Prospettive di sviluppo della microelettronica. La progettazione dei circuiti «VLSI»

L. Dadda (*), G. De Micheli (**), M.G. Sami (*)

Si prospettano i problemi relativi alla progettazione di circuiti e sistemi a grandissimo grado di integrazione (VLSI).

Segue l'esame dello «stato dell'arte» per quanto riguarda la metodologia generale di progetto per la definizione delle specifiche funzionali, per la sintesi, per la simulazione e per il collaudo.

Si illustra poi l'organizzazione complessiva di un sistema di progettazione con l'ausilio del calcolatore, CAD, e si accennerà infine allo sviluppo dei fondamenti teorici che potranno avere importanti effetti sulla concezione dei sistemi VLSI.

1. - Introduzione.

Il rapido incremento del numero di dispositivi elettronici realizzabili su di una singola piastrina di silicio ha già fatto superare la soglia (qualche decina di migliaia) con cui convenzionalmente si indica l'avvento dei circuiti integrati VLSI.

Tale soglia non marca soltanto la capacità tecnologica di realizzare tali circuiti (dipendente in gran parte dalla capacità di ridurre le dimensioni geometriche minime delle maschere) ma anche un modo nuovo di progettare i circuiti stessi. È noto infatti che il microprocessore M68000 (a 16 bit) ha richiesto 52 anni-uomo per il suo progetto, e che un altro microprocessore, lo Intel 8086, ha richiesto 13 anni-uomo per il solo disegno (layout) delle maschere.

La complessità dei circuiti integrati è, inoltre, destinata a crescere ulteriormente nel corso dei prossimi anni. Nel passato si è verificato un raddoppio della complessità dei circuiti ogni anno, secondo la cosiddetta legge di Moore. Si prevede in futuro il raddoppio di tale complessità ogni due anni. La riduzione del tasso di crescita risulterebbe dovuta non tanto ad un rallentamento della capacità tecnologica, ma piuttosto alla crescente difficoltà di progettare circuiti sempre più complessi.

(*) Dipartimento di Elettronica - Politecnico di Milano.

(**) Centro di studio per l'Ingegneria dei Sistemi per l'Elaborazione delle Informazioni - C.N.R. ed University of California, Berkeley.

La necessità di disporre di metodi e strumenti per la progettazione di circuiti integrati di grande complessità con costi e tempi accettabili appare tanto più impellente se si pensa che, mentre alti costi di produzione e tempi di progettazione possono anche accettarsi per prodotti come i microprocessori, destinati a notevoli volumi di produzione e a durare sul mercato parecchi anni, essi impedirebbero di fatto la produzione di circuiti integrati con medi o piccoli volumi di produzione, destinati ad applicazioni specifiche (custom).

La tabella 1 mostra in modo espressivo come i metodi finora adottati per circuiti integrati di piccola e media complessità non possono assolutamente adattarsi per i circuiti VLSI.

Tabella 1

Anno	Dimensioni di griglia	Scala	Dimensioni progetto	Area equivalente
'70	17.5 μm	375 ×	2 m	Muro
'80	8 μm	800 ×	5 m	Pavimento di un grande salotto
anni '80	2 μm	3 200 ×	20 m	Campo da Basketball (15 × 28 m^2)
...	0.4 μm	16.000 ×	100 m	Due campi da Football (120 × 50 × 2 m^2)

Nota: La dimensione di griglia è pari alla somma della larghezza minima dei collegamenti (es. 10 μm) e della loro spaziatura (es. 7.5 μm).

La Progettazione Automatica o CAD (Computer Aided Design) dei circuiti elettronici ha subito una lunga evoluzione, cominciata con l'analisi di circuiti elettronici a componenti discreti e con la progettazione dei circuiti a collegamenti stampati. Alcune insoddisfacenti esperienze compiute talvolta nel passato in tali campi (per inadeguatezza sia di metodologia sia di mezzi di calcolo) devono fare riflettere sulla complessità della analisi e sintesi dei circuiti VLSI. D'altra parte si è constatato che laddove il problema è stato affrontato con sufficiente ampiezza di vedute e mezzi, i risultati non si sono fatti aspettare.

Appare pertanto utile una rassegna critica del settore, per ricavarne indicazioni per i futuri sviluppi.

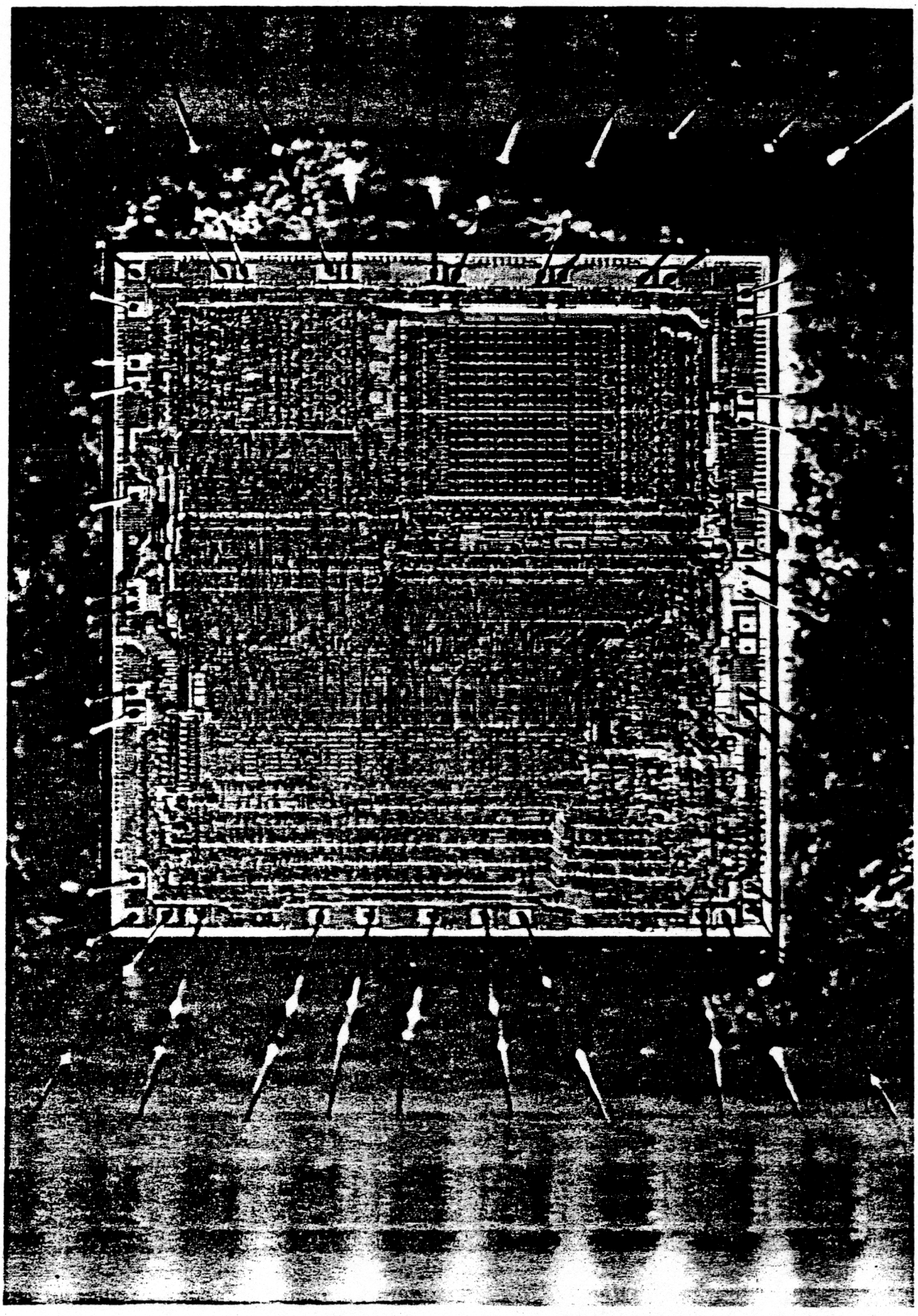

Fig. 1. - Piastrina contenente il microprocessore Z 80 realizzato dalla società SGS Ates.

2. - Metodologie di progetto.

2.1. - *Evoluzione della progettazione microelettronica.*

Per un migliore apprezzamento del problema del progetto dei circuiti VLSI può essere istruttivo riassumerne l'evoluzione.

I primi circuiti integrati furono realizzati disegnandone le maschere su fogli di mylar, sui quali venivano fissate figure geometriche ritagliate da fogli di materiale rosso, (rubylit o «paper dolls» approach, o metodo delle bambole di carta): esse venivano poi fotografate ed il negativo serviva per la preparazione delle maschere con processo fotolitografico.

Al crescere del numero dei componenti si dovette sia aumentare la dimensione complessiva del foglio di mylar sia ridurre le dimensioni dei particolari geometrici. Le dimensioni geometriche complessive crebbero al punto tale da rendere difficile per il disegnatore raggiungere tutte le parti del disegno senza disturbare le parti già disegnate. Uno «strumento» di disegno di nuovo tipo fu costituito da una piattaforma sospesa sopra il disegno sulla quale il disegnatore si distendeva e che poteva muoversi per portarlo nei vari punti.

Successivamente venne introdotto l'uso del calcolatore: i dati di tracciamento delle maschere venivano forniti partendo da schede perforate. In seguito vennero introdotti i digitizzatori, per introdurre nel calcolatore i dati geometrici delle maschere partendo da una rappresentazione grafica.

Il calcolatore poteva presentare maschere o loro parti su un tracciatore, o su un visore, permettendo anche un lavoro interattivo. Il progettista imparò a segmentare il progetto in sottounità, a prospettarle separatamente e poi a riunirle.

Con ciò si resero possibili circuiti integrati sempre più complessi. Ma, con l'aumentare della complessità, emersero nuovi problemi: tra questi quello della verifica finale delle maschere ottenute. Inizialmente essa era puramente visiva, ed al crescere delle dimensioni ciò comportava delle settimane ed anche mesi di lavoro, senza peraltro garantire l'assenza di errori. Furono così messi a punto programmi di verifica automatica e di simulazione. Coi primi si verifica la maschera finale per rivelare la eventuale violazione di regole esprimibili geometricamente (per es. larghezza e spaziatura delle linee metalliche). Coi secondi si verifica il funzionamento logico e elettrico del circuito. Un altro importante strumento di verifica è costituito da programmi capaci di estrarre dalle maschere il circuito logico ed elettrico da sottoporre poi a simulazione.

Il rapido incremento della complessità dei circuiti integrati rende oggi necessario un miglioramento dei programmi in uso, presumibilmente attraverso nuovi e più efficienti algoritmi. Si consideri come esempio il programma di verifica delle maschere: con gli algoritmi oggi in uso, il tempo di calcolo necessario cresce con la potenza 1,5 dei dati elaborati. D'altra parte la verifica dei più complessi circuiti integrati realizzati richiede già qualche giorno di calcolo sui più potenti calcolatori oggi disponibili.

Oltre al miglioramento dei singoli strumenti di progetto e la concezione di nuovi strumenti, si rende peraltro necessaria la concezione di nuove metodologie generali: ciò per far fronte alla evoluzione dei progetti, che sono passati dal semplice ambito logico e circuitale a quello della organizzazione di macchine e che tendono a includere l'architettura stessa di complessi sistemi.

Già nei primi anni del '70 Carver Mead al California Institute of Technology (Caltech) e Lynn Conway alla Xerox di Palo Alto iniziarono a esplorare nuove metodologie per il progetto dei futuri circuiti VLSI. All'inizio i concetti da essi sviluppati non vennero considerati con molta attenzione negli ambienti industriali. Oggi invece il loro testo è ampiamente conosciuto [1].

Oltre che a Caltech, programmi diretti a circuiti VLSI vennero lanciati nelle più importanti università americane: M.I.T. Berkeley, Stanford, Carnegie Mellon. In tutti questi programmi venne messa in gioco una importante partecipazione delle industrie interessate. In essi, inoltre, una parte spesso predominante ricopre il problema della metodologia di progetto e del relativo CAD, e ciò sia per il ruolo che quest'ultimo occupa nella formazione dei nuovi ingegneri elettronici sia perchè il genere di ricerche richiesto bene si inquadra nella ricerca universitaria. Tutte le citate università, inoltre, dispongono (o in casa, o fuori casa tramite opportuni accordi) della possibilità di realizzare i circuiti progettati (le cosiddette «fonderie», silicon foundries).

Sempre negli USA è noto che enorme è stato lo sforzo compiuto da alcune industrie per mettere a punto strumenti e metodi per circuiti VLSI: poco si sa di essi, per la loro natura riservata.

2.2. - *Esempi in atto.*

Anche se tutti coloro che si occupano di progetto VLSI fanno riferimento ad un approccio «top-down» e al «procedimento gerarchico», si trovano in realtà molte e rilevanti differenze nei vari stili di progetto. Può essere interessante confrontare dapprima i criteri adottati nelle due università dove il lavoro specifico nel campo VLSI è più avanzato (Caltech e Berkeley), per poi esaminare le proposte più articolate e sperimentali di Stanford e MIT. L'approccio di Carnegie-Mellon, ha interesse per quanto riguarda il problema della descrizione e delle rappresentazioni a vari livelli.

Le impostazioni scelte a Caltech e Berkeley sono profondamente diverse: più «radicale» quella di Caltech, più legata alla pratica del CAD dei sistemi elettronici quella di Berkeley.

Idealmente, Caltech vorrebbe giungere al cosiddetto «compilatore di silicio», cioè a un insieme di programmi che dalla descrizione architetturale giungono fino alla generazione delle maschere e al controllo dei processi di produzione, senza interventi ulteriori da parte del progettista [2]. Di fatto, al momento, i passi

essenziali — cioè la descrizione dell'architettura, la definizione funzionale delle celle che la costituiscono, il dettaglio circuitale delle celle medesime, l'assemblaggio del «chip» e finalmente la preparazione per la produzione richiedono interazioni da parte del progettista. Questi deve tradurre manualmente la definizione architetturale in uno schema d'interconnessione fra blocchi funzionali, deve definire le «celle elementari» o «foglie» (che vengono poi inserite in una base dati) e costruire a partire da queste mediante regole di composizione i blocchi funzionali; in queste fasi, il supporto automatico è ancora piuttosto limitato, e relativo ad alcune strutture di largo uso come PLA (schiere logiche programmabili), registri ecc., e a schemi architetturali basati sul concetto di bus. Da questo punto in avanti l'automazione è invece molto spinta, e le forme di rappresentazione (come in particolare i cosiddetti «stick diagrams» che costituiscono il passo intermedio fra schema elettrico e disegno della maschera) si sono trasformati in standard universalmente accettati.

L'impostazione del «compilatore di silicio» comporta un'ipotesi fondamentale, e cioè che ad ogni passaggio da un livello di schematizzazione a un livello inferiore (cioè di maggior dettaglio) la traduzione, con il relativo inserimento di nuovi dettagli e nuova informazione, avvenga in modo corretto.

Il controllo rigoroso delle regole che dominano questi passaggi assicura la correttezza e contemporaneamente rende inutili operazioni di verifica, tipicamente affidate altrimenti a programmi di simulazione; si può anche osservare che questo procedimento esclude interventi di scelta e ottimizzazione affidati all'esperienza del progettista e dipendenti dall'applicazione, piuttosto che da regole generali.

Al contrario, la scelta di Berkeley accetta l'uso di questa esperienza, e vuole piuttosto potenziarla, sfruttando le possibilità di ottimizzazione offerte dal particolare caso o dalla classe di applicazioni [3]. Piuttosto di puntare al circuito completamente «su misura» idealmente sintetizzato dal compilatore di silicio, si fornisce al progettista una «scatola di strumenti», un insieme di programmi in parte specifici per particolari schemi logici come la PLA, che egli utilizzerà al meglio componendoli secondo le sue necessità.

Si intuisce che un approccio di questo genere consente di effettuare passi di «traduzione» ottima in modo automatico solo per quelle fasi in cui l'apporto d'informazione del progettista non è determinante ai fini dell'ottimizzazione; programmi di sintesi esistono per strutture ben precise (PLA, in particolare) e poi per la realizzazione e la compattazione dei collegamenti fra i dispositivi, degli schemi geometrici e delle maschere. Altrimenti, il progettista può scegliere fra una varietà di metodi di progettazione — può utilizzare ROM, PLA, gate arrays, può ricorrere a «policelle» etc. — e organizzare passo per passo il proprio sistema.

Si intuisce subito che questo criterio — più pragmatico del precedente — presenta due potenziali problemi. Innanzitutto, il concetto di «correttezza per costruzione» non è più adottabile, ecco quindi intervenire numerosi programmi di simulazione, a vari livelli (logico, elettrico, temporale (timing), misti etc.). I ricercatori di Berkeley accettano infatti come presupposto che un semplice controllo di aderenza alle regole non sia sufficiente quando il progetto si riferisce a strutture complesse, e che invece possono sussistere condizioni d'ingresso e di funzionamento (verificabili in simulazione) che porterebbero a comportamenti errati.

Il secondo problema deriva dalla stessa flessibilità di un insieme di programmi che possono essere liberamente composti e concatenati dall'utente; è ovvio che una traduzione manuale di formati, basi di dati, informazioni trasferite fra moduli etc. porterebbe a un'alta probabilità di errore.

A questa possibilità vuole ovviare la «CAD shell», cioè un ambiente di programmazione che supporti tutti i vari «stili» progettazione senza rendere necessario nè l'intervento umano di traduzione nè la riscrittura delle parti comuni.

Di fronte al coordinamento rigoroso dei lavori sviluppati nelle due università prima citate, può sembrare che le ricerche svolte al MIT e a Stanford non obbediscono a un piano di base. Di fatto, si è di fronte allo sforzo di un gran numero di gruppi di ricerche, che procedono secondo linee e interessi autonomi, ma contemporaneamente esiste la volontà di giungere a un nuovo tipo di progetto e soprattutto a un nuovo approccio al progetto automatico. Evidentemente, gli studi relativi a quest'ultima problematica sono ancora in una fase iniziale, ma questo non li rende meno interessanti.

Al MIT, esiste un insieme di programmi di simulazione (a livello «rete di commutazione»), di layout automatico, di analisi di correttezza delle maschere, di sintesi di strutture basate su celle predefinite [4]. A fronte di ciò, si distinguono due indirizzi fondamentali nella ricerca di nuovi stili di progetto. Il primo coinvolge il gruppo di Intelligenza Artificiale, che assume il progetto VLSI come un particolare caso del processo umano di progettazione e mira a modellare tale processo per fornire al progettista non tanto un sostituto quanto un «assistente di progetto». L'altro, basato sull'osservazione che un progetto di sistemi complessi quali i dispositivi VLSI trova l'unico precedente comparabile nel progetto di grandi sistemi di programmazione, mira ad adottare i metodi e le esperienze sviluppate in quest'area.

L'approccio in termini di Intelligenza Artificiale e la ricerca di un'opportuna rappresentazione della conoscenza sono fra i temi fondamentali anche a Stanford, università dove due istituti (Computer Science e Electrical Engineering) hanno affrontato «in massa» il problema del CAD per circuiti VLSI [5]. Quest'ultimo fatto comporta fra l'altro l'adozione di strutture di calcolo avanzate (prima fra tutte la rete Ethernet), la tendenza a servirsi di linguaggi moderni (Pascal, sue estensioni o linguaggi tipo Pascal)e a utilizzare cri-

teri già collaudati in ambiti più generali (progettazione di strutture di calcolo, studio di basi di dati, etc.). I vari gruppi procedono in parallelo, identificando propri temi ed approcci senza che ci si pongano problemi di sovrapposizione, incompatibilità o traduzione; si vede un problema di comunicazione fra i gruppi (realizzata dalla rete Ethernet) piuttosto che fra programmi.

Probabilmente è proprio grazie alle diversità dei gruppi di ricerca (e dei loro interessi di partenza) che Stanford studia problemi non affrontati in altre università, anche se di grande importanza industriale. Prima fra questi è il problema del «testing», del collaudo cioè di produzione o di accettazione, ben diverso dalla verifica di congruenza del progetto. Si tratta di un problema complesso, che va dal «progetto per la collaudabilità» al progetto delle macchine per il collaudo, e che Stanford affronta nei suoi vari aspetti.

Una predominanza del tema VLSI altrettanto definita non esiste al momento alla Carnegie Mellon University, le cui attività in questo settore paiono piuttosto estensioni e applicazioni dei lavori in atto ormai da molti anni nei campi delle architetture di calcolo e dei linguaggi per la loro descrizione [6]. Le proposte attuali sono rivolte a ridurre il tempo di progetto e ad aumentare la possibilità di confronto fra diverse alternative, piuttosto che ad ottimizzare un particolare singolo progetto; a questo scopo si propone l'adozione del linguaggio ISP per la descrizione ad alto livello e il successivo uso di programmi per la sintesi automatica dei microprogrammi di controllo e delle unità funzionali mediante moduli predefiniti. Nonostante la mancanza di specificità, è interessante osservare questo approccio sistematico all'uso di strumenti di descrizione funzionale e logica.

3. - Il progetto dei circuiti VLSI.

3.1. - *Il processo di progettazione.*

Risulta dall'esame delle metodologie adottate o studiate nei vari centri di ricerca, la tendenza a convergere verso una struttura comune del processo di progettazione.

La gestione di un progetto molto complesso richiede innanzitutto che esso venga strutturato in fasi distinte in modo che in ciascuna fase si debba considerare un'aspetto soltanto del progetto. Naturalmente la suddivisione in fasi deve complessivamente coprire l'intero progetto e le varie fasi debbono essere collegate.

Le fasi in cui può essere distinto un progetto VLSI possono essere le seguenti:

progetto: la descrizione funzionale
il progetto logico
il progetto circuitale
il disegno delle maschere (layout)
la verifica delle maschere
la simulazione
fabbricazione
collaudo

Il progetto detto di tipo top-down inizia dalle specifiche funzionali del circuito (ad alto livello) e procede attraverso un diverso numero di fasi fino alla rappresentazione fisica del circuito stesso (a basso livello).

La verifica o analisi detta di tipo bottom-up inizia dalla rappresentazione reale (fisica) del circuito comprensiva quindi di tutte le grandezze parassite e permette di simulare le prestazioni ad alto livello del circuito in oggetto.

L'eventuale disaccordo tra prestazioni richieste e prestazioni simulate può mettere in luce le parti del circuito da riprogettare prima di passare alla fase di fabbricazione.

In ciascuna delle fasi sopra indicate vengono adottate diverse rappresentazioni del sistema, a diversi gradi di astrazione, decrescenti col proseguire nelle fasi successive.

In ciascuna fase pertanto il progettista concentra la sua attenzione su una parte dei dati.

3.1.1. - *Le specifiche funzionali.*

L'esigenza di formalizzare la descrizione delle funzioni volute da un circuito integrato è andata crescendo col crescere della complessità delle funzioni stesse. Tutti i dati di ingresso al progetto (per es. il tipo di tecnologia usato) possono considerarsi come facente parte, in quanto influenzeranno in qualche modo la struttura finale.

La descrizione funzionale gioca inoltre un suo ruolo nella prima fase del progetto, in cui questo è decomposto funzionalmente in parti interconnesse, in quanto la descrizione deve poter descrivere tale decomposizione nonchè le interconnessioni.

Un esempio di descrizione funzionale è dato da una funzione booleana da realizzare con un circuito integrato (che può essere una ROM o una PLA). Un altro esempio è dato dalla tabella delle transizioni o da un diagramma degli stati, da realizzare con una macchina a stati finiti.

Un esempio particolarmente importante è dato dalla lista delle istruzioni di un microprocessore: il linguaggio ISP della Carnegie Mellon formalizza la descrizione della lista delle istruzioni nonchè delle altre caratteristiche di un microprocessore (lunghezza di parola, spazio di indirizzamento, algoritmi per il calcolo degli indirizzi, ecc.).

Tale linguaggio risulta essere stato generalizzato per la descrizione di sistemi numerici qualsiasi.

La descrizioni in linguaggio ISP viene tradotta automaticamente in un diagramma funzionale a blocchi, come primo passo del processo di sintesi.

Le specifiche funzionali possono essere descritte con vari «linguaggi»: algebrici, grafici, linguaggi «formali».

La descrizione delle specifiche funzionali dovrebbe, idealmente, essere del tutto indipendente dalla struttura logica interna del sistema da realizzare, dovendo limitarsi a fornire le descrizioni del «comportamento» voluto dal sistema stesso, cioè dei legami tra le entrate e le uscite.

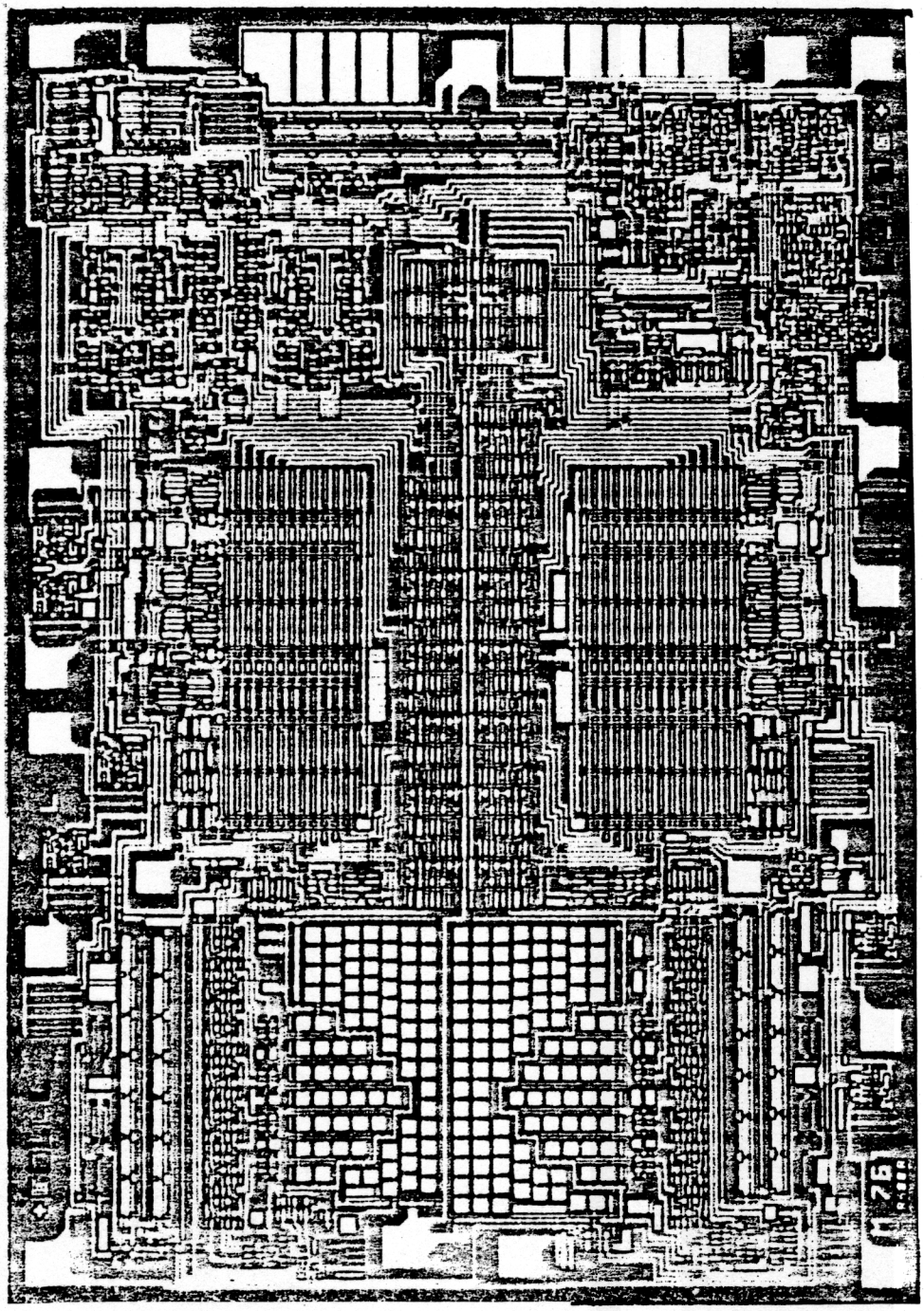

Fig. 2 - Piastrina di un generatore di tono a frequenza multipla M701 realizzato dalla società SGS Ates.

È tuttavia frequente il caso in cui le specifiche funzionali sono fornite con riferimento ad una prefissata struttura logica del sistema, in quanto essa fa parte, in un certo senso, del linguaggio adottato. Come esempio, è comune fornire le specifiche funzionali con un linguaggio che esprime il funzionamento del sistema come trasferimento di informazioni tra registri: si viene così a dettare, insieme al comportamento vero e proprio, e almeno in parte, la struttura interna del circuito, o con altre parole si indica il comportamento «costruendo» un sistema che lo realizza.

Si può considerare tale descrizione come appartenente al campo delle specifiche funzionali, in quanto essa precede il progetto della struttura logica vera e propria, sebbene ne anticipi alcuni caratteri.

Un altro esempio è dato da un linguaggio (SLIM, Stanford) che descrive il comportamento di un circuito in maniera tale che può essere facilmente tradotto in una macchina a stati finiti; esso permette inoltre di descrivere il comportamento delle unità funzionali che interagiscono col circuito in progetto.

Un altro esempio ancora (Stanford) è dato da un linguaggio basato sulle «espressioni regolari», che permette la descrizione delle specifiche funzionali a livello molto alto e che si adatta bene ad alcune classi di sistemi.

Va infine segnalata l'opinione (MIT) che il campo delle specifiche funzionali sia da considerarsi in modo informale, cioè non richiedente l'impiego di linguaggi formali. Esso dovrebbe comprendere tutte le prescrizioni generali cui il progetto deve soddisfare, come le funzioni da calcolare, i vincoli temporali (ritardi, ecc.), la massima potenza dissipata, ecc.

In tali casi, però, si assume come primo passo del progetto la definizione di un diagramma a blocchi, col quale rappresentare il flusso dei dati ed il flusso delle variabili di controllo. Quest'ultimo, è in definitiva, ancora un linguaggio (grafico) per la descrizione delle specifiche funzionali, sotto forma di definizioni di blocchi interconnessi.

3.1.2. - *Dalle specifiche funzionali al disegno delle maschere.*

Alla fase di definizione delle specifiche funzionali del circuito, fa seguito quella detta di «chip planning», cioè la suddivisione del sistema in parti, ciascuna definita funzionalmente e nelle interazioni con le altre parti.

Ciò corrisponde ad un piano di ripartizione della piastrina, con la stima delle aree da assegnare a ciascuna parte e fissando la loro posizione. Tale ripartizione può essere ottimizzata rispetto alla interconnessione tra le parti.

Si noti che la stima delle aree delle varie parti potrà essere fatta sulla base di dati di precedenti progetti.

Alla fase della descrizione e partizione funzionale segue quella del progetto logico, nella quale le funzioni delle parti funzionali costituenti il sistema complessivo vengono realizzate tramite opportuni schemi logici.

Se la suddivisione funzionale è stata molto dettagliata, lo schema logico corrispondente potrà essere elementare, al limite una semplice porta logica. Se invece la suddivisione funzionale è meno dettagliata si avranno schemi logici corrispondenti più complessi, (che possono essere realizzati con PLA, con macchine e stati finiti o con altre strutture più complesse).

Lo schema logico può essere poi tradotto in schema circuitale.

Si noti che in questo verranno adottati opportuni circuiti equivalenti dei vari elementi, ed inoltre, a differenza dello schema logico, si dovranno tenere in conto i carichi elettrici dei vari punti di connessione esterni del blocco considerato.

Il passo successivo è la rappresentazione mediante «stick diagrams»: il circuito è individuato da un insieme di segmenti colorati (sticks) che permettono di definire sia i dispositivi sia la loro posizione realativa nel circuito fisico da realizzare. Vari colori sono messi in corrispondenza coi relativi processi sul silicio, quali la diffusione, l'impianto e la deposizione di linee metalliche o di silicio policristallino.

Successivamente si convertono gli «stick diagrams» in figure geometriche che rappresentano le varie maschere. Questo passo è sovente completamente automatizzato.

È stato creato un linguaggio standard, il CIF, (Caltech Intermediate Form) per la descrizione delle geometrie mediante rettangoli, poligoni e cerchi adiacenti.

Un intero circuito VLSI può essere descritto completamente a livello di CIF. Tale informazione può essere memorizzata su nastro magnetico, trasmessa su una rete di calcolatori del progettista al responsabile della fabbricazione ed infine essere usata, tramite opportuno interprete, per comandare un fotocompositore ottico od un apparecchio per la litografia a fascio di elettroni.

3.2. - *La verifica.*

La descrizione a livello delle geometrie è il punto di partenza per la verifica «bottom-up» dei circuiti VLSI. Appositi programmi sono in grado di estrarre la rappresentazione a livello di circuito elettrico dalle geometrie delle maschere. Questa contiene maggiore informazione riguardo al circuito (ad esempio gli effetti parassiti) e può essere usata come ingresso di simulatori circuitali.

Con l'avvento dei circuiti a larga scala la simulazione è diventata un passo necessario nella fase di progetto. Infatti, così come non è possibile progettare un circuito integrato «con carta e matita», i risultati sperimentali ottenibili con un «bread board» non sono sufficientemente attenibili a causa della diversità della natura fisica del circuito integrato e del suo modello discreto.

Per questo motivo vi è stata negli anni '70 un fiorire di programmi per la simulazione, come per esempio il programma SPICE sviluppato presso l'Università di

California, Berkeley [7] ed il programma ASTAP di proprietà della IBM [8].

Tali simulatori sono stati concepiti per simulare circuiti con un centinaio di dispositivi. Oggi alcuni utenti del programma SPICE simulano con successo circuiti con diverse migliaia di dispositivi.

È però impensabile adottare le tecniche esistenti per circuiti contenenti oltre 100 000 dispositivi, a causa dell'eccessivo tempo di calcolo e spazio di memoria occupato.

Per questo motivo sono allo studio nuovi algoritmi e nuove architetture di sistemi di calcolo. I nuovi algoritmi cercano di sfruttare la struttura del circuito e la sua rappresentazione gerarchica a vari livelli di astrazione. Le nuove architetture sfruttano processori paralleli e diversi tipi di memorie (ad accesso diretto e sequenziale) per permettere l'efficiente esecuzione di speciali simulazioni di circuiti VLSI

Sono stati studiati e realizzati programmi di simulazione per circuiti VLSI relativi ad ogni fase del progetto.

3.2.1. - Simulazione del processo di fabbricazione.

Nella misura in cui la tecnologia di fabbricazione si spinge verso geometrie più piccole, il problema del controllo delle geometrie stesse e dei profili di drogaggio si fa più importante. Modelli matematici ed appositi algoritmi sono stati sviluppati nelle seguenti aree: litografia, incisione, deposizione e processi termici.

Esempi di simulatori di processo sono il programma SUPREM sviluppato dalla Stanford University ed il programma SAMPLE sviluppato presso L'Università della California, Berkeley. Il primo simula la diffusione, l'implantazione e l'ossidazione, mentre il secondo simula la litografia, la deposizione e l'incisione.

3.2.2. - *Simulazione circuitale.*

I simulatori circuitali calcolano la forma d'onda dei vari segnali nel circuito in esame. In generale i simulatori circuitali contengono modelli di vari dispositivi e pertanto sono indipendenti dalla tecnologia usata.

I simulatori circuitali utilizzano i classici algoritmi per la soluzione del sistema sparso di equazioni differenziali non lineari che descrive il funzionamento del circuito.

Varie tecniche sono state realizzate nei simulatori circuitali per diminuirne il tempo di calcolo. Tra queste è stato introdotto l'uso di tabelle per la rappresentazione delle caratteristiche dei dispositivi. Ciò migliora il tempo di esecuzione a spese di una maggiore occupazione di memoria.

Alla Università di California, Berkeley, si è sperimentato con successo la realizzazione con microprogramma delle istruzioni per la soluzione del sistema sparso di equazioni del programma SPICE.

Sono allo studio speciali calcolatori numerici finalizzati alla simulazione circuitale. Nuove architetture utilizzanti molti processori in parallelo possono sfruttare la particolare struttura del circuito da analizzare per ridurre i tempi di calcolo. Per esempio è possibile effettuare una decomposizione a blocchi delle equazioni e risolvere in parallelo le equazioni dei vari blocchi. In questa prospettiva, si sta progettando una nuova versione del programma SPICE adatta a calcolatori vettoriali del tipo CRAY 1.

Sono inoltre allo studio circuiti integrabili dedicati alla soluzione di problemi altamente ricorrenti nelle simulazioni quali la soluzione di sistemi di equazioni algebrici lineari.

3.2.3. - *La simulazione temporale.*

La simulazione circuitale in transitorio di circuiti a larga scala (più di 10 000 dispositivi) risulta problematica per l'eccessivo tempo di esecuzione e lo spazio di memoria necessario. Per questo motivo sono stati realizzati dei simulatori, detti «temporali» (timing simulators), dedicati all'analisi nel dominio del tempo di circuiti digitali a larga scala [9].

Il primo simulatore temporale, il programma MOTIS, fu sviluppato con tecniche rivoluzionarie in due riguardi:

a) si limita il tipo di circuiti da analizzare (circuiti MOS quasi unidirezionali e con un condensatore a terra in ciascun nodo)
b) si scartano sia l'algoritmo di Eliminazione Gaussiana sparsa sia l'iterazione di Newton-Raphson come metodi risolutivi.

Nel programma MOTIS sono usate tecniche di rilassamento per disaccoppiare le equazioni nodali. Gli algoritmi dei simulatori temporali MOTIS-C e SPLICE perfezionano questa tecnica.

In particolare il programma SPLICE unisce l'algoritmo di rilassamento di tipo Gauss-Seidel ad un algoritmo detto di «traccia selettiva» che ordina i nodi del circuito secondo il propagarsi del segnale. Si ottiene così una riduzione del tempo di calcolo di circa due ordini di grandezza rispetto ai normali simulatori circuitali.

3.2.4. - *Simulazione logica e mista.*

La simulazione temporale non è comunque sufficientemente veloce per poter analizzare un intero circuito VLSI. D'altro canto non sempre è necessario conoscere le tensioni nodali di tutto il circuito.

La simulazione logica dà un'informazione sintetica della tensione nodale, ad esempio: alta, bassa, crescente, decrescente ...

Tale informazione può essere sufficiente per verificare il funzionamento di circuiti logici. Al medesimo tempo tale informazione può essere ottenuta con un minimo di memoria (da uno a tre bits per porta logica da analizzare) e con un tempo di analisi molto inferiore alle simulazioni circuitali e temporali.

Per questo motivo alcuni simulatori logici quali il LOGIS sono usati con successo per l'analisi di interi circuiti VLSI digitali.

Grande interesse rivestono i simulatori misti, che permettono la simulazione simultanea di sottocircuiti ciascuno con una diversa rappresentazione. Per esem-

pio una memoria a lettura-scrittura può essere simulata a livello logico nel decodificatore di indirizzo, e con analisi circuitale negli amplificatori differenziali (che hanno alto guadagno e forti reazioni) e con analisi temporali nelle interconnessioni.

Il simulatore misto SPLICE, sviluppato a Berkeley, permette la simulazione a livello logico, temporale e circuitale.

Altri esempi di simulatori misti sono il programma DIANA, sviluppato presso l'Università di Lovanio ed il sistema ADLIB della Stanford University.

3.3. - *Collaudabilità.*

Mentre il problema di un «progetto per la collaudabilità» è da anni sentito nell'industria dei calcolatori e dei sistemi digitali in genere, non si assiste per il momento ad un interesse comparabile negli ambienti universitari dove il problema VLSI viene affrontato. Di fatto, fra le cinque università americane impegnate in questo sforzo solo Stanford dedica un certo impegno al tema collaudo seguendo i due filoni — altrettanto importanti — di un progetto volto a rendere facile il collaudo e dello sviluppo di macchine e procedure per il collaudo. È bene sottolineare che il collaudo di produzione e di accettazione resta indispensabile anche nella ottica di un «progetto corretto per costruzione»; d'altra parte, nelle conferenze specializzate si è ormai largamente affermata la convinzione che tecniche e modelli tradizionali diventano totalmente insoddisfacenti a fronte delle problematiche introdotte dalla VLSI.

Al momento si vedono due tendenze principali a livello di progetto dei dispositivi. La prima orienta il progettista a realizzare una struttura che consenta facilità di collaudo, usando in larga misura tecniche di microprogrammazione per l'unità di controllo (è un criterio già seguito al microprocessore Motorola 68000) e fornendo il dispositivo di comandi capaci di «spezzare» gli anelli di reazione (tipico l'approccio IBM, secondo cui tutte le strutture sequenziali devono poter essere riorganizzate in registri a scorrimento per la fase di collaudo). La seconda tendenza inserisce capacità di partecipazione attiva alla diagnosi nella stessa architettura VLSI, fino a giungere a dispositivi totalmente auto-diagnosticanti e a «tradurre» sulla piastrina le tecniche più collaudate di tolleranza ai guasti (lavori in questo senso sono stati proposti, es., da General Motor e in Europa da LAAS di Tolosa).

Per quanto riguarda le tecniche di collaudo, esiste una generale tendenza verso criteri di tipo funzionale (il collaudo strutturale, oltre a non aver pratica significatività per i tempi richiesti, è spesso impossibile perchè non si conosce la struttura interna con sufficiente dettaglio); non è stata però presentata finora una metodologia generalmente applicabile che consenta anche la produzione automatica dei programmi e delle sequenze di collaudo. Di fatto, anche il problema fondamentale di un modello di guasto, accettabile per un approccio funzionale, è ancora aperto nonostante le proposte interessanti fatte da alcuni ricercatori (in particolare Thatte e Abrahams [10]).

3.4. - *La realizzazione in VLSI di algoritmi «paralleli».*

È noto come la tradizionale struttura dei calcolatori elettronici (naturalmente, anche degli ultimi microprocessori) sia rimasta fondamentalmente la stessa, concepita da Von Neumann, determinata da un «programma» composto da una serie di istruzioni elementari eseguite serialmente. La stessa struttura trova il suo fondamento teorico nella più elementare macchina di Turing, concepita come mezzo concettuale per la definizione degli algoritmi.

Di conseguenza, esiste una stretta parentela tra algoritmi e calcolatori, il che spiega il successo di questi nelle soluzioni di problemi di ogni genere (purchè tradotti in algoritmi). Tutto ciò spiega anche l'abitudine invalsa di concepire gli algoritmi ed i programmi esclusivamente in modo «seriale».

Da qualche tempo si sta considerando il problema della esecuzione «in parallelo» di algoritmi e di programmi, essenzialmente per aumentare la velocità di elaborazione rispetto al caso puramente seriale.

La suddetta questione ha rilevanza anche nella concezione dei sistemi VLSI. Si possono infatti concepire strutture o «stili» diversi, a seconda del modo con cui viene concepita l'esecuzione dell'algoritmo richiesto. Tra i vari «stili» proposti, ne esiste almeno uno che riproduce sostanzialmente la struttura di Von Neumann e nel quale perciò gli algoritmi vengono realizzati tramite un programma seriale registrato in ROM. Con tale sistema potrà, per es. ottenersi l'esecuzione dell'algoritmo della Trasformata veloce di Fourier (FFT).

Lo stesso algoritmo può, tuttavia, essere realizzato con strutture del tutto diverse, in base a schemi ben noti, composti da reti di sommatori e moltiplicatori funzionanti in parallelo.

La scelta dell'una o dell'altra soluzione dipenderà da vari fattori, come la flessibilità di impiego, la facilità di progettazione, la velocità richiesta, ecc.

Quanto sopra è stato detto per mostrare come ricerche già in atto per altre motivazioni (per la programmazione per sistemi in «tempo reale») ed attinenti alla concezione di algoritmi e programmi «in parallelo» possa avere importanti riflessi anche nella concezione di sistemi VLSI, il cui grande numero di componenti può essere utilmente messo a frutto con algoritmi capaci di alto grado di parallelismo.

Ricerche di carattere teorico di base sono peraltro necessarie per tutte le fasi di progetto dei circuiti VLSI.

4. - Gli strumenti per la progettazione automatica di circuiti VLSI.

4.1. - *La stazione di progetto.*

La progettazione di circuiti VLSI negli anni 80 richiederà un uso massiccio di strumenti di calcolo. È opportuno creare quindi un ambiente che mette il progettista in grado di servirsi al massimo dei mezzi di-

sponibili. Ciascun appartenente ad un gruppo di progetto deve poter lavorare indipendentemente su varie parti del circuito ed interagire facilmente coi colleghi per lo scambio di informazioni.

Si pensa che negli anni 80 ciascun progettista debba avere una propria «stazione di progetto», interagente con le altre mediante una rete di comunicazione [3] [11].

Ciascuna stazione di progetto è costituita da un minicalcolatore con una propria memoria di massa (disco) e che colloquia con l'utente mediante un terminale alfanumerico ed un terminale grafico a colori.

Le varie stazioni di progetto sono poi collegate ad un potente calcolatore numerico, ad un certo numero di stampanti e tracciatori (a colori) ed ad un sistema di memorie di massa che possa contenere le librerie di dati e di programmi utili per il progetto.

4.2. - *Il sistema CAD.*

L'insieme dei programmi CAD per il progetto di circuiti VLSI deve costituire un sistema di facile impiego per i progettisti. Si richiede quindi che il sistema sia «amichevole» nei riguardi dell'utente, che abbia un'efficiente base di dati e che si possa effettuare un'agile manutenzione dei programmi. Per questi motivi i programmi CAD devono essere modulari, in modo da poter essere sviluppati e modificati da persone diverse e parallelamente. Inoltre l'intero sistema dev'essere flessibile, facilitando un trasporto su nuove macchine appena queste siano disponibili.

Ci sono due soluzioni a questo problema. Il primo approccio è di sviluppare il software in maniera tale che sia trasportabile tra diversi sistemi operativi con modeste modifiche. Questa soluzione non permette l'uso di programmi che sfruttino le funzioni a basso livello del sistema operativo, e quindi può risultarne una realizzazione inefficiente. Con i recenti sviluppi delle tecnologie dei sistemi operativi, è possibile trasportare facilmente un intero sistema operativo da una macchina ad un'altra. Per questo motivo, una seconda soluzione al problema prevede di sviluppare del software che sfrutti le caratteristiche del sistema operativo, e poi trasportare l'intero sistema operativo contenente il sistema CAD da una macchina ad un'altra. Numerose università e gruppi industriali statunitensi hanno optato per la seconda soluzione, adottando il sistema operativo UNIX (*) come veicolo di sviluppo.

Il sistema di files UNIX, organizzato ad albero, permette un'efficiente struttura dei dati di progetto dei circuiti VLSI. Ogni nodo dell'albero dei files UNIX può essere messo in corrispondenza con un componente, o modulo, del circuito in fase di progetto. Ogni nodo è un indice di files, ciascuno di questi contenente informazione relativa ad un particolare aspetto del modulo.

Ad esempio, ci possono essere files contenenti la descrizione del tracciato della maschera, files di ingresso per il simulatore, ... Lati orientati rappresentano riferimenti a sottomoduli. Cambiare un progetto implica principalmente cambiare la struttura dei lati dell'albero.

I programmi CAD interagiscono con questa semplice base di dati sotto il controllo di un programma supervisore, responsabile di mantenere le relazioni tra i livelli di rappresentazione e di aggiornare le descrizioni dopo aver effettuato un cambiamento.

Per aiutare il progettista ad utilizzare gli strumenti CAD, viene introdotta un'interfaccia ad alto livello, chiamata CADSHELL. In questa «shell» un semplice comando dell'utente si traduce in una chiamata di differenti programmi nel sistema. Con l'aggiunta o la modifica di nuovi programmi nel sistema, oppure cambiamenti di tecnologia, si può modificare l'interfaccia CADSHELL, minimizzando così l'impatto del cambiamento sul progettista.

Sebbene la realizzazione della CADSHELL e di un programma supervisore siano ancora nella fase di ricerca e sperimentazione, diverse particolarità del sistema operativo UNIX come l'uso di «shells» e del programma «MAKE» come supervisore sono impiegate attualmente in questa prospettiva.

5. - Conclusioni.

La progettazione dei circuiti VLSI è molto complessa. Le differenti metodologie di progetto fanno capo comunemente ad un procedimento gerarchico. Il progetto si sviluppa attraverso varie fasi, ad ognuna delle quali compete una rappresentazione ad un diverso livello di astrazione. Il passaggio da una fase ad un'altra è sovente automatizzato e complessi programmi di progettazione automatica svolgono un ruolo predominante nella sintesi «top-down» del circuito e nella sua verifica «bottom-up». Questi programmi fanno parte di un sistema organico di progettazione che risiede su di una apposita stazione di progetto.

Manoscritto pervenuto il 2 aprile 1982.

(*) UNIX è un marchio dei Laboratori Bell.

BIBLIOGRAFIA

[1] Carver Mead, Lynn Conway: «Introduction to VLSI systems». Addison Wesley 1980.

[2] Trimberger, J. Rowsin, C.R. Lang and J.P. Gray: «A Structured Design Methodology and Associated Software Tools». IEEE Trans Circuits Syst vol. CAS 28 pp 618-633 July 81.

[3] A.R. Newton D.O. Pederson A.L. Sangiovanni-Vincentelli and C.H. Sequin: «Design Aids for VLSI: the Berkeley Perspective». IEEE Trans Circuits Syst vol CAS 28 pp 666-680 July 81.

[4] J. Allen and P. Penfild: «VLSI Design Automation Activities at Mit». Trans Circuit Syst vol CAS 28 pp 645-653 July 81.

[5] R.W. Dutton: «Stanford Overview in VLSI Research». IEEE Trans Circuit Syst vol CAS 28 pp 645-653 july 81.

[6] S.E. Director, A.C. Parker, D.P. Siewiorek and P.E. Thomas: «A Design Methodology and Computer Aids for Digital VLSI Systems». IEEE Trans Circuit Syst vol CAS 28 pp 634-644 July 81.

[7] W. Nagel: «SPICE2», A Computer Programm To Simulate Semiconducture Circuits» University of California, Berkely, ERL Memo M520, 1975.

[8] «Advanced Statiatical Analysys Program». Program reference manual. Pub N0 SM20-1118-0 IBM Corp. Data Proc. Div. White Plains, NY 10604.

[9] Giovanni De Micheli, Alberto Sangiovanni-Vincentelli: «Numerical Properties of Algorithms for the Timing Analysis of MOS VLSI Circuits» ECCTD, L'Aia 1981.

[10] S.M. Thatte, J.A. Abraham: «Test Generation for General Microprocessors Architectures». FTCS-9, 1979, pp. 203-210.

[11] G. De Micheli: «Progetto di Sistemi VLSI Assistito da Calcolatore». Pixel n. 2, 1982.