmacher-Funktion folgten bald weitere dieser Richtung.

Darüber hinaus sind auch große Teile der klassischen Theorie der Digital-Hardware entwertet dadurch, daß bei integrierten Schaltungen andere Optimierungs-Kriterien relevant sind: anstelle der Zahl der aktiven Elemente oder "Bausteine" muß der Flächenbedarf auf der Chip-Oberfläche minimiert werden. Verdrahtung, insbesondere mit traditionellen Methoden entworfen,

verbraucht meist viel mehr Chip-Fläche als die Transistoren. Eine Folge davon ist, daß Designer neuen Typs weniger tiefgehende Kenntnisse der Theorie des Logischen Entwurfes benötigt.

Die fast explosionsartige Entwicklung eines rasch wachsenden Milliarden-Marktes für teils neue VLSI-spezifische CAD-Ausrüstungen stimuliert das Wachstum von Entwurfs-Aktivitäten weltweit sehr stark. Die zum Großteil auf die Schrittmacher-Rolle von Universitäten in den USA zurückgehenden CAD-Hilfsmittel haben in den letzten 5 Jahren die Produktivität des Schaltungs-Entwicklers etwa verzehnfacht. Bis gegen Ende der 80er Jahre rechnet man mit einer erneuten Verzehnfachung.

Hierdurch kann die "Entwurfs-Lücke" bald geschlossen werden, die eine Ausschöpfung des Standes der gegenwärtigen (mehr als 100 000 Transistoren auf einem Chip) und zukünftigen (Submikron-) Technologien stark behindert. Eine Ausnahme sind hierbei die Halbleiter-Speicher, deren sehr hohe Stückzahlen hohe Entwurfskosten rechtfertigen. Außerdem sind diese wegen ihrer einfachen Struktur von geringer "Entwurfs-Komplexität".

Neue Infrastrukturen der Fertigung und neue Kooperationsformen zwischen Hersteller und Anwender (mit eigener Entwurfs-Abteilung) machen Voll Kundenschaltungen oft schon bei geringen Stückzahlen lohnend, insbesondere dann, wenn diese einen hohen Innovations-Gehalt haben:

- Preissenkung eines älteren Produktes durch höhere Integrations-Dichte
- Erhöhung der "Intelligenz" eines Produktes durch Mikroelektronik-Einbauten
- Markt-Erschließung für innovative Produkte über den Preis

Solche Innovationen werden meist realisiert durch völligen oder teilweisen Ersatz traditioneller Hardware, oder von Software, durch Anwendung von Kunden-Schaltungen. Voll-Kunden-Schaltungen erreichen wesentlich höhere Integrations-Dichten als Halb-Kunden-Schaltungen (Gate Arrays etc.). Es gibt hier

viele Anwendungen, die deshalb nur durch Voll-Kundenschaltungen realisierbar sind, nicht jedoch durch Halb-Kundenschaltungen. Ein solcher Innovations-Fall soll an folgendem Beispiel gezeigt werden, das einen Sortier-Algorithmus in einen voll-Kundenspezifischen integrierten NMOS-Baustein umsetzt.

Ein praktisches Entwurfs-Beispiel

Der Sortier-Algorithmus *"Bubble Sort"* /2/ eignet sich wegen seiner Einfachheit gut als Demonstrations-Beispiel der Umsetzung eines sequentiellen Programm-Konstruktes in eine spezifische integrierte Schaltung. Bild 1 beschreibt den "Bubble Sort" in einer PASCAL-ähnlichen Notation. Gegeben seien N Datensätze R[1] (R steht für "record") von R[1] bis R[N]. Das Format von R enthält einen Schlüssel-Teil KEY, der dem Auffinden und sortieren der Datensätze dient. Beim Beispiel einer Personal-Datei könnte KEY der Name des Beschäftigten sein, wobei der Sortierprozeß alphabetische Reihenfolge herstellen möge. Bei unserem später folgenden kleinen Trivial-Beispiel (Bilder 5 und 9) dienen Zahlen (1, 3, 4, 7, 8) als Schlüssel KEY.

Sequentielle Realisierung: die Grenzen des Universalrechners

Als Programm eines (Mikro-)Rechners benötigt der "Bubble Sort" viel Rechenzeit. Die innere Schleife ("inner loop", s. Bild 1) des Programmes benötigt fast 100 Primärspeicher-Zugriffe, Befehle holen und den Transport von Daten und Adressen von und zu der Zentraleinheit (*CPU*: central processing unit, zentrale PU) mitgerechnet. Der Kommunikationskanal zwischen CPU und Primärspeicher (vgl. Bild 2) ist relativ langsam und wird daher oft als "von-Neumann-Engpaß" bezeichnet. Dieser und der noch viel langsamere Verkehr mit Sekundär-Speichern (evtl. zusätzlich beeinträchtigt durch *"thrashing"*, d. h. exzessiv häufigen Seitenaustausch zwischen den beiden Speichern) ist die Ursache dafür, daß die *sequentielle Realisierung* (d. h. durch ein Programm) eines Algorithmus bis zu Millionenfach langsamer sein kann im Vergleich zu einer *"planaren Lösung"* (auf der

"planaren" Oberfläche eines "Chip"). So auch beim Bubble Sort, wenn die Datei viele Datensätze umfaßt (z. B. mit N = 1000).

Beim Bubble Sort bearbeitet die *innere Schleife ("inner loop")* bei einem Durchlauf jeweils nur ein Paar benachbarter Datensätze (vgl. Bild 3). Jedes Paar muß N-1 mal bearbeitet werden. Bei N-1 Paaren ergibt dies eine *Komplexität O(N) = N * N* (O steht für "Größenordnung"). Bild 4 verdeutlicht dies durch die Wanderung des "Fensters", durch welches die CPU jeweils während eines Einzeldurchganges durch die innere Schleife "hindurchsieht". Nach Beendigung eines Durchlaufes der inneren Schleife wird das Fenster jeweils um eine Position weiter gerückt. Bild 5 zeigt ein Zahlenbeispiel mit KEY = {1,3,4,7,8}.

Eine "planare" Realisierung (VLSI-Lösung)

Zur Silizium-Realisierung eines Algorithmus gibt es folgende Methoden: Parallelisierung der "inneren Schleife" durch Entwurf eines intelligenten Speichers (*"smart memory"*), Entwurf eines *"systolischen Array"* (mehrere Datenströme begegnen sich auf aktive Weise, für eine Beschreibung des Prinzips ist hier nicht genug Raum), durch Entwurf spezieller *"pipe line"*-Strukturen (insbesondere in der Signal-Verarbeitung), oder über *CAD-generierte spezielle große Schaltnetze* (auch Schaltwerke). Im Folgenden soll die Parallelisierung der "inneren Schleife" des "Bubble Sort" durch Entwurf einer Voll-Kundenschaltung veranschaulicht werden.

Man schafft auf dem"Chip" N Stück Register R, eines für jeden Datensatz. (Bei umfangreichen Datensätzen speichert man in R neben dem Schlüssel KEY nur einen Zeiger.) Dazu wird ein spezialisierter "Prozessor" *sPU* (vgl. Bild 6) entwickelt, der nur ein KEY-Paar vergleichen und einen Tausch *"swap"* ausführen kann. Zum Laden und Entladen des "Chip" wird noch ein *"shift"* (Verschieben des gesamten Datenblock um eine Position) benötigt. Jedes R-Paar bekommt eine sPU zugeordnet (vgl. Bild 7) mit dem Wunsch, daß die N-1 sPUs die N-1 Durchläufe der inneren Schleifen gleichzeitig, also voll parallel ausführen können.

```
loop  J = 2 .. N  begin
  loop   I = 2 .. N  begin
    if  KEY (R [I]) > KEY (R [I-1])
        then swap (R [I], R [I-1])
    endif
  end (* loop I *)  ;
end (* loop J *) ;
```

"inner loop" (innere Schleife)

Format des Wortes R:

KEY (Schlüssel)	Rest von R

Bild 1. Der "Bubble Sort" - Algorithmus

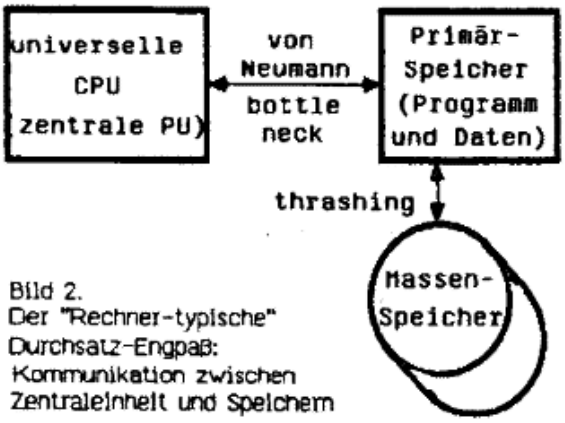


Bild 2.
Der "Rechner-typische" Durchsatz-Engpaß:
Kommunikation zwischen Zentraleinheit und Speichern

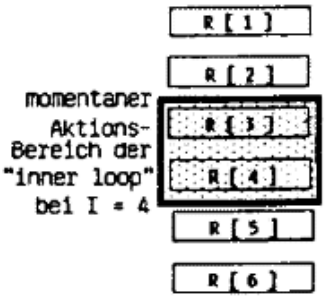


Bild 3. Aktionsbereich eines Lauf der inneren Schleife des Bubble Sort-Algorithmus

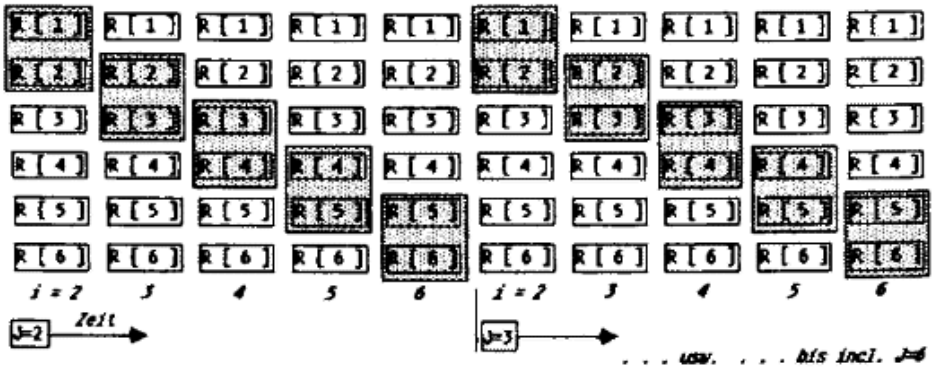


Bild 4. Schnappschüsse aus dem Ablauf des sequentiellen Bubble Sort

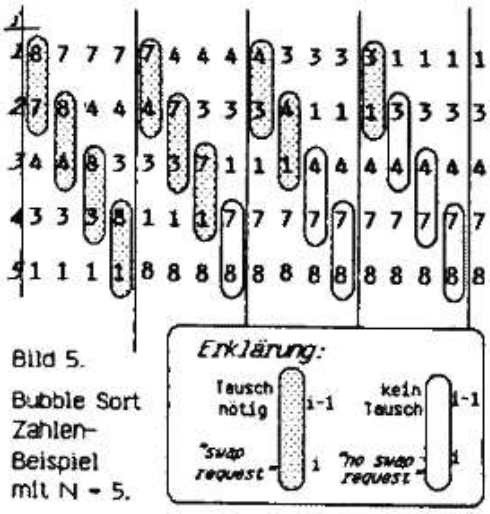


Bild 5.
Bubble Sort Zahlen-Beispiel mit N = 5.

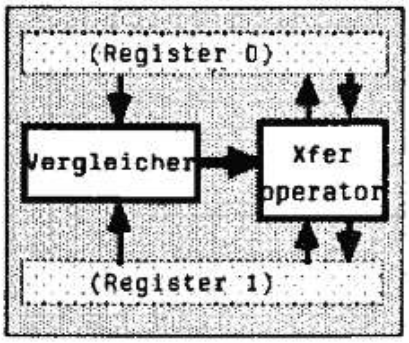


Bild 6. Hypothetische sPU für den "planaren" Bubble Sort

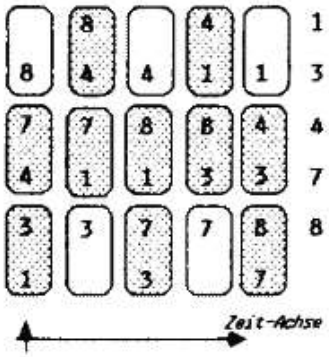


Bild 9. Schnappschüsse bei einem Shuffle Sort-Beispiel mit N = 5 (Datenblockgr.)

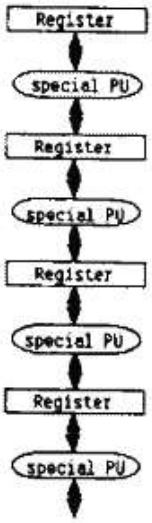


Bild 7.
hypothetische Hardware für den planaren Bubble Sort

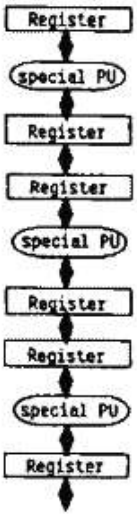


Bild 8.
wie Bild 7, jedoch mit halbierter sPU-Anzahl

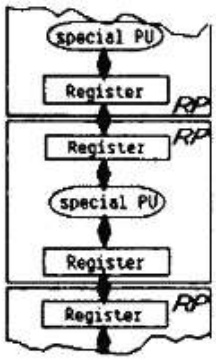


Bild 10. Optimale Partitionierung für strukturierten Entwurf des Sortierspeicherfeldes für den Shuffle Sort ("RP" = req. pair).

Dieser hypothetische Fall ist jedoch nicht realisierbar wegen Register-Zugriffs-Konflikten benachbarter sPUs. Somit kann im Durchschnitt nur die Hälfte der sPUs gleichzeitig aktiv sein. Da Chip-Fläche knapp ist, wird die Hälfte der sPUs eliminiert (s. Bild 8). Daß jedes KEY-Paar die gleiche Chance hat, wird durch Auf- und Ab-Schiften des gesamten Datenblockes im Sortierspeicher erreicht (vgl. Zahlenbeispiel in Bild 9). Da dies an ein Waschbrett erinnert, sei dieser nun veränderte Algorithmus "Shuffle Sort" genannt. Wir halten fest, daß für eine sinnvolle Realisierung auf der Chip-Oberfläche der Algorithmus geändert wurde: "planarer Algorithmus" oder "VLSI-Algorithmus".

Die praktische Durchführung des strukturierten Entwurfes

Nun sei der von obigen Überlegungen ausgehende "strukturierte Entwurf" der "Shuffle Sort"-Schaltung skizziert. Diese Methode wird versucht, die Schaltung durch wiederholte Anwendung weniger Typen einfachster "Zellen" zusammenzusetzen. Da "Verdrahtung" Fläche verschwendet, ist dabei ein wichtiges Ziel die direkte "Zusammensteckbarkeit" der Zellen (*wiring by abutment*, oder: *abutment* statt wiring). Nötig ist Puzzle-artige Kreativität, wobei das geometrische Anschauungsvermögen des Menschen wichtig ist (Geometrie-orientierter Schaltungs-Entwurf). Die Partitionierung nach Bild 10 erwies sich für den Shuffle Sort als günstiger Ausgangspunkt.

Bild 11 zeigt die Lösung des Zellen-Entwurf für den Shuffle Sort: es wurde eine 1-Bit-"Scheibe" ("slice") "RPslice" einer Register-Paar/sPU-Anordnung entworfen, die nur aus drei Sorten Tochter-Zellen *hXf* (half Xfer), *FF* (Flipflop), und *comparator slice* (Vergleicher-Scheibe) zusammengesetzt ist. Der Sortierspeicher wird durch "Zusammenstecken" von W * N Stück dieser Zellen zu einer Matrix-artigen regelmäßigen Anordnung erstellt (W ist die Wortgröße in Bits).

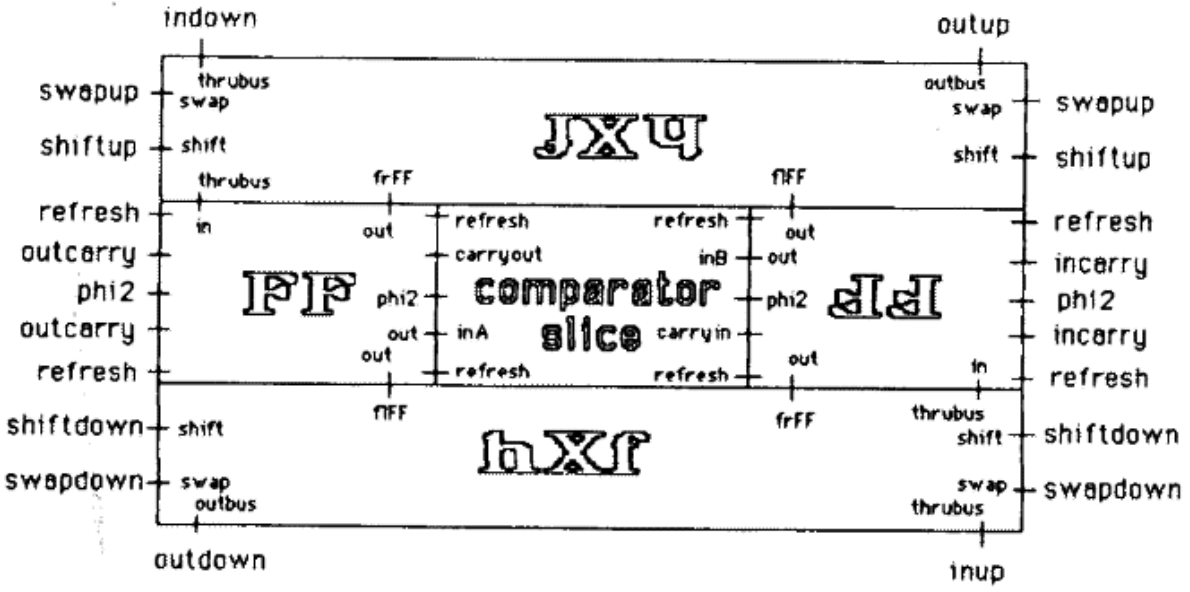
Bereits bei den Tochterzellen wurde der "strukturierte Entwurf" angewandt: schon in einem einzigen "RPslice" kommen

die Zellen *FF* und *hXf* je zweimal vor, und zwar einmal direkt plaziert und einmal um 180 Grad gedreht. Alle Zellen sind "zusammensteckbar" (d. h. sie werden miteinander verbunden durch *wiring by abutment*, s. oben) und sehr einfach aufgebaut. Bild 12 zeigt die Zelle *hXf*, die nur 4 Pass-Transistoren enthält. Bild 13 veranschaulicht die durch die zwei mal 4 Pass-Transistoren (Tranfer-Transistoren) der zwei *hXf* -Zellen realisierten Transport-Operatoren *shift up*, *shift down*, *swap up* und *swap down*. Das Ziel einer möglichst einfachen Zelle wurde hier besonder gut erreicht.

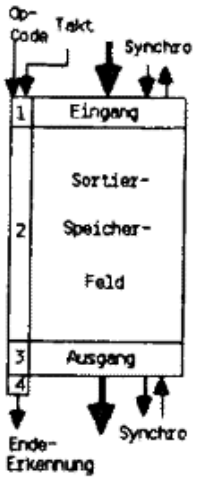
Deutlich sichtbar ist die wichtige Koordination von elektrischem Schaltplan und topologischer Gestalt der Zelle (relative Lage der Anschlüsse an den 4 Seiten der Zelle). Die topologische Gestalt der Zelle (sozusagen noch nicht vermaßte Geometrie) ist ausschlaggebend ist für einen erfolgreichen *strukturierten Entwurf*. Dies ist ein wichtiges Konzept der Vorplanung des späteren Layout-Entwurfes. Dies sind typische Merkmale der im folgenden Abschnitt diskutierten *neuen Arbeitswelt*. Typisch ist hier auch das Überspringen methodologischer Ebenen (vgl. Bild 14): der sehr einfache obige Schaltplan wurde praktisch vom Boden der algorithmischen Ebene aus konzipiert.

Arbeitwelt des strukturierten Entwurfes

Herkömmlich sind zwei Grundtypen der Realisierung von Algorithmen: Programmierung und Hardware-Entwurf. Bei letzterem ist der Logik-Plan auf Baustein-Katalog-Basis wichtigstes Zwischenresultat auf dem Wege zum Endergebnis: der Stück- und Verdrahtungsliste. Wie obiges Beispiel zeigt, ist *strukturierter VLSI-Entwurf* Kennzeichen der *dritten Arbeitswelt* der Algorithmen- und System-Implementierung. Endergebnis ist geometrisches Layout. An die Stelle des Logik-Planes tritt als wichtigstes Zwischenergebnis der *floor plan*, ein Flächenaufteilungs-Konzept, basierend auf einem Satz topologisch vorgeplanter essentieller Zellen. Die Methodik hierzu wird auch als *chip planning* bezeichnet. Ausgehnend von diesem konzeptionellen Vorentwurf kann dann der Physische Entwurf beginnen.



a)



1 globaler Dekodierer
2 lokale Dekodierer-Zeile
3 Datenblock-Ankunft-Erkennung
4 Sortier-Ende-Erkennung

b)

Bild 11. Schuffle Sort: Ergebnis des strukturierten Entwurf. a) die 1-Bit-Scheibe *RPslice* der Registerpaar/Vergleicher-Zelle *RP*. Anschlüsse: große Schrift für die Mutterzelle *RPslice*, kleine Schrift für deren Tochterzellen *comparator slice*, je zwei mal *FF* (Flipflop) und *hXf* (halber Transport-Operator). b) "floor plan" der Gesamtschaltung (ohne Bonding Pads).

Zusammen mit der Weiterentwicklung von Infrastrukturen in der Wirtschaft und an den Hochschulen (u. a. /3 - 8/) verändert diese "Neue Arbeitswelt der Mikroelektronik" in zunehmendem Maße nicht nur die Entwurfsmethoden im engeren Sinne, sondern auch die Arbeitsteilung zwischen den Ingenieuren. Anstelle einer "pipe line aus hineinander geschalteten Spezialisten" tritt ein neuer Designer-Typ. Wie Bild 14 veranschaulicht, ist dieser hierarchisch/organisatorisch denkende Designer in erster Linie ein Wanderer zwischen methodologischen Ebenen, dabei oft Ebenen überspringend. Deshalb und wegen seiner hinzukomenden geringen horizontalen Spezialisierungstiefe (vgl. Einleitung und Bild 14) wird dieser Typ von Designer in den USA gelegentlich auch als *"long thin man"* bezeichnet. Auch obiges Entwurfs- Beispiel versuchte zu zeigen, wie hier eine einzige Person aus der algorithmischen Ebene (Anwendungs-Ebene) weit hinuntertaucht in die Schaltkreis-Ebene und noch tiefer hinunter in die Vorplanung der Zellen-Gestalt und des Layout-Entwurf.

Der *long thin man* ist letztlich ein Anwender, der Layout selbst entwirft, also trotz weitem Horizont mit beiden Beinen fest auf dem Boden des Layout steht. Er sieht den Chip-Entwurf also nicht aus der Vogel-Perspektive, wie der herkömmliche Anwender, sondern aus der Giraffen-Perspektive. Gegenüber herkömmlichen "naiven" Anwendern hat er dadurch einen ungefilterten Blick für das Machbare auf der planaren Chipfläche. Er hat also mehr Innovationskraft verfügbar für die Eroberung der in Bild 15 angedeuteten unerforschten Gebiete der Digitaltechnik.

Wie das Bild auch zeigt, sind herkömmliche Techniken sehr *einseitig* im doppelten Sinne des Wortes: Rechner, auch Parallelrechnersysteme, zusammen mit ihren Speichern, besiedeln nur den Rand des "weiten Landes" des Digitalsystem-Entwurfes. Speicherzellen sind sehr kleine Prozessoren (meist weniger als 10 Transistoren je Bit, nur ganze wenige Operationen: *schreiben, lesen, und behalten*, deshalb *dumme Speicher* oder *dumb memories* genannt), die somit nur den linken Rand des Entwurfsraumes von Bild 15 besiedeln. Universelle (Mikro-)

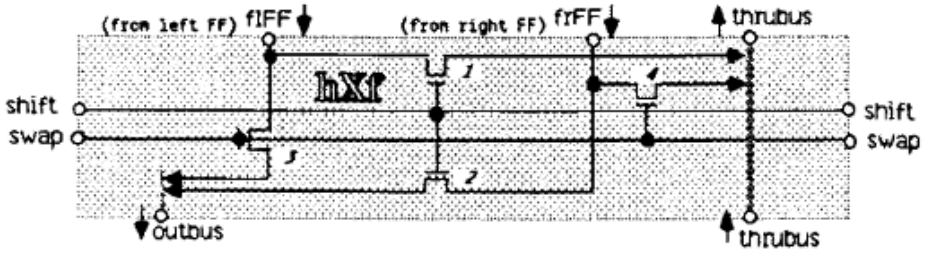


Bild 12. halber Transfer-Operator:Tochterzelle *hXf* der Zelle *RPslice* (1-Bit-Scheibe einer Register-Paar/Vergleicher-Zelle *RP*). Funktion hier (die untere Hälfte): *"swap down"* und *"shift down"* (vgl. auch die Geometrie in Bild 11).

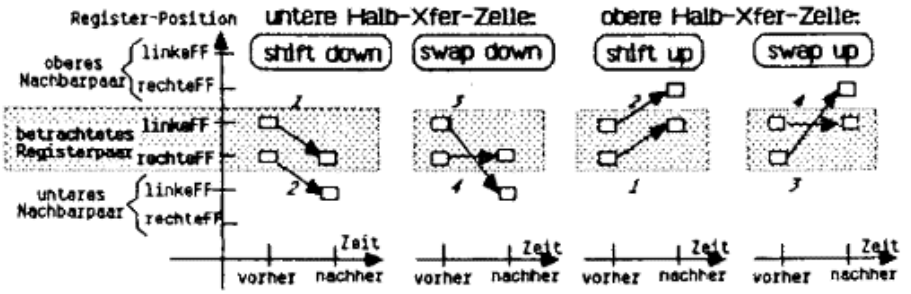


Bild 13. Durch die beiden *hXf*-Zellen eines RP-Modul ausgeführte Transport-Operationen. Kursiv-Ziffern: Transistor-Nummern gemäß Bild 12.

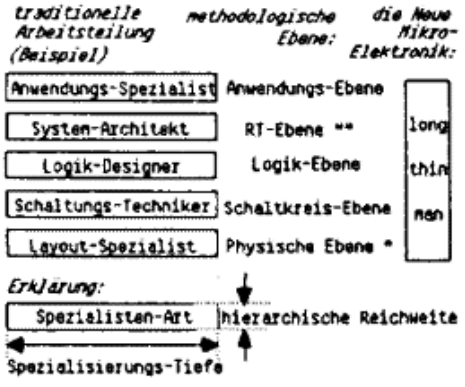


Bild 14. Die Arbeitswelt des "long thin man"; *) Layout-Entwurf, **) "RT" = "register transfer")

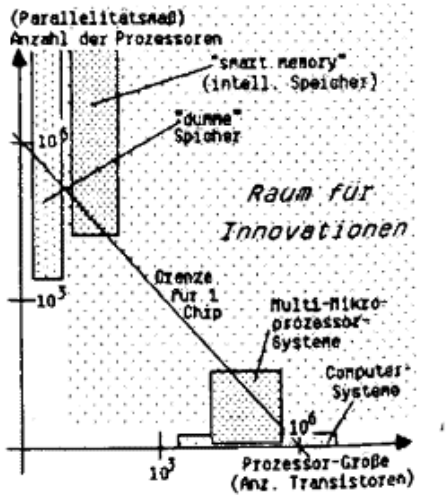


Bild 15. Raum f. VLSI-Algorithmen

Rechner jedoch sind mit oft Hunderttausenden von Transistoren sehr große Prozessoren. Abgesehen von der Forschung werden in geschlossenen Systemen nur relativ wenige davon parallel geschaltet: vom *design space* (Bild 15) wird nur der untere Rand am rechten Ende besiedelt. Es bleibt noch sehr viel Platz für Innovationen übrig, wie Bild 15 deutlich zeigt. Bereits der *smart-memory*-Ansatz (kleine Prozessoren, nur wenig größer als *dumb memories*, Beispiel: obiger *Shuffle Sort*) bringt uns voran beim innovativen Vordringen in weitgehend unerschlossenes Gebiet. Die hier skizzierte *neue Arbeitswelt* der Mikroelektronik macht es möglich, daß sich an diesem Zug viele Menschen beteiligen können, auch im mittelständischen Bereich.

Literatur

/1/ C. Mead, L. Conway: *Introduction to VLSI Sytems;* Addison-Wesley, 1980

/2/ D. Knuth: *The Art of Computer Programming - Volume 3: Sorting and Searching;* Addison-Wesley, 1973

/3/ R. W. Hartenstein: *Die Neue Mikroelektronik in der Informatik: Voraussetzungen und Auswirkungen;* GI-Jahrestagung 1981, (auch als Bericht: Univ. Kaiserslautern, 1981)

/4/ R. W. Hartenstein: *Innovationen der Neuen Mikroelektronik durch Neue Lehrmethoden in Elektrotechnik und Informatik;* Festvortrag zur 25-Jahrfeier des Institut für Technik der Informationsverarbeitung, Universität Karlsruhe, 1983, (auch als Bericht: Universität Kaiserslautern, 1983)

/5/ R. W. Hartenstein: *Shared Cultures: CIF libraries, starting frames and scalable design rules;* NATO Advanced Study Institue on Design Methodologies for VLSI Circuits; Louvain-la-Neuve (Belgien) 1980, in: /6/

/6/ P. G. Jespers, C. H. Sequin, F. van de Wiele: *Design Metholologies for VLSI Circuits;* Stijthoff and Noordhoff, Alphen aan den Rijn (Niederlande), Rockville, Maryland (USA), 1982,

/7/ R. W. Hartenstein: *VLSI-Bausteine in geringen Stückzahlen für Spezialanwendungen;* Elektron. Rechenanlagen, 1980

/8/ R. W. Hartenstein: *Das EIS-Verbundprojekt: Aufbruch in die Neue Mikroelektronik;* Computer-Magazin, 1984